УДК 656.2–50: 519.8

РЕОПТИМИЗАЦИЯ РЕШЕНИЯ ЗАДАЧ О НАЗНАЧЕНИИ

М.П. РЕВОТЮК, П.М. БАТУРА, А.М. ПОЛОНЕВИЧ

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 7 декабря 2010

Рассматривается случай регулярного решения линейных задач о назначении с незначительным изменением матриц стоимостей. Реоптимизация решения асимметричной задачи коммивояжера методом ветвей и границ с ветвлением на задачах о назначении, а также комбинаторной задачи выбора позволила снизить их вычислительную сложность на порядок.

Ключевые слова: задача о назначении, разностная схема, вычислительная сложность.

Постановка задачи

Известно, что классические линейные задачи о назначении (ЛЗН) в виде

$$\min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\cdot x_{ij}\;\middle|\;\sum_{i=1}^{n} x_{ij}=\sum_{j=1}^{n} x_{ij}=1;\; x_{ij}\ge 0;\; i,j=\overline{1,n}\right\} \qquad (1)$$

характеризуются вычислительной сложностью $O(n^3)$ [1]. Случай поиска максимума сводится к (1), если произвести замену $c_{ij}\leftarrow c^{*}-c_{ij},\; i,j=\overline{1,n},\; c^{*}=\max\left\{c_{ij},\; i,j=\overline{1,n}\right\}$.

Достаточно часто требуется пересчет задачи (1) после изменения исходных данных. Например, при решении асимметричной задачи коммивояжера методом ветвей и границ, когда $c_{ij}\ne c_{ji},\; i,j=\overline{1,n},$ эффективный способ ветвления – порождение вариантов решения ЛЗН [2]. Варианты отличаются лишь изменением некоторых элементов строки матрицы. Прямолинейный пересчет очередного варианта задачи потребует $O(n^3)$ операций. Однако итерация расчета для любой отдельной строки имеет вычислительную сложность $O(n^2)$ [1, 2], что побуждает использовать наследование результатов предшествующего решения. Цель работы – построение алгоритмов получения решения новой задачи на основе наследуемых результатов.

Представляющие практический интерес задачи с прямоугольной матрицей

$$\min\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\cdot x_{ij}\;\middle|\;\sum_{i=1}^{m} x_{ij}=\sum_{j=1}^{n} x_{ij}=1;\; x_{ij}\ge 0;\; i=\overline{1,m},\; j=\overline{1,n}\right\} \qquad (2)$$

могут рассматриваться как задачи вида (1) после формального дополнения матрицы строками или столбцами с нулевыми элементами. Очевидно, что сканирование нулевых строк или столбцов будет бесполезным, но необходимым этапом реализации алгоритма венгерского метода. Тем не менее, далее ЛЗН будут рассмотрены в постановке (2), когда $m\le n$. Холостые итерации сканирования нулевых строк будут исключены. Случай $m>n$ легко сводится к решению (2) после транспонирования матрицы и соответствующей замены содержательной интерпретации индексов строк и столбцов.

Алгоритм решения задачи

Наиболее эффективные для решения задачи (1) алгоритмы венгерского метода используют особенности двойственной задачи

$$\max\left\{\sum_{i=1}^{m}u_i+\sum_{j=1}^{n}v_j \;\middle|\; c_{ij}-u_i-v_j\ge 0,\;\; i=\overline{1,m},\; j=\overline{1,n}\right\}. \qquad (3)$$

Здесь неизвестными являются потенциалы строк и столбцов. Значения потенциалов особого интереса не представляют, но определяют решение задачи (2). Отображение решения (3) будем осуществлять на упорядоченный вариант вектора назначений строк столбцам

$$R=\left\{r_j=i \,\middle|\, c_{ij}-u_i-v_j=0,\, i=\overline{1,m},\; j=\overline{1,n}\right\}.$$

Схемы известных версий алгоритмов венгерского метода совпадают, включая быстрый этап инициализации для формирования начального назначения строк и итерационного дополнения решения для оставшихся строк. Вычислительная сложность этапа инициализации – $O(n^2)$. На этом этапе пытаются выполнить назначение строк, используя операцию приведения матрицы задачи. Приведение состоит в вычитании из элементов столбцов минимальных элементов столбцов. Однако этап инициализации можно исключить, совмещая этап начального назначения строк с этапом последовательного поиска решения для всех строк. Такой прием станет ключевым для построения алгоритма реоптимизации.

Действительно, пусть M и N – множества строк и столбцов матрицы ЛЗН (1), а M^* и N^* – множества строк и столбцов текущего назначения, для которых в результате операции приведения выделены элементы с нулевой невязкой: $c_{ij}-u_i-v_j=0,\, i\in M^*,\; j\in N^*$. Если $s=|M^*|<m$ и $t=|N^*|<n$, то, согласно теореме Кенига-Егервари, $s+t<m$.

Определим приращение потенциалов $h=\min\{(c_{ij}-u_i-v_j),\; i\notin M^*, j\notin N^*\}$, а новые значения потенциалов пусть будут $u_i^*=u_i+h\cdot(i\notin M^*)$ и $v_j^*=v_j-h\cdot(j\in N^*)$. Полученные значения допустимы для задачи (3), так как выполняются условия:

1) если $i\in M^*$, то $u_i^*+v_j^*\le u_i+v_j\le c_{ij}$;

2) если $i\notin M^*$, $j\in N^*$, то $u_i^*+v_j^*=u_i+v_j\le c_{ij}$;

3) если $i\notin M^*$, $j\notin N^*$, то $u_i^*+v_j^*=u_i+v_j+h\le c_{ij}$.

Такое новое назначение приведет к увеличению целевой функции задачи (3):

$$\sum_i u_i^*+\sum_j v_j^*=\sum_i u_i+(m-s)h+\sum_j v_j-th=\sum_i u_i+\sum_j v_j+h(m-s-t).$$

Именно это позволят увеличить количество назначенных элементов матрицы, так как $s+t<m$. По индукции, последнее справедливо для любых значений $s=\overline{1,m}$, что подтверждает возможность отказа от этапа инициализации.

Проведенные рассуждения не накладывают ограничений на способ расширения множеств M^* и N^*. Отсюда следует, что последовательное добавление строк, начиная с пустого множества, – допустимый вариант алгоритма решения задачи.

Анализ известных реализаций алгоритма венгерского метода показывает, что итерации расширения множеств M^* и N^* чаще выполняются именно по строкам. Различные версии алгоритма отличаются при этом лишь эвристиками начального назначения потенциалов.

Например, пусть $u_i=0,\, i=\overline{1,m},\;\; v_j=0,\;\; j=\overline{1,n}$. Тогда возможный компактный вариант программы решения ЛЗН представлен на рис. 1. Здесь на каждой итерации вначале уточняются значения потенциалов столбцов, а затем назначается и корректируется потенциал добавляемой строки. Коррекция требуется в случае возникновения конфликтов с ранее просмотренными строками.

```
template <class cost> class LAP {
 cost lap(); // решение задачи о назначении
 void lap(int i); // итерация назначения строки
 cost *u, *v; // потенциалы строк и столбцов
 int  *next; // ссылка на элемент дерева
 cost *slack; // вектор невязок
 int  w; // номер итерации
 int  *color; // вектор счетчика итераций
protected:
 int m, n; // размерность задачи
 int *r; // вектор текущего решения
 cost *c; // матрица стоимости
 operator cost() { // оценка целевой функции задачи
  cost s = 0;
  for (int j=0; j<n; j++) if (r[j]<n) s+=c[r[j]*n+j];
  return s;
 }
 void col(int j); // уточнение потенциала столбца
 void set(int i, int j, cost x); // изменение элемента матрицы задачи
public:
 LAP(int M, int N, cost *C);
 virtual ~LAP();
 static bool is_min(T &was, const T itm) {
  if (was<=itm) return false;
  was=itm;
  return true;
 }
};

template <class cost>
void LAP<cost>::lap(int i) { // итерация назначения строки
 int j,k,I=i,J;
 w++;
 for (j=0; j<n; j++) slack[j]=inf;
 do {
  cost h = inf, *b = c + I*n, U=u[I];
  for (J=j, k=0; k<n; k++)if (color[k]!=w) {
   if (is_min(slack[k],b[k]-U-v[k])) next[k]=J;
   if (is_min(h,slack[k])) j=k;
  }
  if (h) {
   for (k=0; k<n; k++) if (color[k]==w) {
                         u[r[k]]+=h; v[k]-=h;
                       } else slack[k]-=h;
   u[i]+=h;
  }
  color[j]=w; I=r[j];
 } while (I<n);
 for (; (k=next[j])<n; j=k) r[j]=r[k];
 r[j]=i;
}

template <class cost>
 void LAP<cost>::lap() { // решение задачи о назначении
 for (int i=0; i<n; i++) lap(i);
}
```

Рис. 1. Шаблон класса с основными функциями решения задач о назначении

Примечательно, что такой алгоритм при обработке строки $i, i < m$, не учитывает информацию о последующих строках. Этим объясняется большее количество шагов коррекции потенциалов просмотренных строк по сравнению с лучшими реализациями алгоритма венгерского метода, выполняющих построение кратчайшего аугментального пути (Shortest Augmenting Path, SAP) [1]. Однако реализация алгоритма венгерского метода, пригодная для реоптимизации, должна опираться на сохраняемые значения потенциалов строк и столбцов. Алгоритмы метода SAP такие значения в явном виде не используют.

Очевидна схема решения ЛЗН с добавлением новых строк. В [4] подобная схема предложена для решения ЛЗН с расширяемой матрицей задачи (1). Этап расширения матрицы размером $n \times n$ решенной задачи начинается с предварительной оценки потенциалов:

$$v_{n+1} = \min(\min_i(c_{i,n+1} - u_i, i = \overline{1,n}),\ c_{n+1,n+1}),\ u_{n+1} = \min_j(c_{n+1,j} - v_j,\ j = \overline{1,n}).$$

Далее выполняется итерация назначения алгоритма венгерского метода для строки $n+1$ расширенной матрицы. Однако в случае реоптимизации решения изменяться могут произвольные элементы матрицы задачи с сохранением ее размера. Очевидно, что можно выделить строки, в которых изменены элементы, и повторить итерации назначения для таких строк.

Можно заметить, что изменение элементов матрицы влечет необходимость пересмотра результата оптимизации, если только меняется позиция нулевого элемента после операции приведения. Действительно, увеличение элемента $c_{ij} \leftarrow c_{ij}^*$ матрицы задачи (2), когда $x_{ij} = 0$, не меняет решения для любых $i \in \overline{1,m}$, $j \in \overline{1,n}$. Аналогично, уменьшение элемента $c_{ij} \leftarrow c_{ij}^*$, когда $x_{ij} = 1$, не нарушает соотношения $c_{ij} = u_i + v_j$ в задаче (2).

В других случаях, как несложно показать, требуется повтор итераций назначения лишь для строк

$$M^* = \left\{ i \middle| (c_{ij}^* > c_{ij}) \wedge (x_{ij} = 0) \vee (c_{ij}^* < c_{ij}) \wedge (x_{ij} = 1), j = \overline{1,n} \right\}, \quad (4)$$

что влечет, как легко заметить, снижение вычислительной сложности пересчета задачи (1) на величину $\left(m - \left|M^*\right|\right) \cdot O(n^2)$.

Изменение элементов матрицы должно отражаться значениями потенциалов, сбросом признака назначения соответствующей дуги графа и итерацией назначения строки. Если изменены элементы строки i', то ее потенциал должен быть скорректирован:

$$u_{i'} = \min_j(c_{i'j} - v_j, j = \overline{1,n}). \quad (5)$$

В случае изменения элементов столбца j' его потенциал также должен меняться:

$$v_{j'} = \min_i(c_{ij'} - u_i, i = \overline{1,m}). \quad (6)$$

Очевидно, что выбор между строкой или столбцом измененного элемента для реоптимизации решения определяется схемой алгоритма. Если применена схема итерации по строкам (рис. 1), то необходимо получить список строк, для которых столбцы не назначены. Проще всего такой список получить посредством проверки условия $r_j < m,\ j = \overline{1,n}$ (рис. 2).

```
template <class cost> void LAP<cost>::col(int j) {
 cost h=inf;
 for (int i=0; i<m; i++) is_min(h,c[i*n+j]-u[i]);
 v[j]=h;
}

template <class cost> // изменение элемента матрицы задачи
void LAP<cost>::set(int i, int j, cost x) {
 c[i*n+j]=x;
 if (i!=r[j]) {
  col(j);
  i=r[j];
 }
 r[j]=m;
 lap(i);
}
```

Рис. 2. Функции реоптимизации решения задач о назначении

Реоптимизация задачи с прямоугольной матрицей, когда $m < n$, должна учитывать факт включения столбцов в текущее назначение. Изменение элементов для не назначенных столбцов не требует пересчета.

В общем случае коррекцию элементов матрицы удобно проводить групповым методом, совмещая операции обработки подвергшихся изменению строк и столбцов. Однако в некоторых случаях схема исходной задачи предопределяет номер строки, что будет показано далее.

Применение реоптимизации для решения задачи выбора

Исключение этапа инициализации применимо, как следствие, для решения комбинаторной задачи выбора вида

$$\min\left\{\sum_{i\in M}\sum_{j=1}^{n} c_{ij}x_{ij}\left|\sum_{i\in M} x_{ij}=\sum_{j=1}^{n} x_{ij}=1;\ x_{ij}\ge 0;\ i\in M,\ j=\overline{1,n},\ M\subset\overline{1,n}\right.\right\}, \qquad (7)$$

когда решение ЛЗН вида (2) дополнено необходимостью выбора из множества сочетаний индексов исходной матрицы задачи [3, 4].

Прямолинейный способ решения задачи (7) – перебор среди ЛЗН для всех возможных сочетаний. Вычислительная сложность такой процедуры имеет оценку $C_n^m \cdot O(n^3)$. При перечислении сочетаний естественно использовать взаимозависимость последовательно порождаемых ЛЗН и возможность наследования решений задач вида (2) [4]. В этом случае пересчет потребуется лишь для измененных строк матрицы очередного сочетания.

Предпочтительным методом порождения сочетаний для задачи (7) является метод «вращающейся двери» с одиночным расстоянием Хэмминга между множествами M_k^* и M_{k+1}^*, $k=\overline{1,C_n^m}$ [5]. Это означает изменение одной строки матрицы очередной ЛЗН и приводит к выполнению условия $\left|M_k^*\right|\le 1,\ k>1$.

Таким образом, вычислительная сложность процедуры решения задачи (7) будет иметь оценку $C_n^m \cdot O(n^2)$. Это на порядок лучше оценки для прямолинейной реализации схемы полного перебора ЛЗН для всех сочетаний, рассматриваемых независимо. Замена схемы полного перебора на более экономную схему не влечет сокращения времени решения задачи, так как схема порождения сочетаний здесь отражается первым сомножителем.

Поиск глобально оптимального варианта среди множества ЛЗН уместно проводить с прерыванием анализа вариантов, которые наверняка будут хуже лучшего варианта из просмотренной последовательности. Установление бесперспективности варианта для некоторых ЛЗН предлагается проводить, используя оценки нижней границы целевой функции в процессе решения. На основании теории двойственности, нижняя оценка целевой функции $\tilde{Z}^q$ на итерации q решения ЛЗН венгерским методом определяется выражением

$$\tilde{Z}^q=\sum_{i=1}^{m} u_i^q+\sum_{j=1}^{n} v_j^q, \qquad (8)$$

где u_i^q, v_j^q – значения потенциалов строк и столбцов на итерации q. Если $\tilde{Z}^q \ge Z_{rec}$, где Z_{rec} – значение целевой функции лучшего из просмотренных вариантов, то итерации анализа рассматриваемой ЛЗН можно прекратить. Так как на последней итерации нижняя оценка целевой функции совпадает с оценкой оптимального решения, то выражение (8) эффективнее для вычисления по сравнению с прямолинейной реализацией выражения (1).

Однако перебор уместен при наличии дополнительных ограничений на правила образования сочетаний. Наивная схема перебора для решения задачи (7) при отсутствии таких ограничений может быть исключена, если использовать рекуррентный алгоритм реализации венгерского метода решения ЛЗН с итерацией по строкам (рис. 1).

Процесс решения ЛЗН с матрицей размером $m\times n$ в этом случае соответствует итерациям добавления строк в последовательности задач

$$\min\left\{\sum_{i=1}^{k}\sum_{j=1}^{n}c_{ij}x_{ij}\;\middle|\;\sum_{i=1}^{k}x_{ij}=\sum_{j=1}^{n}x_{ij}=1;\; x_{ij}\ge 0;\; i=\overline{1,k},\, j=\overline{1,n},\right\},\quad k=\overline{1,m}\,. \tag{9}$$

При этом $m \le n$, но если транспонировать матрицу задачи (9), то получаем случай (1) с учетом замены содержательного смысла индексов строк и столбцов матриц. Вычислительная сложность решения задачи (9) оценивается величиной $O(n^3)$.

Применение реоптимизации для решения задачи коммивояжера

Конкретизируем схему пересчета ЛЗН [1] при использовании метода ветвей и границ для решения точного решения асимметричных задач коммивояжера [5].

Формальная модель задачи коммивояжера имеет вид:

$$\min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij}\;\middle|\;\begin{array}{l}\sum\limits_{i=1}^{n}x_{ij}=\sum\limits_{j=1}^{n}x_{ij}=1;\; x_{ij}\ge 0,\; i,j=\overline{1,n};\\ u_i-u_j+nx_{ij}\le n-1,\;\; i=\overline{2,n},\;\; j=\overline{2,n},\;\; i\ne j\end{array}\right\} \tag{10}$$

Алгоритм метода ветвей и границ может быть реализован разными способами, различающимися правилами порождения ветвей дерева вариантов. Наиболее успешным для решения задачи (10) оказывается подход, базирующийся на решении ЛЗН, 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.

Процесс релаксации в узле дерева вариантов задачи коммивояжера связан с установкой бесконечных значений элементов матрицы ЛЗН. Очевидна возможность частичного наследования предшествующего решения. Реализацию ветвления на уровне l дерева вариантов задачи коммивояжера предлагается проводить на векторе решения $R^l = \left\{i \mid x_{ij}=1,\; j=\overline{1,n}\right\}$.

Можно заметить, что если k – некоторая вершина решения задачи (2), то последовательность

$$K^l(k)=\left\{K(0)=k,\left\langle K(i)=R^l_{K(i-1)}\,|\,K(i)\ne k\right\rangle\right\}\subseteq R^l$$

только тогда соответствует гамильтонову циклу, когда условием остановки является $K(n-1)=k$, $k\in\overline{1,n}$. Если цикл не гамильтонов, т.е. $K^l(k)\subset R^l$, то необходимо породить множество задач уровня $l+1$. Для этого следует указать цикл минимальной длины, выбрав вершину входа в цикл

$$k^l=\arg\min_k\left\{\left|K^l\left(k\right)\right|,\; k\in\overline{1,n}\right\}.$$

Правило порождения ЛЗН тривиально – для каждой вершины обнаруженного цикла необходимо запретить посещение других вершин этого цикла. Количество порождаемых ЛЗН – $\left|K^l\left(k^l\right)\right|$. При этом матрица очередной ЛЗН отличается от предыдущей одной строкой, в которой часть элементов заменяются значением, не меньшим значения

$$c_{\max}=\max_{i,j}\left\{c_{ij}:i\ne j,\; i=\overline{1,n},\; j=\overline{1,n}\right\}.$$

Построчное изменение матриц проводится по следующему закону:

$$c_{ij}^{l+1}=c_{ij}^{l}+c_{\max}\left(i\in K^l\left(k^l\right)\wedge j\in K^l\left(k^l\right)\right),\; i,j=\overline{1,n}\,. \tag{11}$$

Из выражения (4) следует, что релаксация на множестве ЛЗН при решении задачи коммивояжера методом ветвей и границ соответствует условию $|M^*|=1,\ m=n$. Последнее объясняет сокращение времени решения задачи коммивояжера в n раз.

Из (11) следует, что изменение элементов матриц в любом узле дерева вариантов производится только путем увеличения значения. Учитывая, что потенциал измененной строки будет уточняться на итерации пересчета, предварительное уточнение потенциалов не требуется.

Экспериментальная оценка времени решения задач

В таблице приведены результаты экспериментальной оценки среднего времени решения ЛЗН и реоптимизации результатов решения после изменения одного из случайно выбираемых элементов матриц. Размеры матриц случайных исходных данных, генерируемых по равномерному закону распределения, – n=500…5000, $m=n/2$.

Оценка времени реоптимизации и независимого решения задач о назначении

Размерность, задачи ($m\times n$)	Среднее время решения, мсек (Celeron 1,7 Ггц, 512 Мбайт)		
	Реоптимизация	Метод SAP [1]	Венгерский метод
250×500	0,533	19,167	14,167
500×1000	0,533	98,867	86,000
750×1500	1,033	279,567	344,867
1000×2000	2,033	587,800	757,867
1250×2500	3,067	1088,267	1438,400
1500×3000	5,633	1787,267	2509,767
1750×3500	5,633	2745,167	4044,633
2000×4000	32,200	4006,067	6165,533
2250×4500	33,767	5579,000	8810,333
2500×5000	34,800	7521,700	12043,167

Предложенный алгоритм реоптимизации использует значения потенциалов строк и столбцов предыдущего решения, которые формируются алгоритмом венгерского метода (рис. 1). Результаты эксперимента подтверждают ожидаемое преимущество реоптимизации перед прямолинейным пересчетом, реализуемым даже на основе лучшего среди известных методов решения ЛЗН метода SAP. Алгоритм метода SAP [1] в явном виде значения потенциалов не формирует, поэтому реоптимизация на его основе требует отдельного рассмотрения.

Заключение

Таким образом, реоптимизация решения ЛЗН – эффективный прием решения последовательно возникающих задач. Сокращение времени получения оптимального решения в первом приближении пропорционально объему локальных изменений данных задачи. Дополнительная память для хранения наследуемых значений потенциалов строк и столбцов не превышает объема $O(m+n)$. При этом сохраняется возможность прерывания итераций расчета после установления бесперспективности текущего варианта относительно ранее рассмотренных.

REOPTIMIZATION OF THE LINEAR ASSIGNMENT PROBLEM

M.P. REVOTJUK, P.M. BATURA, A.M. POLONEVICH

Abstract

The problem of the solution of sequence of the linear assignment problems, the initial data different by minor alteration is considered. Inheritance of the result's data of previous problems and its reoptimization allows to decreasing time of reception of the new solution. The algorithm of reoptimization, based on a Hungarian method, is offered.

Литература

1. *Jonker R., Volgenant A.* // Computing, 1987. Vol. 38. P. 325–340.
2. *Fischetti M., Lodi A., Toth P. et al.* // The Traveling Salesman Problem and its Variations. 2002. P. 169–194.
3. *Turkensteen M., Ghosh D., Goldengorin B. et al.* // European Journal of Operational Research. 2007. Vol. 189. P. 775–788.
4. *Toroslu I.H., Üçoluk G.* // Information Sciences 2007. Vol. 177. P. 1523–1529.
5. *Ревотюк М.П., Полоневич А.М.* // Сетевые компьютерные технологии: Сб. тр. III Межд. науч. конф. (Минск, 17–19 октября 2007 г.) Минск. 2007. С. 58–63.