ОПТИМИЗАЦИЯ УПРАВЛЕНИЯ РАСПРЕДЕЛЕНИЕМ РЕСУРСОВ В ПОТОКАХ РАБОТ

Ревотюк М. П., канд. техн. наук
Бруй Н. М., аспирант, e-mail: brujnm@bsuir.by
Калита О.В., аспирант, e-mail: olgakalitav@gmail.com
Сыс А.Д., аспирант, e-mail: sys@bsuir.by

2022

Белорусский государственный университет информатики и радиоэлектроники

Ключевые слова: задачи координации агентов, потоки работ, ЛЗН

Аннотация: Предлагается модель и алгоритмы решения задачи о динамическом назначении планов распределения ресурсов в потоках работ. Учет взаимозависимости работ и выполняющих их агентов позволяет снизить задержки моментов принятия решений по ликвидации проблемных ситуаций.

1. Постановка задачи

Традиционно задачи координации агентов в потоках работ сводятся к известным задачам дискретной оптимизации, таким как линейная задача о назначении (ЛЗН) или задача нескольких странствующих коммивояжеров [1,2]. Однако необходимость учета реальных отношений между агентами и задачами приводит к экспоненциальной сложности алгоритма формирования оптимального назначения. Подобная сложность приводит к задержке момента назначения заданий, снижая эффективность системы агентов. Традиционные приемы использования различного рода аппроксимаций часто неработоспособны из-за недостаточной конкретизации и определенности формируемых решений, а модели реальных систем оказываются сетевыми [2].

Предлагается учесть дискретность процесса формирования портфеля заявок, явно используя понятия наиболее раннего и позднего срока начала решения ЛЗН и уточнения окончательного плана работы агентов. Очевидно, что если процедура назначения дополняет граф оптимального паросочетания при поступлении новых заявок [3], то задержка времени подготовки плана определяется сложностью обработки последней группы заявок.

Рассмотрим пример учета динамики формирования описания ЛЗН, когда строки ее матрицы становятся известными постепенно (рис 1). Здесь представлен случай, когда в момент времени TS необходимо решать ЛЗН, размерность которой NS. Процесс решения такой задачи, например, венгерским методом, отражен сплошной ломаной линией на отрезке времени (IS, T). Излом этой линии в точке TE отражает использование известных эвристик реализации процедур решения ЛЗН для быстрого формирования начального назначения 4.

Можно заметить, что в моменты времени T1, T2, T3 и TS появляется возможность использования инкрементальной схемы алгоритма решения ЛЗН [2,3], когда становятся известными очередные строки матрицы. Процесс работы

алгоритма с реоптимизацией решения для матриц с добавленными строками отражен пунктирной ломаной линией. Примечательно, что после получения последней порции строк матрицы в момент времени IS завершение решения ЛЗН, размерность которой достигает NS, произойдет в момент времени TI. Очевидно, что TI<TC при любых версиях алгоритмов решения ЛЗН, что определяется их полиномиальной вычислительной сложностью.

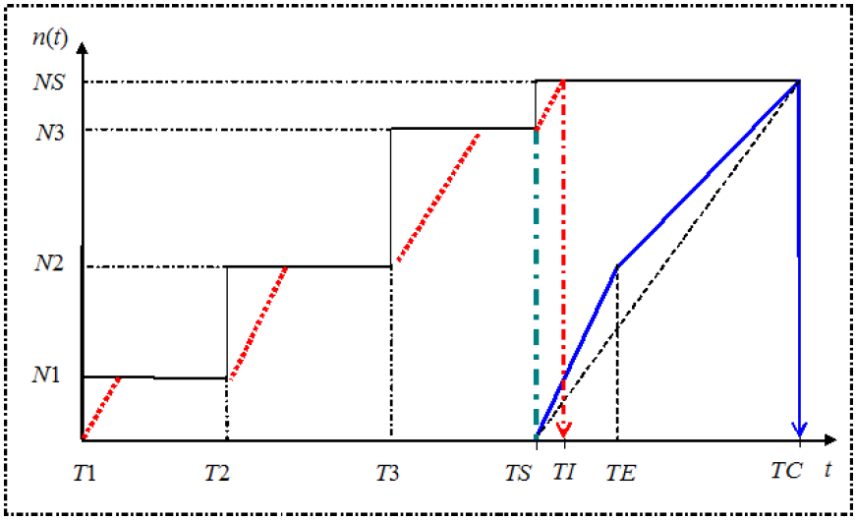

Рисунок 1 – Иллюстрация хода решения ЛЗН

Реализация предлагаемой схемы возможна на рекуррентных сетевых моделях, состояние которых соответствует графу текущего паросочетания с выделением оптимального решения. Переход между состояниями требует решения ЛЗН, задачи коммивояжера и поиска кратчайших путей на графах. На параметры таких задач проецируются особенности процессов обслуживания, включая векторные критерии и разнообразные отношения вложенности [1]. Цель работы – улучшение вычислительной эффективности известных версий инкрементальной схемы алгоритма решения ЛЗН [2,3].

2. Схема инкрементального алгоритма

Решение классических открытых ЛЗН, записываемых в виде

$$\begin{cases} Z_k = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^k x_{ij}^k \to min \\ \sum_{i=1}^{m} x_{ij}^k = 1, j = \overline{1,n}; \sum_{j=1}^{n} x_{ij}^k = 1, i = \overline{1,m} \end{cases} \quad (1)$$

обычно является вектором назначений строк матрицы коэффициентов её столбцам

$$R_k = \{r_j = i | x_{ij}^k = 1, i = \overline{1,m}, j = \overline{1,n}\} \quad (2)$$

Известно, что наиболее эффективные для решения задачи (1) алгоритмы венгерского метода [4] строятся с учетом особенностей двойственной задачи

$$\begin{cases} Z_k = \sum_{i=1}^{m} u_i + \sum_{j=1}^{n} v_j \to max \\ c_{ij} - u_i - v_j, i = \overline{1,m}, j = \overline{1,n}. \end{cases} \quad (3)$$

Предлагается модифицировать инкрементальный алгоритм (3) обновления текущего паросочетания для новой или изменяемой строки i:

```
procedure hlap(i) begin
    u_i = 0, h = ∞; w ← w + n;
    for k ∈ 1,n do
        d_k = c_ik − v_k;
        if h > d_k then
            h = d_k; j = k;
    while true do
        for k ∈ 1,n do
            if s_k == w then
                d_k ← d_k − h;
            else
                v_k ← v_k − h;
                l = r_k; u_l ← u_l − h;
        u_i ← u_i + h; s_j = w; l = r_j;
        if l == 0 then break; ;
        j* = j; u* = u_l; h = ∞;
        for k ∈ 1,n do
            if s_k ≠ w then
                d_k = min{d_k, c_lk − u* − v_k};
                p_k = j* + w;
                if h > d_k then
                    h = d_k; j = k;
        while p_j > w do
            k = p_j − w; r_j = r_k; j = k;
        r_j = i;
```

Решение открытой или закрытой ЛЗН вида (1) реализуется по традиционной схеме:

```
function lap() begin
    w=0;
    for  j ∈ 1,n do
        r_j = 0;  s_j = w;  v_j = 0;  p_j = w;
    for  i ∈ 1,m do  hlap(i) ;
```

Однако после любого изменения элементов матрицы коэффициентов в (1) в строке i необходимо выполнить hlap(i), не повторяя этап инициализации. Это позволяет решать ЛЗН синхронно с поступлением заявок, а для всех имеющихся в портфеле заявок в любой момент времени иметь оптимальное паросочетание агентов и работ.

3. Улучшенный вариант алгоритма

Предлагаемый вариант реализации инкрементального алгоритма итераций метода кратчайшего пополняющего пути (Shortest Augmenting Path, SAP), лучшего среди известных методов решения (1)[4], имеет вид:

```
procedure slap(i) begin
    for  j ∈ 1,n do
        p_j = j;  d_j = c_ij − v_j;
    R = ∅;  S = ∅;  T = {1,n};
    while  true do
        if  |S| = 0 then
            h = min{d_j | j ∈ T};
            S = {j | (d_j = h) ∧ (j ∈ T)};
            T ← T\S;
            for  j ∈ S do
                if r_j == 0 then
                    goto back;
        k = S_1;  S ← S\{k};
        R ← R ∪ {k};  l = r_k;
        for  j ∈ T do
            if  h + c_lj − v_j < d_j then
                d_j = h + c_lj − v_j;  p_j = l;
                if  d_j == h then
                    if r_j == 0 then
                        goto back;
                    S ← S ∪ {j};
                    T ← T\{j};
    back: for  k ∈ R do
        v_k ← v_k + d_k − h;
    repeat
        l = p_j;  p_j = l;  k = j;  j = q_l;  q_l = k;
    until i == l;
```

Его преимущества особенно заметны для разреженных графов.

Заключение

Таким образом, вычислительная сложность поиска оптимального решения – $O\left(mn^{1/2}\log(nC)\right)$, где C – диапазон значений элементов матрицы ЛЗН.

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.

Список использованных источников

1. Gerkey, B. P. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems/ B.P. Gerkey, M.J. Mataric//The International Journal of Robotics Research, _2004. _Vol. 23, no. 9. _P. 939-954.

2. Spivey, M.Z. The Dynamic Assignment Problem/M.Z. Spivey, W.B. Powell//Transportation Science. _2004. _No. 4. _P. 399_419.

3. Toroslu, I.H. Incremental assignment problem/I.H. Toroslu, G. _U_coluk//Information Sciences._2007. _Vol.177. _P. 1523-_1529.

4. Jonker, R., Volgenant, A. A shortest path algorithm for dense and sparse linear assignment problem /R. Jonker½ A. Volgenant//Computing. _1987. _Vol. 38. _P. 325_340.