现代管理科学方法
(第9讲)

郭仁拥 博士/教授/博导

讲授内容

1. 求解CVRP的启发式方法

1. 求解CVRP的启发式方法

求解VRP的精确算法（Exact algorithm）一般是基于TSP的精确算法来设计的

仅仅较小规模算例（例如，包含100个顾客和5辆车）能被优化求解

常被应用到实际当中的启发式方法

- 构造启发式方法
- 两阶段方法：先分组后定路线（Cluster First – Route Second）；先定路线后分组（Route First – Cluster Second
- 改进启发式方法
- 元启发方法

构造启发式方法

- 顺序方式：一次构造一个旅行（tour）
- 并行方式：并行构造多个旅行

两个主要技术

- 合并存在路线（节约准则）
- 将顾客分配到路线上（插入费用准则）

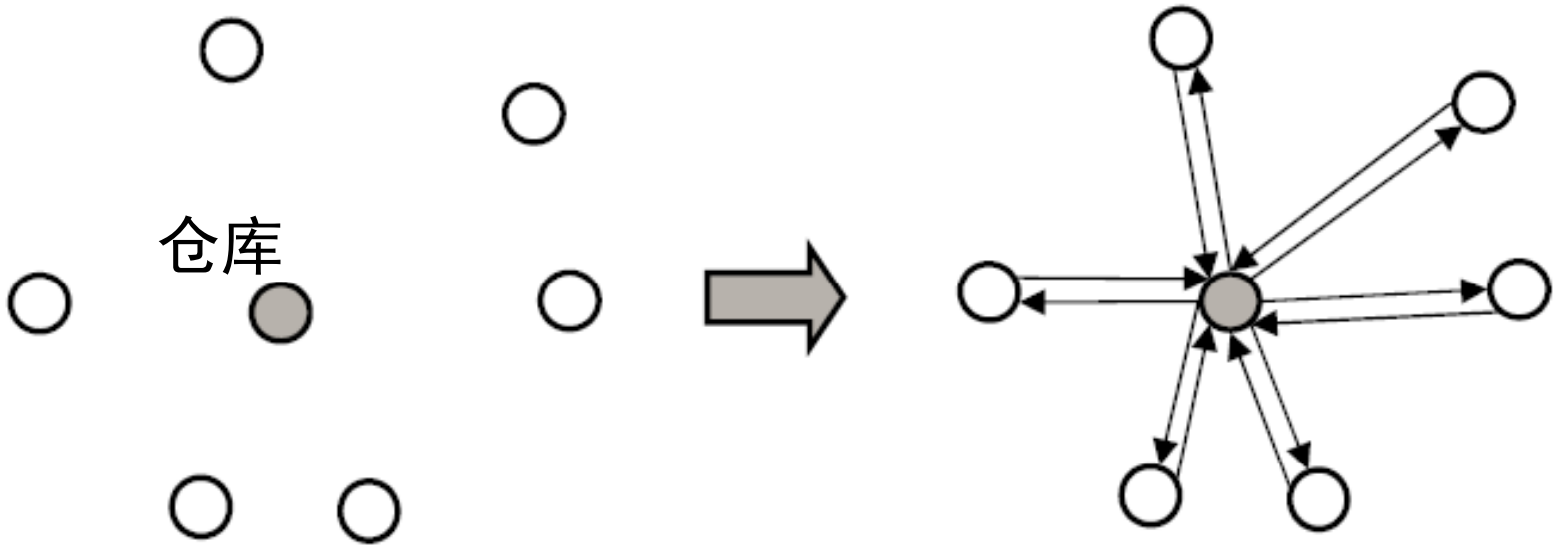
Clark-Wright节约算法

- 顺序版本和并行版本
- 可应用于车辆数是决策变量的情形
- 合并路线对，将一条路线终点与另一条路线起点相连，最大化合并后的距离节约

Clark-Wright节约算法（顺序版本）

初始化

- 对于每个顶点$i=1,2,\cdots,n$，产生一个路线$(0,i,0)$
- 对于每个顶点i，计算费用节约$s_{ij}=c_{i0}+c_{0j}-c_{ij}$，对于$j=1,2,\cdots,n$且$j\neq i$，以非增顺序排列这些费用节约
- 定义路线$(0,1,0)$作为当前路线

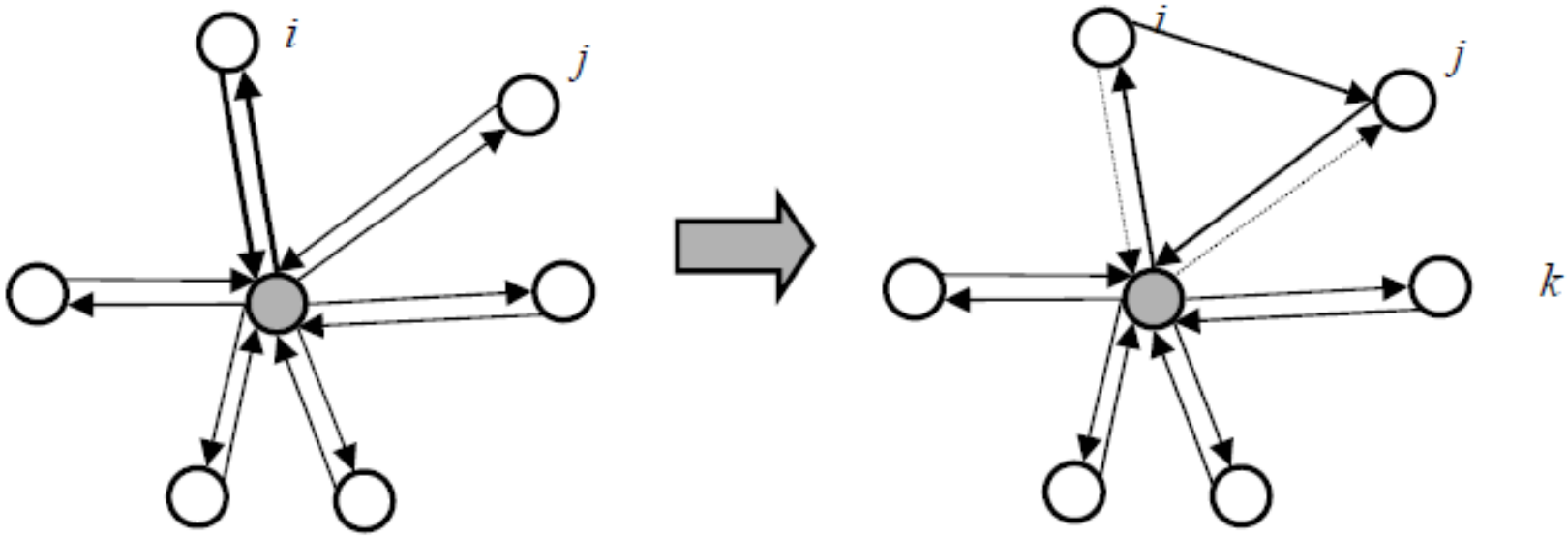


迭代（顺序版本：路线扩展）

- 设路线 $(0, i, \cdots, j, 0)$ 是当前路线
- 确定第一个节约 s_{ki} 或 s_{je}，使得当前路线能与另一条包含弧 $(k, 0)$ 或 $(0, e)$ 的路线可行合并
- 针对当前路线，如果一个可行合并存在，则实施合并
- 否则
  - 如果当前还存在其他未被考虑的路线，则从中选择一条作为当前路线并进行迭代
  - 否则停止（没有更多的可行合并存在）

一个例子

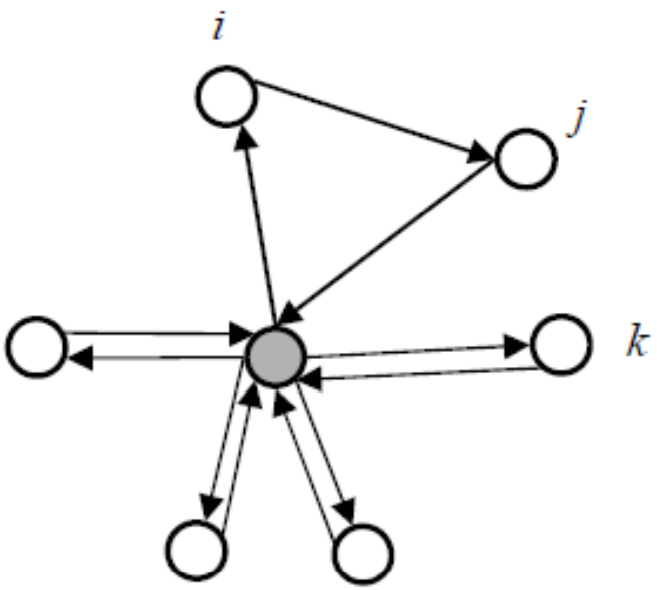
1）当前路线$(0,i,0)$

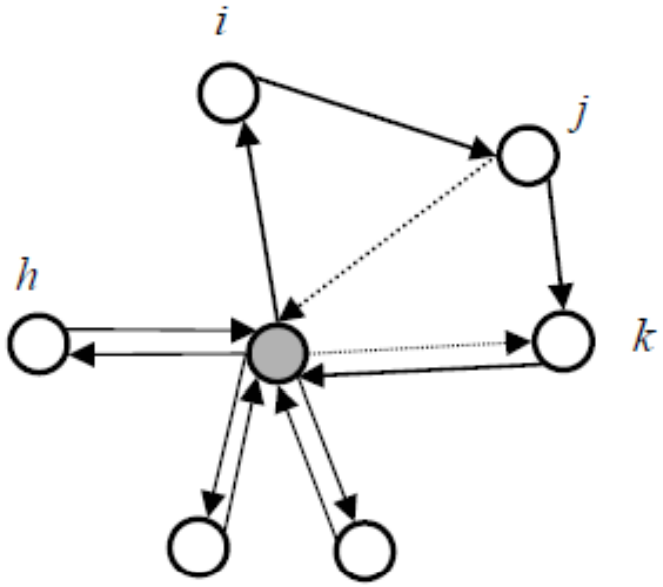


1）最好节约

2）当前路线$(0,i,j,0)$

$$s_{ij} = c_{i0} + c_{0j} - c_{ij}$$





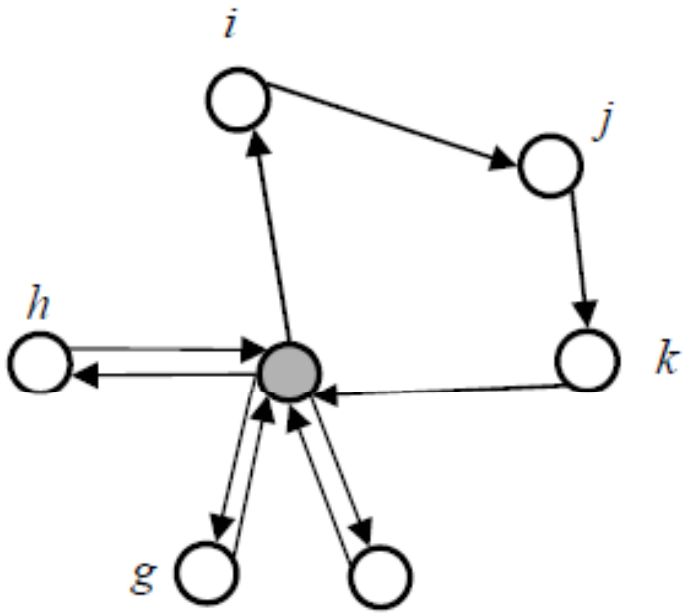
2）当前路线$(0,i,j,0)$

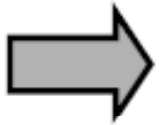
2）最好节约

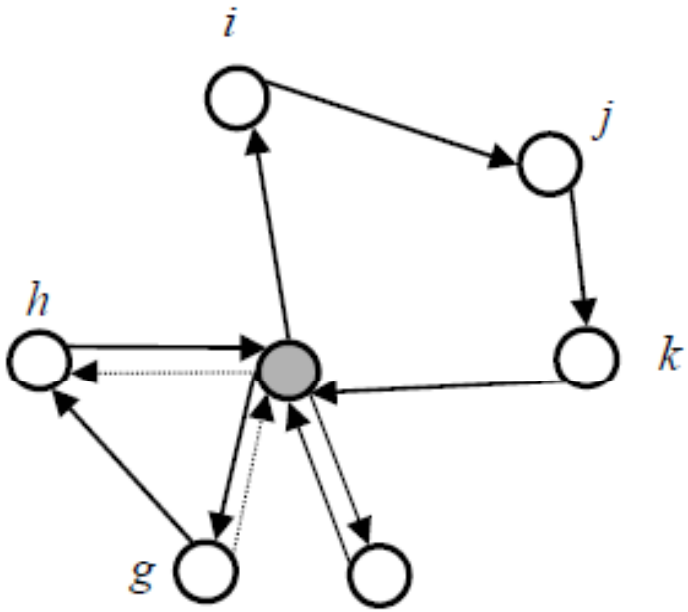
$$s_{jk} = c_{j0} + c_{0k} - c_{jk}$$

3）当前路线$(0,i,j,k,0)$不能被进一步可行扩展

下一个当前路线是$(0,h,0)$







3）当前路线$(0,h,0)$

3）最好节约

$$s_{gh} = c_{g0} + c_{0h} - c_{gh}$$

4）当前路线$(0,g,h,0)$

进一步的合并不可行

一个3-路线解被得到

Clark-Wright节约算法（并行版本）

初始化

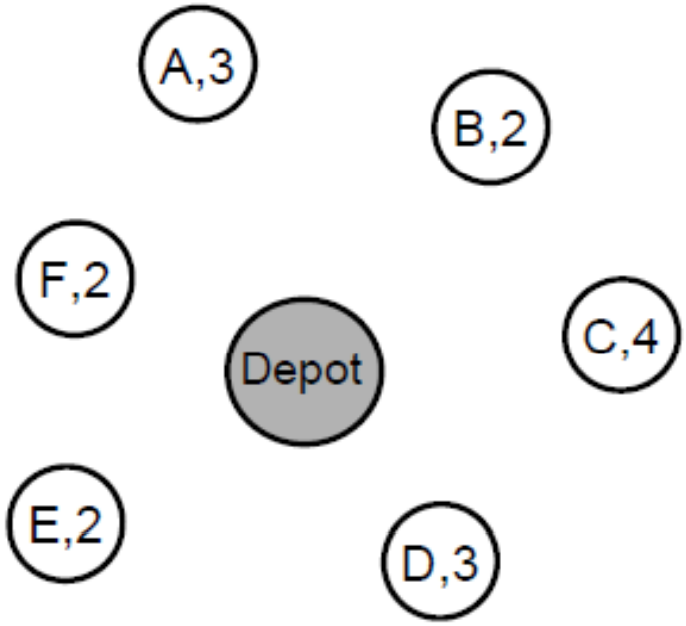
- 对于每个顶点 $i = 1, 2, \cdots, n$，产生一个路线 $(0, i, 0)$
- 对于每个顶点 i，计算费用节约 $s_{ij} = c_{i0} + c_{0j} - c_{ij}$，对于 $j = 1, 2, \cdots, n$ 和 $j \neq i$，以非增顺序排列这些费用节约

迭代（并行版本：最好可行合并）

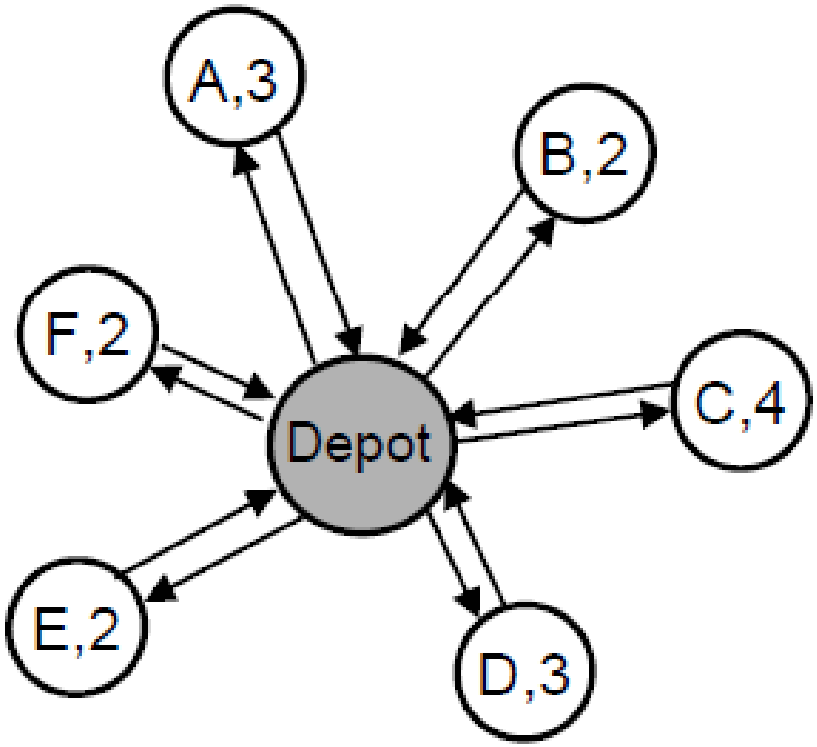
- 如果一个正费用节约存在，则
  - 从最大费用节约 s_{ij} 开始，确定是否包含弧 $(i, 0)$ 的路线和包含弧 $(0, j)$ 的路线可以被可行合并
  - 如果可以，删除 $(i, 0)$ 和 $(0, j)$，合并这两条路线，然后检验下一个费用节约

一个例子

- 对称欧式距离
- 车辆运送能力$C=8$
- 顾客需求在顶点上



c_{ij}	Dep	A	B	C	D	E	F
Dep	0	64	58	54	41	58	41
A	64	0	70	95	103	81	40
B	58	70	0	36	92	113	81
C	54	95	36	0	72	112	91
D	41	103	92	72	0	61	71
E	58	81	113	112	61	0	41
F	41	40	81	91	71	41	0



初始路线和长度

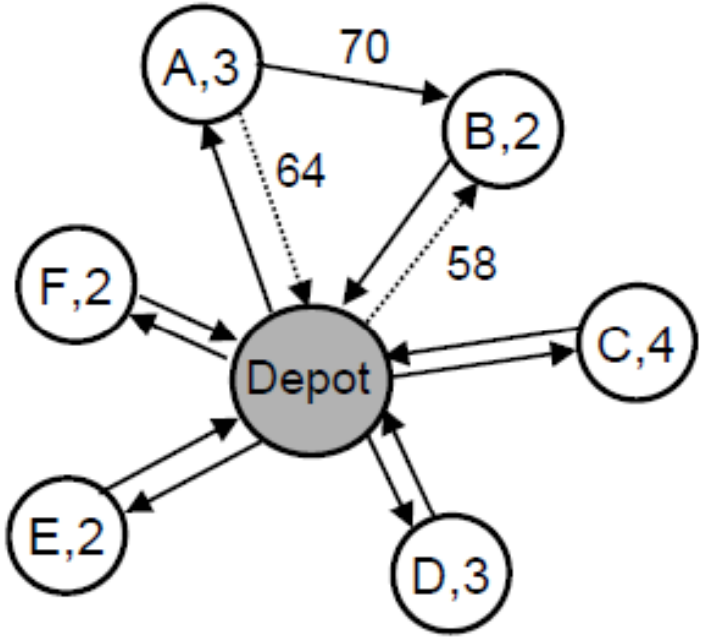
- c(0-A-0) = 128
- c(0-B-0) = 116
- c(0-C-0) = 108
- c(0-D-0) = 82
- c(0-E-0) = 116
- c(0-F-0) = 82

总费用=632

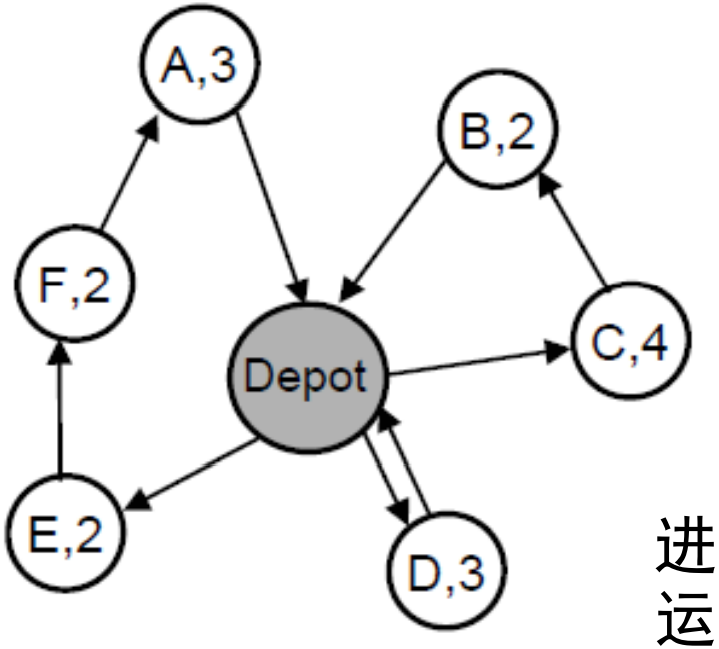
车辆数 = 6

$$s_{AB} = c_{ADep} + c_{DepB} - c_{AB}$$

$$= 64 + 58 - 70 = 52$$



s_{ij}	A	B	C	D	E	F
A	0	52	23	2	41	65
B	52	0	76	7	3	18
C	23	76	0	23	0	4
D	2	7	23	0	38	11
E	41	3	0	38	0	58
F	65	18	4	11	58	0

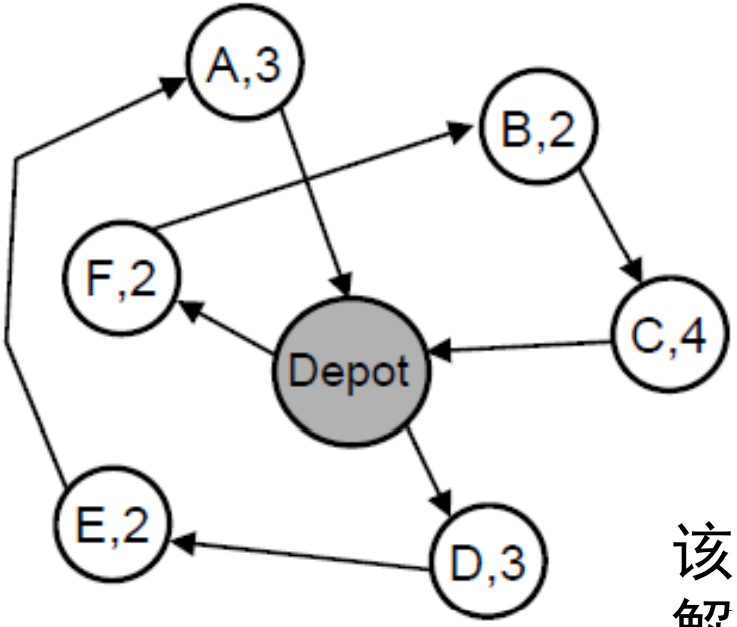


s_{ij}	A	B	C	D	E	F
A	0	52	23	2	41	65
B	52	0	76	7	3	18
C	23	76	0	23	0	4
D	2	7	23	0	38	11
E	41	3	0	38	0	58
F	65	18	4	11	58	0

进一步的合并不可行（否则违反运送能力约束）

3辆车

总费用 = 433



s_{ij}	A	B	C	D	E	F
A	0	52	23	2	41	65
B	52	0	76	7	3	18
C	23	76	0	23	0	4
D	2	7	23	0	38	11
E	41	3	0	38	0	58
F	65	18	4	11	58	0

该启发式方法不能找到一个2-车辆解（总费用 = 459）

两阶段方法：先分组后定路线

阶段1：分组

- 一个分组问题被求解，给每个顾客分配一辆车

阶段2：定路线

- 找出每辆车的路线（求解一个TSP）

方法：

基本分组方法

- 扫描算法（Sweep Algorithm）；Fisher-Jaikumar广义分配（GA）算法；基于位置的启发式方法

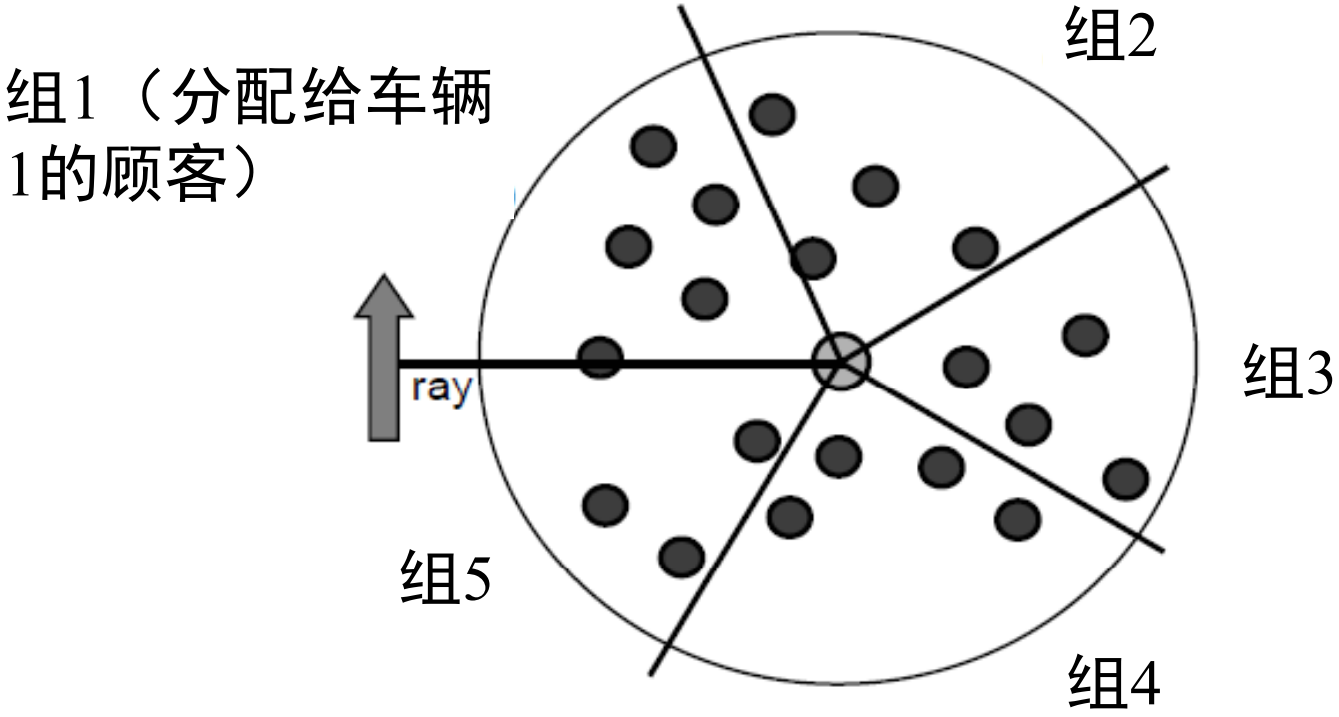
截断分支定界方法

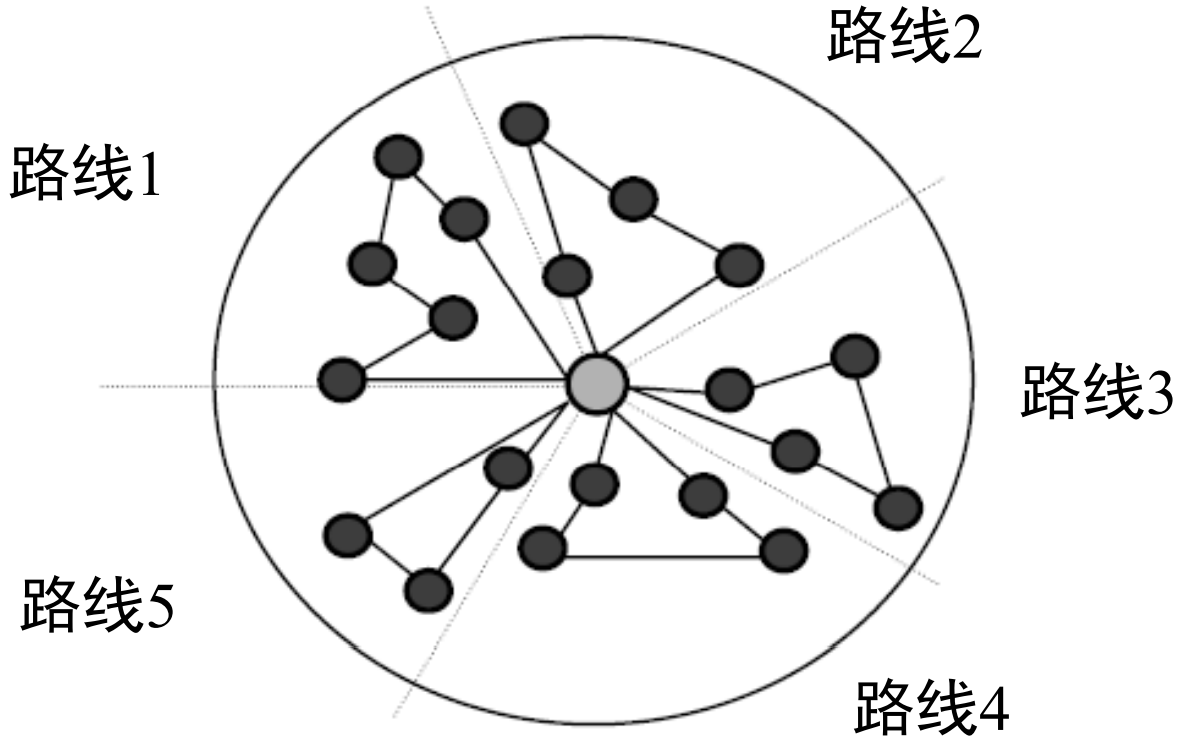
- 搜索树层次⇒车辆路径；每层包含一个由单/多准则（例如，费用节约）产生的可行路线集合；分支⇒路线选择

花瓣算法（Petal Algorithm）

扫描算法

- 平面VRP
- 旋转一条以仓库为中心的射线，得到可行分组
- 对于每组顾客，由求解一个TSP得到一条车辆路线




路线2
路线1
路线3
路线5
路线4

Fisher-Jaikumar基于广义分配算法

第1步（选种）：对于每组顾客$k=1,2,\cdots,K$，选择一个种子顶点j_k（$\in V$）

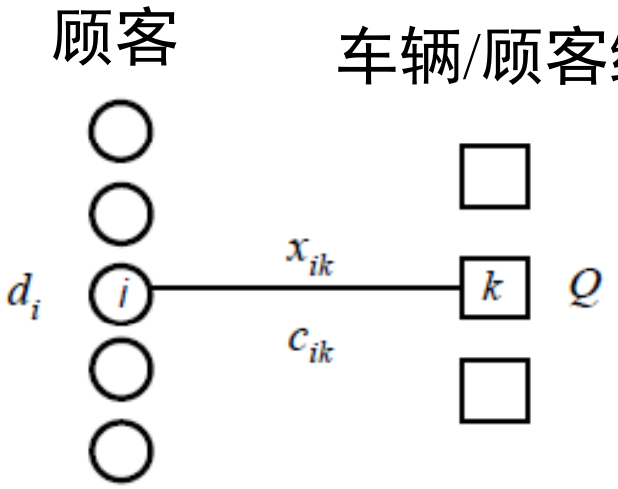
第2步（将顾客分配到种子顶点）：计算分配顾客i到k的费用c_{ik}作为将i插入路线$0-j_k-0$的费用，即

$$c_{ik}=\min\left\{c_{0i}+c_{ij_k}+c_{j_k0},c_{0j_k}+c_{j_ki}+c_{i0}\right\}-\left(c_{0j_k}+c_{j_k0}\right)$$

第3步（广义分配）：求解一个GA问题（具有费用c_{ij}、顾客权重d_i、车辆运送能力Q）

第4步（TSP解）：对于得到的每组顾客，求解一个TSP

广义分配问题（GAP）



$$\min \sum_{i=1}^{n}\sum_{k=1}^{K} c_{ik} \cdot x_{ik}$$

$$(a) \sum_{k=1}^{K} x_{ik} = 1 \qquad \forall i = 1,\ldots,n$$

$$(b) \sum_{i=1}^{n} d_i \cdot x_{ik} \le Q \qquad \forall k = 1,\ldots,K$$

$$x_{ik} \in \{0,1\} \qquad \forall i = 1,\ldots,n; k = 1,\ldots,K$$

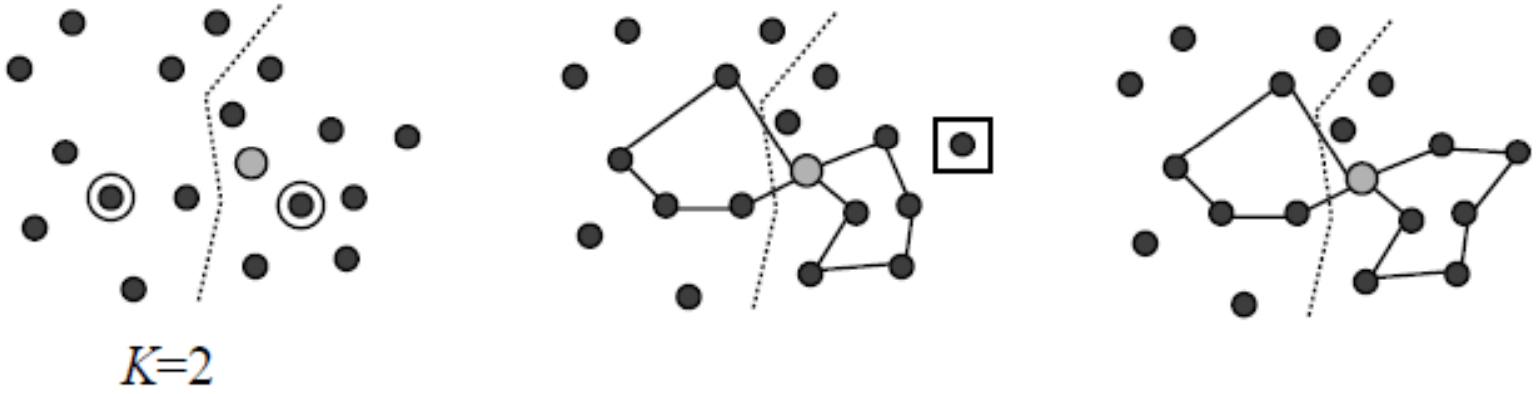
(a) 分配约束
(b) 运送能力约束

GAP是一个强NP难题，精确算法和启发式算法都存在

基于位置的启发式方法

第1步（聚集顶点选择/concentrator selection）：对于每组顾客$k=1,2,\cdots,K$，选择一个聚集顶点j_k（$\in V$），使得分配n个顾客到最近聚集顶点的总距离最小，且每个顾客组的总运送能力不超过Q

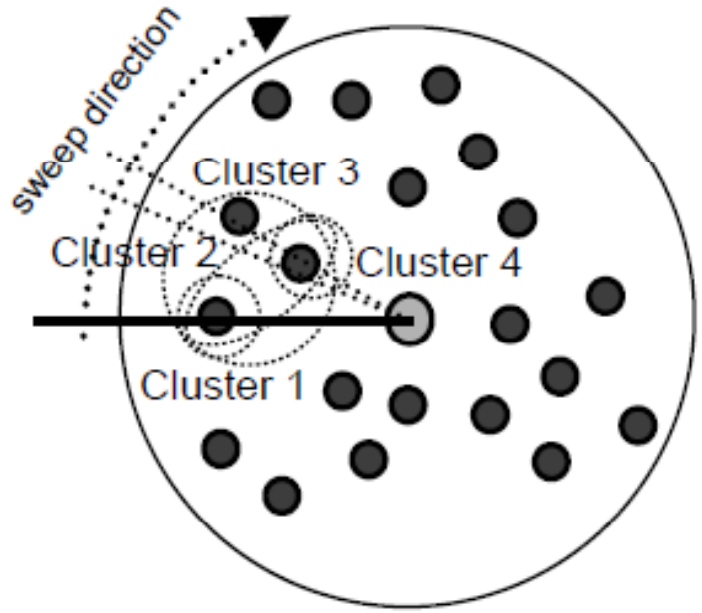
第2步：对于每个顾客组，每次插入一个具有最小（估计）插入费用的顾客，来构造车辆路线



花瓣启发式方法

排序顶点$(v_1, v_2, \cdots, v_n)$，例如利用扫描排序

对于每个顶点v_i，创造顾客组$\{v_i\}$、$\{v_i, v_{i+1}\}$、$\{v_i, v_{i+1}, v_{i+2}\}$、……（每个顾客组可由一辆车可行服务）



对于每个顾客组 k，由求解对应的TSP（精确方法或启发式方法均可），来计算旅行费用 c_k

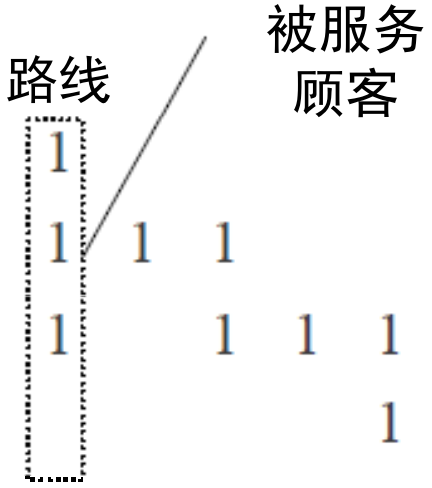
定义和求解一个集合划分问题/SPP（具有一个用户组列和旅行费用 c_k ）

$$\min \sum_{k \in R} c_k x_k$$

$$\sum_{k \in R} a_{ik} x_k = 1 \quad \forall i \in V$$

$$x_k \in \{0,1\} \quad \forall k \in R$$

$$A = [a_{ik}] = \begin{bmatrix} 1 & 1 & 1 & & & & \\ & 1 & 1 & 1 & 1 & & \\ & & 1 & & 1 & 1 & 1 \\ & & & & & & 1 & 1 \end{bmatrix}$$



两阶段方法：先定路线后分组（RFCS）

Beasley算法

阶段1：生成路线—求解一个松弛运送能力约束的单TSP

阶段2：生成顾客组—切割TSP解，产生满足运送能力约束的路线集合

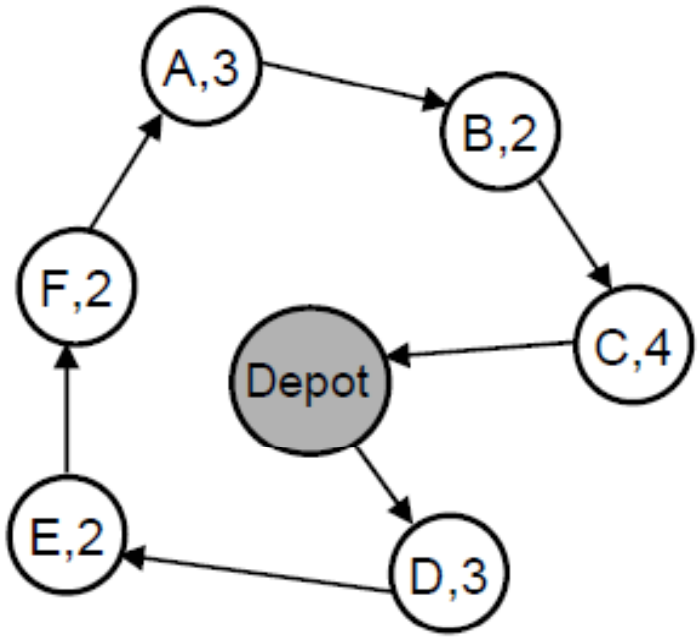
注意：

利用花瓣启发式方法，可在多项式时间内优化求解阶段2

RFCS启发式方法是花瓣启发式方法的特殊形式

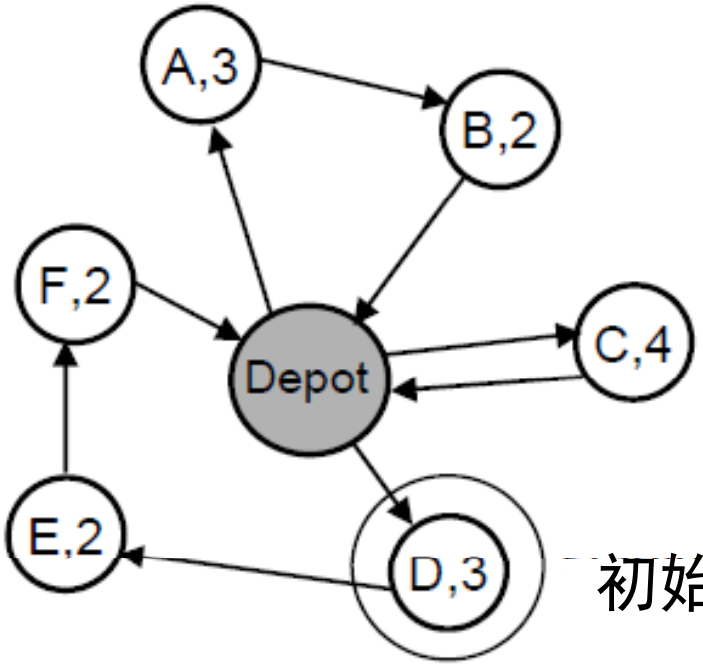
一个例子：车辆运送能力 $C = 8$

TSP解　　路线长度=343（不可行）



c_{ij}	Dep	A	B	C	D	E	F
Dep	0	64	58	54	41	58	41
A	64	0	70	95	103	81	40
B	58	70	0	36	92	113	81
C	54	95	36	0	72	112	91
D	41	103	92	72	0	61	71
E	58	81	113	112	61	0	41
F	41	40	81	91	71	41	0

第一个顾客组/路线集合

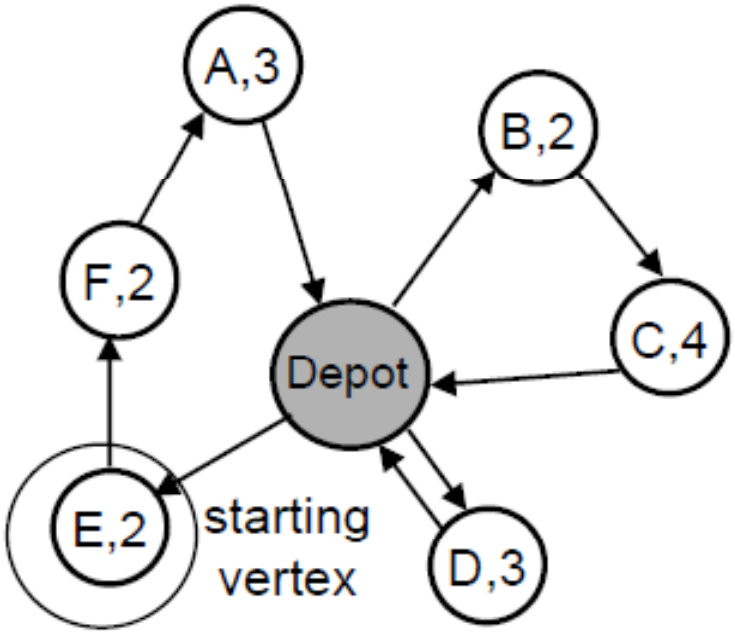


初始顶点

路线总长度=484（可行）

c_{ij}	Dep	A	B	C	D	E	F
Dep	0	64	58	54	41	58	41
A	64	0	70	95	103	81	40
B	58	70	0	36	92	113	81
C	54	95	36	0	72	112	91
D	41	103	92	72	0	61	71
E	58	81	113	112	61	0	41
F	41	40	81	91	71	41	0

第二个顾客组/路线集合



路线总长度=433（可行）

c_{ij}	Dep	A	B	C	D	E	F
Dep	0	64	58	54	41	58	41
A	64	0	70	95	103	81	40
B	58	70	0	36	92	113	81
C	54	95	36	0	72	112	91
D	41	103	92	72	0	61	71
E	58	81	113	112	61	0	41
F	41	40	81	91	71	41	0

……等等

改进启示式方法

基于局部搜索（LS）：搜索解的一个邻域 $N(x)$

利用“移动”构造 $N(x)$

可能的移动：

- 在访问序列中某位置插入一个顾客
- 交换一对顾客的位置
- k-OPT

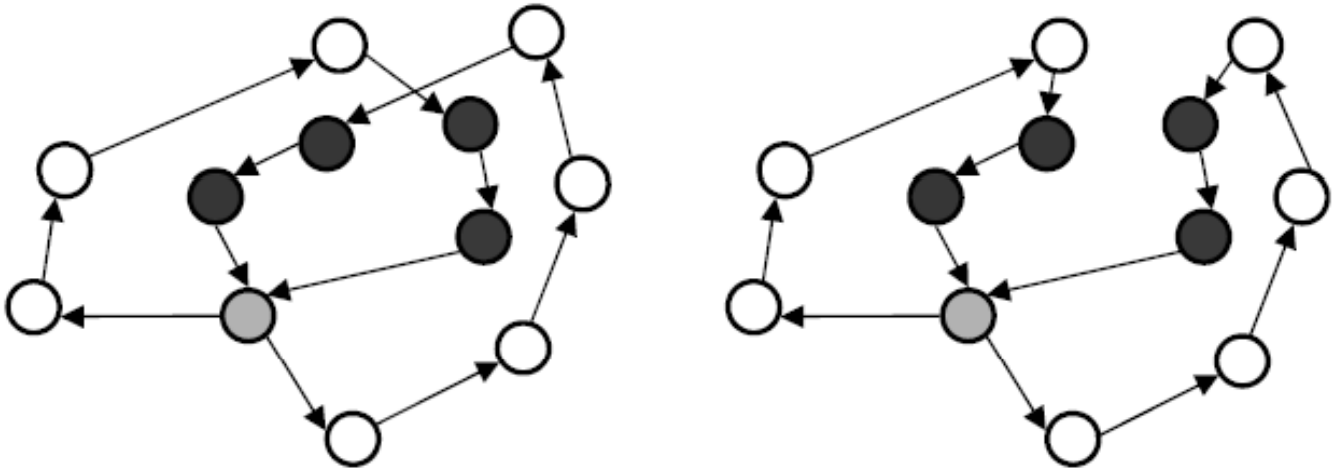
两类方法：

- 单路线改进—分配给路线（车辆）的顾客不变（类似于TSP改进启发式方法）
- 多路线改进—移动可能改变顾客-路线分配

多路线改进：

子路线交叉

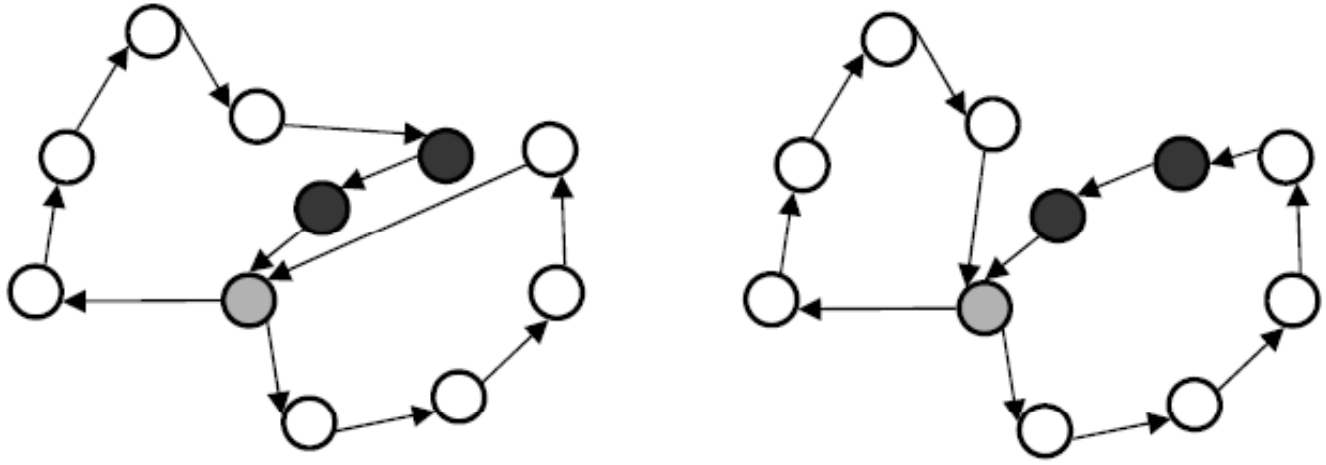
- 由交换两条不同路线中弧的终点，两条顾客链被交换



多路线改进：

子路线迁移

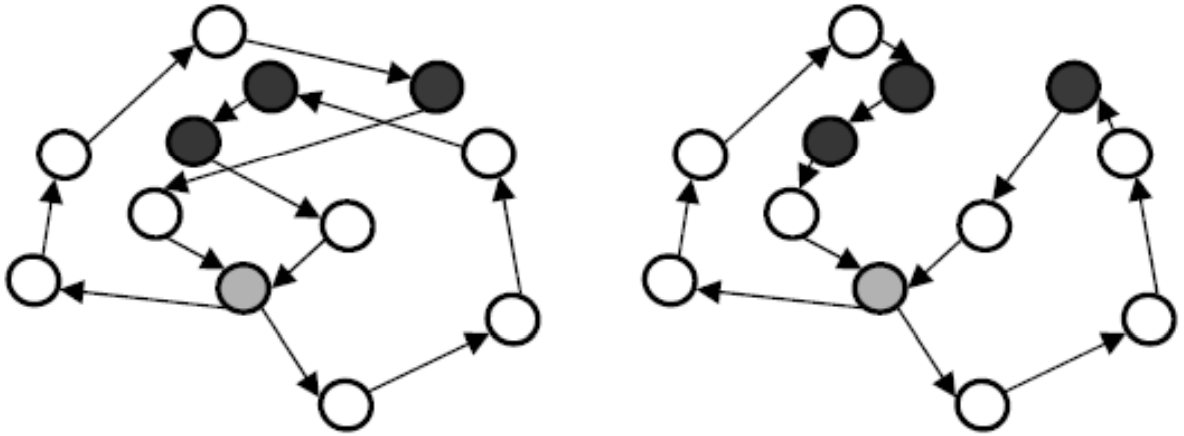
- 将一条包含最多 k 个顾客的顾客链从一条路线移到另一条路线上
- 较大 k 值可能产生更好结果，但邻域搜索时间也增加



多路线改进：

子路线交换

- 包含最多 k 个顾客的两条顾客链在两条路线间交换
- 如果 $k = n$，则子路线交换包含子路线交叉（这是一个较大邻域，计算时间量较大）
- k 值通常取较小的 $\{1, 2, 3\}$



如果 $k_1 > 0$ 和 $k_2 = 0$，则子路线迁移是一个特殊形式

如果 $k_1 = k_2$ 且子路线在路线的两端，则子路线交叉是一个形式