文章编号: 1003- 207(2003) 04- 0041- 06

网络图中多约束条件下的路径选择问题

贺铁林

(中国科学院科技政策与管理科学研究所, 北京 100080)

摘 要: 本文探讨了多约束条件下的路径选择问题。本文的工作在于对多约束条件下的路径选择问题进行了规范的描述和分类, 并针对各类问题讨论了典型的求解算法, 同时对现有算法进行了改进, 使其更适用于求解管理科学领域中的 MCP 问题, 从而拓展了该问题的应用空间。

关键词: 多约束条件; 最短路径算法; NP 完全问题; 启发式算法
中图分类号: O221.7 文献标识码: A

1 引言

在生产流水线管理、交通线路的调度、QoS (Quality of Service) 路由等应用场合下, 我们会面临很多需要考虑多约束条件的路径(Multi-constrained Paths, 简称 MCP) 选择问题。例如, 在进行生产流水线管理时, 我们在选择一条加工路线的时候, 不仅要考虑该路线总的加工时间, 还要考虑这条路线的优质品率。又如, 在进行多媒体通信的时候, 通常需要 QoS 路由的支持, 此时, 衡量一条路径的有效性一般会包含两个约束条件: 延时和带宽。为了方便讨论, 本文只考虑双约束条件下的路径选择问题。如果没有特别指出, 文中 MCP 问题即指双约束条件下的路径选择问题。

Dijkstra[1] 算法、Bellman-Ford[1] 算法作为计算最短路径的经典算法已为大家所理解和接受, 并得到了广泛的应用。但是这两种算法是为解决单约束条件问题而设计的, 并不能直接应用于以上描述的 MCP 问题。

本文根据约束条件对应的权函数性质把 MCP 问题划分Ⅰ类问题和Ⅱ类问题两种类型: Ⅰ类问题较容易求解, 可以通过改进的 Dijkstra 算法、Bellman-Ford 算法直接求解; Ⅱ类问题属于 NP 完全问题[2], 求解比较困难, 目前解决该类问题的通常思路是构造启发式算法[3][4][5], 即通过简化问题求取近似解。

收稿日期: 2002- 10- 18; 修订日期: 2003- 05- 30

作者简介: 贺铁林(1978-), 男, (汉族), 江西莲花人, 中国科学院科技政策与管理科学研究所, 2000 级硕士研究生, 研究方向: 管理科学与工程.

以往, 研究人员对 MCP 问题的关注较多集中在计算机科学领域内, 尤其是网络的 QoS 路由领域内[2][5][7]。本文的贡献在于对 MCP 问题进行了规范的描述和分类, 然后再针对各类问题讨论了典型的求解算法。同时对现有算法进行了改进, 使其更适用于求解管理科学领域中的 MCP 问题。重点讨论了适用于求解Ⅱ类问题的启发式算法。

接下来, 文章按以下结构进行组织: 第 2 部分是对约束条件对应的权函数性质的分析; 第 3 部分根据权函数的性质对 MCP 问题进行了描述和分类; 最后, 第 4 部分讨论了解决各类 MCP 问题的常用算法, 并对现有算法进行了改进。

2 权函数性质分析

本文把约束条件对应的权函数分成了三类: 单性权函数、加性权函数和乘性权函数。权函数的性质决定了路径的总权植与组成该路径各边分权值的关系。

单性权函数指的是, 对于某一路径, 其总权值等于组成该路径各边分权值中的最大值或最小值。生产流水线管理中工序的最大负荷是一个单性权函数, 因为整个流水线的最大负荷是由组成该流水线各工序最大负荷中的最小值决定的。

加性权函数指的是, 对于某一路径, 其总权值是由组成该路径各边的分权值相加而成。生产流水线管理中工序的加工时间就是一个典型的加性权函数。

乘性权函数指的是, 对于某一路径, 其总权值是由组成该路径各边的分权值相乘而成。生产流水线管理中工序的优质品率是一个典型的乘性权函数,

因为整个流水线的优质品率等于各工序优质品率的 乘积。

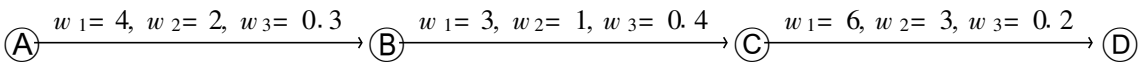


图 1 约束条件的性质

如图 1, 假设路径 $p = A \rightarrow B \rightarrow C \rightarrow D$, w_1, w_2, w_3 分别为单性(最小值)权函数、加性权函数和乘性权函数, 则相对于该路径:

$w_1(p) = \min\{w_1(A, B), w_1(B, C), w_1(C, D)\} = \min\{4, 3, 6\} = 3$

$w_2(p) = w_2(A, B) + w_2(B, C) + w_2(C, D) = 2 + 1 + 3 = 6$

$w_3(p) = w_3(A, B) * w_3(B, C) * w_3(C, D) = 0.3 * 0.4 * 0.2 = 0.024$

3 MCP 问题的分类

本文把 MCP 问题分成两类: Ⅰ类问题的两个约束条件中必须有一个约束条件对应的权函数是单性权函数, 另一个约束条件对应的权函数可以是加性权函数, 也可以是乘性权函数。例如, 生产流水线管理中当给流水线的最大负荷和加工时间同时设定约束条件时, 这个问题就属于Ⅰ类问题。因为工序的最大负荷是单性权函数, 而加工时间是加性权函数。

Ⅱ类问题的两个约束条件对应的权函数必须都是加性权函数或者乘性权函数, 不可以包括单性权函数。生产流水线管理中当给加工时间和优质品率都设定约束条件时, 就是一个典型的Ⅱ类问题。

本文第 4 节将详细讨论分别适用于这两类问题的求解算法。

4 MCP 问题的求解

4.1 适用于Ⅰ类问题的求解方法

我们可以使用改进的 Dijkstra 算法(Modified Dijkstra Algorithm)求解Ⅰ类问题。算法的基本思路是先把网络图中不满足相应于单性权函数约束条件的边去掉, 然后再在剩下的子图中应用 Dijkstra 算法计算出在另一约束条件(相应于加性权函数或乘性权函数)下的可行路径。参考文献[2]详细讨论了使用改进的 Dijkstra 算法求解Ⅰ类问题的步骤。

设 N 表示网络图中节点的数目, 我们可以证明改进的 Dijkstra 算法的计算复杂度为 $O(N^2)$[2], 和 Dijkstra 算法的计算得杂度相同。因此, 使用该算法我们可以在多项式时间内求解Ⅰ类问题。

需要说明的是, 使用改进的 Dijkstra 算法, 求得的解只是满意解, 并不一定是最优解。原因是我们在把网络图中不满足相应于单性权函数约束条件的边去掉时, 使用的是约束条件的最低限值, 假设这个值为 L_{min}。但事实上, 我们完全有可能在放松约束条件 L_{min} 的情况下(设定一个大于 L_{min} 的约束条件 L), 同样求出满足另外一个约束条件(相应于加性权函数或乘性权函数的约束条件) 的满意解。在不知道用户对约束条件偏好的情况下, 我们无法判断设定约束条件 L_{min} 和设定约束条件 L 求得的解中, 哪一组解更符合用户的偏好, 因此也就无法判断那一组解是最优解。

由于改进的 Dijkstra 算法能够在多项式时间内求出Ⅰ类问题的满意解, 而且算法的计算复杂度也不高, 因此该算法是目前求解Ⅰ类问题的常用算法。

4.2 适用于Ⅱ类问题的求解方法

4.2.1 算法描述

Ⅱ类问题属于 NP 完全问题[2]。由于 NP 完全问题目前找不着可以在多项式时间内求解的算法, 通常我们会构造一些与原问题类似的问题, 这些问题能够在多项式时间内求解, 而且又不失原问题的重要特征, 通过求解这些问题我们可以得到原问题的近似解。

下面将要讨论的启发式算法借鉴了参考文献[3]中提出的启发式算法的思想。由于参考文献[3]中提出的启发式算法在原问题有解的情况下, 很有可能得出原问题无解的结论, 这点在管理科学领域中通常是不能接受的。本文对该算法进行了改进, 使其可以确保在原问题有解的情况下能求出原问题的解/近似解。在这里, 近似解表示该解有可能是原问题的解, 也有可能虽然不是原问题的解, 但是原问题解的一个近似。下面, 我们通过一个典型的问题来描述这个算法。

假设一个有向连通图 $G(V, E)$, 设定一个乘性权函数 $w_1: E \rightarrow \mathbf{R}^+$, 一个加性权函数 $w_2: E \rightarrow \mathbf{R}^+$, 两个常数 $C_1 \in \mathbf{R}^+$, $C_2 \in \mathbf{R}^+$。

问题: 定义问题 MCP$(G, s, t, w_1, w_2, c_1, c_2)$

为寻找一条从 s 至 t 的路径 p, 使得 $w_1(p) \leqslant c_1$, $w_2(p) \leqslant c_2$。

由第 3 节的定义, 我们知道该问题属于 Ⅱ 类问题。下面用启发式算法求出该问题的解/近似解。算法包含两个步骤:

Step1: 创建一个新的权函数 $w'_2: E \to Z$

$$w'_2(u, v) = \left\lceil \frac{w_2(u, v) \cdot x}{c_2} \right\rceil \tag{4}$$

其中 x 是一个给定的正整数, $\lceil\ \rceil$ 为向上取整符号。

Step2: 构造两个新的问题 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x)$ 和 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x + N - 1)$, 其中 N 为图 G 中顶点的数目。由 F 面给出的两个定理, 我们知道通过求解这两个问题, 我们可以求出原问题的解/近似解。

定理 1: 问题 $\mathrm{MCP}(G, s, t, w_1, w_2, c_1, c_2)$ 的解必定也是问题 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x + N - 1)$ 的解。

证明: 假设路径 p 是原问题 $\mathrm{MCP}(G, s, t, w_1, w_2, c_1, c_2)$ 一个解, 有 $w_1(p) \leqslant c_1$, $w_2(p) \leqslant c_2$。因此, 要证明 p 也是问题 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x + N - 1)$ 的解, 只需证明 $w'_2(p) \leqslant x + N - 1$。

首先, 由于本问题考虑的图中所有边的权值都是正数, 因此从 s 至 t 的最短路径最多含有 $N - 1$ 条边。根据等式 4 有 $w'_2(u, v) \leqslant \frac{w_2(u, v) \cdot x}{c_2} + 1$, 于是

$$w'_2(p) = \sum_{(u, v) \in p} w'_2(u, v) \leqslant \sum_{(u, v) \in p} \left(\frac{w_2(u, v) \cdot x}{c_2} + 1 \right)$$

$$\leqslant \sum_{(u, v) \in p} \frac{w_2(u, v) \cdot x}{c_2} + N - 1 = \frac{w_2(p) \cdot x}{c_2} + N - 1$$

$$\leqslant \frac{c_2 \cdot x}{c_2} + N - 1 = x + N - 1$$

$w'_2(p) \leqslant x + N - 1$, 定理 1 成立。

定理 2: 问题 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x)$ 的解必定也是问题 $\mathrm{MCP}(G, s, t, w_1, w_2, c_1, c_2)$ 的解。

证明: 假设路径 p 是问题 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x)$ 的一个解, 有 $w_1(p) \leqslant c_1$, $w'_2(p) \leqslant x$。因此, 要证明 p 也是问题 $\mathrm{MCP}(G, s, t, w_1, w_2, c_1, c_2)$ 的解, 只需证明 $w_2(p) \leqslant c_2$。

根据等式 4 有 $w_2(u, v) \leqslant \frac{w'_2(u, v) \cdot c_2}{x}$, 于是

$$w_2(p) = \sum_{(u, v) \in p} w_2(u, v) \leqslant \sum_{(u, v) \in p} \frac{w'_2(u, v) \cdot c_2}{x}$$

$$= \frac{c_2}{x} \leqslant \sum_{(u, v) \in p} w'_2(u, v) = \frac{c_2 \cdot w'_2(p)}{x} \leqslant \frac{c_2 \cdot x}{x} = c_2$$

$w_2(p) \leqslant c_2$, 定理 2 成立。

接下来, 我们采用参考文献[3]中给出的扩展 Dijkstra 算法(EDSP) 在多项式时间内求解问题 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x + N - 1)$, 该算法可以同时计算出满足问题 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x)$ 的解。算法基于以下原理:

由于 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x + N - 1)$ 问题中权函数 w'_2 的取值始终是整型值, 这样对于每一个顶点 $v \in V$, 从 s 至 v 的路径的 w'_2 权值总和 k 必定也是整数, 当 $k \leqslant x + N - 1$ 时, 该路径 w'_2 权值总和满足约束条件 $x + N - 1$。实际上, 在所有从 s 至 v 的路径中, w'_2 权值总和满足约束条件 $x + N - 1$ 的路径不一定是唯一的。但由于 0 至 $x + N - 1$ 的整数集 $[0..x + N - 1]$ 是一个有限的空间, 对于任意整数 $k \in [0..x + N - 1]$, EDSP 算法可以在多项式时间内计算出所有从 s 到 v 的 w'_2 权值总和 k 的可能路径。

同时, 由于对于每一个 k 值, 从 s 至 v 的路径中满足 w'_2 权值总和为 k 的路径不一定是唯一的, EDSP 算法应用了与 Dijkstra 算法类似的思路, 在多项式时间内从这些路径中计算出具有最小 w_1 权值总和(假设为 d)的路径。如果 d 小于 c_1, 那么该路径就是问题 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x + N - 1)$ 的一个可行解, 因为其 w'_2 权值总和 $k \leqslant x + N - 1$, w_1 权值总和 $d \leqslant c_1$。如果该路径 w'_2 权值总和满足 $k \leqslant x$。那么该路径同时也是问题 $\mathrm{MCP}(G, s, t, w_1, w'_2, c_1, x)$ 的解。

在图 2 中, 我们使用伪代码描述了 EDSP 算法。算法开始时会调用 Intialize(G, s), 对算法中需要用到的变量进行初始化, 接下来运行一个 while 循环, 在这个循环中会调用到函数 Relax(u, k, v), 该函数会判断路径是否满足约束条件 $x + N - 1$, 并保留问题所有可能解的路径信息在变量 π 中。

如图 2 所示, 对于每一个顶点 $v \in V$ 和每一个整数 $k \in [0..x + N - 1]$, 当从 s 至 v 的路径的 w'_2 权值总和为 k 时, $d[v, k]$ 表示所有这些路径中 w_1 权值总和最小的估计。假设:

$$\delta(v, k) = \min_{p \in P(v, k)} \{w_1(p)\} \tag{5}$$

其中, $P(v, k) = \{p \mid w'_2(p) = k, p$ 是从 s 至 v 的任何路径$\}$。

最初, $d[v, k]$ 被设为无穷大(∞), 接下来在算

法运行的过程中 $d[v, k]$ 将逐渐接近并最终等于 $\delta(v, k)$。在算法结束时, $d[v, k] = \delta(v, k), v \in V, k \in [0..x + N - 1]$。

在图 2 中, EDSP 算法的实现需要一个集合 Q:

$$Q = \{\langle v, k\rangle \mid d[v, k] > \delta(v, k), v \in V, k \in [0..x + N - 1]\} \quad (6)$$

其中符号 $\langle v, k\rangle$ 表示一个 $v \in V, k \in [0..x + N - 1]$ 的二元组。

```
        Initialize (G, s)
        Begin
            /* V [G] 表示图 G 中顶点的集合, [0.. x+ N- 1] 表示 0 至 x 的所有整数集合* /
(1)         for each v∈ V [G], each i∈ [0.. x+ N- 1] do
(2)             d [v, i] := ∞
(3)             π [v, i] := NIL
(4)         for each i∈ [0.. x+ N- 1] do
(5)             d [s, i] := 0
        end
        Relax (u, k, v)
        begin
(6)             k' := k+ w'2 (u, v)
(7)             if k' ≤x+ N- 1 then
(8)                 if d [v, k'] > d [u, k] * w1 (u, v) then
(9)                     d [v, k'] := d [u, k] * w1 (u, v)
(10)                π [v, k'] := u
        end
        EDSP (G, s)
        begin
(11)    Initialize (G, s)
(12)    Q:= { ⟨u, k⟩ | u∈ V [G], k∈ [0.. x+ N- 1]}
(13)    while Q≠ Ø do
(14)            find (u, k) ∈ Q such that d [u, k] = min_{(u',k')∈Q} {d [u', k']}
(15)            Q:= Q- { ⟨u, k⟩}
                /* 下面的 for 循环遍历节点 u 的所有邻居节点v. * /
(16)            for each outgoing edge of u, (u, v) ∈E do
(17)                Relax (u, k, v)
        end
```

图 2 EDSP 算法的伪码描述

如图 2 所示, 最初 $Q = \{\langle u, k\rangle \mid u \in V[G], k \in [0..x + N - 1]\}$, 接下来在算法的 13- 16 行, 每一次 while 循环都会从 Q 中删除一个元素, 当 $Q = \emptyset$, 算法终止。集合 Q 的使用体现了算法对权函数进行取整的意义: 在图 2 中第 6 行, 算法会设定从 s 到 v 路径的 w'_2 权值总和, 也就是 k' 的值, 由 w'_2 是整数型权函数, 我们知道 k' 必定也是整数, 这样, 加上算法第 7 行的判断我们就可以保证第 8- 9 行中 $d[v, k']$ 相应的二元组 $\langle v, k'\rangle$ 依旧会属于集合 Q。能够保证这一点, $d[v, k]$ 的取值才有意义。

根据上面的论述, 在算法结束时, 我们可以通过下面三个推论来求解原问题。

推论 1: 如果存在 $k \in [0..x]$, 使得 $d[t, k] \leqslant c_1$, 那么一定存在一条从 s 至 t 的路径 p 使得 $w_1(p) \leqslant c_1, w'_2(p) \leqslant x$。这时, 路径 p 就是问题 MCP $(G, s, t, w_1, w'_2, c_1, x)$ 的一个可行解。根据定理 1, 路径 p 同时也是原问题的解。

推论 2: 如果存在 $k \in [x..x + N - 1]$, 使得 $d[t, k] \leqslant c_1$, 那么一定存在一条从 s 至 t 的路径 p 使得 $w_1(p) \leqslant c_1, w'_2(p) \leqslant x + N - 1$。这时, 路径 p 是问题 MCP $(G, s, t, w_1, w'_2, c_1, x + N - 1)$ 的一个可行解, 但不是问题 MCP $(G, s, t, w_1, w'_2, c_1, x)$ 的可行解。根据定理 1、2 我们无法判断该路径是否是原问题的可行解。在这种情况下, 本文把路径 p 作为原问题的一个近似解。

推论 3: 如果对于所有 $k \in [0..x + N - 1]$, $d[t, k] > c_1$, 那么问题 MCP $(G, s, t, w_1, w'_2, c_1, n + N - 1)$ 无解。根据定理 2, 原问题也无解。

其中, 对于每一条可行路径 p, 我们可以通过变量 π 找到这条路径的完整信息。

EDSP 算法的计算复杂度为 $O((x + N)^2 V^2)$。如图 2 所示, 算法中集合 Q 的最大模数为 $(x + N)$

V, 因此, 不考虑 while 循环, 第 14 行的计算能够在 $O((s+N)V)$ 时间内完成。而 while 最多循环 $(x+N)V$ 次, 所以第 14 行总的计算时间复杂度为 $O((x+N)^2V^2)$。另外, 对于每一条边 $(u, v)\in E$, $k\in[0..x+N-1]$, 函数 Relax(u, k, v) 都会被调用一次, 因此第 16- 17 行的 for 总共会循环 $(x+N)E$ 次, 而函数 Relax(u, k, v) 的计算时间复杂度为 $O(1)$, 所以算法中这部分代码的计算时间复杂度为 $O((x+N)E)$。于是, EDSP 算法总的计算时间复杂度为 $O((x+N)^2V^2+(x+N)E)=O((x+N)^2V^2)$。

在上面的讨论中, 原问题中两个约束条件相应的权函数一个是乘性权函数, 另一个是加性权函数。对于其他类的Ⅱ类问题, 我们只需在算法的第 6 行和 9 行把加号"+"和乘号"*"作相应的变化, EDSP 算法同样适用。因此, 本文提出的启发式算法是适用于求解各种Ⅱ类问题的算法。

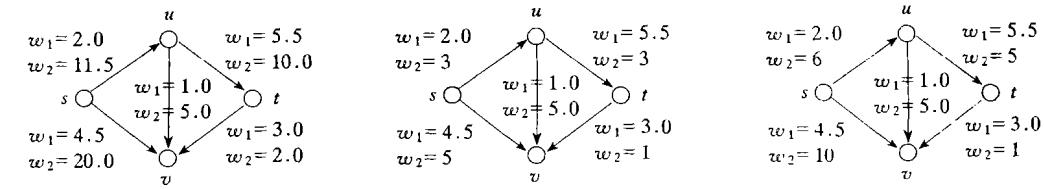
本文提出的启发式算法的意义在于它通过求整构造了两个与原问题类似的问题, 通过求解这两个问题我们可以在多项式时间内求出原本属于 NP 完全问题的原问题的近似解。求解Ⅱ类问题的启发式算法并不是唯一的, 参考文献[4]、[5]也提出了各自的启发式算法, 但这些算法的讨论都是针对 QoS 领域中的特定问题, 在管理科学领域中不具有良好的适用性。

4.2.2　算例分析

在这一节中, 我们将通过具体的算例来验证本文 4.2.1 节论述的启发式算法。我们设计了两个算例, 一个针对原问题有解的情况, 另外一个针对原问题无解的情况。

先讨论原问题有解的情况。如图 3(a)所示, 原问题 MCP$(G, s, t, w_1, w_2, 7.0, 20.0)$ 的解为 $s\to u\to v\to t$。我们使用 4.2.1 论述的启发式算法来求解该问题, 先设 $x=5$, 如图 3(b) 所示。根据算法的要求, 我们需要求解一个新的问题: MCP$(G, s, t, w_1, w_2', 7.0, 8)$。我们知道, 在使用 EDSP 算法求解问题 MCP$(G, s, t, w_1, w_2', 7.0, 8)$ 的同时, 我们可以求解问题 MCP$(G, s, t, w_1, w_2', 7.0, 5)$。在本算例中求解的结果为: 问题 MCP$(G, s, t, w_1, w_2', 7.0, 8)$ 有解, 其解为 $s\to u\to v\to t$, 但是问题 MCP$(G, s, t, w_1, w_2', 7.0, 5)$ 无解。根据本文 4.2.1 节推论 2, 此时, 我们把解 $s\to u\to v\to t$ 作为原问题的一个近似解。实际上, 这个近似解就是原问题的解。

在图 3(c)中, 我们设 $x=10$, 同样使用 EDSP 算法求解得到: 问题 MCP$(G, s, t, w_1, w_2', 7.0, 10)$ 有解, 其解为 $s\to u\to v\to t$。根据本文 4.2.1 节推论 1, 我们可以确定 $s\to u\to v\to t$ 就是原问题的解。

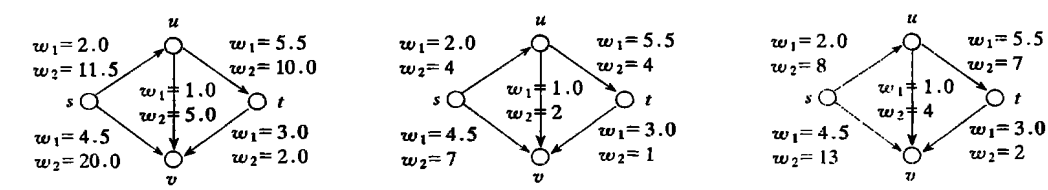


(a) MCP$(G, s, t, w_1, w_2, 7.0, 20.0)$　(b) MCP$(G, s, t, w_1, w_2', 7.0, 8)$　(c) MCP$(G, s, t, w_1, w_2', 7.0, 13)$

图 3　原问题有解的情况

再计论原问题无解的情况。如图 4(a)所示, 原问题 MCP$(Gs, t, w_1, w_2, 7.0, 15.0)$ 无解。设 $x=5$, 如图 4(b)所示, 使用 EDSP 算法求解得到: 问题 MCP$(G, s, t, w_1, w_2', 7.0, 5)$ 无解, 但是问题 MCP$(G, s, t, w_1, w_2', 7.0, 8)$ 有解, 其解为 $s\to u\to v\to t$。根据本文 4.2.1 节推论 2, 此时, 我们把 $s\to u\to v\to t$ 作为原问题的一个近似解。事实上, 这个近似解不是原问题的解。

设 $x=10$, 如图 4(c)所示, 使用 EDSP 算法求解得到: 问题 MCP$(G, s, t, w_1, w_2', 7.0, 13)$ 无解, 根据本文 4.2.1 节推论 3, 我们可以确定原问题无解。



(a) MCP$(G, s, t, w_1, w_2, 7.0, 15.0)$　(b) MCP$(G, s, t, w_1, w_2', 7.0, 8)$　(c) MCP$(G, s, t, w_1, w_2', 7.0, 13)$

图 4　原问题无解的情况

从上面两个算例的讨论我们可以看出: 当 x 取值较大时, 如 x = 10, 我们使用启发式算法求得的解与原问题的解是一致的; 而当 x 取值较小时, 如 x = 5, 我们使用启发式算法求得的近似解与原问题的解不一致。这较好的说明了本文论述的启发式算法的特点: 在使用该启发式算法求解时, 当 x 取值较大, 求得的解比较接近于原问题的真实解; 否则, 我们只能得到原问题的近似解, 这个近似解有可能是原问题的解, 也很有可能不是原问题的解。

但是, 根据本文 4.2 节对算法计算复杂度的分析, 我们知道, x 的值越大, 算法的计算时间复杂度也就越高。因此, 在实际求解时, 我们在选取 x 值的时候, 要在求解成功率和计算复杂度之间作一个权衡。通常, 我们会根据节点的数目以及实际问题对解精确度的要求选择一些经验值。

5 结语

本文探讨了多约束条件下的路径选择问题(MCP 问题)。文章首先根据约束条件对应的权函数性质对 MCP 问题进行了描述和分类, 把 MCP 问题分为 I 类问题和 II 类问题。接下来, 在第 4 部分我们讨论了适用于解决这两类问题的典型算法。重点讨论了适用于求解 II 类问题的启发式算法, 在这个算法中参数 x 的选取很关键, x 越大, 算法求解的成功率就越高, 求得的解也越理想, 但同时计算的时间复杂度也越高。如何在提高算法计算成功率的同时又保持较低的计算复杂度, 是 II 类问题求解下一步研究中关注的重点。

参考文献:

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Introduction to Algorithms(算法导论)[M]. The MIT Press and 高等教育出版社, 2002.

[2] Z. Wang and J. Crowcroft, QoS Routing for Supporting Resource Reservation[J]. IEEE Journal on Selected Areas in communication, September, 1996.

[3] S. Chen and K. Nahrstedt, On Finding Multi-Constrained Paths[C]. IEEE ICC' 98, June 1998.

[4] T. Korkmaz, M. Krunz, Multi-Constrained Optimal Path Selection[C]. IEEE INFOCOM' 01, April 2001.

[5] X. Yuan, X. Liu, Heuristic Algorithms for Multi-Constrained Quality of Service Routing[C]. IEEE INFOCOM' 01, April 2001.

[6] David Eppstein, Finding the k Shortest Paths[C]. IEEE Symposium on Foundations of Computer Science, August 1998.

[7] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, Multicast routing for multimedia communication[J]. IEEE/ACM Trans. Networking, June 1993.

[8] 余祥宣, 崔国华, 邹海明. 计算机算法基础[M]. 华中理工大学出版社, 1998.

On Multi-Constrained Paths in Networks

HE Tie-lin

(Institute of Policy and Management, Chinese Academy of Sciences, Beijing 100080, China)

Abstract: In this paper, we discussed the multi-constrained path(MCP) finding problem. The contributions of this paper lie in illustrating and classifying the MCP problem normatively. And for each kind of problem, it discusses typical algorithms. At the same time, the algorithms are extended to be more appropriate in solving MCP problem in the areas of management sciences.

Key words: multi-constraints; shortest path finding algorithms; NP-Complete; heuristic algorithms