УДК 519.6

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ РЕШЕНИЯ ПРОБЛЕМЫ ВЫПОЛНИМОСТИ В ПРИМЕНЕНИИ К ОПТИМИЗАЦИОННЫМ ЗАДАЧАМ С БУЛЕВЫМИ ОГРАНИЧЕНИЯМИ

О. С. Заикин[1], **И. В. Отпущенников**[1], **А. А. Семенов**[1]

Предложена параллельная технология, которая может использоваться при решении ряда задач дискретной оптимизации. Технология основана на эффективных процедурах сведения задач комбинаторной оптимизации к задачам выполнимости (SAT-задачам). Процесс решения оптимизационной задачи реализован в виде итерационной схемы, каждый этап которой — это решение некоторой SAT-задачи. Получаемые SAT-задачи решаются при помощи различных схем распараллеливания. Для учета информации, накопленной в предыдущих итерациях, реализована техника "Incremental SAT", применяемая в задачах верификации моделей дискретных систем. Разработанная технология была протестирована на некоторых комбинаторных задачах, в частности на квадратичной задаче о назначениях. Работа выполнена при поддержке гранта "Лаврентьевский конкурс молодежных проектов СО РАН" на 2010–2011 гг. и при финансовой поддержке РФФИ (код проекта 11–07–00377-а). Статья рекомендована к публикации Программным комитетом Международной научной конференции "Параллельные вычислительные технологии" (ПАВТ–2011; http://agora.guru.ru/pavt2011).

Ключевые слова: обращение дискретных функций, задача выполнимости (SAT-задача), булевы уравнения, задачи комбинаторной оптимизации.

1. Введение. В настоящей статье рассматриваются некоторые комбинаторные задачи, являющиеся NP-трудными в общей постановке. Многие практически важные области исследований немыслимы без умения решать такие задачи. Речь идет о проблемах синтеза и верификации дискретных управляющих/управляемых систем, проблемах экономики, производственного планирования, логистики и многих других. В данных областях вопросы построения практически эффективных алгоритмов чрезвычайно актуальны. При этом зачастую необходимо уметь находить именно точные решения, поскольку приближенные, даже имеющие малую погрешность, оказываются бесполезны (например, в задачах верификации).

Далее мы рассматриваем так называемый "пропозициональный подход" в применении к решению ряда задач дискретной оптимизации. Данный подход подразумевает нахождение точных решений и состоит из двух составляющих: алгоритмы сведения комбинаторных задач к булевым уравнениям и символьные алгоритмы поиска решений получаемых уравнений. В последние годы интерес именно к такому рассмотрению многих комбинаторных задач заметно усилился в связи с существенным прогрессом в алгоритмике булевых решателей, а также в связи с бурным развитием параллельных вычислительных технологий, поскольку булевы задачи допускают весьма естественные формы параллелизма.

Методам решения булевых уравнений посвящено довольно много источников — от фундаментальной монографии [1] до многочисленных в последние годы работ по SAT-задачам. Под SAT-задачами (от англ. "satisfiability", т.е. "выполнимость") понимаются задачи поиска выполняющих наборов булевых формул, как правило, приведенных к конъюнктивным нормальным формам (КНФ).

Статей, специально посвященных преобразованиям комбинаторных проблем в булевы уравнения, относительно мало (можно сослаться на обзорную статью [2] и список литературы к ней). При этом можно отметить, что в подавляющем большинстве эти результаты имеют характер наглядных примеров и правдоподобных рассуждений — самой строгой в этом смысле продолжает оставаться процедура, фигурирующая в оригинальном и последующих доказательствах теоремы Кука [3, 4]. Особо подчеркнем трудности с применением перечисленных подходов к достаточно общим классам комбинаторных проблем — сведения, описанные в [1], слишком специфичны, а известные варианты теоремы Кука доказаны в отношении машины Тьюринга — модели, которая крайне далека от современных ЭВМ в плане языка и организации вычислений.

[1] Институт динамики систем и теории управления СО РАН, ул. Лермонтова, 134, 664033, г. Иркутск; О. С. Заикин, науч. сотр., e-mail: oleg.zaikin@icc.ru; И. В. Отпущенников, программист, e-mail: otilya@yandex.ru; А. А. Семенов, зав. лабораторией, e-mail: biclop@rambler.ru

В статье [5] были описаны механизмы пропозиционального кодирования программ, вычисляющих дискретные функции на машинах с произвольным доступом к памяти (RAM). Язык данной модели очень естествен — фактически это фрагмент ассемблера современных ЭВМ. Там же в общих чертах описаны основные принципы высокоуровневой трансляции алгоритмов, вычисляющих дискретные функции, в булевы уравнения. Реализацией этих идей стал программный комплекс Transalg [6], архитектура которого и функциональные возможности в применении к решению задач обращения дискретных функций описаны в следующем разделе статьи.

Комплекс Transalg применяется для сведения разнообразных дискретных задач к SAT-задачам. В отношении последних возможно использование различных форм параллелизма, в том числе крупноблочного [7, 8], что особенно перспективно в свете интенсивного развития распределенных вычислений.

В настоящей статье мы описываем механизмы сведения к проблеме выполнимости квадратичной задачи о назначениях QAP (Quadratic Assignment Problem), которая является одной из наиболее трудных комбинаторных задач. Получаемые SAT-задачи решались на вычислительном кластере с применением комбинации различных схем распараллеливания и техники Incremental SAT.

2. Сведение задач комбинаторной оптимизации к задаче о пропозициональной выполнимости (SAT). Программный комплекс Transalg предназначен для сведения к булевым уравнениям (и в том числе к SAT-задачам) задач обращения полиномиально вычислимых дискретных функций. С этой целью алгоритм вычисления функции записывается на специальном C-подобном языке, получившем название язык трансляции алгоритмов (TA-язык), после чего происходит трансляция полученной TA-программы в систему булевых уравнений. На заключительном этапе трансляции система приводится к одной из возможных нормальных форм (“КНФ=1”, “ДНФ=0”, полиномиальные уравнения над полем $GF(2)$). Кроме того, предусмотрена возможность построения И-НЕ-графа [9], представляющего рассматриваемый алгоритм.

Схематично процесс трансляции TA-программ представлен на рис. 1.

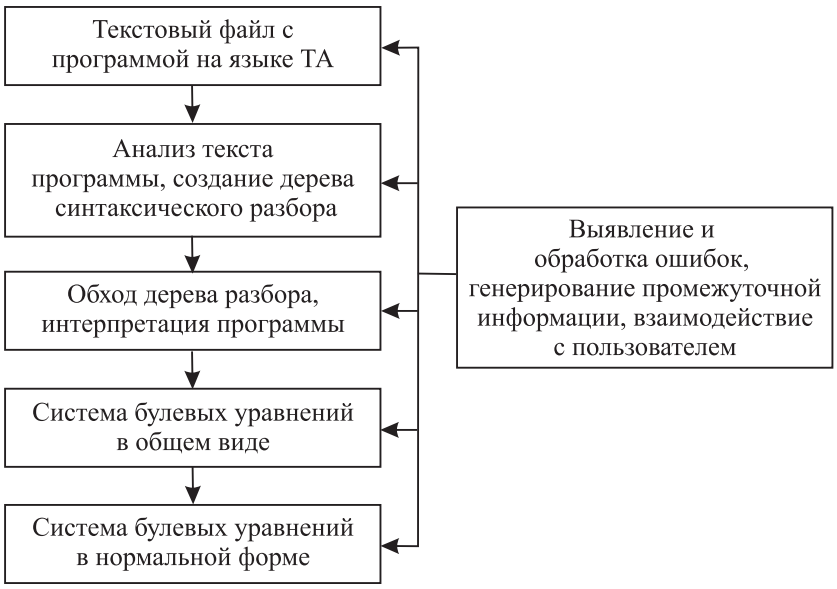

Рис. 1. Общая схема работы программного комплекса Transalg

Фазы анализа текста TA-программы, построения дерева синтаксического разбора и обход полученного дерева с целью интерпретации реализованы стандартным способом (см., например, [10]).

Язык TA представляет собой процедурный язык программирования с блочной структурой и C-подобным синтаксисом. Каждый блок — это конечный список инструкций TA-программы. Программа на языке TA представляет собой набор определений-функций, а также объявлений и определений глобальных переменных и констант. В языке TA реализованы все основные примитивные конструкции, характерные для процедурных языков программирования: объявление/определение переменной или массива переменных; определение именованных констант; оператор присваивания; составной оператор; условный переход; цикл; определение пользовательской функции; возврат из функции; вызов функции.

Следует особо подчеркнуть, что переменные транслируемой TA-программы и переменные пропозиционального кода этой программы представляют разные сущности. Переменные, фигурирующие в тексте транслируемой TA-программы (далее “переменные программы”), понимаются в традиционном смысле — это идентификаторы областей памяти. Переменные, попадающие в пропозициональный код (далее “переменные кода”), понимаются как символы некоторого конечного алфавита. По своему смыслу переменные кода — это переменные итоговой системы булевых уравнений.

В языке TA определен один основной тип данных — тип `bit`. Этот тип служит для объявления переменных программы, связанных с областями памяти ЭВМ, в которых хранятся биты входа, выхода, а также биты, получаемые в процессе вычисления рассматриваемой функции. Это вычисление по своей су-

ти есть последовательность изменений данных в памяти в моменты времени $0, 1, \ldots, e$. В каждый момент i, $i \in \{0, \ldots, e\}$, транслятор сопоставляет рассматриваемой области памяти множество X^i, образованное булевыми переменными (переменными кода), кодирующими содержимое данной области. Тем самым множество X^0 образовано булевыми переменными, кодирующими входные данные, а множество X^e — переменными, кодирующими выходные данные рассматриваемого дискретного преобразования. Множество переменных кода транслируемой программы — это множество $X = \bigcup_{i=0}^{e} X^i$.

Переменные ТА-программы, имеющие тип `bit`, и переменные кода связаны при помощи специальной структуры данных, называемой "`var_object`". Ее назначение будет пояснено далее.

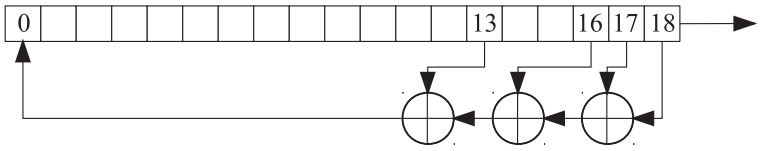

Рис. 2. Схема РСЛОС, реализуемого функцией `shiftReg()`

Кроме типа `bit` в языке ТА используются вспомогательные типы: целочисленный тип `int`, представляющий четырехбайтное знаковое целое число, а также тип `bool` — целое число из множества $\{0, 1\}$.

Пример 1. Рассмотрим ТА-программу, которая реализует регистр сдвига с линейной обратной связью (РСЛОС, [11]), заданной полиномом (над GF(2)) $P(x) = x^{19} + x^{18} + x^{17} + x^{14} + 1$ (рис. 2).

```
_in bit reg[19];
_out bit output[100];
bit shiftReg(){
  bit x = reg[18];
  bit y = reg[18]^reg[17]^reg[16]^reg[13];
  for(int j = 18; j > 0; j = j - 1){
    reg[j] = reg[j - 1];
  }
  reg[0] = y;
  return x;
}
void main(){
  for(int i = 0; i < 100; i = i + 1){
    output[i] = shiftReg();
  }
}
```

Массив булевых переменных `reg` описывает в каждый фиксированный момент времени текущее состояние регистра сдвига. Его содержимое на начальном шаге соответствует входной информации (данный факт отмечен атрибутом `_in`), которая описывается переменными кода, образующими множество входных переменных $X^0 = \{x_1, \ldots, x_{19}\}$.

Представленная программа реализует 100 тактов работы регистра сдвига. В теле основной функции `main()` организован цикл, в котором вызывается функция сдвига регистра `shiftReg()`. Значения, возвращаемые функцией `shiftReg()`, определяют биты выходного слова, которое представлено в программе массивом `output`.

Сдвиг регистра обновляет значения всех элементов массива `reg`. На первом такте данная операция приводит к вводу новых переменных кода, образующих множество $X^1 = \{x_{20}, \ldots, x_{38}\}$. Новые переменные связаны с переменными из множества X^0 следующей системой булевых уравнений:

$$(x_{20} \equiv x_{19} \oplus x_{18} \oplus x_{17} \oplus x_{14}) = 1, \quad (x_{21} \equiv x_1) = 1, \quad (x_{22} \equiv x_2) = 1, \quad \ldots, \quad (x_{38} \equiv x_{18}) = 1.$$

Переменная x_{19} кодирует первый бит ключевого потока, полученный в результате первого сдвига рассматриваемого РСЛОС. Отметим, что локальные переменные `x` и `y` функции `shiftReg()` необходимы лишь для корректной организации вычислений и не связаны с переменными кода программы.

Аналогичным образом можно описывать и преобразовывать в булевы уравнения алгоритмы вычисления дискретных функций из весьма широкого класса. Кроме того, в комплексе Transalg предусмотрена возможность работы с некоторыми задачами, допускающими "непроцедурную" постановку. В частности, к таким задачам могут быть отнесены задачи дискретной оптимизации, в которые требуется найти экстремум некоторой целевой функции на допустимом множестве, определяемом системой ограничений.

Далее мы описываем процесс сведения к SAT-задаче оптимизационных задач из класса 0-1-ЦЛП. Отметим, что общие приемы, используемые в таких сведениях, известны [12], однако в комплексе Transalg дополнительно используется булева минимизация транслируемых функций, которая осуществляется при помощи свободно распространяемой утилиты Espresso [13].

Будем рассматривать систему неравенств

$$Ax \leqslant b, \tag{1}$$

где A — матрица размеров $m \times n$ с целочисленными компонентами, b — вектор длины m, состоящий из целых чисел. Предполагаем, что переменные x_i, $i \in \{1, \ldots, n\}$, могут принимать значения из множества целых чисел $\{0, 1\}$. Требуется при ограничениях (1) минимизировать линейную форму $\langle c, x \rangle$, где c — целочисленный вектор длины n.

Процесс сведения данной задачи к SAT-задаче состоит в преобразовании линейных неравенств, образующих систему (1), в конъюнкции дизъюнктов. При этом можно использовать эквивалентные преобразования исходных ограничений, приводящие к ограничениям вида

$$\langle a', x^\sigma \rangle \leqslant b_0, \tag{2}$$

где a' — вектор длины n с целыми неотрицательными компонентами, x^σ — вектор, образованный литералами над переменными из множества $X = \{x_1, \ldots, x_n\}$, а b_0 — неотрицательное целое число.

Пример 2. Рассмотрим ограничение $3x_1 - 2x_2 + 5x_3 \leqslant 3$. Результатом замены $x_2 = 1 - \overline{x}_2$ является ограничение $3x_1 + 2\overline{x}_2 + 5x_3 \leqslant 5$.

Сказанное означает, что от исходной задачи возможен эффективный переход к задаче минимизации линейной формы $\langle c', x^\sigma \rangle$ при ограничениях вида (2), где c' — вектор, состоящий из неотрицательных целых чисел.

Наиболее простой метод трансляции состоит в том, что псевдобулеву ограничению вида (2) посредством таблицы истинности сопоставляется булева функция $f(x_1, \ldots, x_n)$, принимающая значение 1 тогда и только тогда, когда выполняется ограничение (2). Такой подход возможен при условии, что число n невелико (например, $n \leqslant 20$), т.е. таблица истинности может быть легко построена. В этом случае в систему булевых уравнений добавляется уравнение $\Phi_f(x_1, \ldots, x_n) = 1$, где Φ_f — формула (например, в КНФ), реализующая функцию $f(x_1, \ldots, x_n)$.

В прикладных задачах ЦЛП часто встречаются ограничения, линейная часть которых зависит от сотен переменных, что делает непосредственный переход к таблице истинности невозможным. В этом случае используется представление функции $f(x_1, \ldots, x_n)$ в виде вычисляющего ее алгоритма. Пропозициональный код данного алгоритма — система булевых уравнений, совместная тогда и только тогда, когда выполняется ограничение (2). Данная система строится при помощи комплекса Transalg.

Пример 3. Рассмотрим ограничение $3x_1 + 2\overline{x}_2 \leqslant 5$. На первом шаге для термов $3x_1$ и $2\overline{x}_2$ строятся слова $(x_1\, x_1)$ и $(\overline{x}_2 0)$ (используется тот факт, что числа 3 и 2 представляются двоичными векторами (11) и (10)). Линейной форме $3x_1 + 2\overline{x}_2$ сопоставляется система булевых уравнений

$$(x_4 \equiv x_1) = 1, \quad (x_5 \equiv x_1 \oplus \overline{x}_2) = 1, \quad (x_6 \equiv x_1\overline{x}_2) = 1. \tag{3}$$

Данная система описывает процесс вычисления дискретной функции, которая, получая на входе произвольный двоичный вектор $(x_1\, x_2)$, на выходе выдает число $3x_1 + 2\overline{x}_2$. Данное число кодируется двоичным вектором $(x_4\, x_5\, x_6)$, где x_6 соответствует старшему биту. Тот факт, что $3x_1 + 2\overline{x}_2$ не превосходит числа 5, справедлив тогда и только тогда, когда формула $(\overline{x}_6 \vee x_6\overline{x}_5)$ принимает значение "истина". Таким образом, итоговой системой булевых уравнений, кодирующей ограничение $3x_1 + 2\overline{x}_2 \leqslant 5$, является система (3), дополненная уравнением $(\overline{x}_6 \vee x_6\overline{x}_5) = 1$.

Построенные таким образом системы булевых уравнений приводятся к уравнениям вида "КНФ=1" при помощи преобразований Цейтина [14]. При этом множество переменных разрастается (не более чем полиномиально), однако между множеством решений исходной системы булевых уравнений и множеством решений получаемого уравнения вида "КНФ=1" существует биекция [5].

В комплексе Transalg при построении уравнений вида "КНФ=1" применяются разнообразные приемы оптимизации итогового пропозиционального кода. Было проведено сравнение пропозиционального кода задач 0-1-ЦЛП, полученного с помощью комплекса Transalg, с кодом, полученным известным псевдобулевым решателем MiniSat+ [15]. Сравнение проводилось на серии тестов из библиотеки 0-1-ЦЛП задач [16]. Число переменных в КНФ, получаемых на выходе Transalg, было в среднем меньше на 20%, чем в КНФ, выдаваемых MiniSat+. По числу дизъюнктов, тем не менее, Transalg на данный момент проигрывает MiniSat+ в среднем на те же 20%.

В настоящее время существуют коммерческие решатели задач из класса 0-1-ЦЛП, которые показывают очень хорошие результаты на обширных классах практических тестов [17]. Однако, данные решатели эффективны далеко не на всех задачах дискретной оптимизации. Одной из наиболее трудных комбинаторных задач является квадратичная задача о назначениях (QAP) [18, 19]. Описанная в работе технология сведения к SAT применима к данной задаче. Далее мы приводим общую формулировку задачи QAP и описываем основные этапы ее сведения к SAT-задаче.

Пусть даны некоторые n "объектов", которые размещаются по n "позициям". Дана $(n \times n)$-матрица $D = \|d_{ij}\|$ неотрицательных целых чисел, определяющая расстояния между позициями. Кроме того, дана $(n \times n)$-матрица $F = \|f_{ij}\|$ неотрицательных целых чисел, определяющая "интенсивность потока" между объектами. Предположим, что объекты и позиции пронумерованы натуральными числами от 1 до n и начальное размещение задается тождественной подстановкой: $\begin{pmatrix} 1\,2\ldots n \\ 1\,2\ldots n \end{pmatrix}$.

Рассматривается функция

$$\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{p(i)p(j)}, \tag{4}$$

где p — произвольная подстановка на n-элементном множестве. Требуется найти подстановку $p \in S_n$, доставляющую минимум функции (4).

Для трансляции QAP-задач в SAT был использован программный комплекс Transalg. На начальном шаге трансляции строится $(n \times n)$-матрица $X = \|x_j^i\|$ (так называемая матрица размещений), элементами которой являются булевы переменные. Смысл данной матрицы в том, что она кодирует произвольное размещение объектов по позициям, соответствующее "неопределенной" подстановке p, — булева переменная x_j^i принимает значение "1" тогда и только тогда, когда в результате подстановки p объект с номером i переместился в позицию с номером j. Исходя из этого, можно записать следующую систему булевых ограничений на значения переменных x_j^i, $i, j \in \{1, \ldots, n\}$:

$$\begin{cases} \bigvee\limits_{k=1}^{n} x_k^1 \;\; \&\limits_{j\in\{1,\ldots,n\}\setminus\{k\}} \overline{x}_j^1 = 1, \\ \ldots\ldots\ldots \\ \bigvee\limits_{k=1}^{n} x_k^n \;\; \&\limits_{j\in\{1,\ldots,n\}\setminus\{k\}} \overline{x}_j^n = 1, \end{cases} \qquad \begin{cases} \bigvee\limits_{k=1}^{n} x_1^k \;\; \&\limits_{i\in\{1,\ldots,n\}\setminus\{k\}} \overline{x}_1^i = 1, \\ \ldots\ldots\ldots \\ \bigvee\limits_{k=1}^{n} x_n^k \;\; \&\limits_{i\in\{1,\ldots,n\}\setminus\{k\}} \overline{x}_n^i = 1. \end{cases}$$

Следующий шаг трансляции заключается в построении целевой функции в явном виде, т.е. в виде формулы над переменными x_j^i, $i, j \in \{1, \ldots, n\}$. Для этого транслятор просматривает матрицу F и определяет пары объектов с номерами i_1 и i_2, между которыми интенсивность потока положительна. Для каждой такой пары объектов в матрице X рассматриваются строки с номерами i_1 и i_2 и формируется следующий фрагмент целевой функции (в предположении, что матрица D симметрична):

$$h_{(i_1 i_2)} = f_{i_1 i_2} d_{12}\left(x_1^{i_1} x_2^{i_2} \vee x_1^{i_2} x_2^{i_1}\right) + \ldots + f_{i_1 i_2} d_{(n-1)n}\left(x_{n-1}^{i_1} x_n^{i_2} \vee x_{n-1}^{i_2} x_n^{i_1}\right).$$

Формула, выражающая целевую функцию, имеет следующий вид: $h = \sum\limits_{i_1, i_2 : f_{i_1 i_2} > 0} h_{(i_1 i_2)}$.

Выражение h транслируется в систему булевых уравнений с использованием структур данных и алгоритмов комплекса Transalg.

Следует отметить, что целевая функция в задаче QAP нелинейна по переменным из множества X. Современные коммерческие решатели ЦЛП-задач, комбинирующие методы ветвей, границ и отсечений с методами решения задач линейного программирования над полем рациональных чисел, с такими постановками, как правило, не работают. Сказанное означает перспективность представленной технологии трансляции псевдобулевых задач в булевы уравнения в исследовании задач дискретного программирования, в том числе с нелинейными ограничениями и невыпуклыми целевыми функциями.

На основе описанных выше механизмов трансляции задач дискретной оптимизации в SAT и известного SAT-решателя MiniSat2.0 [20] был создан новый решатель псевдобулевых задач "PBSolver". На вход данный решатель получает описание исходной задачи (0-1-ЦЛП или QAP) в одном из стандартных форматов; процедуры разбора входного файла, а также процедуры трансляции ограничений задачи и целевой функции в КНФ встроены в решатель. Пусть $f(x)$ — целевая функция исходной задачи. Поиск оптимального решения осуществляется итеративным вызовом SAT-решателя. На нулевой итерации SAT-решатель

запускается на КНФ, которая кодирует только ограничения, описывающие допустимое множество исходной задачи. Если КНФ невыполнима (решатель выдает ответ "UNSAT"), то допустимое множество пусто. В противном случае находится некоторая допустимая точка x_0 и строится первое приближение $\big(x_0, f\,(x_0)\big)$. Затем к имеющейся системе добавляется ограничение $f(x) < f\big(x_0\big)$ и на основе полученной системы формируется новая SAT-задача. Тем самым, описанный процесс решения задачи — это схема последовательных приближений, итогом которой является гарантированное нахождение оптимального решения за конечное число итераций.

3. Процедуры распараллеливания задач комбинаторной оптимизации, представленных в форме SAT-задач. Одной из причин построения решателя PBSolver была необходимость разработки параллельных технологий решения задач дискретной оптимизации. Ниже описываются общие принципы распараллеливания таких задач, представленных в форме SAT-задач в соответствии с описанной выше техникой трансляции.

Пусть допустимое множество рассматриваемой задачи не пусто и x_0 — начальное приближение. Наиболее очевидная схема распараллеливания состоит в разбиении интервала $\big[0, f(x_0)\big)$ на непересекающиеся (но покрывающие весь этот интервал) интервалы меньшей длины. Каждому такому интервалу соответствует некоторая SAT-задача. Полученное семейство SAT-задач обрабатывается как параллельный список. Однако для таких задач как QAP более перспективной на наш взгляд является описываемая далее схема, которая близка в идейном плане схемам, использованным при обращении дискретных функций [7, 8]. В соответствии с данной схемой выбирается некоторое множество булевых переменных, варьирование всевозможных значений которых позволяет построить декомпозиционное семейство, образованное SAT-задачами меньшей размерности (в сравнении с исходной). Полученное декомпозиционное семейство обрабатывается как параллельный список. Данная технология была реализована в форме параллельной программы PD-SAT [8], использующей библиотеку MPI. В качестве вычислительного ядра в PD-SAT может использоваться произвольный последовательный SAT-решатель. При решении задач дискретной оптимизации в роли вычислительного ядра в PD-SAT использовался решатель PBSolver.

Далее мы описываем процесс обработки списка заданий в PD-SAT в терминах стандарта MPI (управляющий процесс/вычислительные процессы [21]).

Обрабатываемыми заданиями являются SAT-задачи из декомпозиционного семейства с дополнительными ограничениями, определяющими текущее рекордное значение целевой функции. Вычисления разделены на три этапа.

Этап 1. PD-SAT запущен на n процессах: процесс номер 1 управляющий, процессы с номерами $2, \ldots, n$ — вычислительные. На управляющем процессе решается SAT-задача для КНФ C_{constr}, кодирующей описание допустимого множества исходной задачи. Если КНФ C_{constr} невыполнима, то допустимое множество пусто, вычисления прекращаются и выдается ответ "задача не имеет решений". Если C_{constr} выполнима, то из выполняющего ее набора выделяется вектор x_0 — допустимая точка исходной задачи и формируется начальное приближение $\big(x_0, f(x_0)\big)$.

Этап 2. Управляющий процесс по входным данным формирует список заданий. Число заданий D равно ближайшей справа степени 2 от числа $(n-1)d$. Здесь d — константа, влияющая на загрузку вычислительных процессов. Данная константа определяется эмпирически. С управляющего процесса отсылаются первые $n-1$ заданий из списка: i-е задание ($i \in \big\{1, \ldots, n-1\big\}$) отсылается на вычислительный процесс с номером $i+1$. После этого каждый вычислительный процесс приступает к обработке полученного задания.

Этап 3. После выполнения этапов 1 и 2 управляющий процесс переходит в состояние ожидания решений заданий с вычислительных процессов. Решением задания является либо ответ UNSAT, либо ответ SAT и соответствующее ему значение целевой функции. Получив решение с некоторого вычислительного процесса, управляющий процесс отправляет ему очередное задание из списка. Данное задание — это некоторая КНФ из декомпозиционного семейства и текущее рекордное значение целевой функции.

Кроме того, предусмотрена возможность передавать с управляющего процесса на вычислительные только найденные рекордные значения целевой функции (без передачи КНФ из декомпозиционного семейства). Вычислительные процессы периодически проверяют наличие сообщений с управляющего процесса с обновленными рекордными значениями. В решателе PBSolver предусмотрена возможность осуществления таких проверок за счет применения асинхронных обменов. Если сообщение с обновленным рекордным значением получено и оно меньше, чем текущее значение, найденное в процессе работы решателя, то решение текущей SAT-задачи прерывается и решатель формирует новую SAT-задачу с учетом нового рекордного значения. Таким образом, информация, полученная в ходе решения одного задания, может ускорить процесс решения других заданий. Вычисление останавливается после обработки всего параллельного списка заданий.

Дополнительно при обработке списка заданий применяется техника "Incremental SAT" [22], состоящая в том, что часть конфликтных дизъюнктов, накопленных SAT-решателем при обработке предыдущего задания, конъюнктивно приписывается к КНФ, которая является последующим заданием.

4. Результаты вычислительных экспериментов. Вычислительные эксперименты проводились на задачах 0-1-ЦЛП, взятых из библиотеки [23], а также на некоторых задачах из серии QAP [19]. Сразу отметим, что на задачах 0-1-ЦЛП на данном этапе не удалось превзойти коммерческие решатели, комбинирующие методику ветвей, границ и отсечений с линейным программированием над полем рациональных чисел [17]. Однако, как уже отмечалось, такие решатели не работают напрямую с QAP. Данная задача обладает очень интересными особенностями, которые на наш взгляд хорошо сочетаются с описанной выше схемой распараллеливания, поскольку в матрице X, определяющей допустимое множество задачи QAP, в каждой строке и в каждом столбце находится одна единица. Это означает, что декомпозиционное семейство, построенное, например, по первой строке матрицы X длины n, содержит не 2^n, а n заданий. Аналогично, декомпозиционное семейство, построенное по произвольным k строкам, содержит не 2^{nk}, а $n(n-1)\dots(n-k+1)$ заданий.

В таблице приведены результаты численных экспериментов для некоторых задач QAP из библиотеки [19]. Использовалась описанная выше схема распараллеливания с передачей рекордов от управляющего процесса на вычислительные. Эксперименты проводились на кластере Blackford ИДСТУ СО РАН ([24], 40 четырехъядерных процессоров Intel Xeon 5345 EM64T, 2.33 GHz).

Результаты решения тестов из библиотеки QAP lib [19]

Тест	Последовательное решение	Параллельное решение	
	1 ядро	8 ядер	13 ядер
Chr12a	44 м. 15 с.	9 м. 13 с.	7 м. 24 с.
Chr12b	32 м. 8 с.	17 м. 38 с.	12 м. 12 с.
Chr12c	1 ч. 42 м.	24 м. 38 с.	20 м. 27 с.
		8 ядер	16 ядер
Chr15a	14 ч. 33 м.	9 ч. 2 м.	3 ч. 48 м.
Chr15b	4 ч. 36 м.	3 ч. 9 м.	1 ч. 42 м.

На основе полученных результатов можно сделать выводы о том, что задача QAP продолжает оставаться вычислительно трудоемкой даже при использовании различных техник ее распараллеливания. Наилучшие результаты показывает техника, при которой осуществляется межпроцессорный обмен получаемыми рекордами, однако в этом случае возникает проблема балансировки между числом заданий и их сложностью — из результатов видно, что время решения задачи не зависит от числа вычислительных ядер напрямую.

На данный момент объем численных экспериментов невелик. В ближайшее время планируется расширить класс успешно решенных тестов для QAP, в том числе за счет адаптации используемого SAT-решателя.

5. Заключение. В настоящей статье предложена технология распараллеливания, применимая к обширному классу задач дискретной оптимизации. В рамках данной технологии рассматриваемая оптимизационная задача сводится к SAT-задаче с дополнительными (оптимизационными) условиями на выполняющий набор. Процесс сводимости осуществляется при помощи специального программного комплекса, краткому описанию которого посвящен второй раздел статьи. В заключительной части описана технология распараллеливания SAT-задач, кодирующих задачи дискретной оптимизации. В основе технологии лежат идеи, предложенные в более ранних работах авторов. Особенность технологии в применении к оптимизационным постановкам состоит в необходимости мониторинга процесса обновления рекордных значений целевой функции. Перспективность описанной технологии обосновывается возможностью ее применения к решению оптимизационных задач с различными типами ограничений (в том числе с нелинейными ограничениями типа равенств и неравенств) и с различными типами целевых функций. В качестве модельной задачи дискретной оптимизации, на примере которой демонстрировались характерные черты предложенной технологии, была выбрана квадратичная задача о назначениях (QAP).

СПИСОК ЛИТЕРАТУРЫ

1. *Rudeanu S.* Boolean functions and equations. Amsterdam, London: North-Holland Publishing Company, 1974.
2. *Prestwich S.* CNF encodings // Handbook of Satisfiability / Eds. A. Biere, M. Heule, H. van Maaren, T. Walsh. Amsterdam: IOS Press, 2009. 75–97.
3. *Cook S.A.* The complexity of theorem-proving procedures // Proc. 3rd Ann. ACM Symp. on Theory of Computing (STOC 71). New York: ACM. 1971. 151–159.
4. *Garey M.R., Johnson S.* Computers and intractability: A guide to the theory of NP-completeness. New York: W. H. Freeman, 1979.

5. *Семёнов А.А.* Трансляция алгоритмов вычисления дискретных функций в выражения пропозициональной логики // Прикладные алгоритмы в дискретном анализе. Сер. Дискретный анализ и информатика. 2008. Вып. 2. Иркутск: Изд-во ИГУ. 70–98.
6. *Отпущенников И.В., Семенов А.А.* Инструментальное средство трансляции алгоритмов вычисления дискретных функций в выражения исчисления высказываний. Свидетельство о государственной регистрации программы для ЭВМ № 2011611151 (03.02.2011).
7. *Заикин О.С., Семенов А.А.* Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. № 1. 43–50.
8. *Заикин О.С.* Реализация процедур прогнозирования трудоемкости параллельного решения SAT-задач // Вестник УГАТУ. 2010. **14**, № 4(39). 210–220.
9. AIG Format (http://fmv.jku.at/aiger/).
10. *Ахо А., Сети Р., Ульман Дж.* Компиляторы. Принципы, технологии, инструменты. М., СПб, Киев: Вильямс, 2001.
11. *Menezes A., Oorschot P., Vanstone S.* Handbook of applied cryptography. Boca Raton: CRC Press, 1996.
12. *Een N., Sorensson N.* Translating pseudo-Boolean constraints into SAT // J. Satisfiability, Boolean Modeling and Computation. 2006. **2**. 1–25.
13. Espresso heuristic logic minimizer (http://embedded.eecs.berkeley.edu/pubs/downloads/espresso).
14. *Цейтин Г.С.* О сложности вывода в исчислении высказываний // Записки научных семинаров ЛОМИ АН СССР. 1968. **8**. 234–259.
15. MiniSat+ solver (http://minisat.se/MiniSat.html).
16. *Beasley J.E.* Or-library: distributing test problems by electronic mail // J. Oper. Res. Soc. 1990. **41**, N 11. 1069–1072.
17. CPLEX solver for linear and mixed integer programming (http://www.aimms.com/features/solvers/cplex).
18. *Cela E.* The quadratic assignment problem: theory and algorithms. Berlin: Springer, 1998.
19. QAPLIB — A Quadratic Assignment Problem Library (http://www.seas.upenn.edu/qaplib/).
20. Minisat SAT Solver (http://www.minisat.se).
21. *Гришагин В.А., Свистунов А.Н.* Параллельное программирование на основе MPI. Учебное пособие. Нижний Новгород: Изд-во ННГУ, 2005.
22. *Disch S., Scholl C.* Combinational equivalence checking using incremental SAT solving, output ordering, and resets // ASP-DAC 2007. 938–943.
23. MIPLIB — Mixed Integer Problem Library (http://miplib.zib.de).
24. Суперкомпьютерный центр ИДСТУ СО РАН (http://www.mvs.icc.ru).

Поступила в редакцию
06.04.2011