УДК 656.2-50: 519.8

БЫСТРАЯ ОЦЕНКА ИНТЕРВАЛОВ УСТОЙЧИВОСТИ РЕШЕНИЯ ЛИНЕЙНЫХ ЗАДАЧ О НАЗНАЧЕНИИ

М.П. РЕВОТЮК, М.К. КАРОЛИ, П.М. БАТУРА

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

Поступила в редакцию 20 апреля 2013

Предложен эффективный алгоритм оценки интервалов устойчивости решений открытых и закрытых линейных задач о назначении, основанный на итерациях пересмотра результатов оптимизации. Экономный одношаговый переход к ближайшей вершине политопа задачи для всех ребер оптимального совершенного паросочетания позволяет практически на порядок снизить вычислительную сложность оценки устойчивости ее решения.

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

Постановка задачи

Решение классических закрытых линейных задач о назначении (ЛЗН) [1,2], формально записываемых в виде

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

обычно есть вектор назначений строк матрицы коэффициентов ее столбцам:

$$R = \left\{ r_j = i \,\middle|\, x_{ij} = 1,\; i,\, j = \overline{1,n} \right\}. \qquad (2)$$

Во многих случаях требуется оценка устойчивости назначения. При этом для каждого элемента c_{ij} в (1) необходимо вычислить интервал (a_{ij}, b_{ij}), в котором значения таких элементов могут быть изменены без нарушения структуры существующего оптимального назначения (2). Очевидно, что $c_{ij} \in (a_{ij}, b_{ij}),\; i,\, j = \overline{1,n}$, однако значения границ таких интервалов не являются непосредственным результатом оптимизации назначения. Симплекс-метод – формальная основа лучших из известных алгоритмов решения ЛЗН, принципиально не гарантирует прохождения всех вершин политопа задачи (1).

Базовый алгоритм решения задачи

Рассматриваемая задача – специальный случай оценки чувствительности решений задач линейного программирования. Алгоритм определения интервалов устойчивости решения закрытых ЛЗН вида (1) предложен в [3] на основе модификации процедуры известного венгерского метода. При этом для каждого элемента матрицы $(c_{ij},\; i,\, j = \overline{1,n})$ интервал устойчивости определятся путем оценки изменения функции критерия оптимизации после инверсии вхождения такого элемента в оптимальное совершенное паросочетание. Формально

ключевая процедура оценки интервалов устойчивости – реоптимизация решения ЛЗН после целенаправленного возмущения значения каждого элемента матрицы.

В настоящей работе рассмотрен способ улучшения предложенного в [3] алгоритма в направлении снижения его вычислительной сложности и распространения на случай открытых ЛЗН. Для этого предварительно проведем ревизию хорошо известных схем венгерского метода с целью эффективного выполнения этапа реоптимизации решений ЛЗН [1,2].

Открытые (асимметричные) ЛЗН с прямоугольными матрицами

$$\min\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\cdot x_{ij} \;\middle|\; \sum_{i=1}^{m} x_{ij} = \sum_{j=1}^{n} x_{ij} = 1;\; x_{ij}\ge 0;\; i=\overline{1,m},\; j=\overline{1,n}\right\} \qquad (3)$$

часто решают как задачи вида (1) после дополнения матрицы строками или столбцами с нулевыми элементами. Сканирование нулевых строк или столбцов является бесполезным, но необходимым этапом алгоритма венгерского метода [1].

Далее ЛЗН будут рассмотрены в постановке (3), когда $m\le n$. При этом холостые итерации сканирования нулевых строк будут исключены. Случай $m>n$ легко сводится к решению (2) после транспонирования матрицы $(c_{ij},\ i,j=\overline{1,n})$ и соответствующей замены содержательной интерпретации индексов строк и столбцов.

Предварительно отметим, что для открытой ЛЗН вида (3), используемое далее понятие оптимального паросочетания соответствует паросочетанию степени $m<n$. С целью удобства выделения оптимального паросочетания вектор назначения строк столбцам определим так:

$$R=\left\{ r_j = i\cdot(t_j=1)+(m+1)\cdot(t_j=0),\;\; i=\left\{k\,\middle|\, x_{kj}=1,\; k=\overline{1,m}\right\},\; t_j=\sum_{k=1}^{m} x_{kj},\;\; j=\overline{1,n}\right\}.$$

Так как по определению $|R|=n$, то такой вектор формируется путем отображения оптимального решения на элементах, первоначально имеющих значения $r_j=m+1,\;\; j=\overline{1,n}$.

Известно, что наиболее эффективные для решения задачи (1) алгоритмы венгерского метода строятся с учетом особенностей двойственной задачи

$$\max\left\{\sum_{i=1}^{m} u_i + \sum_{j=1}^{n} v_j \;\middle|\; c_{ij}-u_i-v_j\ge 0,\;\; i=\overline{1,m},\;\; j=\overline{1,n}\right\}. \qquad (4)$$

Здесь неизвестными являются потенциалы строк и столбцов. Значения потенциалов особого интереса не представляют, но определяют решение задачи (2). Схемы известных версий алгоритмов венгерского метода решения ЛЗН включают быстрый этап формирования начального назначения строк и итерационного дополнения решения для оставшихся строк. На начальном этапе пытаются выполнить назначение строк, используя операцию приведения матрицы задачи. Приведение состоит в вычитании из элементов столбцов минимальных элементов столбцов. Различные версии известных алгоритмов решения ЛЗН отличаются лишь приемами начального частичного назначения строк [1, 2].

Однако при реализации алгоритма решения ЛЗН с ориентацией на систематическую реоптимизацию решений этап формирования исходного решения не является узким местом. Инициализация векторов потенциалов строк и столбцов любой ЛЗН принципиально может быть реализована итерациями формирования оптимального паросочетания инкрементной версией алгоритма венгерского метода [4]. При этом удается учесть чувствительность целевой функции к изменениям переменных ЛЗН (см. рис. 1).

Предлагаемый шаблон класса решения ЛЗН аккумулирует ряд приемов сокращения холостых шагов при обработке последствий изменения размера и элементов матрицы задачи. Например, можно заметить, что изменение элементов матрицы влечет необходимость пересмотра результата оптимизации, если только меняется позиция нулевого элемента после операции приведения. Действительно, увеличение элемента $c_{ij}\leftarrow c_{ij}+0$ 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 $c_{ij} \leftarrow c_{ij} - 0$, когда $x_{ij} = 1$, не нарушает соотношения $c_{ij} = u_i + v_j$ в задаче (2).

В других случаях требуется повтор итераций назначения строк. Любое изменение элементов матрицы должно отражаться значениями потенциалов. Если изменены элементы строки i, то ее потенциал должен соответственно меняться: $u_i = \min\{c_{ij} - v_j, j = \overline{1,n}\}$. В случае изменения элементов столбца j его потенциал также должен меняться: $v_j = \min\{c_{ij} - u_i, i = \overline{1,m}\}$.

```
template <class T> struct item { T a,b; }; // Интервал устойчивости

template <class T>
 struct to_min: public item<T> { // Класс стратегии поиска минимума
 static T inf() { return numeric_limits<T>::max()/2; }
 static bool lt(T &a, const T b) {
 if (a<=b) return false;
 a=b;
 return true;
 }
 void hide(T x) { a=-inf(); b=x; }
 void show(T x) { a=x, b=inf(); }
};

template <typename E> struct Solution { // Результат решения ЛЗН
 vector<E> u,v; // вектор потенциалов строк и столбцов
 vector<int> q,r; // вектор текущего решения
 Solution(int M,int N): u(M),v(N),q(M),r(N) {}
 void operator=(Solution &obj) { u=obj.u; v=obj.v; q=obj.q; r=obj.r; }
};

template <typename E, template <typename T> class D>
class LAP: public Solution<E> { // Решение и реоптимизация ЛЗН
 vector<int> prec, mark;
 vector<E> s;
 E maxs;
 int stp; // номер итерации решения задачи
 vector<E> b; // локальная копия матрицы стоимости
 void lap(int i);
public:
 int m, n; // размерность задачи
 vector<E *> c; // указатели строк матрицы стоимости
 void lap();
 E setx(int i,int j, E x) { // изменение элемента матрицы
 E cij=c[i][j]; c[i][j]=x; r[q[i]]=n;
 lap(i); // назначение модифицированной строки
 return cij;
 }
 LAP(int M,int N,E *C): Solution(N,N),m(M),n(N),c(N),
 prec(N),s(N),mark(N), stp(N*N*N),b(N*M), maxs(D<E>::inf()) {
 for (int i=0, k=0; i<m; i++) {
 c[i]=&b[i*n];
 for (int j=0; j<n; j++) c[i][j]=C[k++];
 }
 lap();
 }
}
```

Рис. 1. Шаблон базового класса решения задач о назначении

Выявление отмеченных и других особых ситуаций предлагается совместить с расширением назначения и повтором назначения модифицированных строк (рис. 1).

Очевидно, что в случае, когда $m \le n$, требуется организация итерации по строкам, для

которых столбцы не назначены. Список таких строк можно получить проверкой условия $r_j \le m,\ j = \overline{1,n}$. Нетрудно заметить, что подобная проверка не потребуется, если использовать вектор отображения назначения строк столбцам

$$Q = \left\{ q_i = j \,\middle|\, r_j = i,\ j = \overline{1,n},\ i = \overline{1,m} \right\}.$$

Векторы Q и R удобно формировать на каждой итерации алгоритма венгерского метода, совмещая операции обработки последствий подвергшихся изменению строк и столбцов согласно (5) или (6). Это допускает проведение реоптимизации решения ЛЗН в любой момент синхронно с коррекцией элементов матрицы (рис. 2).

```
template <typename E, template <typename T> class D>
 void LAP<E,D>::lap() { // назначение строк
 for (int j=0; j<n; j++) { v[j]=0; r[j]=n; mark[j]=prec[j]=stp; }
 for (int i=0; i<m; i++) lap(i); // назначение реальных строк
 if (m<n) for (int i=m, j=0; j<n; j++) if (r[j]==n) {
 u[i]=-v[j]; q[r[j]=i]=j; i++; // назначение виртуальной строки
 }
 }

template <typename E, template <typename T> class D>
 void LAP<E,D>::lap(int i) { // назначение строки
 int j=0, k,row,col; E *b=c[i], h=maxs;
 for (k=0; k<n; k++) if (D<E>::lt(h,s[k]=b[k]-v[k])) j=k;
 if (u[i]=h) for (k=0; k<n; k++) s[k]-=h;
 for (stp-=n; (row=r[j])<n; ) {
 mark[j]=stp; h=maxs; b=c[row]; E uk=u[row]; col=j+stp;
 for (k=0; k<n; k++) if (mark[k]!=stp) {
 if (D<E>::lt(s[k],((row<m)? b[k]:0)-uk-v[k])) prec[k]=col;
 if (D<E>::lt(h,s[k])) j=k;
 }
 if (h) for (u[i]+=h, k=0; k<n; k++) if (mark[k]==stp) {
        u[r[k]]+=h; v[k]-=h;
 } else s[k]-=h;
 }
 while ((k=prec[j]-stp)<n) { q[r[j]=r[k]]=j; j=k; }
 q[r[j]=i]=j;
}
```

Рис. 2. Основные функции класса решения задачи о назначении

Оценка интервалов устойчивости решения

Элементы матрицы ЛЗН можно разделить на два вида, отражая их принадлежность текущему оптимальному решению. Элементы такого решения соответствуют назначенным ребрам графа оптимального паросочетания

$$E_m = \left\{ (r_j, j) \,\middle|\, (r_j \le m),\ j = \overline{1,n} \right\} = \left\{ (x_i, y_i),\ i = \overline{1,m} \right\}.$$

Оставшиеся элементы матрицы дополняют граф задачи (3):

$$E_u = \left\{ (i, j),\ i = \overline{1,m},\ j = \overline{1,n} \right\} \setminus E_m .$$

Очевидно, что $|E_m| = m$, а $|E_u| = m \cdot (n-1)$, что приходится учитывать при оценке вычислительной сложности алгоритма. Обозначим $\delta_m(x,y)$ и $\delta_u(x,y)$ – допустимое изменение веса любого ребра $x \to y$, где $(x,y) \in E_m \cup E_u$.

Интервал значений веса назначенного ребра, в котором назначение ребра остается неизменным, для задачи минимизации вида (3) легко записать из элементарных соображений:

$$I^m_{\min}(x,y) = (-\infty, c_{xy} + \delta_m(x,y)],\ (x,y) \in E_m;\ I^u_{\min}(x,y) = [c_{xy} - \delta_u(x,y), +\infty),\ (x,y) \in E_u.$$

Последнее означает, что существующий вес можно увеличить для назначенных ребер или уменьшить для остальных ребер без нарушения структуры текущего решения задачи минимизации.

При решении задачи максимизации границы интервалов зеркально отражаются относительно начала координат:

$$I^m_{\max}(x,y)=[c_{xy}-\delta_m(x,y),+\infty),(x,y)\in E_m;\ \ I^u_{\max}(x,y)=(-\infty,\ c_{xy}+\delta_u(x,y)],(x,y)\in E_u.$$

Будем далее рассматривать случаи задачи минимизации. В таких случаях говорят, что ребро графа оптимального паросочетания скрыто, если его вес увеличен так, чтобы ребро больше не являлось частью существующего решения. Скрытие формально реализуется назначением веса из интервала значений $(c_{xo}+\delta_m(x,y),+\infty)$. Как известно, для элементов оптимального паросочетания справедливо условие $c_{xy}=u_x+v_y,\ (x,y)\in E_m$.

Пусть оценка оптимального решения ЛЗН есть

$$Z^0=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}\cdot x_{ij}=\sum_{i=1}^{m}u_i+\sum_{j=1}^{n}v_j\ . \qquad (5)$$

Рассмотрим ребра, принадлежащие оптимальному решению E_m. Можно показать, что если ребро $x\to y$ скрыто, то реоптимизация решения может быть проведена относительно строки x (при условии наличия других связей вершины x). Легко заметить, что процесс реоптимизации, начинающийся в вершине x, завершится в вершине y, потенциал которой тоже не изменится. Меняется только потенциал строки u_x (рис. 2). Из выражения (5) следует, что изменение оценок решения оценивается выражением

$$u^1_x-u^0_x=Z^1-Z^0,\ x\in\overline{1,m}. \qquad (6)$$

Здесь нулевой верхний индекс использован для пометки исходного, а единичный – нового решения. Далее покажем, что искомое значение $\delta_m(x,y)=Z^1-Z^0$. Очевидно, что назначение нового веса $c_{xy}\leftarrow c_{xy}+\delta_m(x,y)$ приведет к изменению оценки решения – $Z^1\leftarrow Z^0$. Новое решение без ребра $x\to y$ имеет оценку $Z^1=Z^0+\delta_m(x,y)$. Если после этого вес ребра $x\to y$ уменьшить так, что $\delta_m(x,y)\leftarrow\delta_m(x,y)-0$, то последствия такого шага очевидны – условие $Z^1\ge Z^0$ останется справедливым, но ребро станет частью исходного оптимального паросочетания. Последнее означает, что с учетом (5)

$$I^m_{\min}(x,y)=(-\infty,c_{xy}+Z^1-Z^0]=(-\infty,c_{xy}+u^1_x-u^0_x],\ \ (x,y)\in E_m. \qquad (7)$$

Отсюда следует алгоритм построения интервала безопасного изменения значений веса ребра. Установим гарантированно приводящее к скрытию ребра значение $c_{xy}=+\infty$. Фиксируя в соответствии с (6) потенциалы строки x до и после проведения реоптимизации, получим $\delta_m(x,y)=u^1_x-u^0_x$. Использование разности значений потенциалов исключает необходимость наивного прямолинейного вычисления оценок решений задачи (1), требующего $m+n$ шагов.

Следуя [3], рассмотрим ребра, не принадлежащие оптимальному решению, когда $c_{xy}\ge u_x+v_y,\ (x,y)\in E_u$. Если вес таких ребер менять в интервале $(u_x+v_y,+\infty)$, то структура решения остается неизменной.

Обозначим далее $\varepsilon_u(x,y)=c_{xy}-\delta_u(x,y)$. Будем считать, что ребро графа паросочетания открыто, если его вес уменьшен так, что это ребро становится частью нового оптимального решения. Открытие ребра наступает в случае, когда $c_{xy}\leftarrow\varepsilon_u(x,y)$.

С целью определения $\varepsilon_u(x,y),(x,y)\in E_u$, построим вспомогательный граф G_a, образуемый из графа оптимального паросочетания путем удаления всех дуг, инцидентных вершинам x и y. В таком графе будет (m–2) ребра графа оптимального паросочетания. Если

далее выполнить уменьшение веса $c_{xy} \leftarrow \varepsilon_u(x,y)$, то получим для графа G_a, дополненного ребром (x,y), оптимальное паросочетание степени m с оценкой Z^a. Если для рассматриваемого ребра выполняется условие $c_{xy} < Z^0 - Z^a$, то оценка решения, содержащее такое ребро, будет удовлетворять условию $c_{xy} + Z^a < Z^0$. Это противоречит предположению о том, что исходное оптимальное паросочетание было совершенным. Таким образом, значение $\varepsilon_u(x,y) = Z^0 - Z^a$ есть нижняя граница интервала устойчивости, поэтому

$$I^u_{\min}(x,y) = [Z^0 - Z^a, +\infty),\ (x,y) \in E_u. \qquad (8)$$

Схема ускорения процедуры оценки устойчивости

Однако реализация алгоритма построения интервала значений веса ребер на основе (8) не является эффективной. Для каждого из $m \cdot (n-1)$ ребер придется решать ЛЗН, размер которых $(m-1)(n-1)$. Учитывая дискретный характер процесса перемещения по вершинам симплекса при реализации алгоритма решения ЛЗН, предлагается вместо выражения (8) воспользоваться выражением (7), инвертируя направление шагов процесса построения интервала. Конечная граница интервала $I^m_{\min}(x,y)$ после этого станет начальной границей интервала $I^u_{\min}(x,y)$. Нулевой шаг в (7) становится решением ЛЗН для гарантированно приводящего к открытию ребра значения $c_{xy} = -\infty$, а единичный шаг соответствует решению ЛЗН с исходной матрицей. В результате получаем

$$I^u_{\min}(x,y) = (-\infty + Z^0 - Z^1, +\infty] = (-\infty + u^o_x - u^1_x, +\infty],\ (x,y) \in E_u. \qquad (9)$$

Таким образом, оценка выражения (9) требует лишь одношаговой реоптимизации исходной задачи (см. рис. 3). Отметим, что интервал бесконечных значений $(-\infty, +\infty)$ для удобства программной реализации выражений (7) и (9) может безопасно отображаться на интервал $(c_{\min} - 1, c_{\max} + 1)$, где $c_{\min} = \min\{c_{ij}, i = \overline{1,m},\ j = \overline{1,n}\}$, $c_{\max} = \max\{c_{ij}, i = \overline{1,m},\ j = \overline{1,n}\}$.

```
template <class E, template <class T> class D>
class ILAP: public LAP<E,D> { // Оценка интервалов устойчивости
 vector<D<E>> ab;
 Solution<E> os;
 void back(int i,int j, E x) { c[i][j]=x; *(Solution *)this=os; }
 void hide(int i,int j) { // скрытие ребра оптимального паросочетания
 E cij=setx(i,j,maxs);
 ab[i*n+j].hide(cij+u[i]-os.u[i]);
 back(i,j,cij);
 }
 void show(int i,int j) { // открытие ребра оптимального
паросочетания
 E cij=setx(i,j,-maxs);
 ab[i*n+j].show(-maxs+os.u[i]-u[i]);
 back(i,j,cij);
 }
public:
 ILAP(int M,int N,E *C):LAP(M,N,C),ab(m*n),os(n,n) {
 os=*this;
 for (int i=0; i<m; i++) { // построение матрицы интервалов
 int k=os.q[i];
 for (int j=0; j<k; j++) show(i,j);
 hide(i,k);
 for (int j=k+1; j<n; j++) show(i,j);
 }
 }
};
```

Рис. 3. Шаблон класса оценки интервалов устойчивости решения задач о назначении

Пример решения задачи о назначении с оценкой его устойчивости

Исходные данные и результаты решения примера задачи о назначении с оценкой интервалов его устойчивости представлены в таблице. Элементы таблицы – тройки (a_{ij}, c_{ij}, b_{ij}), где интервал устойчивости задан парами (a_{ij}, b_{ij}), $i = \overline{1,m}$, $j = \overline{1,n}$. Оптимальное назначение строк столбцам соответствует выделенным подчеркиванием элементам таблицы.

Исходные данные и результат решения открытой задачи о назначении (*m* = 4, *n* = 6)

i \ *j*	1	2	3	4	5	6
1	33, 87, +∞	46, 62, +∞	57, 64, +∞	39, 43, +∞	57, 72, +∞	-∞, 57, 61
2	-∞, 52, 76	41, 68, +∞	52, 94, +∞	36, 63, +∞	52, 76, +∞	50, 83, +∞
3	32, 77, +∞	-∞, 48, 52	55, 64, +∞	34, 54, +∞	55, 59, +∞	55, 68, +∞
4	32, 58, +∞	45, 49, +∞	56, 71, +∞	-∞, 42, 46	56, 66, +∞	56, 89, +∞

Заключение

Таким образом, определение интервалов устойчивости открытых и закрытых ЛЗН может проводиться посредством реоптимизации текущего оптимального решения, если инвертировать принадлежность дуг графа задачи соответствующему совершенному паросочетанию и учесть эту принадлежность направлением нумерации состояний. Инверсия реализуется заменой веса дуги на максимальное значение весов для дуг оптимального паросочетания или на минимальное значение для остальных дуг. Время получения оценок устойчивости на основе разности потенциалов изменяемых строк в первом приближении сокращается не менее, чем в n раз, по сравнению с известной версией [3]. Решение вспомогательных задач о назначении не требуется. Дополнительная память для хранения наследуемых значений потенциалов строк и столбцов не превышает объема $O(m+n)$.

QUICK EVALUATION OF THE INTERVAL STABILITY OF THE LINEAR ASSIGNMENT PROBLEM SOLUTIONS

M.P. REVOTJUK, M.K. QARALEH, P.M. BATURA

Abstract

An efficient algorithm for evaluating the interval of solutions stability of opened and closed linear assignment problems based on a review of the results of optimization iterations is proposed. Economical one-step transition to the nearest vertex of the polytope of problem one for all the edges of the optimal perfect matching is almost an order of magnitude reduces the computational complexity of evaluating the stability of the current optimal solution.

Список литературы

1. *David W. Pentico*. // European Journal of Operational Research. 2007. № 176 (2). P. 774–793.
2. *Bijsterbosch J., Volgenant A.* // Annals of Operations Research. 2010. № 181 (1). P. 443–462.
3. *Lantao Liu, Dylan A. Shell*. // The International Journal of Robotics Research. 2011. № 30 (7). P. 936–953.
4. *Ревотюк М.П., Батура П.М., Полоневич А.М.* // Докл. БГУИР. 2011. № 1 (55). С. 55–62.