УДК 004.832.23; 519.245

ПРИМЕНЕНИЕ МЕТОДА МОНТЕ-КАРЛО К ПРОГНОЗИРОВАНИЮ ВРЕМЕНИ ПАРАЛЛЕЛЬНОГО РЕШЕНИЯ ПРОБЛЕМЫ БУЛЕВОЙ ВЫПОЛНИМОСТИ

О. С. Заикин[1], **А. А. Семенов**[2]

Рассматривается применение метода Монте-Карло к планированию решения сложных вариантов задачи о булевой выполнимости (SAT, Boolean Satisfiability) в параллельных вычислительных системах. Распараллеливание SAT-задачи является результатом выделения в множестве булевых переменных исходной конъюнктивной нормальной формы некоторого подмножества, называемого декомпозиционным множеством. Для декомпозиционных множеств можно естественным образом определить ряд параметров, характеризующих "качество" декомпозиции. Для оценки этих параметров предлагается использовать вычислительную схему метода Монте-Карло. В частности, данный метод применен для поиска декомпозиционного множества с наименьшим прогнозным временем решения исходной задачи. Реализована параллельная MPI-программа, с помощью которой на вычислительном кластере был получен прогноз времени решения задачи логического криптоанализа шифра Bivium. Успешно осуществлен логический криптоанализ нескольких ослабленных версий шифра Bivium, проведено сравнение реального времени криптоанализа с прогнозным.

Ключевые слова: задача выполнимости булевых формул, метод Монте-Карло, поиск с запретами, MPI, криптоанализ, шифр Bivium.

1. Введение. Многие сложные комбинаторные задачи могут быть эффективно сведены к задаче о выполнимости булевых формул (SAT, Boolean Satisfiability). В подавляющем большинстве случаев под SAT понимается задача поиска выполняющего набора либо доказательство невыполнимости конъюнктивной нормальной формы (КНФ). Программы, решающие данную задачу, называются SAT-решателями. Несмотря на то что SAT-задача является NP-трудной (NP-hard), прогресс в алгоритмике SAT-решателей, достигнутый за последнее десятилетие, настолько существен, что позволяет использовать SAT-подход для решения комбинаторных задач из целого ряда областей (символьная верификация, криптография, биоинформатика, комбинаторика и др.).

Поскольку SAT-задачи являются вычислительно трудными, для их решения оправдано применение параллельных вычислений. Статья [1] была одной из первых работ, в которой рассматривались общие проблемы параллельной реализации алгоритма DPLL (Davis-Putnam-Logemann-Loveland Algorithm), являющегося базовым алгоритмом для большинства современных эффективных SAT-решателей. Некоторые общие стратегии распараллеливания SAT по данным впервые, по-видимому, были представлены в статье [2]. В дальнейшем проблемам решения SAT посредством разбиения области поиска на непересекающиеся подобласти с последующей параллельной обработкой полученных подобластей был посвящен ряд работ этого же коллектива [3–5].

В настоящей статье мы описываем подход, который так же, как и методы работ [2–5], базируется на распараллеливании по данным, но существенно отличается от этих методов. В этом подходе используется этап препроцессинга, на котором осуществляется минимизация специальным образом определяемой прогнозной функции. Это позволяет найти такое разбиение пространства поиска на подобласти, которое имеет относительно хорошее прогнозное время параллельной обработки полученных подобластей. Для ряда трудных SAT-задач посредством данного подхода удалось найти декомпозиции, позволившие реально решать такие задачи на современных параллельных вычислительных системах.

Основа используемого нами подхода была заложена в работах [6–9]. В них параллельные алгоритмы решения SAT были применены к задачам криптоанализа ряда поточных шифров, самым известным из которых является генератор ключевого потока A5/1. В статье [10] были приведены теоретические основы подхода к статистическому оцениванию качества декомпозиции SAT-задачи. Этот подход базируется на

[1] Институт динамики систем и теории управления СО РАН (ИДСТУ СО РАН), ул. Лермонтова, 134, 664033, Иркутск; науч. сотр., e-mail: zaikin.icc@gmail.com

[2] Институт динамики систем и теории управления СО РАН (ИДСТУ СО РАН), ул. Лермонтова, 134, 664033, Иркутск; зав. лабораторией, e-mail: biclop.rambler@yandex.ru

широко используемой вычислительной схеме, известной как "метод Монте-Карло" [11]. В статье [12] были приведены результаты первых вычислительных экспериментов по реализации этого подхода, а также результаты его апробации на задаче поиска декомпозиционного множества для параллельного логического криптоанализа генератора A5/1.

В настоящей статье мы описываем также новый алгоритм построения декомпозиционных множеств для SAT-задач, в основе которого лежит стратегия поиска с запретами (tabu search). Полученная реализация метода оказалась существенно более эффективной, чем вариант, описанный в [12], где для поиска декомпозиционного множества с "хорошим" значением прогнозной функции использовался метод имитации отжига. Предложенная схема была реализована в виде MPI-приложения для вычислительного кластера. В вычислительных экспериментах была рассмотрена задача логического криптоанализа известного поточного шифра Bivium [13].

Статья имеет следующую структуру. В разделе 2 описана общая процедура оценивания качества декомпозиционного множества, базирующаяся на методе Монте-Карло. Раздел 3 содержит описание нового алгоритма минимизации прогнозной функции, использующего стратегию поиска с запретами. В разделе 4 описаны детали программной реализации данного алгоритма. Кроме того, в разделе 4 приведены результаты вычислительных экспериментов по применению предложенного метода к задаче логического криптоанализа поточного шифра Bivium.

2. Метод Монте-Карло прогнозирования времени решения SAT-задач. В данном разделе приведены краткие математические основы используемого нами метода, при этом мы придерживаемся терминологии работ [5] и [8]. Пусть C — произвольная конъюнктивная нормальная форма (КНФ) над множеством булевых переменных X. Рассматриваем задачу выполнимости КНФ C, т.е. задачу поиска решений (или доказательства их отсутствия) булевого уравнения $C = 1$.

Сопоставим КНФ C множество формул $C \cdot F_i$, $i \in \{1, \ldots, S\}$, таких, что для любых i, $j : i \neq j$, формула $C \cdot F_i\, F_j$ невыполнима и $C \equiv C \cdot F_1 \vee \ldots \vee C \cdot F_S$.

Множество формул $\{C \cdot F_i\}_{i=1}^{S}$ называется декомпозиционным семейством. Несложно понять, что исходная SAT-задача не имеет решений тогда и только тогда, когда все формулы из декомпозиционного семейства являются невыполнимыми. SAT-задачи для формул из декомпозиционного семейства могут быть решены независимо друг от друга на параллельной вычислительной системе.

Существуют разные подходы к построению декомпозиционных семейств. В основе ряда техник лежит использование информации, которую выдает решатель на исходной SAT-задаче [5, 14, 15]. К сожалению, данные техники в общем случае не дают возможности эффективно оценить время, необходимое для обработки всего декомпозиционного семейства. Идейная основа метода, позволяющего строить такого рода оценки, была описана в ряде статей, посвященных логическому криптоанализу некоторых поточных шифров. В статьях [16, 17] было предложено строить прогноз времени решения исходной SAT-задачи на основе знания времени решения некоторого относительно небольшого множества задач, выбранных из декомпозиционного семейства случайным образом. В дальнейшем этот же подход был использован для оценок времени криптоанализа шифра Bivium в работах [18, 19].

В соответствии с указанным методом, из множества переменных исходной КНФ выбирается некоторое подмножество $\widetilde{X} = \{x_{i_1}, \ldots, x_{i_d}\}$, $\widetilde{X} \subseteq X$, называемое декомпозиционным множеством. Формулы F_i, $i \in \{1, \ldots, 2^d\}$, — это все возможные полные конъюнкты над множеством $\widetilde{X}$. Затем решаются $N \ll 2^{|\widetilde{X}|}$ SAT-задач для КНФ вида $C \cdot F_{i_j}$, $j \in \{1, \ldots, N\}$, таких, что $F_{i_1}, \ldots, F_{i_N}$ случайным образом выбраны из множества $\{F_1, \ldots, F_{2^d}\}$. Зная среднее время решения этих задач, мы можем оценить общее время, необходимое для решения исходной SAT-задачи.

В работе [10] было показано, что эта общая идея может быть вполне строго формализована в рамках вычислительной схемы, известной как метод Монте-Карло. Далее мы кратко останавливаемся на этих моментах.

Итак, рассматриваем задачу выполнимости произвольной КНФ $C = C(X)$, где $X = \{x_1, \ldots, x_n\}$ — множество входящих в C булевых переменных. Пусть $\widetilde{X} = \{x_{i_1}, \ldots, x_{i_d}\}$, $\widetilde{X} \subseteq X$, — произвольное декомпозиционное множество. Через $\{0,1\}^d$, $d \in N$, обозначается множество всех двоичных последовательностей длины d. Обозначим через $C\big[\widetilde{X}/(\alpha_1, \ldots, \alpha_d)\big]$, $\alpha_j \in \{0,1\}$, $j \in \{1, \ldots, d\}$, такую КНФ, которая является результатом подстановки в C значений $x_{i_1} = \alpha_1, \ldots, x_{i_d} = \alpha_d$. Множество всех КНФ вида $\Delta\big(C, \widetilde{X}\big) = \Big\{C\big[\widetilde{X}/(\alpha_1, \ldots, \alpha_d)\big]\Big\}_{(\alpha_1, \ldots, \alpha_d) \in \{0,1\}^d}$, как легко видеть, является декомпозиционным семейством. Далее называем его декомпозиционным семейством, порожденным множеством $\widetilde{X}$.

Рассмотрим некоторый алгоритм A решения SAT-задач. Полагаем, что A — детерминированный и полный, т.е. останавливающийся на любых входах, алгоритм. Обозначим через $t_A\big(C, \widetilde{X}\big)$ время, которое

потребуется алгоритму A для решения всех SAT-задач из семейства $\Delta\big(C,\widetilde{X}\big)$. Основная исследуемая далее проблема состоит в получении численных оценок величины $t_A\big(C,\widetilde{X}\big)$.

Зададим на $\{0,1\}^d$ равномерное распределение. Каждому набору $(\alpha_1,\ldots,\alpha_d)$, выбираемому случайно из $\{0,1\}^d$, сопоставим число $\xi_A(\alpha_1,\ldots,\alpha_d)$ — время работы алгоритма A на входе $C\big[\widetilde{X}/(\alpha_1,\ldots,\alpha_d)\big]$. Тем самым задана случайная величина $\xi_A\big(C,\widetilde{X}\big)=\big\{\xi_A(\alpha_1,\ldots,\alpha_d)\big\}_{(\alpha_1,\ldots,\alpha_d)\in\{0,1\}^d}$, имеющая некоторое вероятностное распределение. Поскольку A — полный алгоритм, то $\xi_A\big(C,\widetilde{X}\big)$ имеет конечное математическое ожидание $\mathrm{M}\Big[\xi_A\big(C,\widetilde{X}\big)\Big]$ и конечную дисперсию $\mathrm{D}\Big[\xi_A\big(C,\widetilde{X}\big)\Big]$. Так как A — детерминированный алгоритм, то можно рассматривать N независимых наблюдений значения $\xi_A\big(C,\widetilde{X}\big)$ как одно наблюдение N независимых случайных величин, распределенных по тому же закону, что и $\xi_A\big(C,\widetilde{X}\big)$.

Несложно показать [10], что имеет место равенство

$$t_A\big(C,\widetilde{X}\big)=2^d\,\mathrm{M}\Big[\xi_A\big(C,\widetilde{X}\big)\Big]. \tag{1}$$

Декомпозиционное множество $\widetilde{X}\in 2^X$, для которого величина (1) минимальна, далее называется оптимальным.

В соответствии с методом Монте-Карло [11, 20], для приближенного вычисления математического ожидания $\mathrm{M}[\xi]$ случайной величины ξ используется вероятностный эксперимент, представляющий собой серию из N независимых наблюдений величины ξ. Пусть $\xi^1,\ldots,\xi^N$ — результаты соответствующих наблюдений, которые можно рассматривать как однократную реализацию N независимых, но одинаково распределенных случайных величин, т.е. $\mathrm{M}[\xi]=\mathrm{M}\big[\xi^1\big]=\ldots=\mathrm{M}\big[\xi^N\big]$, $\mathrm{D}[\xi]=\mathrm{D}\big[\xi^1\big]=\ldots=\mathrm{D}\big[\xi^N\big]$.

Если $\mathrm{M}[\xi]$ и $\mathrm{D}[\xi]$ конечны, то из центральной предельной теоремы [21] следует основная формула метода Монте-Карло:

$$\Pr\left\{\left|\frac{1}{N}\sum_{j=1}^{N}\xi^j-\mathrm{E}[\xi]\right|<\frac{\delta_\gamma\cdot\sigma}{\sqrt{N}}\right\}=\gamma. \tag{2}$$

Здесь $\sigma=+\sqrt{\mathrm{D}(\xi)}$ — среднеквадратическое отклонение, γ — уровень значимости, $\gamma=\Phi(\delta_\gamma)$, где Φ — функция Лапласа. Соотношение (2) означает, что при всех сделанных предположениях величина (выборочное среднее) $\overline{\xi}=\dfrac{1}{N}\cdot\sum\limits_{j=1}^{N}\xi^j$ хорошо приближает $\mathrm{M}[\xi]$ при достаточно большом числе наблюдений N. Для каждого конкретного значения N качество этого приближения зависит от величины $\mathrm{D}[\xi]$. На практике в роли оценки $\mathrm{D}[\xi]$ обычно используется следующая величина (исправленная выборочная дисперсия): $s^2=\dfrac{1}{N-1}\sum\limits_{j=1}^{N}\big(\xi^j-\overline{\xi}\big)^2$. В нашем случае особенно важно, что N может быть существенно меньше, чем 2^d. Это дает возможность использовать этап препроцессинга для оценивания величины (1).

Итак, в соответствии со сказанным, процесс приближенного вычисления величины (1) для конкретного $\widetilde{X}$ выглядит следующим образом. Выбираем случайным образом N наборов значений переменных, входящих в $\widetilde{X}$:

$$\Theta\big(\widetilde{X}\big)=\Big\{\big(\alpha_1^1,\ldots,\alpha_d^1\big),\ldots,\big(\alpha_1^N,\ldots,\alpha_d^N\big)\Big\}. \tag{3}$$

Будем рассматривать случайные величины $\xi^j=\xi_A\big(\alpha_1^j,\ldots,\alpha_d^j\big)$, $j=1,\ldots,N$, и вычислим величину $F_{A,C}\big(\widetilde{X}\big)=2^d\left(\dfrac{1}{N}\sum\limits_{j=1}^{N}\xi^j\right)$. В силу сказанного выше, при большом N значение $F_{A,C}\big(\widetilde{X}\big)$ является хорошим приближением величины (1).

Функцию $F_{A,C}$ далее называем прогнозной функцией. Отметим, что прогнозную функцию можно вычислять на обычном компьютере или вычислительном кластере, дополнительно необходим лишь качественный генератор псевдослучайных чисел. Тем самым мы можем от задачи поиска оптимального декомпозиционного множества перейти к задаче поиска декомпозиционного множества с минимальным значением прогнозной функции.

3. Алгоритм минимизации прогнозной функции. Несмотря на естественную природу прогнозной функции, проблема ее минимизации имеет ряд специфических особенностей. Во-первых, легко понять, что $F_{A,C}$ не всегда вычисляется эффективно — можно построить такое небольшое $\widetilde{X}$, что необходимое

для вычисления $F_{A,C}\big(\widetilde{X}\big)$ время будет сравнимо с временем, необходимым для решения исходной SAT-задачи. Во-вторых, $F_{A,C}\big(\widetilde{X}\big)$ — это время работы вычислительной среды на случайной выборке задач из семейства, порождаемого декомпозиционным множеством $\widetilde{X}$. Таким образом, функция $F_{A,C}$ не задана в виде формулы и, следовательно, методы, оперирующие аналитическими свойствами целевой функции, не могут быть применимы для ее минимизации.

В статье [12] для минимизации функции $F_{A,C}$ был использован метод имитации отжига. Далее предлагается новый алгоритм минимизации $F_{A,C}$, основанный на стратегии локального поиска с запретами. Данный алгоритм имеет существенно более высокую эффективность, чем описанный в [12]. Причины этого будут объяснены далее.

Поскольку функция $F_{A,C}$ не задана выражением, самый естественный подход к ее минимизации — это стратегия последовательных улучшений значений $F_{A,C}$. Согласно этой стратегии на первом шаге мы должны построить начальное декомпозиционное множество $\widetilde{X}_0$, для которого значение $F_{A,C}\big(\widetilde{X}_0\big)$ может быть вычислено быстро. После этого мы пытаемся улучшить это значение, проверяя окрестность точки, соответствующей $\widetilde{X}_0$, в некотором пространстве поиска. Таким образом, минимизация функции $F_{A,C}$ — это итерационный процесс, состоящий из переходов от точки к точке в некотором конечном пространстве, обозначаемом далее через $\Re$.

В ситуациях, когда значение минимизируемой функции эффективно вычисляется в любой точке пространства поиска, допускается многократное ее вычисление в одной и той же точке. Исходя из упомянутых выше причин, в нашем случае это крайне нежелательно. В предлагаемом далее алгоритме мы храним все точки, в которых значение прогнозной функции уже было посчитано. Это естественным образом соответствует базовой идее поиска с запретами [22].

Начнем с определения пространства поиска. Отметим, что произвольное множество $\widetilde{X} \in 2^X$ может быть описано при помощи булевого вектора

$$\chi\big(\widetilde{X}\big) = \chi = (\chi_1, \ldots, \chi_n), \quad \chi_i = \begin{cases} 1, & x_i \in \widetilde{X}, \\ 0, & x_i \notin \widetilde{X}, \end{cases} \quad i = 1, \ldots, n. \tag{4}$$

В этом случае пространство поиска $\Re$ является n-мерным булевым гиперкубом $E^n = \{0,1\}^n$. Для произвольной точки $\chi \in E^n$ окрестность $\mathcal{N}_\rho(\chi)$ радиуса ρ определяется как множество векторов χ' из E^n, таких, что $\mathrm{dist}_H(\chi', \chi) \leqslant \rho$, где $\mathrm{dist}_H(\chi_1, \chi_2)$ — расстояние Хэмминга между χ_1 и χ_2. Проколотая окрестность точки χ — это множество $\mathcal{N}^*_\rho = \mathcal{N}_\rho(\chi) \setminus \{\chi\}$. Далее для сокращения записи через $F(\chi)$ обозначается $F_{A,C}\big(\widetilde{X}\big)$, где $\chi = \chi\big(\widetilde{X}\big)$ определяется в соответствии с (4). Мы рассматриваем задачу поиска минимума функции F над E^n.

Простая стратегия локального поиска [23] обычно останавливается после нахождения локального экстремума. Существуют разнообразные техники, которые позволяют выходить из таких точек. В соответствии с основным принципом поиска с запретами, осуществляется переход от точки локального экстремума к некоторой точке, которая не включена в текущий список запретов. При этом может оказаться, что значение целевой функции в новой точке хуже, чем известное лучшее ее значение (текущий рекорд). После перехода к новой точке запускается этап локального поиска в проколотой окрестности этой точки.

Существующие способы построения и использования списков запретов весьма разнообразны и, как правило, определяются индивидуальными особенностями задачи. Для наших целей удобно хранить пройденные точки в виде двух списков L_1 и L_2. Список L_1 хранит такие точки $\chi \in E^n$, что для любой $\chi' \in \mathcal{N}_\rho(\chi)$ значение $F(\chi')$ уже было посчитано. В списке L_2 мы храним точки $\chi \in E^n$, такие, что $F(\chi)$ было посчитано, но существует $\chi' \in \mathcal{N}_\rho(\chi)$, для которой $F(\chi')$ еще не вычислялось. Для отображения этой информации каждая точка в L_2 представлена двумя булевыми векторами: вектором $\chi \in E^n$ и вектором $\theta(\chi)$ длины $\sum\limits_{i=1}^{\rho} \binom{n}{i}$. Данный вектор хранит информацию о всех точках из проколотой окрестности точки χ: любая компонента $\theta(\chi)$, равная 1, отмечает точку из $\mathcal{N}^*_\rho(\chi)$, значение F в которой было вычислено; нулевые компоненты соответствуют точкам, значение F в которых не вычислялось.

Каждый новый центр окрестности локального поиска выбирается из окрестности некоторой точки, находящейся в списке L_2. После вычисления значения F во всех точках из $\mathcal{N}_\rho(\chi)$ точка χ удаляется из списка L_2 и переносится в список L_1. Такая организация данных дает возможность для каждой новой точки быстро проверять, вычислялось ли в ней значение прогнозной функции на предыдущих итерациях.

Собственно процесс минимизации прогнозной функции в описываемом далее алгоритме разделен на две стадии. На начальной стадии значение прогнозной функции вычисляется для всех точек из рассматриваемой окрестности $\mathcal{N}_\rho(\chi)$. Это целесообразно ввиду того, что при удачном выборе $\widetilde{X}_0$ почти все

SAT-задачи в выборке являются очень простыми, поскольку в исходную КНФ подставляются значения относительно большого числа переменных. Тем самым, имеет смысл проверить всю рассматриваемую окрестность и лишь затем перейти к точке с лучшим значением прогнозной функции из этой окрестности (либо, если рекорд улучшить не удалось, перейти к некоторой точке из списка L_2). На второй стадии, когда число подставляемых переменных уменьшается и SAT-задачи в выборке усложняются, каждый раз осуществляется переход к точке с новым рекордным значением прогнозной функции (либо же проверяется вся окрестность и делается переход к следующей точке из L_2). Условия перехода от первой стадии ко второй могут быть различными. В предлагаемом алгоритме переход ко второй стадии осуществляется, если мощность очередного рекордного декомпозиционного множества больше, чем мощность декомпозиционного множества с предыдущим рекордом.

В псевдокоде алгоритма используются следующие обозначения:

χ_{start} — стартовая точка алгоритма;

χ_{current} — центр окрестности, в которой на текущей итерации осуществляется локальный поиск;

F_{record} — текущее рекордное значение функции F;

χ_{record} — точка, в которой было вычислено F_{record};

SecondStage — булева переменная, равная TRUE, если алгоритм перешел на стадию № 2;

weight (χ) — вес Хэмминга двоичного вектора χ;

length$\big(\theta(\chi)\big)$ — длина вектора $\theta(\chi)$.

Минимизация прогнозной функции с помощью поиска с запретами

Входные данные: КНФ C, начальная точка χ_{start}

Выходные данные: Прогноз F_{record} для SAT-задачи C

```
1  χ_record ← χ_current ← χ_start
2  F_record ← F(χ_start)
3  L_1 ← ∅
4  L_2 ← {χ_start}
5  SecondStage ← FALSE
6  while TRUE do
7      while weight(θ(χ_current)) < length(θ(χ_current)) do
8          χ ← случайно выбранная точка из N*_ρ(χ_current)
9          вычислить F(χ)
10         L_2 ← L_2 ∪ {χ}
11         foreach χ' ∈ N*_ρ(χ) do // обновить списки запретов
12             if χ' ∈ L_2 then
13                 отметить в θ(χ'), что χ проверена
14                 отметить в θ(χ), что χ' проверена
15                 if weight(θ(χ')) = length(θ(χ'))
16                     L_2 ← L_2 \ {χ'}
17                     L_1 ← L_1 ∪ {χ'}
18         if F(χ) < F_record then // если найден новый рекорд
19             if weight(χ) > weight(χ_record) then
20                 SecondStage ← TRUE
21             χ_record ← χ_current ← χ
22             F_record ← F(χ)
23             if SecondStage = TRUE then
24                 break
25     if timeExceeded() or L_2 = ∅ then
26         return F_record
27 χ_current ← точка из L_2 согласно эвристике
```

В вычислительных экспериментах (раздел 4) рассматривались окрестности с $\rho = 1$. Выбор χ_{current} из L_2 осуществлялся в соответствии с эвристиками, подобранными в процессе вычислительных экспериментов. Наилучшие результаты на данный момент показала следующая эвристика. При необходимости выбрать новую точку из L_2; в этом списке сначала выделяются все точки, находящиеся на минимальном расстоянии Хэмминга от точки с текущим рекордом. Затем все эти точки разбиваются на классы по весу представляющих их булевых векторов χ (векторы одинакового веса попадают в один класс). Далее случайным образом выбирается класс, а из него случайным образом выбирается новая точка χ_{current}.

Эффективность описанного процесса поиска можно существенно ускорить, если использовать следующее простое наблюдение. Основное время, необходимое для вычисления F, тратится на вычисление значений $\xi^j = \xi_A\big(\alpha_1^j, \ldots, \alpha_d^j\big)$, $j = \{1, \ldots, N\}$. Предположим, что F_{record} — текущее рекордное значение функции и после вычисления значений $\xi^{i_1}, \ldots, \xi^{i_k}$, $\{i_1, \ldots, i_k\} \subset \{1, \ldots, N\}$, в некоторой точке χ' имеет место следующее неравенство:

$$\frac{2^d}{N} \sum_{r=1}^{k} \xi^{i_r} > F_{\text{record}}. \tag{5}$$

Тогда очевидно, что $F(\chi') > F_{\text{record}}$. В этой ситуации мы можем прервать процесс вычисления $F(\chi')$ и перейти к следующей точке пространства поиска. Использование такого приема позволяет на второй

стадии работы алгоритма (когда SAT-задачи в выборке становятся трудными) прерывать по условию (5) вычисления более чем в 98% точек.

Как уже говорилось выше, в статье [12] вместо представленной здесь схемы для минимизации прогнозной функции использовался метод имитации отжига. К сожалению, для данного метода описанная техника прерывания вычислений не может быть использована, поскольку для нахождения вероятности перехода из точки χ в новую точку χ' в методе имитации отжига требуется знать значение целевой функции в обеих этих точках.

Возможность предложенного алгоритма строить хорошие декомпозиционные множества за относительно небольшое время существенно зависит от выбора начального декомпозиционного множества $\widetilde{X}_0$. Как уже было отмечено ранее, нам следует выбирать $\widetilde{X}_0$ так, чтобы значение $F\big(\widetilde{X}_0\big)$ вычислялось быстро. В общем случае мы всегда можем положить $\widetilde{X}_0 = X$. Однако для многих SAT-задач возможно выбрать $\widetilde{X}_0$, $\widetilde{X}_0 \subset X$, так, что $\big|\widetilde{X}_0\big| \ll |X|$ и значение $F\big(\widetilde{X}_0\big)$ может быть вычислено эффективно. В частности, для SAT-задач, кодирующих задачи обращения криптографических дискретных функций, таким свойством обладают SUPBS-множества (Strong Unit Propagation Backdoor Set) [24]. В случаях, когда $\widetilde{X}_0$ является таким множеством, мы можем минимизировать прогнозную функцию только на подмножествах $\widetilde{X}_0$, существенно сокращая тем самым первоначальное пространство поиска.

4. Вычислительные эксперименты. Алгоритм минимизации прогнозной функции, представленный в разделе 3, был встроен в параллельный решатель PDSAT [25, 26], являющийся MPI-программой. Для организации взаимодействия между процессами в этом решателе используется модель Master–Slave (один процесс управляющий, остальные вычислительные). На управляющем MPI-процессе в PDSAT хранятся списки L_1 и L_2, а также осуществляется выбор точек пространства поиска в соответствии с описанной выше схемой поиска с запретами. Кроме того, на управляющем процессе для каждой новой точки $\widetilde{X} = \{x_{i_1}, \ldots, x_{i_d}\}$ создается множество $\Theta\big(\widetilde{X}\big) \subset \{0,1\}^d$, $\Big|\Theta\big(\widetilde{X}\big)\Big| = N$ (см. (3)) с использованием стандартного генератора псевдослучайных чисел “Mersenne Twister”.

Для вычисления значений прогнозной функции используются вычислительные MPI-процессы. После получения с управляющего процесса набора $\big(\alpha^j_{i_1}, \ldots, \alpha^j_{i_d}\big) \in \Theta\big(\widetilde{X}\big)$, $j \in \{1, \ldots, N\}$, вычислительный процесс начинает решать SAT-задачу для КНФ $C\Big[\widetilde{X}/\big(\alpha^j_{i_1}, \ldots, \alpha^j_{i_d}\big)\Big]$. Множество $\left\{C\Big[\widetilde{X}/\big(\alpha^j_{i_1}, \ldots, \alpha^j_{i_d}\big)\Big]\right\}_{j=1}^{N}$ называется выборкой для точки $\widetilde{X}$. Для решения SAT-задач на вычислительных процессах используется SAT-решатель MiniSat версии 2.2 [27], в котором была дополнительно реализована возможность прерывания вычисления по условию (5). Суммарное время, потраченное на обработку выборки для $\widetilde{X}$, отслеживается на управляющем процессе. Если срабатывает условие (5), то обработка данной выборки на вычислительных процессах прерывается управляющим процессом при помощи отправки асинхронных сообщений [28]. В функцию `search()` в решателе MiniSat была добавлена периодическая проверка наличия сообщений с управляющего процесса. При получении такого сообщения функция `search()` досрочно прерывает свою работу, а на вход SAT-решателю подается новая задача, полученная с управляющего процесса.

Особо отметим, что фактически значение прогнозной функции, так как оно определено выше, является прогнозом времени обработки декомпозиционного семейства одним экземпляром рассматриваемого алгоритма решения SAT-задач. Иными словами, значение прогнозной функции вычисляется для одного ядра многопроцессорной системы. Экстраполировать полученное значение на случай решения задачи в многоядерной системе не представляет сложности — необходимо это значение разделить на соответствующее число ядер (так как ядра обрабатывают декомпозиционное семейство независимо).

Далее приводятся результаты вычислительных экспериментов, в которых программа PDSAT использовалась для криптоанализа шифра Bivium [13]. Эта задача была сведена к SAT-задаче при помощи программного комплекса TransAlg [29]. Напомним, что криптоанализ, рассматриваемый как SAT-задача, называется логическим [30].

В шифре Bivium используются два сдвиговых регистра специального вида, первый из которых состоит из 93 ячеек, второй — из 84 ячеек. В работе [31] было показано, что криптоанализ данного шифра “на основе известного открытого текста” имеет смысл рассматривать в следующей постановке. По известному фрагменту ключевого потока требуется восстановить 177 бит, которые соответствуют внутреннему состоянию регистров перед началом генерации этого ключевого потока. В представленных ниже экспериментах в качестве известного фрагмента использовались первые 200 бит ключевого потока. На рис. 1 представлена схема работы шифра Bivium.

Сначала мы применили программу PDSAT для оценивания времени решения задачи логического

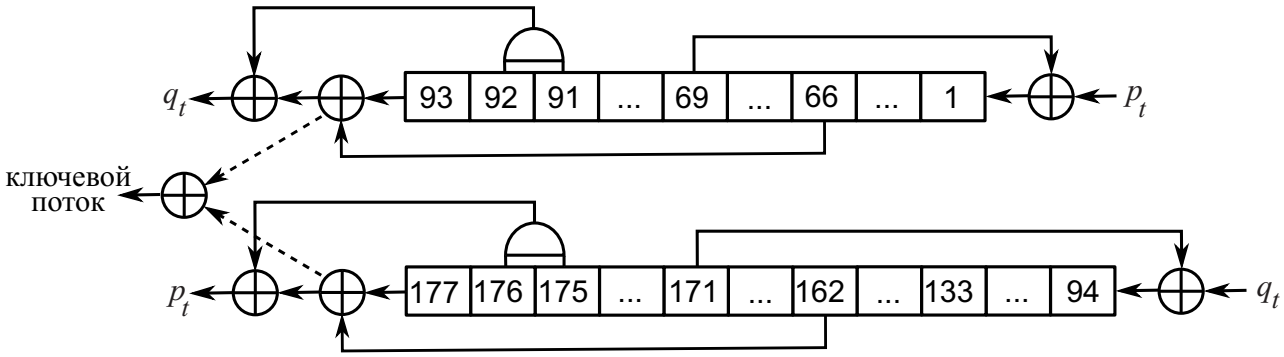

Рис. 1. Схема работы шифра Bivium

криптоанализа шифра Bivium. В качестве $\widetilde{X}_0$ использовалось множество SUPBS, состоящее из 177 переменных, кодирующих состояние регистров генератора Bivium после стадии инициализации. Прогнозная функция минимизировалась на подмножествах множества $\widetilde{X}_0$. Для каждой новой точки пространства поиска обрабатывалась выборка объемом $N = 100\,000$ КНФ.

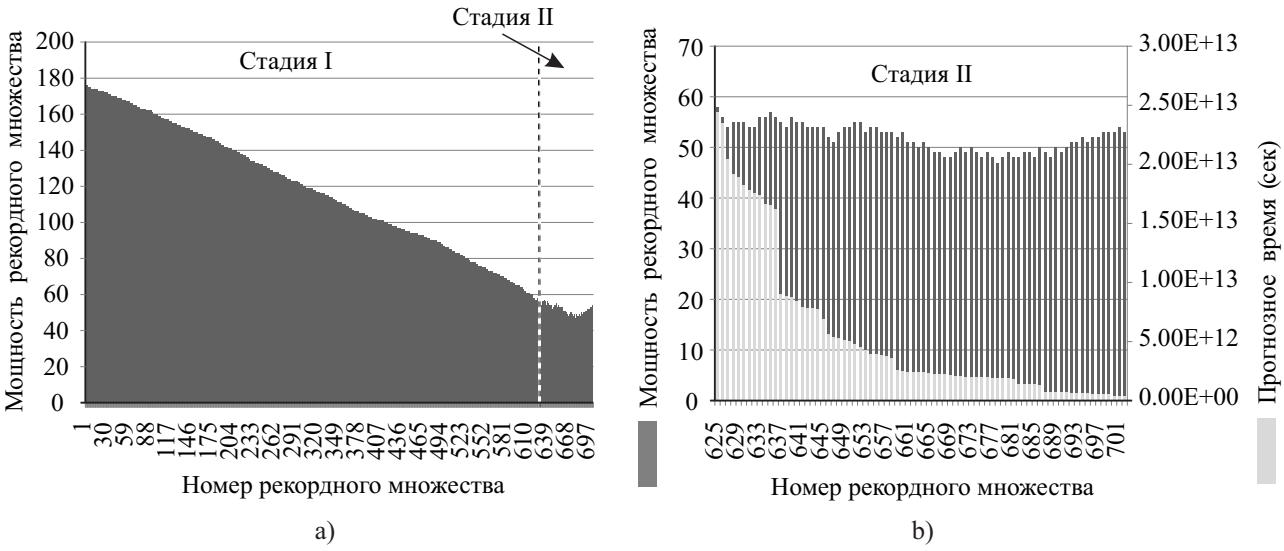

Рис. 2. Процесс минимизации прогнозной функции для криптоанализа шифра Bivium: a) динамика изменения мощности рекордных декомпозиционных множеств, b) график прогнозной функции на второй стадии минимизации

Программа PDSAT была запущена на 1 сутки на 480 ядрах вычислительного кластера “Академик В.М. Матросов” Иркутского суперкомпьютерного центра СО РАН [32]; использовался 1 управляющий процесс и 479 вычислительных. На рис. 2a представлена динамика изменения мощности рекордных декомпозиционных множеств в процессе минимизации прогнозной функции. Всего было найдено 702 рекордных множества (соответствующих рекордным значениям прогнозной функции), из них 625 на первой стадии работы алгоритма. На рис. 2b отображен процесс минимизации прогнозной функции и динамика изменения мощности рекордных декомпозиционных множеств на второй стадии работы алгоритма. Здесь и далее значение времени приводится в секундах. Найденное в результате работы программы PDSAT множество X_{record}, состоящее из 53 переменных, отмечено на рис. 3 серой заливкой.

Проблема оценивания времени логического криптоанализа шифра Bivium рассматривалась ранее в работах [16–19]. В частности, в [17] были проанализированы несколько фиксированных видов декомпозиционных множеств (в терминологии [17] “стратегий”). В [18, 19] фактически был использован подход Монте-Карло (без формального обоснования), при помощи которого оценивались не сами декомпозиционные множества, а значения параметра d, т.е. их размерность. Новизна подхода, предлагаемого нами, заключается в том, что для поиска хорошего декомпозиционного множества используется итерационная схема минимизации прогнозной функции в специальном пространстве поиска.

Далее приведено сравнение полученных нами результатов с результатами работы [17], поскольку в

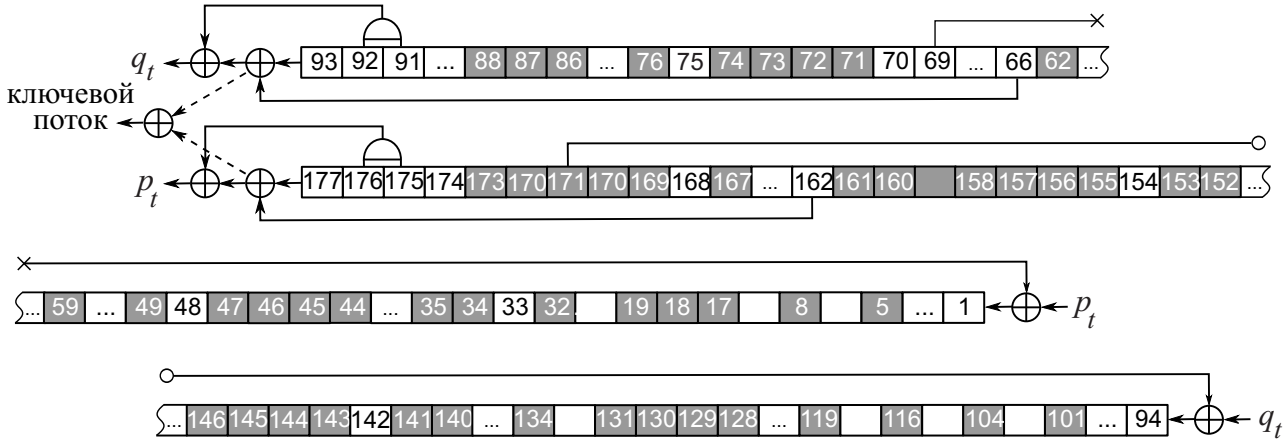

Рис. 3. Декомпозиционное множество из 53 переменных, найденное с помощью программы PDSAT

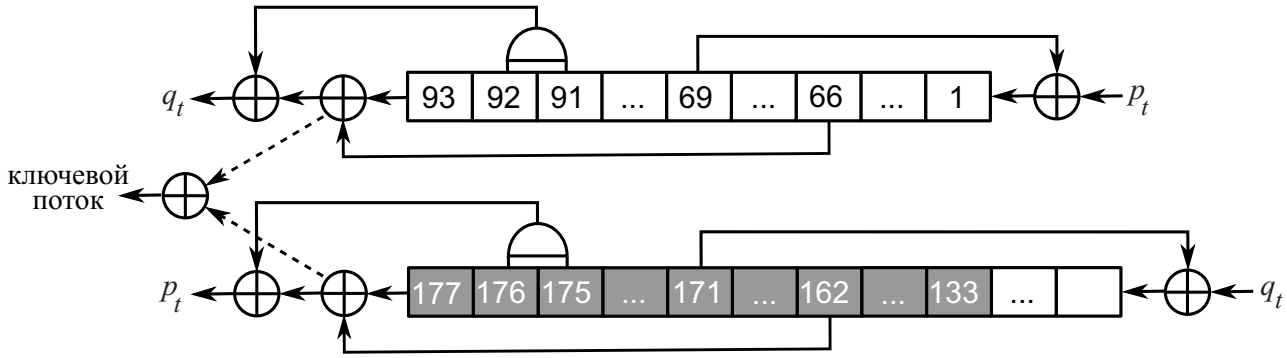

Рис. 4. Декомпозиционное множество из 45 переменных, найденное в [17]

ней декомпозиционные множества представлены в явном виде, а эксперименты легко воспроизводятся. Лучшее декомпозиционное множество, найденное в [17], состоит из 45 переменных и представлено на рис. 4. В табл. 1 приведены качественные параметры декомпозиционных множеств с рис. 3 и 4. Мин., Макс. и Сред. обозначают минимальное, максимальное и среднее время решения SAT-задач для КНФ из выборки, соответствующей итоговому декомпозиционному множеству. Через s^2 обозначена исправленная выборочная дисперсия (раздел 2), через F — значение прогнозной функции, посчитанное для одного ядра процессора AMD Opteron 6276 (именно такие процессоры установлены на узлах кластера "Академик В.М. Матросов").

Таблица 1

Качественные параметры декомпозиционных множеств с рис. 3 и 4

X_{record}	$\lvert X_{\text{record}}\rvert$	Мин.	Макс.	Сред.	s^2	F для 1 ядра
Рис. 3	53	0.00001	0.92871	0.00002	0.00001	3.33600×10^{11}
Рис. 4	45	0.17063	23.91759	4.88732	9.27706	1.71957×10^{14}

Кроме того, мы рассмотрели задачи логического криптоанализа ослабленных версий шифра Bivium. Далее используется обозначение "BiviumK", если имеется в виду ослабленный вариант шифра Bivium с известными значениями K последних ячеек второго регистра генератора после стадии инициализации.

В табл. 2 представлены результаты прогнозов, каждый из которых был получен в течение 1 суток на 480 ядрах кластера "Академик В.М. Матросов". Значения F в табл. 2 посчитаны для 1 ядра процессора AMD Opteron 6276. В этой таблице приведен также результат для неослабленного варианта шифра Bivium, подробно разобранный выше. Для всех рассмотренных задач указана мощность найденного рекордного декомпозиционного множества и процент времени, потраченного на выполнение второй стадии минимизации.

Помимо режима прогнозирования программа PDSAT может работать в режиме решения. В этом

Таблица 2

Прогнозное время криптоанализа различных вариантов ослабленного шифра Bivium

Задача	F для 1 ядра	$\lvert X_{\text{record}}\rvert$	% на 2 стадию
Bivium20	9.96004×10^5	33	96.35
Bivium18	3.89756×10^6	35	94.40
Bivium16	1.25220×10^7	38	96.35
Bivium14	7.13668×10^7	37	95.99
Bivium12	2.04054×10^8	41	93.98
Bivium10	6.24878×10^8	42	96.45
Bivium8	2.26222×10^9	45	96.10
Bivium6	1.07919×10^{10}	47	94.01
Bivium4	3.72146×10^{10}	49	96.09
Bivium2	1.35281×10^{11}	52	95.76
Bivium	3.33600×10^{11}	53	95.97

случае на вход PDSAT подается булев вектор, определяющий декомпозиционное множество X_{record}, т.е. множество с наименьшим значением прогнозной функции, найденное на момент остановки процедуры минимизации. На управляющем процессе генерируются все $2^{|X_{\text{record}}|}$ наборов значений переменных из множества X_{record}, каждому из которых соответствует SAT-задача. Для уменьшения затрат на передачу данных SAT-задачи объединяются в пакеты. Число пакетов SAT-задач равно ближайшей справа степени двойки от числа $k\,p$, где k — число вычислительных MPI-процессов, а p — константа, влияющая на равномерность нагрузки MPI-процессов (подбирается в ходе вычислительных экспериментов). Далее использовалось $p = 16$.

Таблица 3

Прогнозное и реальное время решения задач на кластере “Академик В.М. Матросов”

Задача	F для 480 ядер	Выполняющий набор			Все семейство		
		Мин.	Макс.	Сред.	Мин.	Макс.	Сред.
Bivium20	2079	53	2337	912	1973	2426	2314
Bivium18	8137	2176	7662	4584	10 223	10 907	10 601
Bivium16	26 142	2071	39 908	21 352	46 940	48 192	47 729
Bivium14	148 991	1377	144 539	84 547	156 752	161 072	158 643

В табл. 3 приведены результаты логического криптоанализа четырех ослабленных вариантов шифра Bivium: Bivium20, Bivium18, Bivium16, Bivium14. Для каждого варианта было решено 10 тестов на 480 ядрах кластера “Академик В.М. Матросов”. Для каждой из задач BiviumK, $K \in \{20, 18, 16, 14\}$, указано минимальное, максимальное и среднее значения времени поиска выполняющего набора и времени обработки всего декомпозиционного семейства (по 10 тестам). Прогноз времени решения на 480 ядрах был вычислен на основе рекордного значения прогнозной функции F, найденного для одного ядра процессора AMD Opteron 6276. Особо отметим, что для ослабленных тестов процедура прогнозирования может занимать больше времени, чем последующая обработка на кластере декомпозиционного семейства. Вследствие этого процедура прогнозирования запускалась лишь для одного теста из 10. Для каждой задачи найденное декомпозиционное множество было использовано для решения всех 10 тестов. Для получения

подробной статистики процесс решения не прерывался после нахождения выполняющего набора; таким образом, в каждом тесте обрабатывалось все декомпозиционное семейство.

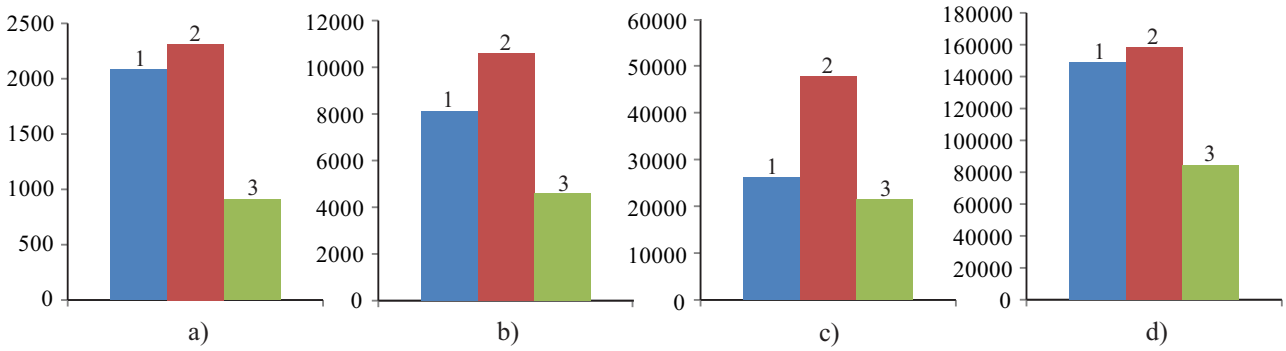

Рис. 5. Прогнозное и реальное среднее время решения задач на 480 ядрах кластера “Академик В.М. Матросов”: a) Bivium20, b) Bivium18, c) Bivium16, d) Bivium14

На рис. 5 представлено сравнение прогнозного времени (помечено цифрой 1) с реальным средним временем обработки всего декомпозиционного семейства (помечено цифрой 2) и реальным средним временем нахождения выполняющего набора (помечено цифрой 3) (по 10 тестам). Следует отметить, что для каждой задачи прогнозируется именно время обработки всего декомпозиционного семейства.

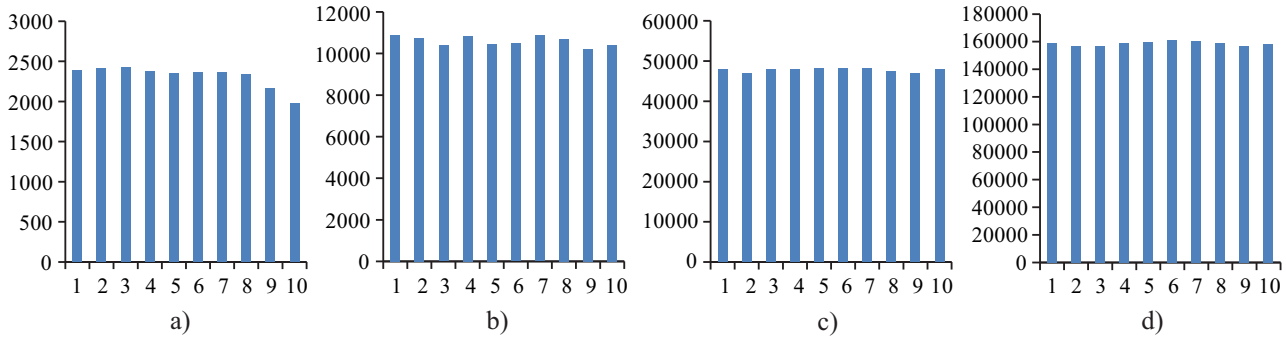

Рис. 6. Время обработки всего декомпозиционного семейства на серии из 10 тестов для ослабленных вариантов шифра Bivium: a) Bivium20, b) Bivium18, c) Bivium16, d) Bivium14

На рис. 6 отображено изменение времени обработки декомпозиционного семейства по 10 тестам для каждого ослабленного варианта шифра Bivium. Особо отметим, что несмотря на использование во всех тестах одного и того же декомпозиционного множества (найденного для первого теста), изменение общего времени обработки декомпозиционного семейства во всех задачах весьма незначительно.

Интересной особенностью рассматриваемого класса тестов является тот факт, что подавляющее большинство (около 99,9%) задач из декомпозиционного семейства решаются либо без угадываний переменных (только за счет использования техники Unit Propagation [33]), либо с крайне незначительным числом угадываний. Для иллюстрации этого факта в табл. 4 представлена статистика решения 10 тестов задачи криптоанализа Bivium20. Напомним, что в качестве X_{record} использовалось множество из 33 переменных, т.е. декомпозиционное семейство состояло из 8 589 934 592 SAT-задач. Кроме минимального, максимального и среднего времени решения SAT-задач из декомпозиционного семейства в табл. 4 приведены данные о времени решения SAT-задач, в которых был найден выполняющий набор (поле “Вып. набор”). Значение поля “$\%\,(a, b)$” — это процент SAT-задач из декомпозиционного семейства, которые были решены за время t секунд, где $a \leqslant t \leqslant b$. Для остальных рассмотренных ослабленных вариантов шифра Bivium наблюдается похожая картина.

5. Заключение. В работе представлен алгоритм, позволяющий получать прогнозы времени решения SAT-задач при их декомпозиции по переменным. Данный алгоритм основан на методе Монте-Карло — каждому рассматриваемому способу декомпозиции, т.е. декомпозиционному множеству, сопоставляется случайная выборка упрощенных SAT-задач, решение которых позволяет оценить время обработки всего декомпозиционного семейства. Соответствующая оценка является значением прогнозной функции.

Новизна предложенного подхода состоит в том, для нахождения декомпозиционного множества с

Таблица 4

Данные по решению 10 тестов задачи криптоанализа Bivium20

№	Мин.	Макс.	Сред.	Вып. набор	% (0, 0.0001)	% (0.0001, 0.01)	% (0.01, 1)
1	0.00001	0.53229	0.00013	0.07185	99.73513	0.11832	0.14655
2	0.00001	0.56710	0.00013	0.04826	99.73520	0.11825	0.14655
3	0.00001	0.58066	0.00013	0.01694	99.74308	0.11100	0.14592
4	0.00001	0.47563	0.00012	0.04464	99.73529	0.11823	0.14648
5	0.00001	0.63290	0.00012	0.01300	99.74433	0.10987	0.14580
6	0.00001	0.60373	0.00012	0.05827	99.74391	0.10963	0.14646
7	0.00001	0.58299	0.00012	0.05846	99.73532	0.11937	0.14531
8	0.00001	0.67404	0.00012	0.03516	99.73724	0.11708	0.14569
9	0.00001	0.62390	0.00011	0.04597	99.75312	0.10159	0.14529
10	0.00001	0.69061	0.00010	0.05315	99.76164	0.09454	0.14382

хорошим значением прогнозной функции используется итеративный алгоритм поиска с запретами в специальным образом определяемом пространстве. Предложенный алгоритм был реализован в виде MPI-программы, которая была использована на вычислительном кластере для получения прогноза времени параллельного решения задачи логического криптоанализа известного шифра Bivium. Кроме того, были получены прогнозы времени логического криптоанализа для ряда ослабленных вариантов шифра Bivium. Для этих вариантов был осуществлен их успешный логический криптоанализ на вычислительном кластере. Из полученных результатов следует, что прогнозное время хорошо согласуется с реальным. Отметим, что некоторые из приведенных в статье результатов в предварительной форме были представлены в препринте [34]. Основной новый результат настоящей статьи — полный криптоанализ ряда ослабленных версий шифра Bivium, выполненный на вычислительном кластере.

Работа выполнена при частичной финансовой поддержке РФФИ (проект № 14–07–00403а) и Совета по грантам Президента РФ для поддержки молодых ученых (стипендия СП–1855.2012.5).

СПИСОК ЛИТЕРАТУРЫ

1. *Böhm M., Speckenmeyer E.* A fast parallel SAT solver — efficient workload balancing // Annals of Mathematics and Artificial Intelligence. 1996. **17**, N 3/4. 381–400.
2. *Hyvärinen A.E.J., Junttila T.A., Niemelä I.* A distribution method for solving SAT in grids // Lecture Notes in Computer Science. Vol. 4121. Heidelberg: Springer, 2006. 430–435.
3. *Hyvärinen A.E.J., Junttila T.A., Niemelä I.* Partitioning SAT instances for distributed solving // Lecture Notes in Computer Science. Vol. 6397. Heidelberg: Springer, 2010, 372–386.
4. *Hyvärinen A.E.J., Junttila T.A., Niemelä I.* Grid-based SAT solving with iterative partitioning and clause learning // Lecture Notes in Computer Science. Vol. 6876. Heidelberg: Springer, 2011, 385–399.
5. *Hyvärinen A.E.J.* Grid based propositional satisfiability solving. Ph.D. Thesis. Department of Information and Computer Science. Aalto (Finland): Aalto Univ. Press, 2011.
6. *Заикин О.С., Семенов А.А.* Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. № 1. 43–50.
7. *Посыпкин М.А., Заикин О.С., Беспалов Д.В., Семенов А.А.* Решение задач криптоанализа поточных шифров в распределенных вычислительных средах // Тр. Института системного анализа РАН. 2009. **46**. 119–137.
8. *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. Vol. 6873. Heidelberg: Springer, 2011, 473–483.
9. *Semenov A., Zaikin O., Bespalov D., Posypkin M.* Parallel algorithms for SAT in application to inversion problems of some discrete functions (arXiv: 1102.3563 [cs.DC]; Cornell Univ. Library), 2011 (http://arxiv.org/abs/1102.3563).
10. *Семенов А.А., Заикин О.С.* Алгоритмы построения декомпозиционных множеств для крупноблочного распараллеливания SAT-задач // Известия Иркутского гос. ун-та. Серия “Математика”. 2012. № 4. 79–94.
11. *Metropolis N., Ulam S.* The Monte Carlo method // J. of the American Statistical Association. 1949. **44**, N 247. 335–341.

12. *Заикин О.С., Семенов А.А., Посыпкин М.А.* Процедуры построения декомпозиционных множеств для распределенного решения SAT-задач в проекте добровольных вычислений SAT@home // Управление большими системами. Вып. 43. М.: Ин-т проблем управления РАН, 2013. 138–156.
13. *Canniére C.D.* Trivium: A stream cipher construction inspired by block cipher design principles // Lecture Notes in Computer Science. Vol. 4176. Heidelberg: Springer, 2006, 171–186.
14. *Zhang H., Bonacina M.P., Hsiang J.* PSATO: a distributed propositional prover and its application to quasigroup problems // J. of Symbolic Computation. 1996. **21**. 543–560.
15. *Heule M., Kullmann O., Wieringa S., Biere A.* Cube and conquer: guiding CDCL SAT solvers by lookaheads // Lecture Notes in Computer Science. Vol. 7261. Heidelberg: Springer, 2011, 50–65.
16. *Mcdonald C., Charnes C., Pieprzyk J.* Attacking Bivium with MiniSat. Technical Report 2007/040. ECRYPT Stream Cipher Project, 2007 (http://eprint.iacr.org/2007/129).
17. *Eibach T., Pilz E., Völkel G.* Attacking Bivium using SAT solvers // Lecture Notes in Computer Science. Vol. 4996. Heidelberg: Springer, 2008. 63–76.
18. *Soos M., Nohl K., Castelluccia C.* Extending SAT solvers to cryptographic problems // Lecture Notes in Computer Science. Vol. 5584. Heidelberg: Springer, 2009. 244–257.
19. *Soos M.* Grain of salt — an automated way to test stream ciphers through sat solvers // Proc. of the Workshop on Tools for Cryptanalysis. Egham: Univ. of London, 2010. 131–144.
20. *Ермаков С.М.* Метод Монте-Карло и смежные вопросы. М.: Наука, 1971.
21. *Феллер В.* Введение в теорию вероятностей и ее приложения. Т. 2. М.: Мир, 1984.
22. *Glover F., Laguna M.* Tabu search. Norwell: Klüwer, 1997.
23. *Пападимитриу Х., Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир, 1984.
24. *Järvisalo M., Junttila T.A.* Limitations of restricted branching in clause learning // Constraints. 2009. **14**, N 3. 325–356.
25. *Заикин О.С.* Реализация процедур прогнозирования трудоемкости параллельного решения SAT-задач // Вестник Уфимского гос. авиационного техн. ун-та. 2010. **14**, № 4. 210–220.
26. *Заикин О.С., Отпущенников И.В., Семенов А.А.* Параллельные алгоритмы решения проблемы выполнимости в применении к оптимизационным задачам с булевыми ограничениями // Вычислительные методы и программирование: Новые вычислительные технологии. 2011. **12**, № 1. 205–212.
27. *Eén N., Sörensson N.* An extensible SAT-solver // Lecture Notes in Computer Science. Vol. 2919. Heidelberg: Springer, 2003. 502–518.
28. *Гришагин В.А., Свистунов А.Н.* Параллельное программирование на основе MPI. Изд-во Нижегородского гос. ун-та им. Н.И. Лобачевского, 2005.
29. *Отпущенников И.В., Семенов А.А.* Технология трансляции комбинаторных проблем в булевы уравнения // Прикладная дискретная математика. 2011. № 1. 96–115.
30. *Massacci F., Marraro L.* Logical cryptanalysis as a SAT problem // J. of Automated Reasoning. 2000. **24**, N 1/2. 165–203.
31. *Maximov A., Biryukov A.* Two trivial attacks on Trivium // Lecture Notes in Computer Science. Vol. 4876. Heidelberg: Springer, 2007. 36–55.
32. Вычислительный кластер "Академик В.М. Матросов" [Электронный ресурс] // Иркутский суперкомпьютерный центр Сибирского отделения РАН (http://hpc.icc.ru/hardware/matrosov.php).
33. *Dowling W., Gallier J.* Linear-time algorithms for testing the satisfiability of propositional Horn formulae // J. of Logic Programming. 1984. **1**, N 3. 267–284.
34. *Semenov A., Zaikin O.* On estimating total time to solve SAT in distributed computing environments: application to the SAT@home project (arXiv: 1308.0761 [cs.AI]; Cornell Univ. Library), 2013.

Поступила в редакцию
29.10.2013

Application of the Monte Carlo Method for Estimating the Total Time of Solving the SAT Problem in Parallel

O. S. Zaikin [1] **and A. A. Semenov** [2]

[1] *Institute for System Dynamics and Control Theory, Russian Academy of Sciences; ulitsa Lermontova 134, Irkutsk, 664033, Russia; Ph.D., Scientist, e-mail: zaikin.icc@gmail.com*

[2] *Institute for System Dynamics and Control Theory, Russian Academy of Sciences; ulitsa Lermontova 134, Irkutsk, 664033, Russia; Ph.D., Head of Laboratory, e-mail: biclop.rambler@yandex.ru*

Received October 29, 2013

Abstract: The application of the Monte Carlo method to plan the solving process for hard examples of the Boolean satisfiability problem (SAT) on parallel computing systems is discussed. The parallelization of the SAT problem is reached as a result of choosing the subset of the set of variables of an original CNF (Conjunctive Normal Form). This set is called a decomposition set. For such sets, we can naturally define a number of parameters to measure the "quality" of decomposition. In order to evaluate these parameters, we use the computational scheme based on the Monte Carlo method. In particular, this method is used to search for the decomposition set with minimal predicted time of solving the original problem. Using the implemented parallel MPI-program, it is possible to obtain a prediction of time required to perform the logical cryptanalysis of the Bivium cipher on a computing cluster. We successfully performed the logical cryptanalysis of several weakened versions of the Bivium cipher. The computing cost of such a cryptanalysis is in agreement with the predicted one.

Keywords: Boolean satisfiability problem (SAT), Monte Carlo method, tabu search, MPI, cryptanalysis, Bivium cipher.

References

1. M. Böhm and E. Speckenmeyer, "A Fast Parallel SAT Solver — Efficient Workload Balancing," Ann. Math. Artif. Intell. **17** (3/4), 381–400 (1996).

2. A. E. J. Hyvärinen, T. A. Junttila, and I. Niemelä, "A Distribution Method for Solving SAT in Grids," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2006), Vol. 4121, pp. 430–435.

3. A. E. J. Hyvärinen, T. A. Junttila, and I. Niemelä, "Partitioning SAT Instances for Distributed Solving," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2010), Vol. 6397, pp. 372–386.

4. A. E. J. Hyvärinen, T. A. Junttila, and I. Niemelä, "Grid-Based SAT Solving with Iterative Partitioning and Clause Learning," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2011), Vol. 6876, pp. 385–399.

5. A. E. J. Hyvärinen, *Grid Based Propositional Satisfiability Solving*, Ph.D. Thesis (Aalto Univ., Aalto, 2011).

6. O. S. Zaikin and A. A. Semenov, "Technology of Large-Block Parallelism in SAT Problems," Problemy Upravl., No. 1, 43–50 (2008).

7. M. A. Posypkin, O. S. Zaikin, D. V. Bespalov, and A. A. Semenov, "Solving the Cryptanalysis Problems of Stream Ciphers on Distributed Computer Systems," Tr. Inst. Systems Anal., Ross. Akad. Nauk **46**, 119–137 (2009).

8. A. Semenov, O. Zaikin, D. Bespalov, and M. Posypkin, "Parallel Logical Cryptanalysis of the Generator A5/1 in BNB-Grid System," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2011), Vol. 6873, pp. 473–483.

9. A. Semenov, O. Zaikin, D. Bespalov, and M. Posypkin, *Parallel Algorithms for SAT in Application to Inversion Problems of Some Discrete Functions*, arXiv preprint: 1102.3563 [cs.DC] (Cornell Univ. Library, Ithaca, 2011), available at http://arxiv.org/abs/1102.3563.

10. A. A. Semenov and O. S. Zaikin, "Algorithms of Constructing Decomposition Sets for Large-Block Parallelization of SAT Problems," Izv. Irkutsk Gos. Univ. Ser. Mat., No. 4, 79–94 (2012).

11. N. Metropolis and S. Ulam, "The Monte Carlo Method," J. Am. Stat. Assoc. **44** (247), 335–341 (1949).

12. O. S. Zaikin, A. A. Semenov, and M. A. Posypkin, "Procedures of Constructing Decomposition Sets for the Distributed Solution of SAT problems in the SAT@home Project," in *Large System Control* (Inst. Problem Upravl., Moscow, 2013), Issue 43, pp. 138–156.

13. C. D. Canniére, "Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2006), Vol. 4176, pp. 171–186.

14. H. Zhang, M. P. Bonacina, and J. Hsiang, "PSATO: A Distributed Propositional Prover and Its Application to Quasigroup Problems," J. Symbolic Comput. **21**, 543–560 (1996).

15. M. Heule, O. Kullmann, S. Wieringa, and A. Biere, "Cube and Conquer: Guiding CDCL SAT Solvers by Lookaheads," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2011), Vol. 7261, pp. 50–65.

16. C. McDonald, C. Charnes, and J. Pieprzyk, *Attacking Bivium with MiniSat*, Technical Report 2007/040, ECRYPT Stream Cipher Project (2007), available at http://eprint.iacr.org/2007/129.

17. T. Eibach, E. Pilz, and G. Völkel, "Attacking Bivium Using SAT Solvers," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2008), Vol. 4996, pp. 63–76.

18. M. Soos, K. Nohl, and C. Castelluccia, "Extending SAT Solvers to Cryptographic Problems," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2009), Vol. 5584, pp. 244–257.

19. M. Soos, "Grain of Salt — An Automated Way to Test Stream Ciphers Through SAT Solvers," in *Proc. ECRYPT Workshop on Tools for Cryptanalysis, June 22–23, 2010* (Univ. of London, Egham, 2010), pp. 131–144.

20. S. M. Ermakov, *The Monte Carlo Method and Related Problems* (Nauka, Moscow, 1971) [in Russian].

21. W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 2 (Wiley, New York, 1966; Mir, Moscow, 1984).

22. F. Glover and M. Laguna, *Tabu Search* (Klüwer, Norwell, 1997).

23. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Prentice-Hall, Englewood Cliffs, 1982; Mir, Moscow, 1984).

24. M. Järvisalo and T. A. Junttila, "Limitations of Restricted Branching in Clause Learning," Constraints **14** (3), 325–356 (2009).

25. O. S. Zaikin, "Implementation of Prediction Procedures to Evaluate the Complexity of Parallel Solution of SAT Problems," Vestn. Ufimsk. Gos. Aviats. Tekhn. Univ. **14** (4), 210–220 (2010).

26. O. S. Zaikin, I. V. Otpuschennikov, and A. A. Semenov, "Parallel Algorithms for Solving SAT Problems in Application to Optimization Problems with Boolean Constraints," Vychisl. Metody Programm. **12**, 205–212 (2011).

27. N. Eén and N. Sörensson, "An Extensible SAT-Solver," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2003), Vol. 2919, pp. 502–518.

28. V. A. Grishagin and A. N. Svistunov, *Parallel Programming Based on MPI* (Nizhni Novgorod Gos. Univ. Press, Nizhni Novgorod, 2005) [in Russian].

29. I. V. Otpuschennikov and A. A. Semenov, "A Technology for the Translation of Combinatorial Problem in Boolean equations," Prikl. Discret. Mat., No. 1. 96–115 (2011).

30. F. Massacci and L. Marraro, "Logical Cryptanalysis as a SAT Problem," J. Autom. Reasoning **24** (1/2), 165–203 (2000).

31. A. Maximov and A. Biryukov, "Two Trivial Attacks on Trivium," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2007), Vol. 4876, pp. 36–55.

32. *Irkutsk Supercomputing Center*, available at http://hpc.icc.ru/hardware/matrosov.php.

33. W. Dowling and J. Gallier, "Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae," J. Logic Programm. **1** (3), 267–284 (1984).

34. A. Semenov and O. Zaikin, *On Estimating Total Time to Solve SAT in Distributed Computing Environments: Application to the SAT@home Project*, arXiv preprint: 1308.0761 [cs.AI] (Cornell Univ. Library, Ithaca, 2013), available at http://arxiv.org/abs/1308.0761.