Реализация процедур прогнозирования трудоемкости параллельного решения SAT-задач

О.С. Заикин

Разработана и реализована в виде MPI-программы крупноблочная параллельная технология решения SAT-задач (задач поиска решений уравнений вида «КНФ = 1», где КНФ – конъюнктивная нормальная форма) в распределенных вычислительных средах. В рамках данной технологии осуществляется декомпозиция исходной SAT-задачи на семейство подзадач. Используется процедура статистического прогнозирования трудоемкости параллельного решения SAT-задач, которая позволяет определить оптимальные (по прогнозу) параметры декомпозиции. Использование параметров декомпозиции, найденных с помощью процедур прогнозирования, позволяет успешно решать SAT-задачи, кодирующие задачи обращения ряда криптографических дискретных функций.

1. Введение

Многие значимые в практическом отношении комбинаторные проблемы допускают эффективные сводимости к задачам поиска решений булевых уравнений вида «КНФ=1» (КНФ – конъюнктивная нормальная форма) (см. [1]). Задачи поиска решений таких уравнений называются SAT-задачами, для их решения используются специальные программные комплексы, называемые SAT-решателями (см. [2]). В последнее время высокими темпами развиваются параллельные SAT-решатели (см., например, [3] и [4]). Как правило, в таких SAT-решателях используется концепция мелкозернистого параллелизма (см. [5]). Такой поход вполне оправдывает себя на многих классах тестов, например для КНФ, кодирующих задачи верификации в микроэлектронике (см. [6]). Применительно же к задачам обращения дискретных функций (например, к задачам криптоанализа поточных систем шифрования) высокую эффективность показала представленная в серии работ [7–11] крупноблочная параллельная технология решения SAT-задач в распределенных вычислительных средах (РВС). В рамках данной технологии осуществляется декомпозиция исходной SAT-задачи на семейство подзадач. Наилучшие результаты показали различные варианты декомпозиции по переменным, кодирующим вход дискретной функции, задачу обращения которой требуется решить. Для определения наилучших (по прогнозу) параметров декомпозиции используется процедура прогнозирования трудоемкости параллельного решения SAT-задач. Каждому варианту значений параметров декомпозиции соответствует случайная выборка SAT-задач. Прогноз заведомо быстро вычисляется для одной из выборок, затем он итеративно улучшается при обработке остальных выборок. Решение некоторых SAT-задач может быть прервано при превышении порогового значения.

Изначально данная технология была реализована в виде пакета прикладных программ (ППП) D-SAT [12], который функционирует под управлением инструментального комплекса DISCOMP (см. [10]). Функциональное наполнение ППП D-SAT включает процедуры решения SAT-задач и процедуры прогнозирования трудоемкости решения SAT-задач. Дальнейшим развитием данной параллельной технологии стала ее реализация в виде MPI-программы. Подробности данной реализации рассматриваются в настоящей статье.

В MPI-программе в режиме прогнозирования все SAT-задачи по всем случайным выборкам объединяются в единый параллельный список. В результате достигается равномерная загрузка РВС, но усложняется обработка данных. Для своевременного прерывания решений SAT-задач используются неблокирующие обмены (см. [13]), что позволяют каждому процессу эффективно использовать свое рабочее время: управляющий процесс занимается отправкой заданий и обработкой решений, вычислительные процессы решают SAT-задачи.

В данной работе впервые приведены результаты параллельного логического криптоанализа суммирующего генератора на основе четырех регистров сдвига с линейной обратной связью. Также приведены улучшенные результаты логического криптоанализа ряда других генераторов.

2. Крупноблочная параллельная технология решения SAT-задач

Далее приведено краткое описание крупноблочной параллельной технологии решения SAT-задач, представленной в работах [7–9].

Под распределенной вычислительной средой (РВС) понимается совокупность вычислительных единиц, объединенных коммуникационной сетью. В качестве вычислительной единицы РВС выступает программно-аппаратный ресурс, требуемый для решения некоторой вычислительной задачи. В качестве вычислительной единицы далее рассматривается одно ядро процессора, часть общей оперативной памяти и памяти жесткого диска, а также системное программное обеспечение.

Рассматривается произвольная конъюнктивная нормальная форма (КНФ) C над множеством булевых переменных $X = \{x_1,...,x_n\}$. В множестве X выбирается некоторое подмножество $\{x_{i_1},...,x_{i_d}\}, \{i_1,...,i_d\} \subseteq \{1,...,n\}, d \in \{1,...,n\}$. Множество $X' = \{x_{i_1},...,x_{i_d}\}$ называется *декомпозиционным множеством*, а d *– размерностью декомпозиционного множества*. Дополнительно полагается, что при $d = 0$ декомпозиционное множество пусто. Декомпозиционному множеству $X' : |X'| = d, d > 0$ ставится в соответствие множество $Y(X') = \{Y_1,...,Y_k\}$, состоящее из $k = 2^d$ различных двоичных векторов длины d, каждый из которых является набором значений переменных из множества X'. *Декомпозиционным семейством*, порожденным из КНФ C множеством X', называется множество $\Delta_{X'}(C)$ КНФ, полученных подстановками в C векторов $Y_j, j \in \{1,...,k\}$: $\Delta_{X'}(C) = \{C_1 = C|_{Y_1},...,C_k = C|_{Y_k}\}$, $\Delta_{\varnothing}(C) = \{C\}$. КНФ, полученная подстановкой в C вектора Y_j, обозначается через $C_j = C|_{Y_j}$.

Пусть $\Delta_{X'}(C) = \{C_1,...,C_k\}$ – декомпозиционное семейство КНФ, порожденное из КНФ C некоторым декомпозиционным множеством X' мощности d. Всякому набору, выполняющему исходную КНФ C, соответствует набор, выполняющий некоторую КНФ из семейства $\Delta_{X'}(C)$. Наоборот, произвольному набору, выполняющему некоторую КНФ из $\Delta_{X'}(C)$, соответствует единственный набор, выполняющий КНФ C. Следовательно, исходная КНФ C выполнима тогда и только тогда, когда выполнима хотя бы одна КНФ семейства $\Delta_{X'}(C)$. Таким образом, решение исходной SAT-задачи для КНФ C сводится к решению, вообще говоря, $k = 2^d$ SAT-задач для КНФ $C_1,...,C_k$ соответственно. Если исходная КНФ C выполнима, то по набору, выполняющему некоторую КНФ семейства $\Delta_{X'}(C)$, можно эффективно перейти к набору, выполняющему исходную КНФ C.

Пусть имеется РВС, состоящая из $r \in \mathrm{N}$ вычислительных единиц. Возможны следующие два случая.

1) $k \le r$ – число КНФ в семействе $\Delta_{X'}(C)$ не превосходит числа вычислительных единиц РВС. В этом случае для каждой КНФ из семейства $\Delta_{X'}(C)$ SAT-задача решается на отдельной вычислительной единице РВС.

2) $k > r$ – число КНФ в семействе $\Delta_{X'}(C)$ больше числа вычислительных единиц РВС.

Крупноблочное распараллеливание SAT-задач для случая $k \le r$ рассматривается, например, в работе [14]. Для случая $k > r$ предлагается следующая процедура.

Процедура 1. Каждому вектору $Y_j, j \in \{1,...,k\}$ ставится в соответствие натуральное число N_j, двоичным представлением которого является вектор Y_j. Данное число назовем натуральным индексом КНФ C_j. Семейство КНФ $\Delta_{X'}(C)$ упорядочивается некоторым образом (например, по возрастанию натуральных индексов соответствующих векторов). Произвольная КНФ из $\Delta_{X'}(C)$ называется *связанной*, если в рассматриваемый момент времени SAT-задача для нее либо уже решена, либо решается на некоторой вычислительной единице РВС. Остальные КНФ называются *свободными*. Выбираются первые r КНФ $C_1,...,C_r$ из семейства $\Delta_{X'}(C)$. Для каждой из выбранных КНФ $C_1,...,C_r$ решается SAT-задача на отдельной вычислительной единице

РВС. Как только освобождается некоторая из r вычислительных единиц РВС, на ней запускается процедура решения SAT-задачи для первой (в смысле введенного выше порядка) свободной КНФ семейства $\Delta_{X'}(C)$. Данный процесс продолжается до тех пор, пока не будет найден выполняющий набор некоторой КНФ из $\Delta_{X'}(C)$, либо пока не будет доказана невыполнимость всех КНФ из $\Delta_{X'}(C)$. Описанная процедура решает SAT-задачу для произвольной КНФ C корректно.

Пусть выбрано некоторое декомпозиционное множество X'. Представляет интерес построение такого $X^{\tilde{}} \subset X'$, использование которого в качестве декомпозиционного множества делает декомпозицию более эффективной, чем на основе X'. Данная проблема весьма нетривиальна. Если мощность $X^{\tilde{}}$ мала, то SAT-задачи, получаемые при декомпозиции КНФ, как правило, весьма сложны. Если мощность $X^{\tilde{}}$ велика, то велика и мощность декомпозиционного семейства $\Delta_{X^{\tilde{}}}(C)$, и в этом случае простота SAT-задач КНФ данного семейства мало что дает. Для решения данной задачи предлагается следующая процедура статистического прогнозирования (см. [7]).

Процедура 2. Используется натуральное число R_0, от значения которого зависит, имеется необходимость формирования случайной выборки или нет. Например, за R_0 можно принять число вычислительных единиц в РВС. Если при некотором $X^{\tilde{}} \subseteq X', |X^{\tilde{}}| = d$ мощность семейства $\Delta_{X^{\tilde{}}}(C)$ слишком велика, то представление о времени соответствующего параллельного вычисления можно составить на основе знания среднего времени решения SAT-задач для серии КНФ, выбранных случайным образом из $\Delta_{X^{\tilde{}}}(C)$. Через q_d обозначаем объем такой выборки. Через Y^d обозначается множество, образованное всеми различными векторами значений переменных из $X^{\tilde{}} : |X^{\tilde{}}| = d$. Каждому значению параметра $d \in \{0,1,...,|X'|\}$ такому, что $2^d > R_0$, ставится в соответствие множество векторов $\{Y_{j_1},...,Y_{j_{q_d}}\}$, выбираемых из $Y(X')$ в соответствии с равномерным распределением, а также выборка КНФ $\Theta_d = \{C_{j_1} = C|_{Y_{j_1}},...,C_{j_{q_d}} = C|_{Y_{j_{q_d}}}\}$. Каждому значению параметра $d \in \{0,1,...,|X'|\}$ такому, что $2^d \le R_0$, ставится в соответствие множество $Y(X')$ и множество КНФ $\Theta_d = \Delta_{X'}(C)$. Множество выборок $\{\Theta_d\}_{d\in\{0,1,...,|X'|\}}$ обозначается через Θ. Фиксируется SAT-решатель S. Обозначим через $t(C')$ время работы SAT-решателя S на произвольном входе C'. Вводится в рассмотрение функция

$$\tau_S : \Theta \to \mathrm{N}, \tau_S(\Theta_d) = \sum_{C'\in\Theta_d} t(C'),$$

значением которой при каждом фиксированном $d \in \{0,1,...,|X'|\}$ является суммарное время работы SAT-решателя S по всем КНФ из Θ_d. При некоторых значениях параметра d (например, при $d = 0$) КНФ из Θ_d могут оказаться очень сложными для SAT-решателя, и в этом случае время подсчета соответствующего значения прогнозной функции может превысить разумные границы. Для учета данного факта вводится в рассмотрение специальная функция $g(C) = p(m \cdot n)$, здесь m – число дизъюнктов в КНФ C, а $p(\cdot)$ – некоторый полином, степень которого больше 1.

Допустим, что в соответствии с перечисленными правилами построено семейство выборок $\Theta = \{\Theta_d\}_{d\in\{0,1,...,|X'|\}}$ (при фиксированном R_0). Прогнозная функция определяется следующим образом.

$$T(\Theta_d)=\begin{cases}\dfrac{2^d}{q_d}\cdot\tau_S(\Theta_d),\ 2^d>R_0,\ \tau_S(\Theta_d)<g(C);\\ \tau_S(\Theta_d),\ 2^d\le R_0,\ \tau_S(\Theta_d)<g(C);\\ \\ \infty,\ \tau_S(\Theta_d)\ge g(C).\end{cases}$$

Запись «$T(\Theta_d)=\infty$» означает, что функция не определена на выборке Θ_d. Рациональное число $T(\Theta_d)$ является прогнозом времени, требуемого для решения исходной SAT-задачи при декомпозиции КНФ C на семейство КНФ, порожденное множеством X^d. Тем самым задача прогнозного планирования оптимального по трудоемкости параллельного вычисления сводится к задаче минимизации функции T на множестве $domT\subseteq\Theta$. Идея оптимизации функции T состоит в том, что значение $T(\Theta_{|X'|})$ вычисляется заведомо эффективно. Затем значение T итеративно улучшается при обработке остальных выборок из $domT$. Если время обработки выборки превышает пороговое значение, обработка данной выборки прерывается. Результатом работы описанной процедуры является наилучшее (по прогнозу) значение $d_*\in domT$ мощности декомпозиционного множества X', а также соответствующее прогнозное время $T(\Theta_{d_*})$.

Эффективность процедуры 1 существенным образом зависит от структуры декомпозиционного множества. Выбор декомпозиционного множества – это отдельная нетривиальная проблема. Некоторые общие стратегии построения декомпозиционных множеств с ориентацией на задачи криптоанализа генераторов ключевого потока были рассмотрены в [10].

3. Описание MPI-программы PD-SAT

Приведенная в разделе 2 технология была реализована в виде MPI-программы PD-SAT, которую можно также назвать параллельным SAT-решателем. PD-SAT может функционировать в режиме решения SAT-задачи (см. раздел 3.1) и в режиме прогнозирования трудоемкости решения SAT-задачи (см. раздел 3.2). Следует особо отметить принципиальные различия данных, обрабатываемых в указанных режимах. Если в режиме решения заданиями являются списки SAT-задач, то в режиме прогнозирования заданиями являются конкретные SAT-задачи.

В режиме решения SAT-задачи декомпозиционное семейство разбивается на непересекающиеся подсемейства КНФ. Каждое такое подсемейство образует задание, которое обрабатывается на фиксированной вычислительной единице РВС. Под решением задания понимается решение SAT-задач для всех КНФ из соответствующего подсемейства: ответ на задание «UNSAT», если все КНФ из подсемейства невыполнимы; ответ «SAT», если хотя бы одна КНФ из подсемейства выполнима.

Режим прогнозирования реализует описанную в разделе 2 процедуру статистического прогнозирования трудоемкости решения SAT-задачи. В данном режиме заданиями являются SAT-задачи для КНФ, образующих обрабатываемую случайную выборку. Тем самым каждой такой выборке сопоставляется *выборка заданий*.

Все сказанное позволяет выделить следующие классы заданий:

- *Свободные задания* – задания, процесс решения которых на текущий момент не был запущен;
- *Связанные незавершенные задания* – задания, которые решаются на текущий момент;
- *Связанные завершенные задания* – задания, которые на текущий момент уже решены.

3.1 Реализация режима решения SAT-задачи

Данный режим основан на процедуре 1, приведенной в разделе 2. Вычисления разделены на три этапа.

Этап 1. PD-SAT запущен на n процессах: процесс номер 1 управляющий, процессы с номерами 2,…,n – вычислительные. Управляющий процесс по входным данным формирует список заданий, вычислительные процессы при этом простаивают. Число заданий D равно ближайшей справа степени двойки от числа $(n-1)\cdot C$. Здесь C – константа, влияющая на загрузку вычислительных процессов. Данная константа определяется эмпирически; в вычислительных экспериментах, описанных в разделе 4, использовалась $C = 4$.

Пусть, например, дана РВС, состоящая из четырех вычислительных единиц. В MPI-программе один управляющий процесс и три вычислительных. Пусть $C = 2$. В соответствии со сказанным выше управляющий процесс сгенерирует $D = 8$ заданий.

Этап 2. С управляющего процесса отсылаются первые n-1 свободных заданий из списка: i-ое задание (i=1,…,n-1) отсылается на вычислительный процесс с номером i+1 (каждое такое задание становится связанным незавершенным). Каждый вычислительный процесс приступает к обработке полученного задания.

Пример: 1 управляющий процесс, 3 вычислительных процесса, 8 заданий. Схема выполнения второго этапа для данного примера приведена на Рис. 1, управляющий процесс обозначен «УП», вычислительные – «ВП».

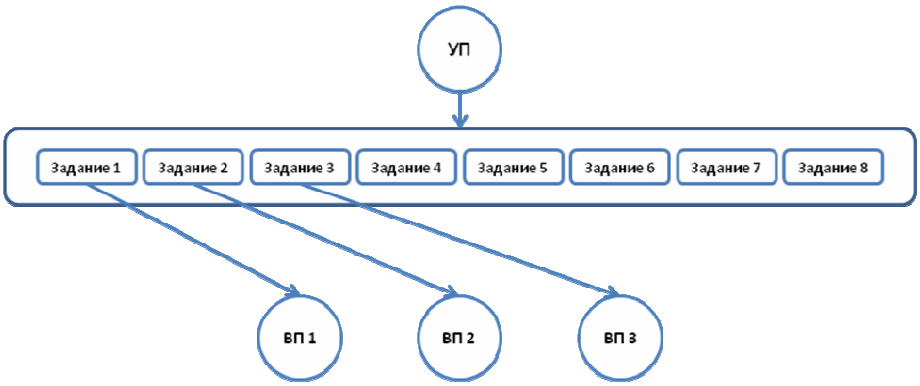

Рис. 1. Пример выполнения этапа 2 режима решения SAT-задачи

Первые два этапа являются подготовительными, выполняются очень быстро, и время их выполнения не вносит значительного вклада в общее время работы.

Этап 3. После выполнения этапов 1–2 управляющий процесс переходит в состояние ожидания решений заданий с вычислительных процессов. Если на управляющий процесс приходит ответ «UNSAT», то задание, ответ на которое был прислан, становится связанным завершенным. На приславший данный ответ вычислительный процесс отправляется очередное свободное задание из списка. Программа завершают свою работу, если на управляющий процесс приходит ответ «SAT» (в этом случае исходная КНФ выполнима) или если получены ответы «UNSAT» на все задания (в этом случае исходная КНФ невыполнима).

В этапах 1–3 используются только блокирующие функции обмена MPI_Send и MPI_Recv (см. [15]). Безусловно, приведенная схема решения проста и присуща многим задачам, допускающим крупноблочное распараллеливание, но ее описание полезно для понимания работы PD-SAT в режиме прогнозирования (см раздел 3.2).

3.2 Реализация режима прогнозирования трудоемкости параллельного решения SAT-задачи

Данный режим основан на процедуре 2, приведенной в разделе 2. Как и в режиме решения SAT-задачи, вычисления в режиме прогнозирования разделены на три этапа. Основные отличия режимов – в функционировании на третьем этапе.

Еще раз отметим, что в режиме прогнозирования заданием является конкретная SAT-задача (в отличие от режима решения, где заданием является список SAT-задач).

Далее используются обозначения, введенные в работе [11]. Через $d^{\min}$ и $d^{\max}$ обозначаются натуральные числа, определяющие соответственно нижнюю и верхнюю границы интервала,

в котором изменяются значения d , где d – размерность декомпозиционного множества (см. раздел 2). Через q обозначается жестко заданное значение, определяющее число КНФ в произвольной случайной выборке, а через r – число вычислительных единиц РВС, относительно которого строится прогноз. В режиме прогнозирования на вход PD-SAT подаются параметры $d^{\min}$, $d^{\max}$, q, r.

Этап 1. Как и в режиме решения SAT-задачи, PD-SAT использует *n* процессов. На управляющем процессе формируется список заданий. Для каждого значения $d \in \{d^{\min},...,d^{\max}\}$ строится отдельная выборка заданий. В случае $2^d > q$ в выборку заданий включаются SAT-задачи для q случайным образом выбранных КНФ из декомпозиционного семейства. Если $2^d \le q$, то в выборку включаются SAT-задачи для всех КНФ из декомпозиционного семейства. Задания из всех выборок объединяются в единый параллельный список, притом первыми в списке располагаются задания из выборки, полученной при $d = d^{\max}$. Далее задания располагаются по убыванию значения d, последними в списке расположены задания из выборки, полученной при $d = d^{\min}$.

Этап 2. Подобно режиму решения SAT-задачи, на данном этапе с управляющего процесса отсылаются первые *n*-1 свободных заданий из списка: *i*-ое задание (*i*=1,…,*n*-1) отсылается на вычислительный процесс с номером *i*+1.

Этап 3. На данном этапе осуществляется параллельная обработка различных выборок заданий с целью построения прогнозов трудоемкости решения исходной SAT-задачи при использовании соответствующей декомпозиции. Основной на данном этапе является процедура прогнозирования GetPredict, работающая на управляющем процессе. Данная процедура, во-первых, определяет параметры лучшего на текущий момент прогноза трудоемкости решения исходной SAT-задачи. Во-вторых, она определяет, обработка каких выборок заданий должна быть прервана, ввиду превышения соответствующими процессами текущих ограничений на время работы. Процедура GetPredict запускается через малые временные интервалы (на практике использовался двухсекундный интервал). На входе GetPredict получает следующие массивы:

- cnf_real_time_arr (в данном массиве содержится получаемая от вычислительных процессов информация о времени обработки связанных завершенных заданий);
- cnf_appr_time_arr (данный массив строится на управляющем процессе и содержит информацию о времени обработки связанных незавершенных заданий);
- cnf_status_arr – массив статусов заданий;
- set_status_arr – массив статусов выборок заданий.

Статус задания и статус выборки заданий – это динамически изменяющиеся параметры, которые в различные моменты вызова GetPredict могут принимать различные значения.

В текущий момент статус задания может принимать следующие значения:

- WAIT, если задание свободное или связанное незавершенное;
- STOP, если задание находится в выборке, обработка которой прерывается. Задания, получившие статус STOP (в том числе и свободные на текущий момент), в дальнейшем не обрабатываются;
- UNSAT, если задание связанное завершенное и соответствующая ему КНФ оказалась невыполнимой;
- SAT, если задание связанное завершенное и соответствующая ему КНФ оказалась выполнимой.

В текущий момент статус выборки заданий может принимать следующие значения:

- WAIT, если в выборке имеются задания со статусом WAIT, но нет ни одного задания со статусом SAT;
- SAT, если хотя бы одно задание из выборки получило статус SAT;
- STOP, если счет для выборки прерван;
- UNSAT, если все задания из выборки имеют статус UNSAT.

На выходе GetPredict выдает измененные массивы cnf_appr_time_arr, cnf_status_arr, set_status_arr, а также массив cnf_to_stop_arr, содержащий номера вычислительных процессов, на которых должна быть прервана обработка текущих заданий (данный массив может быть пустым).

Прерывание обработки заданий достигается за счет отправки с управляющего процесса неблокирующих сообщений (используются функции MPI_Isend) о прерывании на вычислительные процессы с номерами из массива cnf_to_stop_arr (если данный массив не пуст).

От вычислительных процессов требуется не только получать и решать задания, но и периодически проверять наличие сообщений о прерываниях. В используемые SAT-решатели были внесены изменения, позволяющие осуществлять такую проверку за счет применения неблокирующих функций MPI_Iprobe. Если сообщение о прерывании есть, то работа SAT-решателя досрочно завершается, выдается ответ «UNSAT». Даже если КНФ была на самом деле выполнимой, для прогнозирования это не важно. Сообщение с ответом «UNSAT» отправляется на управляющий процесс, после чего принимается следующее задание.

Между периодическими запусками процедуры прогнозирования управляющий процесс переходит в состояние ожидания ответов от вычислительных. Если от вычислительного процесса присылается ответ «UNSAT», на приславший этот ответ вычислительный процесс отправляется очередное свободное задание из списка, время решения SAT-задачи заносится в массив cnf_real_time_arr. Процедура прогнозирования завершает свою работу, если управляющий процесс получил ответ «SAT» (в этом случае SAT-задача для исходной КНФ решена в режиме прогнозирования, исходная КНФ выполнима) или если для всех заданий получены ответы «UNSAT».

Применение неблокирующих обменов позволяют каждому процессу эффективно использовать свое рабочее время: управляющий процесс занимается отправкой заданий и обработкой решений, вычислительные процессы решают SAT-задачи.

Дополнительно отметим, что в PD-SAT предусмотрена процедура отслеживания «опоздавших» сообщений о прерывании: такие сообщения могут возникать вследствие того, что за время обработки данных управляющим процессом на некотором вычислительном процессе было решено задание из выборки, обработку которой необходимо было прервать (но этого не было сделано из-за загруженности управляющего процесса). В этом случае сообщение о прерывании от управляющего процесса может быть некорректно интерпретировано. Такого рода ситуации исключаются за счет дополнительной проверки статусов сообщений, поступающих на вычислительные процессы от управляющего.

В PD-SAT используются следующие SAT-решатели, основой которых является известный решатель Minisat (см. [16]):

- dminisat, основанный на MiniSat-C_v1.14.1. (версия на языке C), оптимизирован для решения SAT-задач, кодирующих задачи обращения дискретных функций (см. [10]);
- minisat2, без существенных изменений;
- minisat2_mod, основан на minisat2, внесены изменения в ключевые параметры-константы, добавлено увеличение активности ядровых переменных.

Изначально SAT-решатели семейства Minisat предназначены только для работы под Unix-подобными операционными системами (ОС). В исходный код всех используемых в PD-SAT SAT-решателей были внесены изменения, обеспечивающие платформонезависимость (в смысле переносимости на уровне исходного кода, см. [17]). Тем самым, PD-SAT может функционировать как под управлением Unix-подобных ОС, так и под управлением ОС семейства Windows.

4. Вычислительные эксперименты

В данном разделе приведены результаты криптоанализа ряда генераторов ключевого потока, полученных, в том числе, с использованием программы PD-SAT.

Последовательный логический криптоанализ, реализованный на обычном персональном компьютере (ПК), оправдал себя применительно к генераторам Геффе и Вольфрама (см. [18], [19]). Применение комплекса TransAlg (см. [20]) позволило получить более экономные КНФ-

представления ряда криптографических алгоритмов в сравнении с полученными ранее посредством LC-комплекса (см. [21]). Данный факт, а также адаптация SAT-решателей к задачам обращения дискретных функций (см. [10]), позволили осуществить последовательный логический криптоанализ суммирующего генератора (см. [22], [23], [24]) на основе трех регистров сдвига с линейной обратной связью (РСЛОС), задаваемых следующими полиномами обратной связи: $X^{19}+X^{18}+X^{17}+X^{14}+1;\ X^{22}+X^{21}+1;\ X^{23}+X^{22}+X^{21}+X^{8}+1$. Длина инициализирующей последовательности составляет 66 бит (64 бита – начальное заполнение РСЛОС 1–3 и 2 бита – начальное заполнение регистров сумматора), анализировался фрагмент ключевого потока длиной 180 бит. В таблице 1 приведены результаты последовательного логического криптоанализа суммирующего генератора данной конфигурации для всех трех используемых в PD-SAT SAT-решателей (см. раздел 3).

Таблица 1. Результаты последовательного логического криптоанализа 66-битного суммирующего генератора на основе трех РСЛОС.

Время решения / SAT-решатель	Минимальное	Максимальное	Среднее
minisat2	1 мин.	4 ч. 40 мин.	1 ч. 17 мин.
minisat2_mod	4 мин.	1 ч. 30 мин.	58 мин.
dminisat	21 мин.	8 ч. 37 мин.	2 ч. 51 мин.

Несмотря на все сказанное, в отношении описанных в таблице 2 генераторов последовательный логический криптоанализ по-прежнему неэффективен.

Таблица 2. Описание генераторов.

Описание генератора	РСЛОС	Длина фрагмента ключевого потока	Размер КНФ	
			Переменных	Дизъюнктов
Пороговый 5 РСЛОС, 72 бита	$X^{11}+X^{9}+X^{4}+X^{2}+1;$ $X^{13}+X^{4}+X^{3}+X+1;$ $X^{15}+X^{5}+X^{4}+X^{2}+1;$ $X^{16}+X^{6}+X^{4}+X+1;$ $X^{17}+X^{6}+X^{4}+X^{2}+1$	150	972	15000
Пороговый 5 РСЛОС, 80 бит	$X^{13}+X^{10}+X^{8}+X^{5}+1;$ $X^{15}+X^{13}+X^{3}+X+1;$ $X^{16}+X^{13}+X^{8}+X^{2}+1;$ $X^{17}+X^{6}+X^{4}+X^{2}+1;$ $X^{19}+X^{18}+X^{17}+X^{14}+1$	150	980	15000
Суммирущий, 4 РСЛОС, 63 бита	$X^{13}+X^{4}+X^{3}+X+1;$ $X^{15}+X^{5}+X^{4}+X^{2}+1;$ $X^{16}+X^{6}+X^{4}+X+1;$ $X^{17}+X^{6}+X^{4}+X^{2}+1$	180	1683	19266

Для параллельного логического криптоанализа перечисленных генераторов была использована описанная в разделе 3 MPI-программа PD-SAT. Вычислительные эксперименты осуществлялись на кластере Blackford Multicore ИДСТУ СО РАН (см. [25]), который имеет следующие основные характеристики: 20 вычислительных узлов; 40 четырехъядерных процессоров Intel Xeon Quad-Core E5345 2.33 GHz; пиковая производительность – 1,493 TFlops; наивысшая производительность по Linpack – 924,4 GFlops; интерконнект 2 x Gigabit Ethernet; ОС Gentoo Linux.

При проведении вычислительных экспериментов кластер Blackford Multicore рассматривался как РВС (далее «РВС Blackford»), вычислительная единица которой состоит из одного ядра процессора Intel Xeon Quad-Core E5345 2.33 GHz, общей оперативной памяти и жесткого диска. В целях единообразного представления далее приводятся сравнительные результаты последовательного решения SAT-задачи на одной вычислительной единице РВС Blackford и результаты параллельного решения SAT-задачи при помощи программы PD-SAT в данной РВС.

При параллельном решении SAT-задач, кодирующих криптоанализ 63-битного суммирующего генератора на основе четырех РСЛОС, использовались 129 вычислительных единиц РВС Blackford. Для остальных генераторов использовались 72 вычислительные единицы.

В таблице 3 приведены результаты прогнозирования трудоемкости параллельного логического криптоанализа рассматриваемых генераторов и соответствующие прогнозу параметры декомпозиции.

Таблица 3. Результаты прогнозирования.

SAT-решатель / Генератор/ Число используемых единиц РВС	minisat2	minisat2_mod	dminisat
Пороговый, 5 РСЛОС, 72 бита / 72 единицы РВС	7 мин. 38 сек. 13 переменных	8 мин. 17 сек. 16 переменных	6 мин. 55 сек. 15 переменных
Пороговый, 5 РСЛОС, 80 бит / 72 единицы РВС	4 ч. 20 мин. 14 переменных	4 ч. 9 мин. 24 переменных	4 ч. 4 мин. 26 переменных
Суммирующий, 4 РСЛОС, 63 бита / 129 единиц РВС	42 ч. 4 мин. 28 переменных	14 ч. 18 мин. 25 переменных	13 ч. 9 мин. 24 переменных

Для каждого генератора представлены примеры оптимизации прогнозной функции (см. Рис. 2–4), полученные при использовании в PD-SAT SAT-решателя dminisat (он оказался лучшим по прогнозу среди SAT-решателей). Заштрихованные сеткой столбцы означают, что соответствующие вычисления прогнозной функции были прерваны из-за превышения текущего порогового значение (см. раздел 2). С использованием параметров декомпозиции, найденных при помощи процедур прогнозирования (см. таблицу 3), осуществлен криптоанализ перечисленных генераторов, результаты приведены в таблице 4.

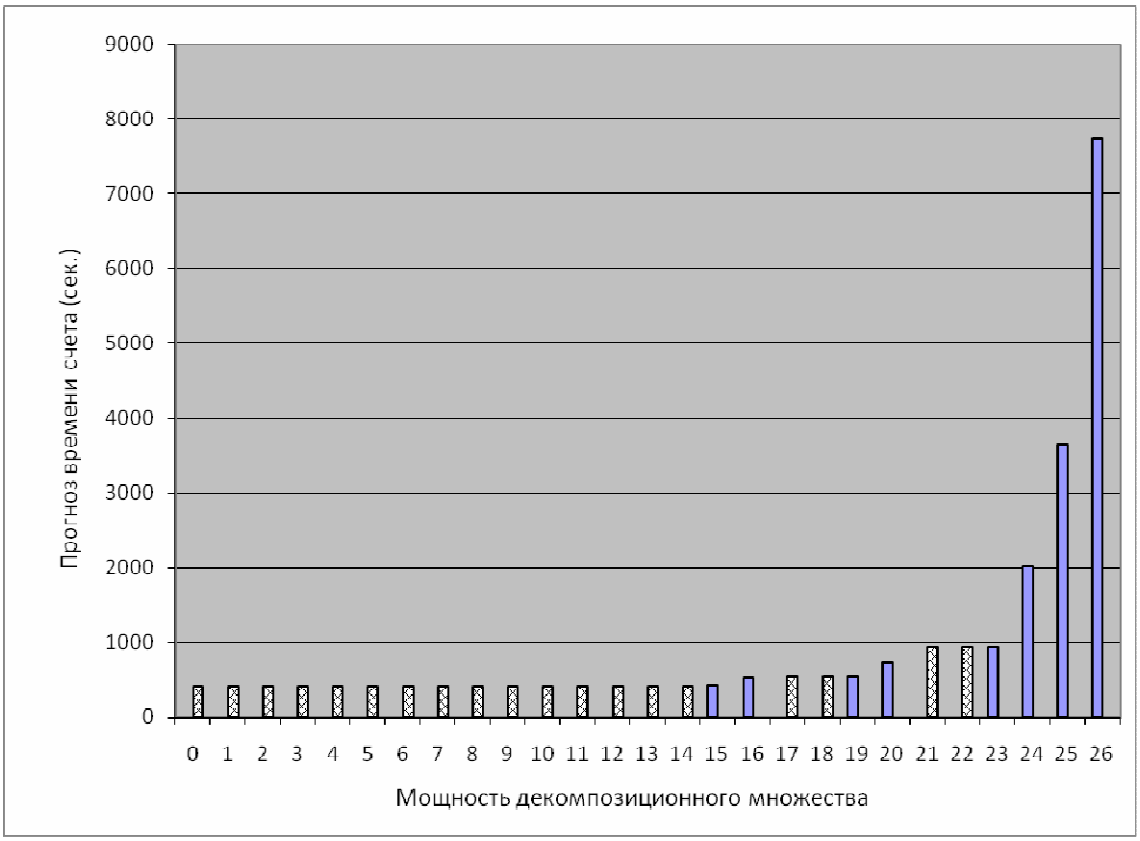

Рис. 2. Пример оптимизации прогнозной функции для 72-битного порогового генератора (один тест)

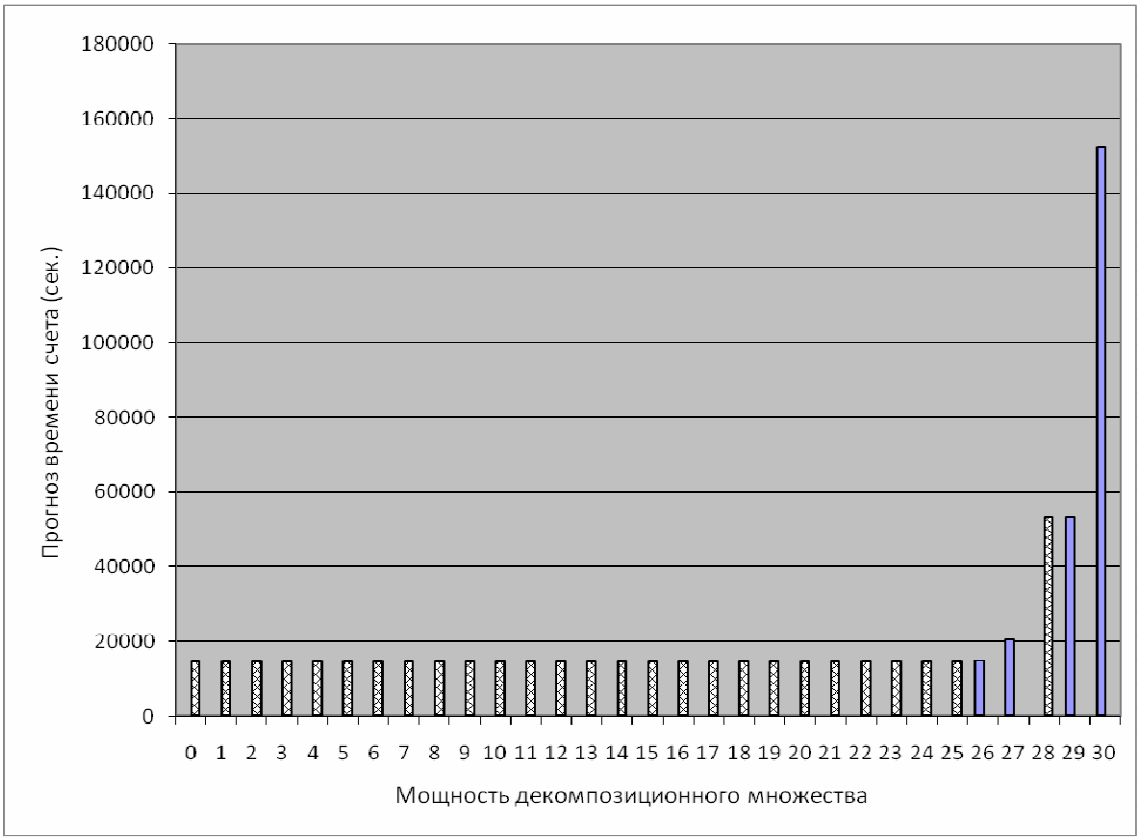

Рис. 3. Пример оптимизации прогнозной функции для 80-битного порогового генератора (один тест)

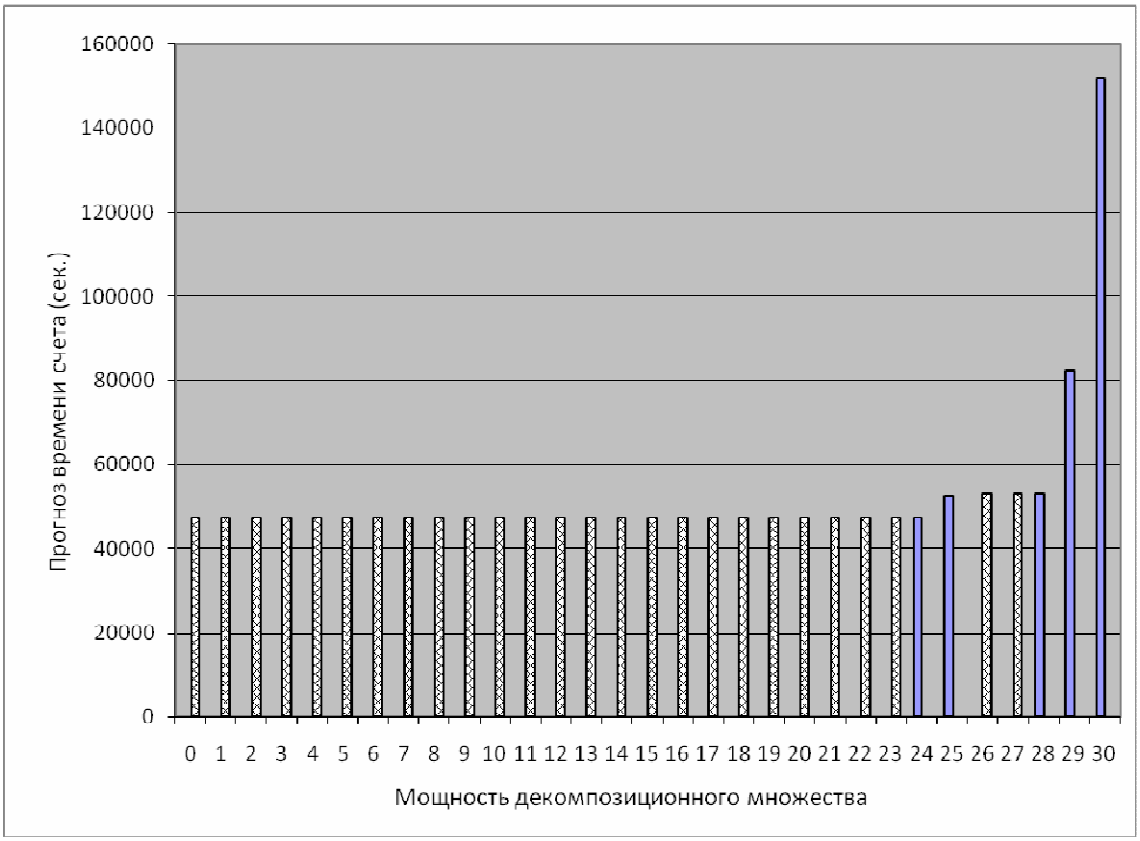

Рис. 4. Пример оптимизации прогнозной функции для 63-битного суммирующего генератора на основе четырех РСЛОС (один тест)

Таблица 4. Результаты криптоанализа некоторых генераторов (серия тестов).

Генератор / Число используемых единиц РВС	Время параллельного решения			Время последовательного решения
	Минимальное	Максимальное	Среднее	
Пороговый, 5 РСЛОС, 72 бита / 72 единицы РВС	3 мин. 57 сек.	13 мин. 4 сек.	6 мин. 29 сек.	> 1 суток (вычисление прервано)
Пороговый, 5 РСЛОС, 80 бит / 72 единицы РВС	34 мин.	3 ч. 17 мин.	1 ч. 34 мин.	> 1 суток (вычисление прервано)
Суммирующий, 4 РСЛОС, 63 бита / 129 единиц РВС	12 мин.	5 ч.	2 ч. 6 мин.	> 2 суток (вычисление прервано)

Заключение

В работе представлена реализация крупноблочной параллельной технологии решения SAT-задач в виде MPI-программы PD-SAT. Данная программа позволяет осуществлять прогнозирование трудоемкости решения SAT-задач и их непосредственное решение в рамках любой распределенной вычислительной среды с установленной коммуникационной MPI-средой.

На серии численных экспериментов продемонстрировано успешное использование PD-SAT в решении задач логического криптоанализа ряда поточных систем шифрования, последовательный логический криптоанализ в отношении которых не дал приемлемых результатов.

Предполагается дальнейшее развитие представленной в работе технологии и ее применение в параллельном логическом криптоанализе других систем шифрования.

Автор благодарит Семенова Александра Анатольевича за внимание к работе и участие в обсуждении основных результатов.

Литература

1. Семенов А.А. О сложности обращения дискретных функций из одного класса // Дискретный анализ и исследование операций. 2004. Т. 11. № 4. С. 44-55.
2. Семенов А.А., Беспалов Д.В. Технологии решения многомерных задач логического поиска // Вестник Томского гос. ун-та. – Приложение. – 2005. – № 14. – С. 61-73.
3. Luís Gil, Paulo Flores, Luís Miguel Silveira. PMSat: a parallel version of MiniSAT. Journal on Satisfiability, Boolean Modeling and Computation, 2008. –Volume 6. –71-98.
4. Tobias Schubert, Matthew Lewis, Bernd Becker. PaMiraXT: Parallel SAT Solving with Threads and Message Passing // Journal on Satisfiability, Boolean Modeling and Computation, 2009. – Volume 6. –P. 203-222.
5. Бандман О.Л. Мелкозернистый параллелизм в вычислительной математике // Программирование. –2001. –№ 4. –С. 5-20.
6. Miroslav N. Velev, Randal E. Bryant. Effective use of Boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors // Journal of Symbolic Computation, 2003. –Volume 35, Issue 2. –P. 73-106.
7. Заикин О.С., Семенов А.А. Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. №1. С. 43-50.
8. Семенов А.А., Заикин О.С. Неполные алгоритмы в крупноблочном параллелизме комбинаторных задач // Вычислительные методы и программирование. – 2008. т. 9. – С. 108-118.
9. Заикин О.С., Семенов А.А., Сидоров И.А., Феоктистов А.Г. Параллельная технология решения SAT-задач с применением пакета прикладных программ D-SAT. // Вестник ТГУ. – Приложение. 2007. – № 23. – С. 83-95.

10. Семенов А.А., Заикин О.С., Беспалов Д.В., Буров П.С., Хмельнов А.Е. Решение задач обращения дискретных функций на многопроцессорных вычислительных системах // Труды Четвертой Международной конференции «Параллельные вычисления и задачи управления» PACO'2008 (Москва 26-29 октября 2008). – С. 152-176.

11. Заикин О.С. Декомпозиционные представления данных в крупноблочном параллелизме SAT-задач // Прикладные алгоритмы в дискретном анализе. Иркутск: ИГУ, 2008. – Серия: Дискретный анализ и информатика, вып. 2. – С. 49-69.

12. Заикин О.С. Пакет прикладных программ Distributed-SAT: Свидетельство об официальной регистрации программы для ЭВМ № 2008610423. – М.: Федеральная служба по интеллектуальной собственности, патентам и товарным знакам, 2008.

13. Гришагин В.А., Свистунов А.Н. Параллельное программирование на основе MPI. Учебное пособие. – Нижний Новгород: издательство ННГУ им. Н.И. Лобачевского, 2005.

14. Опарин Г.А., Богданова В.Г., Сидоров И.А. Интеллектуальный решатель задач в булевых ограничениях в распределенной вычислительной среде // Информационные и математические технологии в науке и управлении. – Иркутск: ИСЭМ РАН, 2007. – С. 32-40.

15. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.

16. SAT-решатель MiniSat: [http://minisat.se/MiniSat.html], 10.12.2009.

17. Олейников А.Я. Методика тестирования на соответствие стандартам, обеспечивающим переносимость прикладных программ (POSIX). – Москва, 1999. – 31 с.

18. Семенов А.А. Логико-эвристический подход в криптоанализе генераторов двоичных последовательностей // Труды международной научной конференции ПАВТ'07. Челябинск, ЮУрГУ, 2007. – Т. 1, С. 170-180.

19. Семенов А.А., Заикин О.С., Беспалов Д.В., Ушаков А.А. SAT-подход в криптоанализе некоторых систем поточного шифрования // Вычислительные технологии. 2008. – Т. 13, № 6. – С. 134-150.

20. Буров П.С., Игнатьев А.С., Отпущенников И.В. Программная трансляция алгоритмов в логические выражения в задачах диагностирования дискретных систем // Материалы IX школы-семинара «Математическое моделирование и информационные технологии», Иркутск, 2007. – С. 33-35.

21. Буранов Е.В. Программная трансляция процедур логического криптоанализа симметричных шифров // Вестник Томского гос. ун-та. – Приложение. – 2004. – № 9 (1). – С. 60-65.

22. Rueppel R.A., Correlation immunity and the summation combiner. In Lecture Notes in Computer Science 218; Advances in Cryptology: Proc. Crypto'85, H. C. Williams Ed., Santa Barbara, CA, Aug. 18-22, 1985, P. 260-272. Berlin: Springer-Verlag, 1986.

23. Menezes A., Van Oorschot P., Vanstone S. Handbook of Applied Cryptography. – CRC Press, 1996. – 657 с.

24. Поточные шифры. Результаты зарубежной открытой криптологии. – М.:Мир, 1997. – 389 с.

25. Суперкомпьютерный центр ИДСТУ СО РАН [http://www.mvs.icc.ru], 10.12.2009.