Параллельные алгоритмы решения SAT в применении к оптимизационным задачам с булевыми ограничениями*

О.С. Заикин, И.В. Отпущенников, А.А. Семенов

Институт динамики систем и теории управления СО РАН

Предложена параллельная технология, применимая к целому ряду задач дискретной оптимизации, и использующая концепцию крупноблочного параллелизма. Технология основана на эффективных процедурах сведения задач комбинаторной оптимизации к SAT-задачам. Процесс решения исходной оптимизационной задачи реализован в виде итерационной схемы, каждый этап которой – это решение некоторой SAT-задачи. Получаемые SAT-задачи решаются при помощи крупноблочных параллельных алгоритмов. Для учета информации, накопленной в предыдущих итерациях, реализована техника «Incremental SAT», применяемая в задачах верификации многих дискретных систем. Разработанная технология была протестирована на решении задач 0-1-ЦЛП в распределенных вычислительных средах.

1. Введение

Как известно, многие NP-трудные задачи возникли из совершенно конкретных практических постановок. В ряде направлений без умения решать данные задачи невозможно обойтись. Речь идет о проблемах синтеза и верификации дискретных управляющих/управляемых систем, проблемах экономики, производственного планирования, логистики и многих других. В этих областях вопросы построения практически эффективных алгоритмов решения соответствующих комбинаторных задач чрезвычайно актуальны. Большое число исследований посвящено построению приближенных алгоритмов решения NP-трудных проблем. Однако в некоторых приложениях, например, в задачах верификации микросхем приближенные алгоритмы (даже обладающие малой погрешностью) совершенно бесполезны. С другой стороны, необходимость находить точные решения существенно сужает класс методов – методы непрерывной математики, генетические алгоритмы, разнообразные «эволюционные эвристики» в общем случае не дают точных решений.

В настоящей работе приведены основы «пропозиционального подхода» к решению комбинаторных задач из весьма широкого класса. Данный подход предполагает нахождение точных решений и включает две составляющих. Во-первых, это алгоритмы сведения комбинаторных задач к булевым уравнениям, и, во-вторых, это символьные алгоритмы поиска решений получаемых уравнений. В последние годы интерес именно к такому рассмотрению 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.

Библиография по решению булевых уравнений весьма обширна – от фундаментальной монографии С. Рудяну [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 сведению комбинаторных проблем к булевым уравнениям, сравнительно мало (можно сослаться на обзорную статью [2] и список литературы к ней). Удивительно то, 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]. Следует отметить, что реализовать на основе перечисленных результатов кон-

* Работа выполнена при поддержке гранта «Лаврентьевский конкурс молодежных проектов СО РАН» 2010-2011.

кретные процедуры сведения, применимые к достаточно широким классам комбинаторных задач, крайне затруднительно: сведения, описанные в [2], 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.

В работе [5] описаны механизмы пропозиционального кодирования программ, вычисляющих дискретные функции на машинах с неограниченными регистрами (МНР). Язык данной модели очень естествен – фактически это аналог 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, архитектура которого и функциональные возможности в применении к решению задач обращения дискретных функций описаны в следующем пункте.

Комплекс Transalg применяется для сведения различных дискретных задач к SAT-задачам, в отношении которых возможно использование различных техник распараллеливания [6], [7]. В настоящей работе Transalg используется для сведения к SAT-задачам задач 0-1-целочисленного линейного программирования (0-1-ЦЛП).

В заключительной части работы описаны общие принципы распараллеливания SAT-задач, кодирующих задачи дискретной оптимизации. Здесь же приведены результаты программной реализации соответствующей технологии и результаты численных экспериментов на задачах 0-1-ЦЛП.

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 С-подобном языке (ТА-язык), после чего происходит трансляция полученной ТА-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)). Также предусмотрена возможность построения И-НЕ-графа [8], представляющего рассматриваемый алгоритм.

Схематично процесс трансляции ТА-программ представлен на **Рис. 1**.

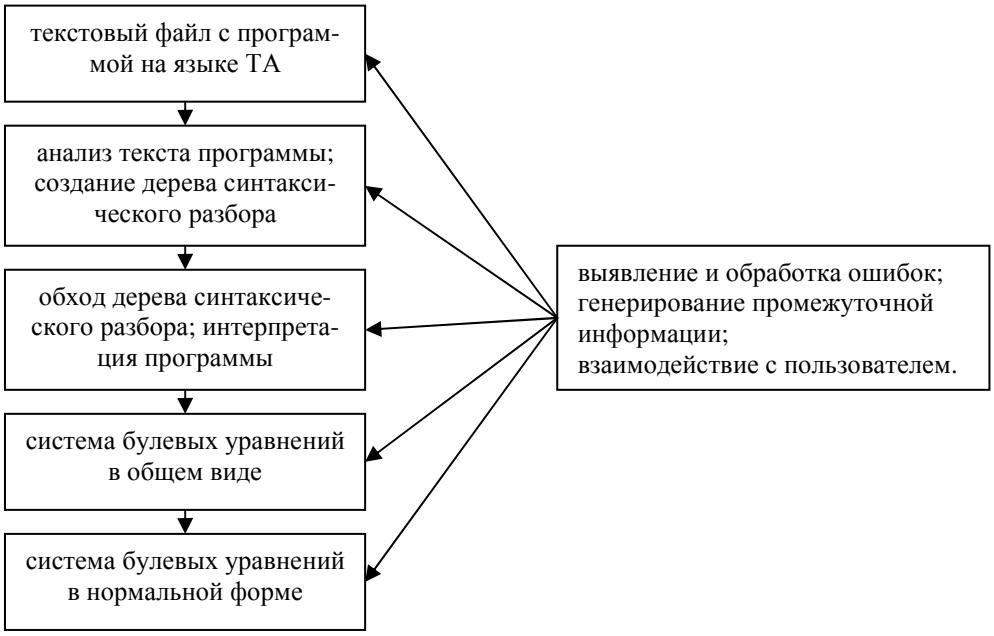

Рис. 1. Общая схема работы программного комплекса Transalg

Фазы анализа текста ТА-программы, построения дерева синтаксического разбора и обход полученного дерева с целью интерпретации реализованы стандартным способом (см., например [9]).

Язык ТА представляет собой процедурный язык программирования с блочной структурой и С-подобным синтаксисом. Каждый блок – это конечный список инструкций ТА-программы. Программа на языке ТА представляет собой набор определений функций, а также объявлений и определений глобальных переменных и констант. В языке ТА 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.

В языке ТА поддерживается два основных типа данных. Переменные целочисленных типов хранят параметры транслируемой программы, не зависящие от данных, подаваемых ей на вход. Например, это могут быть длины входного и выходного слов, количество итераций в циклах, целочисленные константы, используемые при вычислении значения дискретной функции. Тип данных bit используется для объявления булевых переменных, кодирующих входную информацию транслируемой программы, а также информацию, возникающую в процессе работы этой программы.

Особо подчеркнем, что переменные транслируемой ТА-программы и переменные пропозиционального кода этой программы представляют, вообще говоря, разные сущности. Переменные, фигурирующие в тексте транслируемой ТА-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. Тем не менее, эти два вида переменных тесно связаны. Каждой переменной программы соответствует специальная структура данных «var_object». Эта структура позволяет связывать переменные программы типа bit с переменными кода. При интерпретации инструкций, содержащих переменные программы, транслятор проверяет соответствующие структуры var_object на наличие в них связи с переменными кода.

Кроме перечисленных, в тексте ТА-программы могут встречаться переменные, необходимые для хранения результатов промежуточных вычислений. Структура var_object таких переменных не связывает их с переменными кода. Далее такие переменные называются фиктивными. Проиллюстрируем все сказанное на следующем примере.

Пример 1. Рассмотрим ТА-программу, которая реализует регистр сдвига с линейной обратной связью (РСЛОС, [10]), заданной полиномом (над GF(2)) $P(x) = x^{19} + x^{18} + x^{17} + x^{14} + 1$.

```
__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 $X^0 = \{x_1,...,x_{19}\}$.

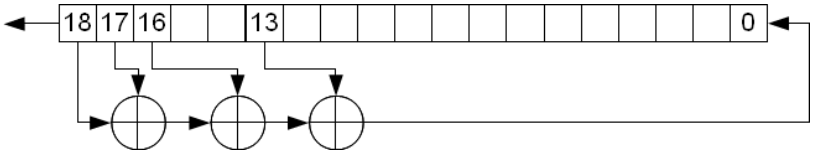

Рис. 2. Схема РСЛОС, реализуемого функцией shiftReg()

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 $X^1 = \{x_{20},...,x_{38}\}$. Новые переменные связаны с переменными из множества X^0 следующей системой булевых уравнений:

$$\begin{cases} (x_{20} \equiv x_{19} \oplus x_{18} \oplus x_{17} \oplus x_{14}) = 1 \\ (x_{21} \equiv x_1) = 1 \\ (x_{22} \equiv x_2) = 1 \\ ..... \\ (x_{38} \equiv x_{18}) = 1 \end{cases}$$

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. В контексте вышесказанного x и y – фиктивные переменные.

Аналогичным образом можно описывать и преобразовывать в булевы уравнения алгоритмы вычисления дискретных функций из весьма широкого класса. Далее мы описываем процесс сведения к SAT оптимизационной задачи из класса 0-1-ЦЛП. Специализированные для данной задачи процедуры трансляции в SAT были описаны, например, в работе [11]. В нашем случае для трансляции 0-1-ЦЛП в SAT использовался комплекс Transalg.

Рассматривается система неравенств

$$A \cdot x \le b, \qquad (1)$$

где A – $m \times n$-матрица с целочисленными компонентами, b – вектор длины m, состоящий из целых чисел. Предполагаем, что переменные $x_i, i \in \{1,...,n\}$, могут принимать значения в множестве 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 состоит в преобразовании линейных неравенств, образующих систему (1), в конъюнкции дизъюнктов. При этом можно использовать эквивалентные преобразования исходных ограничений, приводящие к ограничениям вида

$$\langle a', x^{\sigma} \rangle \le b_0, \qquad (2)$$

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

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

Сказанное означает, что от исходной задачи возможен эффективный переход к задаче минимизации линейной формы $\left\langle c', x^{\sigma} \right\rangle$ при ограничениях

$$A' \cdot x^{\sigma} \le b',$$

где A' – $m \times n$-матрица с неотрицательными целыми компонентами, b', c' – векторы, состоящие из неотрицательных целых чисел, x^{σ} – вектор литералов над X.

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

$$\Phi_f(x_1,...,x_n) = 1,$$

где Φ_f – формула (например, в КНФ), реализующая функцию $f(x_1,...,x_n)$.

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

$$\begin{cases} (x_4 \equiv x_1) = 1 \\ (x_5 \equiv x_1 \oplus \bar{x}_2) = 1 \\ (x_6 \equiv x_1 \cdot \bar{x}_2) = 1. \end{cases} \qquad (3)$$

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

Построенные таким образом системы булевых уравнений приводятся к уравнениям вида «КНФ=1» при помощи преобразований Цейтина [12]. 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.

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+ [13]. Сравнение проводилось на серии тестов из библиотеки 0-1-ЦЛП задач [14]. 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%.

На основе описанных выше механизмов трансляции задач 0-1-ЦЛП в SAT и известного SAT-решателя MiniSat2.0 был создан новый решатель псевдобулевых задач «PBSolver». На вход решателю поступает файл с задачей 0-1-ЦЛП на минимум в формате «LP». Процедуры разбора данного файла, а также процедуры трансляции ограничений задачи и целевой функции в КНФ встроены в решатель. Пусть $f(x)$ – целевая функция исходной задачи 0-1-ЦЛП. Поиск оптимального решения осуществляется итеративным вызовом SAT-решателя. На нулевой итерации SAT-решатель запускается на КНФ, которая кодирует только ограничения исходной задачи 0-1-ЦЛП. Если ответ «UNSAT», то допустимое множество пусто. В противном случае находится некоторая допустимая точка x_0 и строится первое приближение $(x_0, f(x_0))$. Затем к имеющейся системе добавляется ограничение $f(x) < f(x_0)$, и на основе полученной системы формируется новая SAT-задача. Тем самым, такой процесс решения 0-1-ЦЛП-задачи – это фактически схема последовательных приближений, итогом которой является гарантированное нахождение оптимального решения.

3. Процедуры распараллеливания задач комбинаторной оптимизации, представленных в форме SAT-задач

Одной из причин построения нового решателя псевдобулевых задач была проблема разработки параллельных технологий решения таких задач. Ниже описываются общие принципы распараллеливания задач комбинаторной оптимизации, представленных в форме SAT-задач. Исходная задача рассматривается при этом как задача минимизации некоторой функции, принимающей целые неотрицательные значения, на допустимом множестве, определяемом системой ограничений. Полагаем, что построена SAT-задача, кодирующая систему ограничений, определяющую допустимое множество. Если полученная КНФ невыполнима, то допустимое множество пусто и процесс останавливается. В противном случае решается серия SAT-задач – выполняющий набор каждой очередной SAT-задачи определяет точку в допустимом множестве, значение целевой функции в которой меньше, чем в точке, полученной на предыдущей итерации. При этом возможны различные техники распараллеливания. Наиболее очевидная схема состоит в разбиении интервала $[0, f(x_0))$ 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.

На настоящий момент реализована другая схема распараллеливания, которая близка в идейном плане схемам, использованным при обращении дискретных функций [6], [7]. 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. Далее в терминах стандарта MPI (управляющий процесс/вычислительные процессы, см., например, [15]) описывается собственно процесс обработки списка заданий параллельным решателем PD-SAT [7].

Обрабатываемыми заданиями являются SAT-задачи из декомпозиционного семейства с дополнительными ограничениями, определяющими текущее рекордное значение целевой функции. Все задания обрабатываются псевдобулевым решателем PBSolver, который был описан в разделе 2. Далее используется следующая классификация заданий (см. [7]): 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. Вычисления разделены на три этапа.

Этап 1. PD-SAT запущен на n процессах: процесс номер 1 управляющий, процессы с номерами $2,...,n$ – вычислительные. На управляющем процессе решается SAT-задача для КНФ C_{constr}, кодирующей только ограничения исходной 0-1-ЦЛП задачи. Если КНФ C_{constr} невыполнима, то допустимое множество пусто, вычисления прекращаются и выдается ответ «исходная 0-1-ЦЛП задача не имеет решений». Если C_{constr} выполнима, то допустимое множество

не пусто. В этом случае из выполняющего набора КНФ C_{constr} выделяется вектор x_0 – допустимая точка исходной 0-1-ЦЛП задачи и формируется начальное приближение $(x_0, f(x_0))$.

Этап 2. Управляющий процесс по входным данным формирует список заданий. Число заданий D равно ближайшей справа степени 2 от числа $(n-1)\cdot C$. Здесь C – константа, влияющая на загрузку вычислительных процессов. Данная константа определяется эмпирически. С управляющего процесса отсылаются первые $n-1$ свободных заданий из списка: i-е задание ($i \in \{1,...,n-1\}$) отсылается на вычислительный процесс с номером $i+1$ (каждое такое задание становится связанным незавершенным). Каждый вычислительный процесс приступает к обработке полученного задания.

Этап 3. После выполнения этапов 1–2 управляющий процесс переходит в состояние ожидания решений заданий с вычислительных процессов. Если на управляющий процесс приходит ответ, то соответствующее задание становится связанным завершенным. На приславший данный ответ вычислительный процесс отправляется очередное свободное задание из списка. Данное задание – это некоторая КНФ из декомпозиционного семейства и текущее рекордное значение целевой функции.

Помимо передачи заданий предусмотрена возможность передавать с управляющего процесса на вычислительные найденные рекордные значения целевой функции. Каждое новое рекордное значение отправляется на вычислительные процессы, даже если на данный момент на них не передается очередное задание. Вычислительные процессы периодически проверяют наличие сообщений с управляющего процесса с обновленными рекордными значениями. В используемые SAT-решатели были внесены изменения, позволяющие осуществлять такую проверку за счет применения асинхронных обменов. Если сообщение с обновленным рекордным значением получено и оно меньше, чем текущее значение, найденное в процессе работы псевдобулевого решателя, то решение текущей SAT-задачи прерывается и псевдобулев решатель формирует и решает 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.

Также при обработке списка заданий используется техника «Incremental SAT» [16], состоящая в том, что часть конфликтных дизъюнктов, накопленных SAT-решателем при обработке предыдущего задания, конъюнктивно приписывается к КНФ, которая является последующим заданием.

На данный момент объем численных экспериментов невелик. Проведенные эксперименты использовали примеры из библиотек тестов [17, 18]. На ряде примеров при распараллеливании было получено ускорение, близкое к линейному, однако эффекта сверхлинейного ускорения, в отличие от задач обращения некоторых дискретных функций [19], ни на одном тесте добиться не удалось.

Заключение

В статье предложена технология распараллеливания, применимая к обширному классу задач дискретной оптимизации. Предварительно 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 пункт работы. 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 в возможности 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 задача 0-1-целочисленного линейного программирования.

Литература

1. Rudeanu S. Boolean functions and equations, Amsterdam-London: North-Holland Publishing Company, 1974. 442 p.
2. Prestwich S. CNF encodings. In Handbook of Satisfiability (editors: A. Biere, M.Heule, H. van Maaren, T. Walsh). IOS Press, 2009. pp. 75-97.
3. Cook S.A. The complexity of theorem-proving procedures // Proc. 3rd Ann. ACM Symp. on Theory of Computing (STOC 71). ACM. 1971. pp. 151-159.
4. M.R. Garey and S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman, 1979, 340 p.
5. Семенов А.А. Трансляция алгоритмов вычисления дискретных функций в выражения пропозициональной логики // Прикладные алгоритмы в дискретном анализе. Серия: Дискретный анализ и информатика, Вып. 2. 2008. Иркутск: Изд-во ИГУ. С. 70-98.
6. Заикин О.С., Семенов А.А. Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. №1. С. 43–50.
7. Заикин О.С. Реализация процедур прогнозирования трудоемкости параллельного решения SAT-задач // Вестник УГАТУ. 2010. Т. 14, №4(39) С. 210-220.
8. Формат AIG.
   URL: http://fmv.jku.at/aiger/ (дата обращения: 11.12.2010).
9. Ахо А., Сети Р., Ульман Дж. Компиляторы. Принципы, технологии, инструменты. – М.: СПб, Киев, «Вильямс», 2001. 768 с.
10. Menezes A., Oorschot P., Vanstone S. Handbook of Applied Cryptography. CRC Press. 1996. 657 p.
11. Een N., Sorensson N. Translating Pseudo-Boolean Constraints into SAT // Journal on Satisfiability, Boolean Modeling and Computation. 2006. Vol. 2. pp. 1-25.
12. Цейтин Г.С. О сложности вывода в исчислении высказываний // Записки научных семинаров ЛОМИ АН СССР. 1968. Т.8. С. 234-259.
13. MiniSat+ solver.
    URL: http://minisat.se/MiniSat.html (дата обращения: 11.12.2010).
14. Beasley J.E. Or-library: distributing test problems by electronic mail // J. Oper. Res. Soc. 1990. Vol. 41(11). pp. 1069–1072.
15. Гришагин В.А., Свистунов А.Н. Параллельное программирование на основе MPI. Учебное пособие. – Нижний Новгород: издательство ННГУ им. Н.И. Лобачевского, 2005.
16. Stefan Disch, Christoph Scholl: Combinational Equivalence Checking Using Incremental SAT Solving, Output Ordering, and Resets. ASP-DAC 2007. pp. 938-943.
17. Pseudo-Boolean Competition 2010.
    http://www.cril.univ-artois.fr/PB10/ (дата обращения: 11.12.2010).
18. MIPLIB - Mixed Integer Problem Library.
    URL: http://miplib.zib.de (дата обращения: 11.12.2010).
19. Семенов А.А., Заикин О.С. Неполные алгоритмы в крупноблочном параллелизме комбинаторных задач // Вычислительные методы и программирование. 2008. Т. 9. №1. С. 112–122.