

# АГЕНТНАЯ СИСТЕМА РЕШЕНИЯ КОМБИНАТОРНЫХ ЗАДАЧ РАЗМЕЩЕНИЯ

*М.П. Ревотюк, О.Г. Терехова*

Белорусский государственный университет информатики и радиоэлектроники,  
кафедра информационных технологий автоматизированных систем  
6, П.Бровки, Минск, Беларусь  
телефон: + 375(17)293-86-58; e-mail: rmp@bsuir.by

**Обсуждается схема решения комбинаторных задач размещения производственных объектов, использующая возможности распределения вычислений на вычислительных сетях общего назначения. Предлагается учитывать взаимозависимость этапов параллельного решения локальных задач, что приводит к повышению быстродействия решения исходной задачи.**

**Ключевые слова** - задача размещения, агентные системы, параллельные вычисления.

## ПОСТАНОВКА ЗАДАЧИ

Рассмотрим одну из известных разновидностей задач размещения производственных объектов, относящихся к классу транспортных.

Пусть имеется  $n$  пунктов потребления некоторой однородной продукции, характеризуемых вектором потребления  $B = \{b_j, j = \overline{1, n}\}$ . Для удовлетворения потребностей в продукции требуется разместить  $m$  однородных объектов с объемом производства  $A = \{a_i, i = \overline{1, m}\}$  с учетом выполнения условия баланса:

$$\sum_{i=1}^m a_i = \sum_{j=1}^n b_j.$$

Для простоты предположим, что производственные объекты могут быть размещены в любом из пунктов, пронумерованных от 1 до  $k$ , а транспортные затраты характеризуются матрицей  $C = [c_{ij}, i = \overline{1, k}, j = \overline{1, n}]$ .

Обозначим  $R$  – множество мест размещения объектов производства,  $|R|=k$ , а  $r$  – подмножество, представляющее конкретный вариант,  $|r|=m$ .

При фиксации варианта  $r \subset R$  величина транспортных расходов на перевозку продукции составит

$$C(r) = \sum_{i \in r} \sum_{j=1}^n x_{ij} \cdot c_{ij}, \quad (1)$$

где  $x_{ij}$  – объем продукции, поставляемой из пункта  $i$  в пункт  $j$ .

Полагается, что объемы поставки и потребления при этом для каждого отдельного варианта сбалансиированы:

$$\begin{cases} \sum_{i \in r} x_{ij} = b_j, j = \overline{1, n}; \\ \sum_{j=1}^n x_{ij} = a_i, i = \overline{1, m}. \end{cases} \quad (2)$$

Задача минимизации  $C(r)$  при последних условиях – классическая транспортная задача. Цель решения исходной задачи размещения – найти

$$r^* = \arg \min_{r \subset R} C(r), \quad (3)$$

что означает экспоненциально сложный выбор сочетания  $m$  номеров из  $k$ .

Возможен подход к решению такой задачи в рамках целочисленного линейного программирования. Постановка задачи существенно определяется схемой порождения подмножеств  $r \subset R$ . Если описание такой схемы связано с трудно формализуемыми правилами, то метод перебора остается вполне приемлемым [1].

Рассматриваемая задача относится к задачам выбора из конечного множества вариантов. Конкретную задачу выбора  $Z$  в рамках объектно-ориентированного подхода можно характеризовать тройкой

$$Z = (V, P, S), \quad (4)$$

где  $V$  – множество вариантов, подлежащих оценке по заданному критерию;  $P$  – процедура получения оценки качества для отдельного варианта из множества  $V$ ;  $S$  – процедура реализации вычислительной схемы решения  $Z$ , определяющая порядок применения  $P$  к элементам множества  $V$ .

Структуризация рассматриваемой задачи в форме (4) очевидна. Вариант  $v$  представлен классической транспортной задачей, а множество вариантов  $V$  определяется набором всех сочетаний из  $k$  элементов множества  $R$  по  $m$  элементов. Процедура  $P$  оценки отдельного варианта – одна из хорошо известных процедур решения транспортной задачи линейного программирования. Процедура  $S$  порождения вариантов может представлять перебор всех допустимых комбинаций образования сочетаний.

## СХЕМА РЕШЕНИЯ ЗАДАЧИ

Анализ структуры задачи выбора показывает, что их решение возможно в рамках распределенной системы на вычислительной сети, где основным активным элементом выступает агент, выявляющий доступность и готовность к решению задачи узла сети. Известно, что “жадный” алгоритм распределения загрузки узлов сети является оптимальным по критерию минимума времени решения задачи  $Z$ .

Предположим, что доступные для кооперации ЭВМ оснащены агентами, ожидающими получения описания локальной задачи [1]. Получив задание, агент активизирует процесс ее решения, продолжая следить за общей обстановкой на сети. Слежение заключается в проверке условия бесперспективности анализируемого варианта посредством сравнения оценки его целевой функции с рекордным значением.

Агент решения задачи должен выполнять посылку широковещательного сообщения группе кооперируемых ЭВМ сети. Такая посылка легко реализуема посредством применения протокола UDP и группового обмена на уровне гнездовых соединений [1].

Идея объектно-ориентированного представления функциональной части агента в форме шаблона класса на языке C++, согласно [1], имеет вид:

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

Здесь предполагается решение задачи минимизации целевой функции  $F$ . Кроме того, процедура  $P$  обладает возможностью получения в процессе работы нижних оценок значения функции  $F$ , а  $R$  – текущее значение рекордной оценки, является общедоступным для всех агентов.

Переход к реальной программной реализации задачи требует дальнейшей конкретизации функций-элементов шаблона представленного выше класса агента.

Анализ известных алгоритмов решения транспортных задач показывает, что наиболее удобным для решения задач выбора вида (1)-(3) на сетях является алгоритм венгерского метода. Доводы в его пользу:

имеется возможность получения нижней оценки целевой функции и прерывания итераций анализа варианта при установлении его бесперспективности;

этап инициализации решения допускает исключение пересчета смежных вариантов размещения.

С целью использования последней особенности, алгоритм генерации сочетаний построен на основе метода “вращающейся двери”, характеризующегося порождением сочетаний в порядке минимального изменения [2].

## ЗАКЛЮЧЕНИЕ

Рассмотренная схема решения задач размещения использована для проектирования и реализации реальной системы решения задач выбора. Настройка производительности системы осуществляется выбором количества кооперируемых ЭВМ. Открытость системы для учета дополнительных ограничений на варианты размещения обеспечена полиморфизмом операторов в функциях выборки сочетаний. Выборка исходных данных производится посредством использования сервиса Microsoft Virtual Earth, модуль доступа к которому реализован на языке программирования JavaScript.

## ЛИТЕРАТУРА

- [1] Кузнецова Н.В. Агентная система кооперации ресурсов вычислительной среды для решения задач выбора/Ревотюк М.П., Кузнецова Н.В./Известия Белорусской инженерной академии, № 1(15)/1, 2003. – с. 265-268.
- [2] Кишкевич, А.П. Генерация подмножеств сочетаний в порядке минимального изменения/А.П. Кишкевич, М.П. Ревотюк//Дистанционное обучение – образовательная среда XXI века: Материалы VI Междунар. научно-методической конф. (Минск, 22-23 ноября 2007 г.) – Мин.: БГУИР, 2007. – С. 232-234