УДК 621.350.6

РЕВОТЮК М.П., КИШКЕВИЧ А.П., ХАДЖИНОВА Н В.

СИНХРОНИЗАЦИЯ АГЕНТОВ КООПЕРАТИВНЫХ ВЫЧИСЛЕНИЙ

Объект рассмотрения – механизмы синхронизации объектно-ориентированной реализации шаблонов агентной системы решения вычислительно сложных задач на локальных сетях. На примере задачи коммивояжера показано, что для синхронизации работы агентов предпочтительнее оказываются разностные схемы определения локальных задач.

Привлекательная альтернатива решения эпизодически возникающих задач повышенной сложности - утилизации потенциально доступных узлов локальных сетей [1]. Доступные для кооперации ЭВМ оснащаются агентами, ожидающими получения описания задачи и ее варианта. Получив задание, агент активизирует процесс решения в фоновом режиме, продолжая следить за общей обстановкой на сети. Однако реализация жадного алгоритма обработки множества вариантов порождает задачу синхронизации агентов, особенно значимую при случайном характере подключения ЭВМ пользователями сети. Предмет обсуждения – уточнение объектной спецификации процедур анализа отдельного варианта [1] с целью построения самосинхронизируемой системы агентов.

Идея объектно-ориентированной реализации функциональной части агента пусть представлена шаблоном абстрактного класса на языке C++ [1]:

```
template <class Criterial, class Variant> class Agent {
 volatile Criterial R; // Глобальный объект рекордной оценки
 virtual Criterial FirstStep() = 0;
 virtual bool NextStep(Criterial &) = 0;
public:
 virtual void P(Variant &V) {
  Criterial F = V.FirstStep(); // Начальная оценка варианта
  if (R>F) { // Организация анализа варианта
   while (V.NextStep(F)) if (R<F) return;
   if (R>F) R = F; // Фиксация нового рекорда
  }
 }
};
```

Здесь предполагается решение задачи минимизации критериальной функции *F* с возможностью получения нижних оценок значения *F* на отдельной итерации анализа варианта *V*, а *R* – доступное всем агентам текущее значение рекордной оценки**.**

Предлагаемый способ синхронизации агентов основан на известном методе проверки и установки, применяемый к разделяемым на сети данным представления потока вариантов – значений *V* и *R*.

Условие отказа от анализа варианта из-за установления его бесперспективности должно быть представлено оператором сравнения *R* и *F*, который должен быть определен в классе Criterial.

Связь агента с внешней средой организуется через область данных *R*, которая кэширует известное на сети значение рекордной оценки. Выполняя оператор присваивания *R = F*, агент должен послать широковещательное сообщения группе кооперируемых ЭВМ. Такая посылка легко реализуема посредством применения протокола UDP и группового обмена на уровне гнездовых соединений. Базовый класс

обеспечения обмена, который наследуется классом Criterial, должен создать потоки передачи и приема, обеспечивающие асинхронный режим записи и чтения значения R. Синхронизация доступа реализуется использованием критической секции функциями записи и чтения значения R.

С целью минимизации трафика, первое сообщение активизируемого агента – $R = \infty$, должно заинтересовать лишь одну ЭВМ. Уместно предоставить приоритет ответа владельцу рекорда, если имеются просмотренные варианты.

Полиморфизм операторов присваивания предлагается применить для синхронизации и процессов выборки вариантов V. Однако при этом следует выделять различные случаи взаимозависимости вариантов, что позволяет посредством явного учета рекуррентных схем связи вариантов снизить нагрузку на сеть.

Для независимых вариантов, например, в комбинаторных задачах размещения транспортного типа или квадратичной задачи о назначениях, приходится синхронизировать начальные состояния генераторов перестановок или сочетаний [2,3]. Локальные задачи агентов полностью определены в этом случае матрицами весов, получаемыми однократно в момент получения описания задачи.

Взаимозависимость вариантов характерна для задач, решаемых методом ветвей и границ [4,5]. Древовидный характер динамически порождаемого пространства поиска часто можно и удобно представить в разностной форме. Это позволяет сократить объем памяти стека порождаемых вариантов.

Например, алгоритм Литтла решения задачи коммивояжера [4], как известно, основан на порождении и анализе бинарного дерева вариантов [5], представляемых матрицами расстояний фиксированного размера.

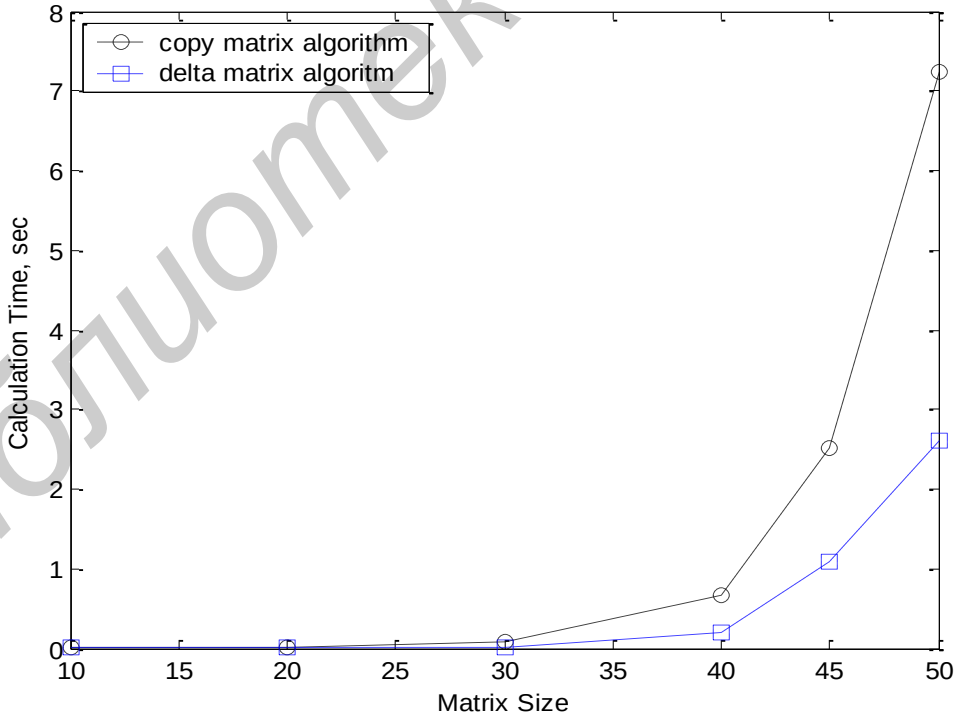

Рис. 1. Зависимость времени решения задачи коммивояжера от размерности

Изменение матрицы расстояний в алгоритме Литтла при рекурсивном переходе от текущего узла дерева вариантов к другим узлам включает:

приведение матрицы - вычитание наименьшего элемента из каждой строки и столбца;

вычеркивание строк и столбцов с целью запрета преждевременных замыканий и выбранного на данном этапе перехода для «правой» ветви дерева.

Чтобы избежать переписывания матрицы (передачи между агентами) в случае ее приведения, будем хранить текущие наименьшие элементы по строкам и столбцам. Аналогично, чтобы избежать переписывания матрицы в случае вычеркивания строк и столбцов, будем обращаться к соответствующей строке и столбцу через списки активных строк и столбцов. Измененные элементы этих списков, а также запреты на раннее замыкание и на переход в случае правого ветвления будем помещать в стек обхода дерева вариантов.

Таким образом, чтобы вернуться к рассмотрению какой-либо правой ветки, необходимо восстановить из стека только измененные элементы матрицы и списков номеров строк и столбцов.

Очевидно, что объем памяти для хранения описания изменений матрицы существенно меньше ее размера, что следует использовать для выбора алгоритма синхронизации кэшируемых отдельными агентами данных.

Результаты эксперимента по оценке эффективности реализации алгоритма Литтла рекурсивной процедурой обхода узлов дерева на основе копирования матриц и их модификации с откатом (рис.1) подтверждают преимущество разностных схем. Таким образом, при выборе структур данных представления задачи для распределенных вычислений в случае взаимозависимых вариантов следует отдать предпочтение разностным схемам с целью минимизации трафика на определение задач агентам.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ревотюк М.П., Кузнецова Н.В. Агентная система кооперации ресурсов вычислительной среды для решения задач выбора//Известия Белорусской инженерной академии, № 1(15)/1, 2003. – С. 265-268.

2. Кишкевич А.П., Ревотюк М.П. Порождение подмножеств сочетаний//Известия Белорусской инженерной академии, № 1(17)/1, 2004. – С. 78-81.

3. Ревотюк М.П., Кишкевич А.П., Дарадкех Ю.И. Генерация подмножеств перестановок вариантов//Дистанционное обучение – образовательная среда XXI века: Материалы IV Международной научно-методической конференции (Минск, 10-12 ноября 2004 г.) – Мн.: БГУИР, 2004. – С.276-279.

4. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ/Пер. с англ. – М.: МЦМНО, 2002. – 960 с.

5. Седжвик Р. Фундаментальные алгоритмы на C++. Части 1-4. Анализ. Структуры данных. Сортировка. Поиск – Киев: ДиаСофт, 2002. – 687 с.

Ревотюк Михаил Павлович
Профессор кафедры информационных технологий автоматизированных систем, к.т.н.
Белорусский государственный университет информатики и радиоэлектроники, г. Минск
Тел.: +375(17) 239-86-58
E-mail: rmp@bsuir.unibel.by
Кишкевич Андрей Павлович
Инженер ИП “Центр компетенции ”Эвклид”
Тел.: (+37529)643-88-83
E-mail: a-kishkevitch@mdtvision.com
Хаджинова Наталья Владимировна
Аспирант кафедры информационных технологий автоматизированных систем
Белорусский государственный университет информатики и радиоэлектроники, г. Минск
Тел.: +375(17) 231-49-31
E-mail: kafitas@bsuir.unibel.by