ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ АВТОМАТИЗАЦИИ ИССЛЕДОВАНИЙ КОМБИНАТОРНЫХ АЛГОРИТМОВ ЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ

В.И. Романов

Объединенный институт проблем информатики Национальной Академии Наук Беларуси
г. Минск, Республика Беларусь
телефон: + (375 17) 2842182; факс: + (375 17) 2842175; e-mail: rom1954@tut.by
web: www.uiip.bas-net.by

Описывается программный комплекс для решения комбинаторных задач логического распознавания и искусственного интеллекта «LOGIC», служащий основой для проведения разработок, ориентированных на автоматизацию исследований комбинаторных алгоритмов логического проектирования. Показываются основные направления таких разработок, связанные с развитием языка управления вычислениями и механизмами проведения серийных экспериментов решения задач.

Ключевые слова – анализ параметрически настраиваемых алгоритмов, решение логико-комбинаторных задач, управление процессом вычислений.

1 ПРОГРАММНЫЙ КОМПЛЕКС «LOGIC»

В области дискретной математики существует достаточно широкий класс комбинаторных задач. Такие задачи часто возникают при проектировании цифровых технических систем и в технической диагностике электронных схем. Для этих областей применения характерной особенностью является массовое использование логических данных, представленных булевыми векторами и матрицами, а также их производными, такими как троичные матрицы, миноры и т.п. Подобные задачи далее будем называть логико-комбинаторными. Качество проектов, полученных при помощи промышленных САПР, существенным образом зависит от качества и эффективности программной реализации алгоритмов решения логико-комбинаторных задач.

В настоящее время существуют специализированные системы выполнения математических расчетов, подобные MathCAD или MatLab [1]. Однако область дискретной математики в известных системах представлена недостаточно полно как по набору решаемых задач, так и по составу методов их решения. Такое состояние дел инициировало разработку программного комплекса для интерактивного решения логико-комбинаторных задач «LOGIC» [2].

Архитектура данного программного комплекса базируется на типовом для подобных систем компонентном наборе, включающем:

- язык проведения вычислений;
- среду вычислений;
- библиотеку математических функций;
- пользовательский интерфейс.

Набор основных целей разработки комплекса «LOGIC» также типичен для систем проведения математических расчетов и включает:

- проведение математических вычислений;
- создание новых алгоритмов;
- моделирование описываемых процессов, объектов, явлений;
- анализ данных, их исследование и визуализация.

В программном комплексе "LOGIC" обеспечивается решение задач из следующих разделов:

- булевы вычисления;
- булевы функции;
- логические уравнения;
- графы;
- логическое распознавание.

Задачи раздела «Булевы вычисления» классифицируются на основании обслуживаемых объектов:

- булевы объекты (булевы вектора, строки и колонки булевых матриц, булевы матрицы целиком и их миноры);
- троичные объекты (троичные вектора, строки троичных матриц, троичные матрицы целиком);
- прочие задачи.

Задачи раздела «Булевы функции» [3] охватывают разнообразные комбинаторные задачи, важные при выполнении логического проектирования, и классифицируются по обрабатываемым объектам:

- преобразования полностью определенной булевой функции;
- преобразования частично определенной булевой функции;
- преобразования системы полностью определенных булевых функций;
- преобразования системы частично определенных булевых функций.

Перечисленные преобразования включают в себя в первую очередь различные задачи минимизации и упрощения указанных объектов.

Задачи раздела «Логические уравнения» позволяют находить решения систем как линейных, так и нелинейных логических уравнений.

В состав раздела «Графы» в настоящей версии системы включены задачи проведения раскраски графов, задача поиска максимального разреза графа и задача нахожде-

ния покрытия графа максимальными разрезами. Решение этих задач осуществляется на основании оригинальных алгоритмов.

Раздел «Логическое распознавание» содержит в себе ряд задач [4], связанных с программной реализацией метода логического распознавания, предложенного А.Д. Закревским [5].

2 ТЕХНОЛОГИЯ РЕШЕНИЯ ЗАДАЧ

Архитектурно программный комплекс «LOGIC» можно рассматривать как совокупность ядра, содержащего функции, которые предназначены для решения определяемых задач, и оболочки, при помощи которой обеспечивается пользовательский интерфейс.

Оболочка системы представляет инструментальные и сервисные средства для наблюдения за ходом вычислений и управления им:

- настройка на конкретный раздел вычислений;
- формирование среды, связанной с решением конкретной логико-комбинаторной задачи;
- анализ и интерпретация L-программы проекта;
- обслуживание данных;
- настройка выполняющих вычисления программ ядра при помощи значений введенных параметров управления;
- настройка параметрически определяемых элементов пользовательского интерфейса;
- выдача справок о математическом обеспечении системы и о наборе типовых задач выбранного раздела.

Типовыми называются задачи, для решения которых пользователю достаточно осуществить их выбор в меню.

При разработке программного комплекса осуществлялся отбор лучших методов решения типовых задач. При наличии конкурентноспособных методов в меню встраивались позиции, позволяющие уточнить у пользователя выбор конкретной альтернативы.

Каждый раздел в LOGIC определяется совокупностью:

- набора типовых задач;
- множеством пользовательских объектов, в терминах которых осуществляется формулировка решаемых задач;
- библиотекой стандартных функций, предназначенных для программной реализации типовых задач;
- набором дополнительных параметров, определяющих режимы выполнения стандартных функций.

Управление процессом решения задачи в LOGIC осуществляется путем интерпретации его описания в виде программы на специальном, крайне простом, L-языке. Такой подход позволяет решать как типовые задачи, так и произвольные, определяемые путем задания соответствующей программы вычислений.

Реализуемые в LOGIC основные технологические цепочки отличаются для случаев решения типовой и произвольной задач.

Технология решения типовой задачи предполагает следующую последовательность этапов:

1). Выбор решаемой задачи в меню.

При реализации команды на решение типовой задачи LOGIC сформирует набор объектов, участвующих в ее решении, и текст L-программы, выполняющей это решение.

2). Построение содержимого объектов, выступающих в роли исходных данных задачи.

В LOGIC данные могут быть определены путем генерирования псевдослучайных значений, путем импорта из заранее подготовленных файлов или путем ручного редактирования.

3). Выполнение L-программы, решающей задачу, по команде "Run".

По завершению выполнения программы в одном из полей на панели состояния будет показано время выполнения программы. Результаты решения можно наблюдать на панели данных, выбрав интересующий объект из списка на панели объектов. Пример окна, полученного после решения одной из задач минимизации системы полностью определенных булевых функций, представлен на рис.1.

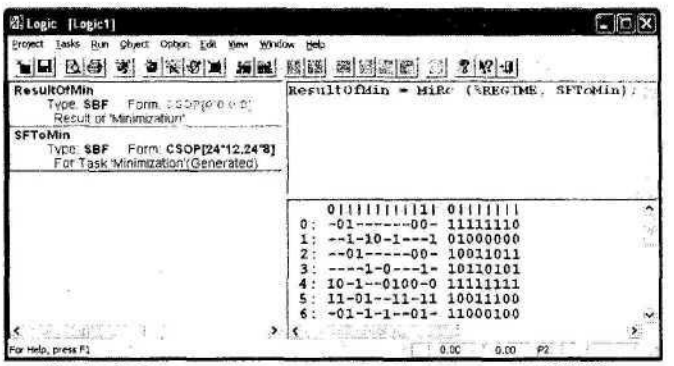

Рис.1. Рабочее окно программного комплекса «LOGIC».

Технология решения произвольной задачи предполагает следующую последовательность этапов:

1). Определение и создание множества объектов, участвующих в решении рассматриваемой задачи.

Для создания объекта используется команда "Create" из меню "Object".

2). Построение содержимого объектов, выступающих в роли исходных данных задачи.

3). Формирование текста программы на L-языке.

Возможности редактирования текста на панели программы обеспечивают удобные средства определения требуемой цепочки операторов.

4). Выполнение L-программы, решающей задачу, по команде "Run".

При желании можно указать режим пошагового выполнения, когда после очередного выполненного оператора процесс приостанавливается и могут быть выполнены любые команды работы с объектами или проектом. Этот режим особенно удобен для проведения отладки и анализа разрабатываемых программ. По завершению выполнения программы в одном из полей панели состояния

будет показано время выполнения программы. В соседнем поле демонстрируется время выполнения очередного оператора. Результаты решения можно наблюдать на панели данных, выбрав интересующий объект в списке на панели объектов.

3 РАЗВИТИЕ ЯЗЫКА УПРАВЛЕНИЯ ПРОЦЕССОМ ВЫЧИСЛЕНИЙ

Анализ возможностей, предоставляемых программным комплексом «LOGIC», показал, что не все заявленные цели его разработки обеспечены в равной степени. В частности, реализованная версия L-языка, обеспечивающая построение только линейных программ вычислений, недостаточна с точки зрения поддержки процессов разработки новых комбинаторных алгоритмов.

Другим недостатком комплекса является «неравномерность» заполнения задачами по разным разделам вычислений. Это в первую очередь связано с ограниченными ресурсами разработчиков комплекса. Именно поэтому его дальнейшее развитие предлагается проводить только в области логического проектирования, набор типовых задач и объектов которой строится как объединение объектов и задач разделов «булевы вычисления» и «булевы функции» программного комплекса «LOGIC».

Разработку новых алгоритмов решения задач логического проектирования целесообразно проводить с использованием следующей технологии:

1). Определить место задачи, алгоритм решения которой разрабатывается, в общей иерархии задач комплекса.

2). Декомпозировать алгоритм решения задачи на более мелкие компоненты с целью сведения части из них к набору готовых алгоритмов, представленных в библиотеке готовых программных функций.

3). Пополнить библиотеку программных функций новыми уникальными компонентами.

4). Выполнить отладку программной реализации нового алгоритма.

Ориентируясь на представленную технологию необходимо отметить такую общую особенность разрабатываемых алгоритмов: в большинстве случаев критерием качества алгоритма выступает скорость решения задачи. В этой связи становится особенно существенно быстрое исполнение самых внутренних программных циклов, в то время как организация самого внешнего управления на эффективность конечного результата влияет слабо.

Приведенные обстоятельства служат обоснованием возможности разработки внешнего управления непосредственно в L-языке. Анализ модулей верхнеуровневого управления показал, что в состав языка должны быть добавлены средства условного перехода (ветвления) и средства организации циклического выполнения.

Для организации ветвления предлагается добавить в состав языка оператор

```
IF <условие> THEN
  <Блок действий 1>
[ELSE
  <Блок действий 2>]
```

Такого рода конструкции присутствуют практически во всех существующих языках программирования с минимальными отличиями в синтаксисе определения.

Что касается организации циклов, то в состав L-языка предлагается добавить пару операторов:

```
FOR <начальная установка параметра цикла>
TO <условие завершения>
[STEP <шаг изменения параметра>
  <Блок действий>]
```

- так определяется цикл с параметром, и

```
WHILE <условие > DO
  <Блок действий>
```

- определение цикла с предусловием.

4 АВТОМАТИЗАЦИЯ ИССЛЕДОВАНИЙ

Вместе с разработкой новых алгоритмов подобные системы проведения вычислений нуждаются в средствах автоматизации проведения экспериментов по оценке эффективности используемых комбинаторных алгоритмов.

В первую очередь это связано с тем, что трудоемкие задачи часто нельзя решить точными методами и используются приближенные алгоритмы, основанные на применении тех или иных эвристик.

Для того, чтобы дать оценку примененной эвристики, определить области ее применимости и/или предпочтения, необходимо проведение многих серий экспериментов.

Все параметры, определяющие один конкретный эксперимент, можно условно разделить на три группы. К первой группе относятся параметры, некоторым образом описывающие исходные данные задачи, подаваемые для проведения вычислений. По сути дела такие параметры чаще всего используются для определения поведения алгоритмов генерирования псевдослучайных значений создаваемых объектов.

В программном комплексе «LOGIC» широко представлены возможности параметрически управляемого генерирования различных объектов. Пример диалоговой панели, служащей для настройки параметров генерирования одного из популярных объектов - системы полностью определенных булевых функций, представлен на рис.2.

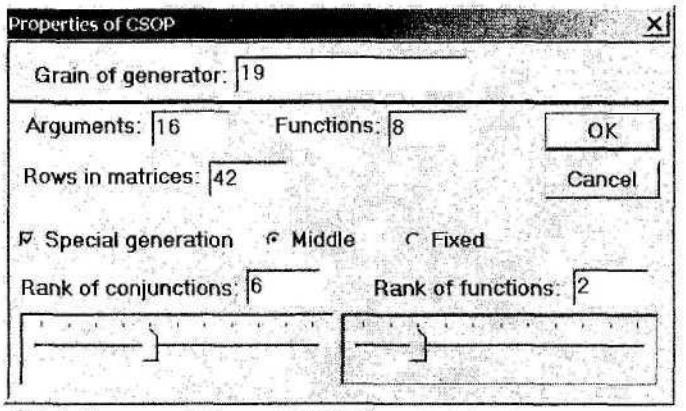

Рис.2. Пример диалога для настройки параметров генерирования системы полностью определенных булевых функций.

Вторая группа параметров служит для определения поведения собственно алгоритма решения задачи, задавая, например, допустимое отклонение решения от точного или допустимую глубину проводимого комбинаторного поиска. Так, переключатели Middle / Fixed, представленные на рис.2, определяют поведение алгоритма генерирования: будет ли указанный ранг конъюнкций равен заказанному в среднем по всем строкам (Middle) либо строго для каждой из них (Fixed).

Третья группа параметров представляет собой совокупность разнообразных оценок исследуемых алгоритмов. В ее состав, кроме безусловно необходимой оценки времени, затраченного на поиск решения, попадают различные количественные характеристики этого решения.

Рассмотрим в качестве примера задачу минимизации булевых функций многих переменных в классе ДНФ [6]. Эксперименты по исследованию соответствующего алгоритма проводились на множестве псевдослучайных булевых функций, определяемых на основе трех параметров:

g – начальное значение используемого датчика псевдослучайных чисел;

n – число переменных булевой функции;

r – плотность единиц в ее векторном представлении (размер множества единичных значений функции).

В качестве результирующих параметров-оценок для данного алгоритма использовались:

N – число единичных значений в векторе функции (число импликант в совершенной ДНФ этой функции);

C – число импликант в полученной ДНФ;

S – длина полученной ДНФ (сумма рангов импликант);

Q_k - число импликант ранга k в полученной ДНФ;

It – число выполненных итераций алгоритма;

T – затраченное время в с.

При проведении исследований данного алгоритма были зафиксированы значения параметров n и r, а изменению подвергался только параметр g. Таким образом оценивался разброс получаемых решений на потоке подобных по характеристикам исходных данных. Для данной серии можно вычислить средние значения результирующих оценок. Применяя те же правила изменения параметра g при других значениях параметров n и r в работе [6] делается вывод о допустимости использования рассматриваемого алгоритма при различных комбинациях указанных управляющих параметров – формируются наборы возможных значений характеристик обрабатываемых исходных данных.

Приведенная классификация параметров и существующие средства генерирования псевдослучайных объектов служат основой для создания типовых механизмов автоматизации серийных испытаний комбинаторных алгоритмов логического проектирования. Данный подход к организации проведения серийных экспериментов может оказаться полезным и для других классов разрабатываемых алгоритмов.

ЛИТЕРАТУРА

[1] Steinhause S. Comparison of mathematical programs for data analysis [Electronic resource]. – February 24. 2008. – Mode of access: http://www.scientificweb.de/ncrunch . –Date of access: 20.09.2008.

[2] Романов, В.И. Программный комплекс для решения логико-комбинаторных задач "LOGIC"/ В.И. Романов // Информационные технологии в промышленности (ITI*2008): тезисы докладов Пятой Междунар. научн. - техн. конф., Минск, 22-24 октября 2008 г. / ОИПИ НАН Беларуси; науч. редактор Е.В. Владимиров – Минск, 2008 - С. 198-199.

[3] Романов, В.И. Задачи раздела «булевы функции» в вычислительной системе LOGIC / В.И. Романов, Н.Р. Торопов // Проблемы проектирования и производства радиоэлектронных средств: сборник материалов V Междунар. научн. – техн. конф.: в 3-х т., Новополоцк, 29–30 мая 2008 г. / ПГУ; под общ. ред. С.В. Абламейко, М.Л. Хейфица – Новополоцк, 2008. – Т. 3 - С.64 – 67.

[4] Zakrevskij, A.D. Tasks of logical recognition in system LOGIC / A.D. Zakrevskij, V.I. Romanov // Neural Networks and Artificial Intelligence: proc. of the Fifth Intern. Conf. Minsk, 27–30 May 2008. – Minsk, 2008. – P. 260–263.

[5] Zakrevskij, A.D. A common logic approach to data mining and pattern recognition. – Triantaphyllou, E. and G. Felici (Eds.). Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques. - Massive Computing Series, Springer, Heidelberg, Germany, 2006, pp. 1-43.

[6] Закревский, А.Д. Минимизация булевых функций многих переменных в классе ДНФ – итеративный метод и программная реализация / А.Д. Закревский, Н.Р. Торопов. // Прикладная дискретная математика. – 2009, № 1(3). – С. 5-14.