Math-Net.Ru

Общероссийский математический портал

С. В. Чернышев, Локальная оптимизация путей в задачах маршрутизации автотранспорта с временными окнами, *УМН*, 2009, том 64, выпуск 1, 165–166

DOI: <https://doi.org/10.4213/rm9229>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 3.85.73.92

26 апреля 2023 г., 11:04:33

Локальная оптимизация путей в задачах маршрутизации автотранспорта с временными окнами

С. В. Чернышев

В задачах маршрутизации автотранспорта одним из критериев оптимизации является уменьшение суммарных транспортных расходов [1]. В задачах такого типа транспортные расходы являются второстепенными параметрами оптимизации, поэтому их учет производится в заключительных фазах оптимизации. В данной работе приводится алгоритм минимизации транспортных расходов при фиксированном порядке посещения контрольных пунктов. Предполагается, что транспортные расходы можно выразить в виде линейной функции множества ребер, входящих в маршрут. Таким образом решается задача минимизации длины маршрута при заданных временных ограничениях.

Дано множество контрольных пунктов – набор вершин графа $\{v_0, \dots, v_k\}$, для каждой из которых задано время работы $[t_s(v_j), t_f(v_j)]$. Дано множество ребер E_j , соединяющих пару вершин (v_{j-1}, v_j) . Для каждого ребра e указана длина $d(e)$ и время $t(e)$. Необходимо построить путь минимальной длины, проходящий последовательно через вершины $\{v_0, \dots, v_k\}$ так, чтобы были выполнены ограничения по времени. Таким образом, нужно выбрать k ребер $\{e_1, \dots, e_k\}$, $e_j \in E_j$, так, чтобы были выполнены следующие условия:

$$\sum_{j=1}^k d(e_j) \rightarrow \min,$$

$$t_s(v_0) + \sum_{j=1}^i t(e_j) \in [t_s(v_i), t_f(v_i)], \quad i = \overline{1, k}.$$

Приведенные формулы не отражают того факта, что прибытие в контрольный пункт может быть раньше времени начала работы. На практике эта ситуация допустима и означает либо простой автотранспорта, либо снижение скорости движения на этом участке. Мы не будем заострять внимание на этом обстоятельстве, так как эта ситуация не влияет на суть излагаемого метода и может быть легко в нем учтена.

Корректным путем назовем путь, удовлетворяющий ограничениям по времени. Разобьем отрезок $[t_s(v_0), t_f(v_k)]$ на n равных отрезков точками

$$t_i = \frac{(n-i)t_s(v_0) + it_f(v_k)}{n}.$$

Чтобы по заданному значению t быстро вычислять индекс i такой, что $t_i \leq t < t_{i+1}$, введем функцию $\text{Index}(t)$:

$$\text{Index}(t) = \left\lceil \frac{n(t - t_0)}{t_n - t_0} \right\rceil.$$

Обозначим $d[i, j]$ длину кратчайшего корректного пути, соединяющего последовательно вершины $\{v_0, \dots, v_j\}$, время движения вдоль которого не превосходит t_i . Если указанного пути не существует, будем считать, что функция не определена. Значение функции будем вычислять динамически:

$$d[i, j] = \min_{e \in E_j} d[i'_e, j-1] + d(e), \quad i'_e = \text{Index}(t_i - t(e)).$$

Полученное рекуррентное соотношение означает, что оптимальный путь $\{v_0, \dots, v_j\}$ мы получаем из пути $\{v_0, \dots, v_{j-1}\}$, выбирая ребро $e \in E_j$ подходящим образом. Индекс i'_e выбирается однозначно для каждого ребра $e \in E_j$, так как $d[i, j]$ – невозрастающая функция по первому аргументу.

Алгоритм нахождения оптимальных путей

```

1  d[0, 0] = 0
2  for j = 1 to k do
3    for i = Index(ts(vj)) + 1 to Index(tf(vj)) do
4      for e ∈ Ej do
5        i' = Index(ti - t(e))
6        if (d[i', j - 1] ∈ ℝ) then
7          d[i, j] = min{d[i, j], d[i', j - 1] + d(e)}

```

Строки 4–7 алгоритма требуют $O(|E_j|)$ времени выполнения при фиксированных значениях i, j . Общее количество итераций по переменной i не превосходит n . Таким образом, общее время работы не превосходит

$$O\left(\sum_{j=1}^k \sum_{i=0}^n |E_j|\right) = O\left(\sum_{i=0}^n \sum_{j=1}^k |E_j|\right) = O(nl),$$

где $l = \sum_{j=1}^k |E_j|$ – общее количество ребер.

Для больших значений n функция $d[i, j]$ при фиксированном параметре j приобретает характер явно выраженной кусочно постоянной функции. Это обстоятельство позволяет хранить не все значения i , а только те, при которых происходит “скачок” функции. Пусть m – общее количество таких скачков. Тогда время работы алгоритма при использовании сбалансированных деревьев будет $O(ml \log m)$.

Уменьшение шага дискретизации позволяет получить решение задачи с любой наперед заданной точностью. На практике n нужно выбирать максимально большим, но так, чтобы этот этап оптимизации выполнялся достаточно быстро.

Список литературы

[1] Е. В. Панкратьев, А. М. Чеповский, Е. А. Черепанов, С. В. Чернышев, *Фундам. и прикл. матем.*, **9:1** (2003), 235–251; англ. пер.: E. V. Pankratiev, A. M. Chepovskii, E. A. Cherepanov, S. V. Chernyshev, *J. Math. Sci. (N. Y.)*, **128:6** (2005), 3487–3495.

С. В. Чернышев (S. V. Chernyshev)

Московский государственный университет

им. М. В. Ломоносова

E-mail: stcherny@adde.math.msu.su

Представлено А. В. Михалёвым

Принято редколлегией

20.06.2008