РАЗНОСТНАЯ СХЕМА ПРЕСТАВЛЕНИЯ СОСТОЯНИЙ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА

Кишкевич А.П., Ревотюк М.П., БГУИР, Минск

Предмет рассмотрения – реализация алгоритма решения задачи коммивояжера [1] с минимальными изменениями исходной матрицы при переходе от шага к шагу. Потребность учета изменения представления задачи возникает в случае ее распараллеливания и синхронизации агентов распределенных вычислений [2]. Цель исследования – сокращение расхода памяти и, как следствие, объема передаваемых данных между узлами вычислительной сети.

Задача коммивояжера в классической постановке формулируется следующим образом: задана матрица расстояний (стоимости переезда) $C = \left\| c_{ij}, i, j = \overline{1,n} \right\|$ между любым из *n* городов, необходимо найти цикл минимальной длины однократного посещения каждого города.

Если решение задачи коммивояжера представить матрицей булевых переменных $X = \left\| x_{ij}, i, j = \overline{1,n} \right\|$, где единица означает включение в оптимальный цикл дуги $i \to j$, то формальная модель оптимизации имеет вид:

$$
\begin{aligned}
&Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \Rightarrow \min; \\
&\sum_{i=1}^{n} x_{ij} = 1; \\
&\sum_{j=1}^{n} x_{ij} = 1; \\
&x_{ij} \geq 0, i, j = \overline{1,n}; \\
&u_i - u_j + n x_{ij} \leq n - 1, \quad i = \overline{2,n}, \quad j = \overline{2,n}, \quad i \neq j.
\end{aligned}
\tag{1}
$$

Модель (1) не является конструктивной для построения эффективной вычислительной схемы решения задачи коммивояжера. Среди точных методов ее решения одним из эффективных считается метод ветвей и границ. Схема алгоритма метода ветвей и границ может быть реализована разными способами, различающимися правилами порождения ветвей дерева вариантов. Наиболее успешным считается подход, базирующийся на решении задач о назначении, анализе получающихся замкнутых циклов и, если таких циклов более одного, последующем переборе вариантов разрыва циклов. Рекурсия обхода дерева строится на матрице расстояний, где разрывы циклов задаются назначением бесконечных значений длин запрещаемых дуг.

Однако далее, для простоты оценок предлагаемых решений, остановимся на классическом варианте алгоритма Литтла [1], основанного на порождении бинарного дерева. В некоторых случаях он не теряет привлекательности [3].

Изменение матрицы расстояний в алгоритме Литтла при рекурсивном переходе от текущего узла дерева вариантов к другим узлам включает [1]:

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

вычеркивание строки и столбца для выбранного на данном этапе перехода и запрет преждевременных замыканий частичного пути.

Чтобы избежать переписывания матрицы (передачи между агентами) в случае ее приведения, предлагается 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.

На этапе возврата к рассмотрению какой-либо правой ветки дерева, необходимо восстановить из стека только измененные элементы матрицы и списков номеров строк и столбцов. Оценка трудоемкости восстановления – $O(n)$.

Достаточно просто оценить объем памяти для рассмотренной схемы. При размерности задачи *n* максимально возможная глубина стека не превосходит значения *n.* Необходимое количество элементов памяти составит:

$$M_{\Delta C} = n^2 + 2n * (n+3), \qquad (2)$$

где первое слагаемое представляет исходную матрицу *C*, а второе – изменения в векторах текущих минимумов, решении и матрице расстояний на каждом шаге, соответственно (множитель 2 появляется из-за необходимости хранения адреса измененного элемента). Отметим, что выражение (2) представляет оценку сверху, а в реальных случаях объем памяти будет практически ниже.

Несложно показать, что объем памяти, необходимой для прямолинейной реализации классического алгоритма Литтла [1]:

$$M_C = n * \left(n^2 + 2n\right). \qquad (3)$$

Сравнивая (2) и (3), легко видеть, что объем памяти в предложенном варианте на один порядок меньше зависит размерности задачи.

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

Результаты эксперимента по оценке эффективности реализации алгоритма Литтла рекурсивной процедурой обхода узлов дерева на основе копирования матриц и их модификации с откатом подтверждают преимущество разностных схем. Потери времени на восстановление состояния узла дерева решений оказались незначительными. Таким образом, при выборе структур данных представления задачи для распределенных вычислений в случае взаимозависимых вариантов [2] следует отдать предпочтение разностным схемам с целью минимизации трафика на определение задач агентам.

Литература

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

2. 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.

3. Gutin G., Yeo A. Assignment problem based algorithms are impractical for the generalized TSP. Australasian J. Combinatorics, vol. 27, 2003. – pp. 149-154.