网络设计问题

(Network Design Problem)

东北大学 系统工程研究所

2011.09

网络设计问题

- 网络设计问题属于一类重要的工程优化问题
- 任何具有网状拓扑结构的问题，均可归结为网络设计问题，涵盖的内容非常广泛。
- 很多实际问题可以构造为网络设计问题。
- 近年来，一些从事遗传算法研究的学者，对网络设计的优化问题倾注了极大的关注。

各类网络设计问题

- 基本网络设计问题
  - 最短路径问题 (Shortest Path Problem (SPP))
  - 最大流问题 (Maximum Flow (MXF) Problem)
  - 最小费用流 (Minimum Cost Flow (MCF) Problem)
  - 双准则网络设计问题 (Bicriteria Network Design Problem (BNP))
  - 多准则网络设计问题 (Multi-criteria Network Design Problem)
- 最小生成树问题
  - 约束最小生成树问题
  - 二次最小生成树问题
  - 度约束最小生成树问题
  - 双目标最小生成树问题
  - 多目标最小生成树问题 (Multicriteria Minimum Spanning Tree (mc-MST))
  - 叶约束最小生成树问题 (Leaf-constrained Minimum Spanning Tree)

各类网络设计问题

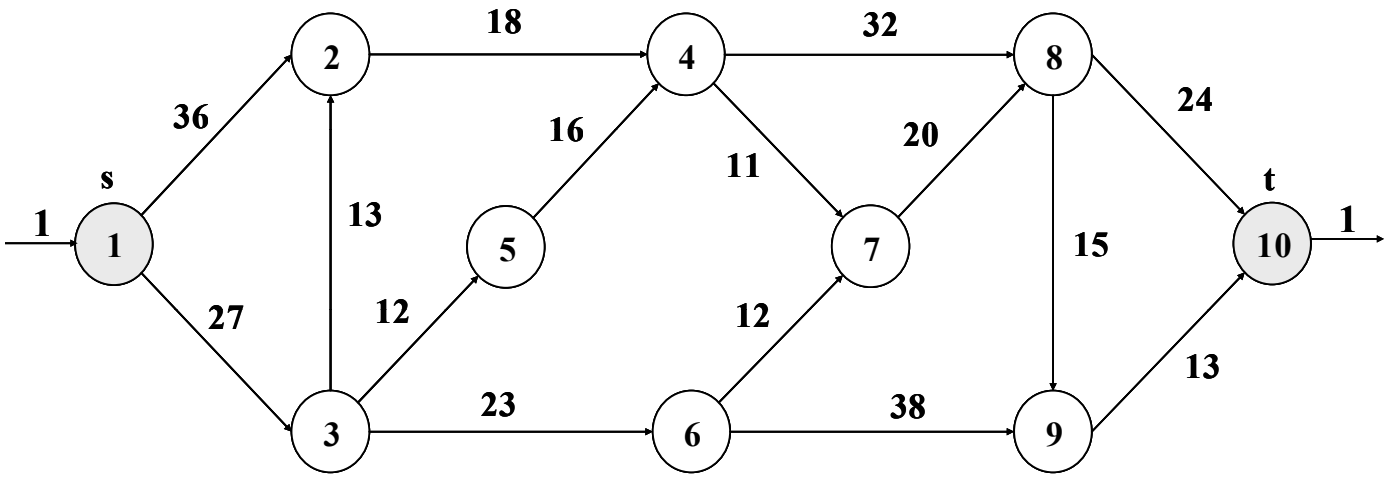
- 物流网络设计问题 (Logistic Network Design)
  - 基本运输问题 (Transportation Problem)
  - 扩展运输问题 (Extension of Transportation Problem)
  - 双阶段运输问题 (Two-stage Transportation Problem)
  - 多阶段物流链网络 (Multi-stage Logistic Chain Network)
  - 供应链网络设计 (Supply Chain Network Design)
  - 多仓储 VRP 问题 (Multi-depot VRP)
- 通信网络和局域网设计问题
  - 有适应能力的网络路由 (Adaptive Network Routing)
  - 集中式网络设计 (Centralized Network Design)
  - 计算机网络扩展(Computer Network Expansion Problem)
  - 主干网设计 (Backbone Network Design)
  - 双准则局域网拓扑设计 (Bicriteria LAN Topological Design)

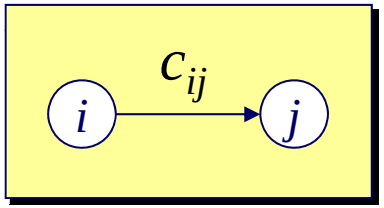
最短路径问题

(Shortest Path Problem)

最短路径问题

- 最短路径问题也许是所有网络设计中最简单的问题。
- 问题描述：
  - 假设每条边 $(i, j) \in A$ 有一个相关的费用（或距离） c_{ij}，目标就是要寻找一条从指定的起点 s 到指定的终点 t 的一条路径，使总费用（或距离）最小。

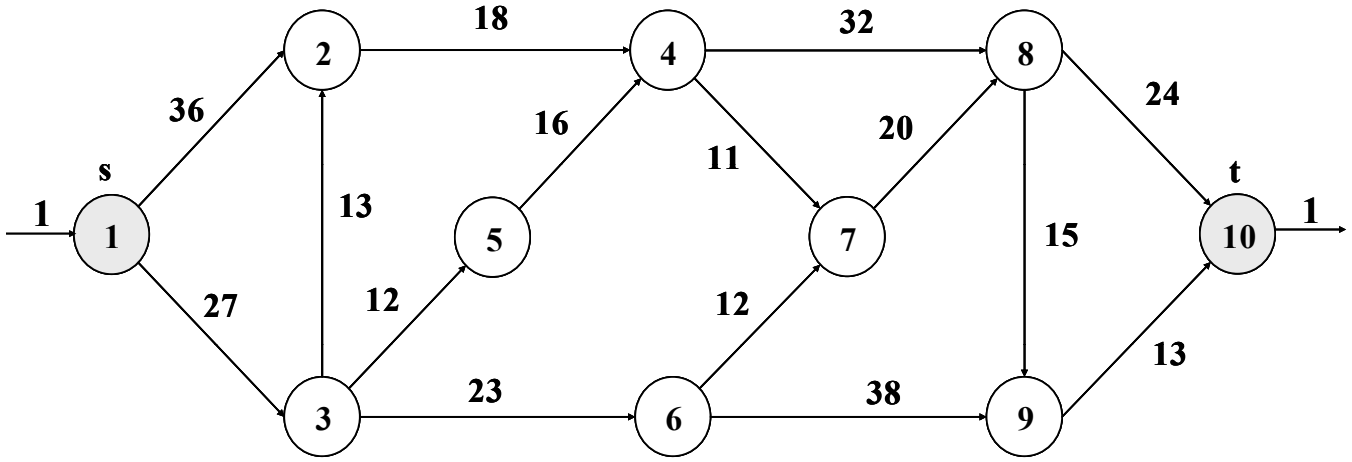


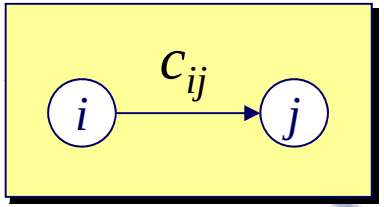


网络模型

- 网络模型可描述为：
  - 有向图：$G=(V, A)$，其中 V 是节点的集合， A 是连接的集合
  - c_{ij} 是和每个连接 (i, j) 相关的费用
  - 起点：节点 1
  - 终点：节点 m
  - 指示变量：

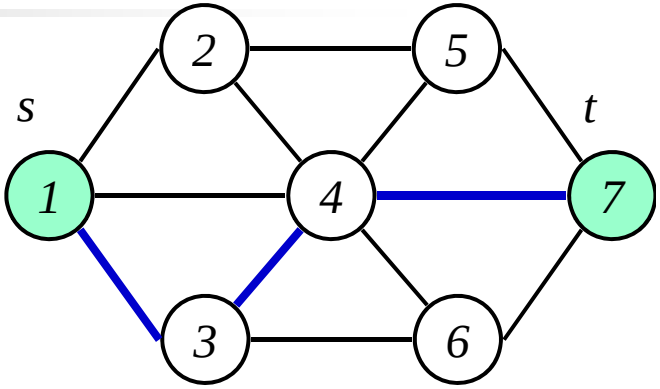
$$x_{ij} = \begin{cases} 1, & \text{if link } (i,j) \text{ is included in the path} \\ 0, & \text{otherwise} \end{cases}$$





模型描述

- 用数学公式可描述为：



$$\min \quad z = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij}$$

由点 i 出去的边之和

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} 1 & (i=1) \\ 0 & (i=2,3,\cdots,m-1) \\ -1 & (i=m) \end{cases}$$

进入点 i 的边之和

$$x_{ij} = 0 \text{ or } 1 \quad (i, j = 1, 2, \cdots, m)$$

where,

x_{ij}：从点 i 到点 j 的连接；

c_{ij}：和每个连接 (i, j) 相关的费用；

- 约束条件的含义是：
  - 节点 1 只有一条出去的边；
  - 由节点 i 出去的边和进入节点 i 的边是相等的；
  - 进入节点 m 的边只有一条。

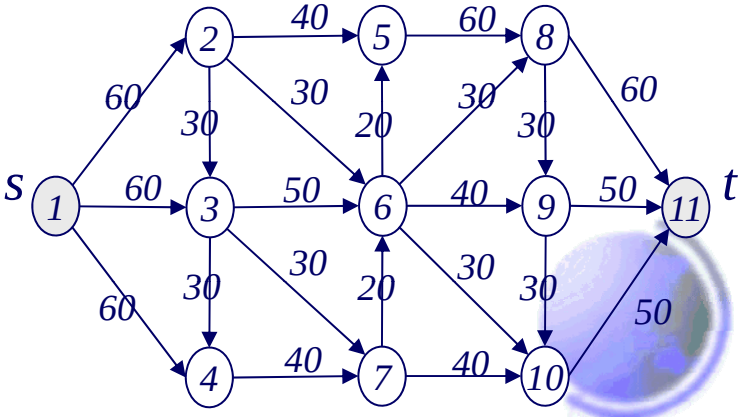
特点

- 一类特殊的 0-1 线性规划问题
- NP-难题

应用

- 与交通路网有关的求“距离”最短的问题
  - 例如公路、铁路运输问题、城市交通问题。还有：火灾、地震、道路应急救援问题，120急救车、配电网故障抢修等都涉及到求最短路的问题。
- 飞机最优航线确定问题
  - 完成一次飞行任务要中转多个机场时，如何确定最优的航线，使得航行距离最短或费用最少。
- 大型船只的航线确定问题
  - 首先要建立航路点，确定航线网络。这时必须要考虑的问题是要劈开海上障碍区，例如岛礁、浅地、沉船、养殖区等。
- GPS导航
  - 这也是最短路的典型应用。

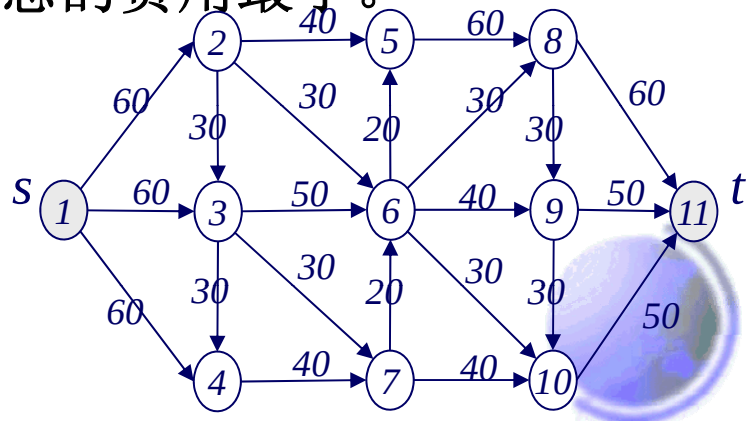
确定一条出发点到目的地终点之间的最短（或最快）行驶路线。



应用

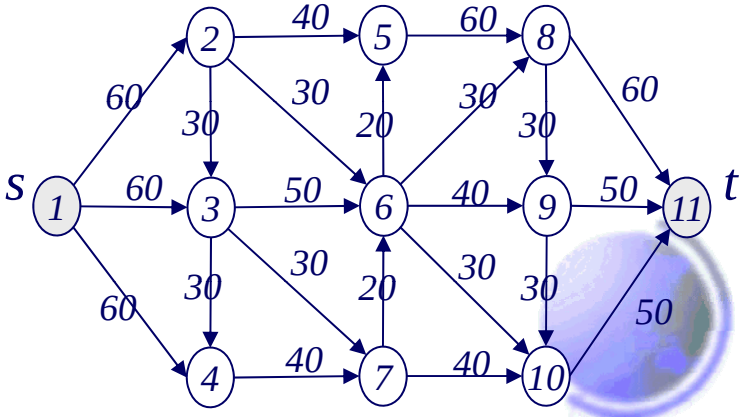
- 移动机器人路径规划
  - 确定由起点到终点的一条路径，使得耗能最小、路径最短、或时间最小。
- IP路由选择问题
  - 寻找一条将数据报从源节点（信源机）发送到接收节点（信宿机）的最佳传输路径的问题，而传输路径由一系列路由器组成
- 管道铺设
  - 例如成品油管道优化设计时，通常考虑各种因素指定若干个点作为泵站的候选点→节点。在不同的泵站之间铺设管道的话，会产生不同的费用（例如:线路费用、运行维护费用等）→边上的权值。目标是如何铺设管道，能使总的费用最小。
- 医院向导系统
  - 各科室作为节点，科室间的直接通道作为边，距离作为边的权重。

那么就可以求出任一科室到另一科室的最短距离。



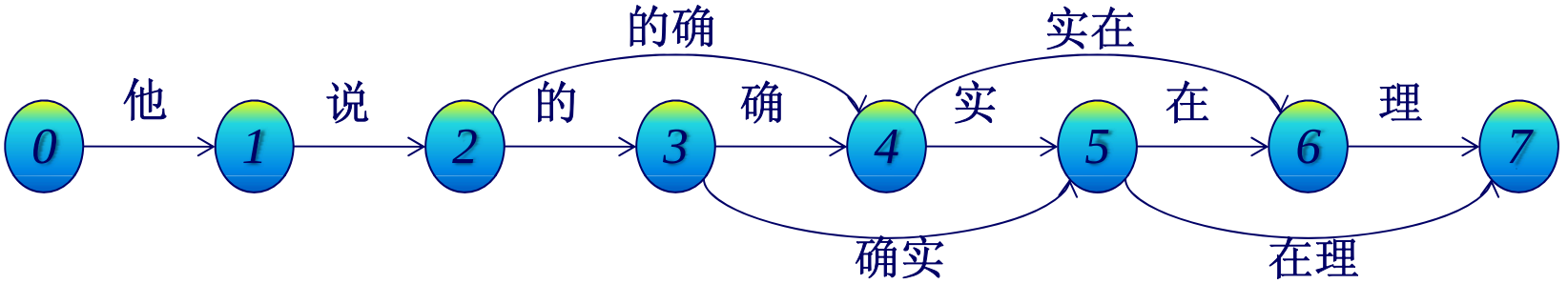
应用

- 大型停车场的车位诱导
  - 找到从入口到所有空闲车位的最短路，从中确定一个距离最短的停车位。
- 高速公路收费系统
  - 收费的多少是以起点到终点之间的最短路为基础的。
- 旅游线路设计问题
  - 对旅游者而言,仅考虑景点间的相互距离肯定是不够的，还要考虑景点的知名度、时间限制等
- 中文自然语言处理中的“中文分词”问题
- 多阶段决策问题
  - 设备更新问题
  - 原料选用
  - 投资预算
- …

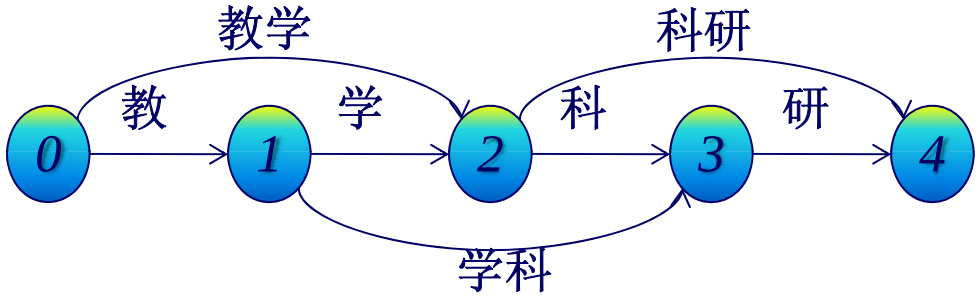


中文分词问题

- 对中文字符串通过原子切分、找出原子之间所有可能的组词方案后形成一个特殊的有向图
- 例如：
  - 对汉字串“他说的确实在理”进行有向图构造



  - 对汉字串“教学科研”进行有向图构造

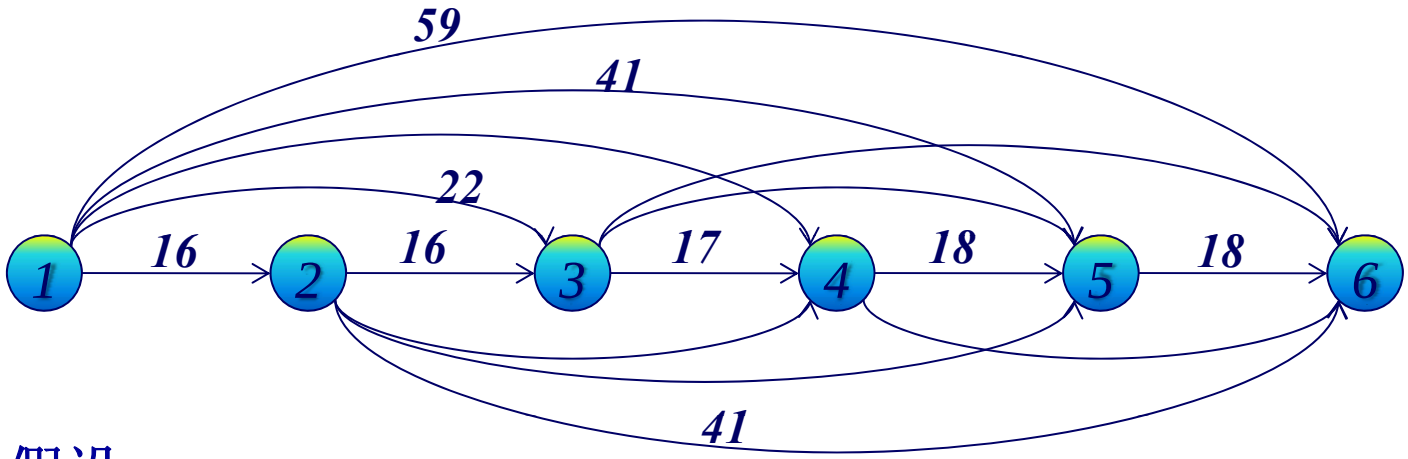


- 求最短路实际上就是找一句话中最少的分词。

设备更新问题

- 年初购买新设备及使用不同年限的设备维修费用：

年限	第1年	第2年	第3年	第4年	第5年
购买单价（万元）	11	11	12	13	13
使用年限	**0~1**	**1~2**	**2~3**	**3~4**	**4~5**
维修费用（万元）	5	6	8	11	18



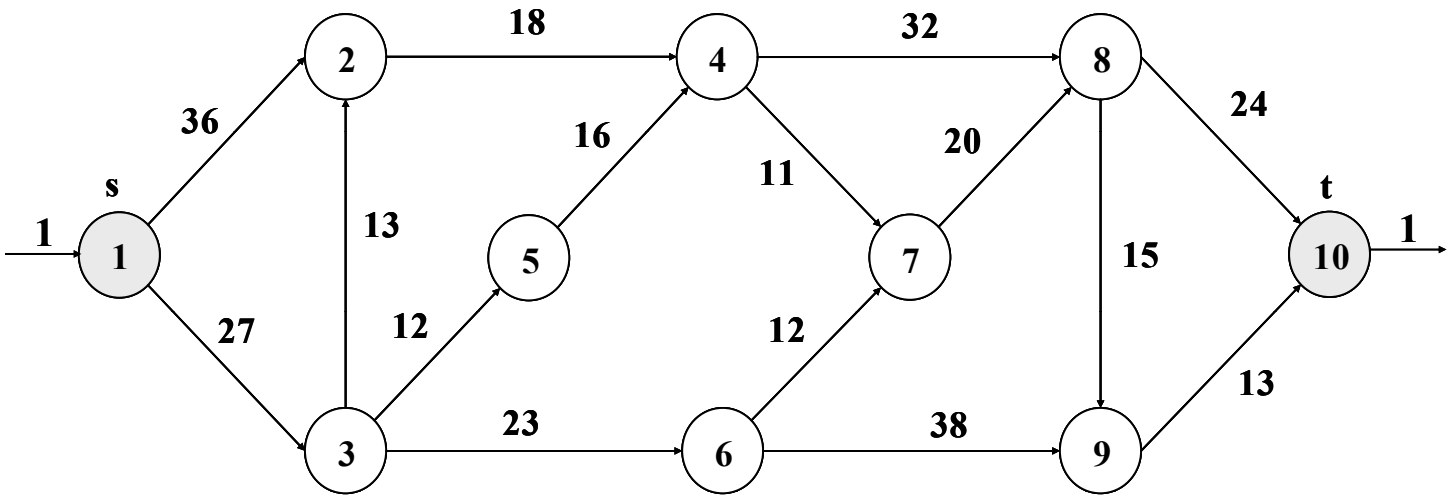
- 图中假设：
  - 点 *i*，表示第 *i* 年年初，同时也表示第 *i-1* 年的年末。例如…
  - 由 *i~j* 的边，表示由第 *i* 年年初购买设备，并使用到第 *j* 年年初。例如..
  - 边上的权重表示设备购买和维修费用之和，不同的边费用不同。例如..

一般的求解方法

- 穷举法
- 动态规划法
  - 分阶段逐步求出各节点到终点的最短路。
- **E.W.Dijkstra 最短路算法**
  - 从起点逐步求出到每一个节点的最短路。
- **Floyd-Warshall Algorithm**
  - **Floyd-Warshall algorithm is an algorithm to solve the all pairs shortest path problem in a weighted, directed graph by multiplying an adjacency-matrix representation of the graph multiple times.**
- **Bellman-Ford算法**
  - 也称为反向搜索算法或距离向量算法，它是从目的点出发反向计算的。它是找到一个给定的源顶点出发的最短路径，限制条件是径上最多只有一条链路；然后再找到一些最短路径，限制条件是路径上最多只有2条链路，其余类似。

GA求解

- 如何对网络中的路径进行编码，是设计GA的关键
- 难点在于：
  - 由起点到终点的一个路径包含可变个节点
  - 边的随机序列通常不能对应一条路径



- 路径**1**：1→2→4→8→10　　目标函数：z=110
- 路径**2**：1→2→4→7→8→10　　目标函数：z=109
- 路径**3**：1→3→5→4→7→8→10　　目标函数：z=110

GA求解

- 编码要解决的关键问题：
  - 染色体的合法性
    - 染色体是否描述了给定问题的一个解
    - 通常采用修复技术将非法染色体转换为合法染色体
  - 染色体的可行性
    - 由染色体解码得到的解是否在问题的可行域内
    - 有四种处理约束的技术
  - 映射的唯一性
    - 染色体到解的映射 通常存在三种形式：**1:1, *m*:1, 1:*n***
    - **1:1** 映射是最好的
    - **1:*n*** 映射是最不期望的
    - **m:1** 映射会减少种群的多样性

- 编码方法：
  - 基于优先级的编码
  - 可变长编码
  - 定长编码

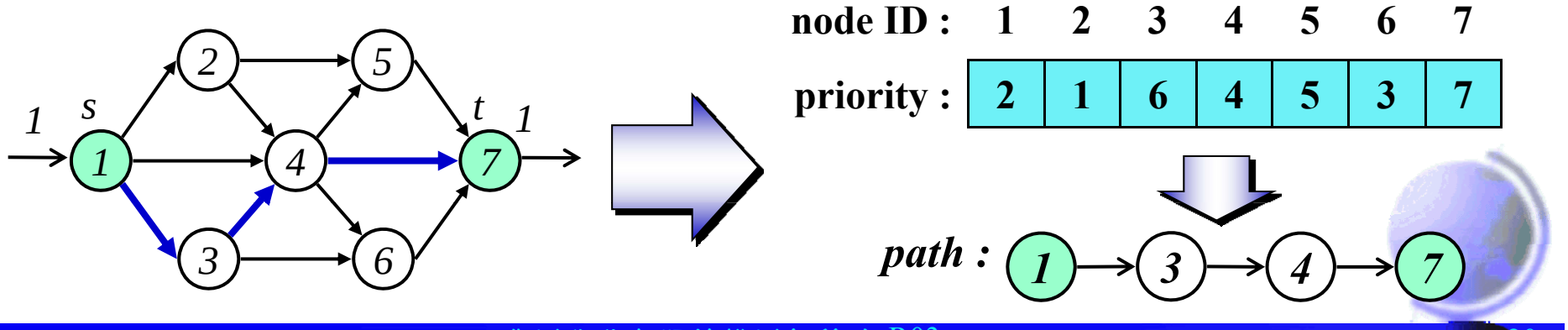
GA求解

- 基于优先级的编码
  - 这种编码方式最早由 **Cheng & Gen, 1997** 年提出，用于求解资源受限项目调度问题。
  - 并非直接对解编码（不存在直接的映射关系），而是采用一种间接的方法：通过对路径的导向性信息进行编码，构建染色体中的路径而不是路径本身。
  - 在基于优先级的编码方法中：
    - 基因的位置代表节点编号，有多少节点就有多少个基因；
    - 基因的值代表节点的优先权，不同节点的优先权各不相同。
    - 解码时，从节点1开始依次选择路径，在可选的节点中只有优先权最高的节点加入路径中。

node ID :	1	2	3	4	5	6	7
priority :	2	1	6	4	5	3	7

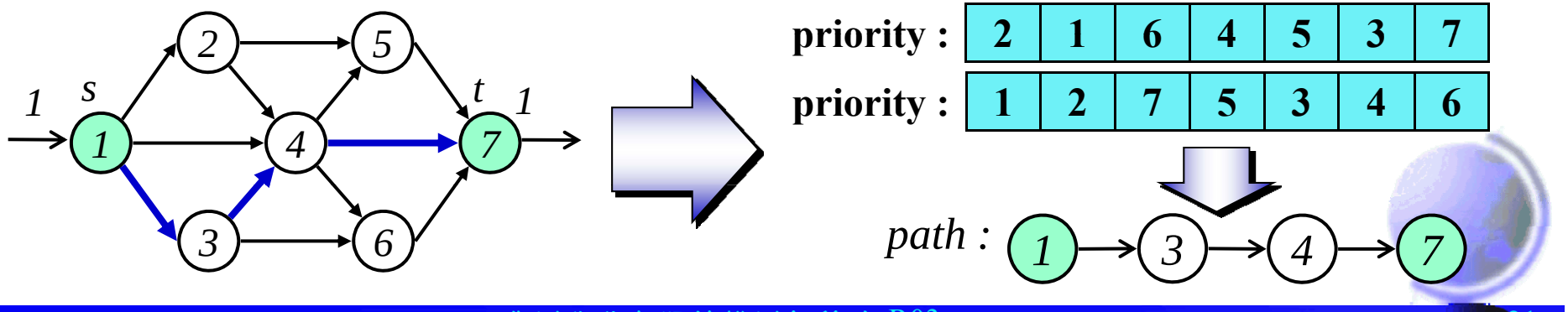
GA求解

- 基于优先级的编码
  - 实例
    - 假设要寻找一条节点 1 到 7 的路径。编码如图所示。
    - 首先，寻找与节点 1 相邻的节点。节点 2 ，3 和 4 在此都是适合的，但节点 3 具有更高的优先权，因此，被纳入路径。
    - 然后，寻找与节点 3 相邻的节点 (4, 6)，并将具有最高优先权的节点纳入路径 (节点 4)。
    - 重复这些步骤，直至得到一条完整的路径 (1, 3, 4, 7)。



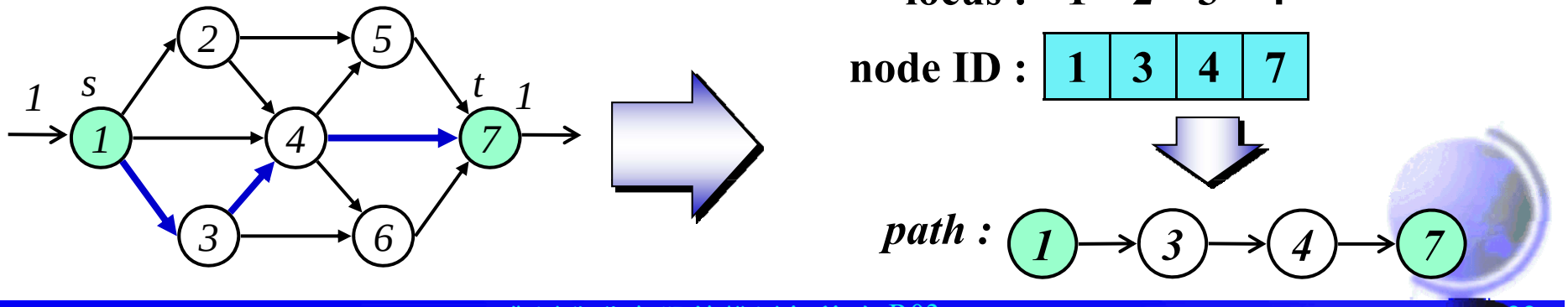
GA求解

- 基于优先级的编码
  - 优点：
    - 任何顺序的编码对应着一条路径 (合法性)，因此，绝大多数已有的遗传操作可以轻易地在这种编码上运用。
    - 同样，任何一条路径拥有相应的编码 (完备性)，因此，遗传搜索能到达解空间中的任意一点。
  - 缺点：
    - 编码与路径之间的映射是 ***m*:1** 的，这意味着不同的染色体可能对应着同一条路径，但概率较低。不会影响**GA**操作。



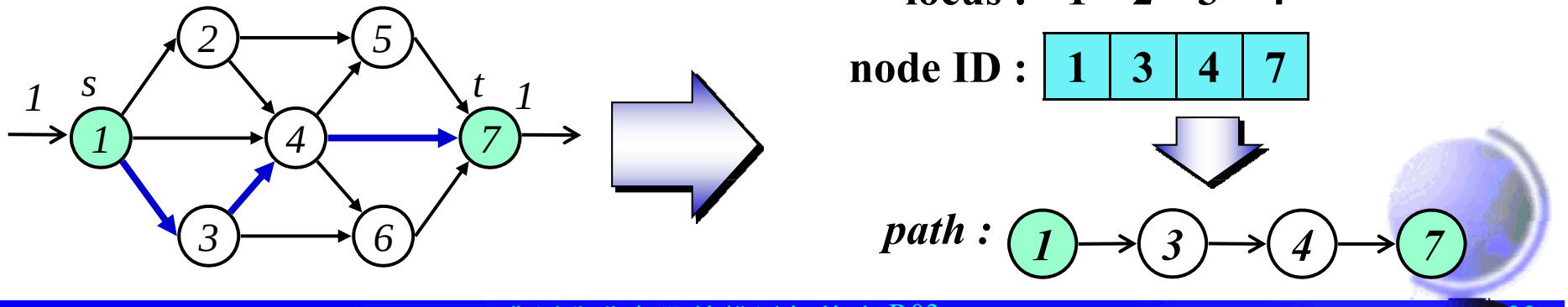
GA求解

- 可变长编码
  - 由 **Munemoto et al., 1998** 提出，用于求解有线或无线环境下网络路由问题。
  - 编码由正整数序列组成：
    - 染色体中的基因位表示节点在路径中的位置
    - 染色体的值，表示路径中的节点编号
    - 染色体的长度是可变的，但不应该超过最大长度 **n**



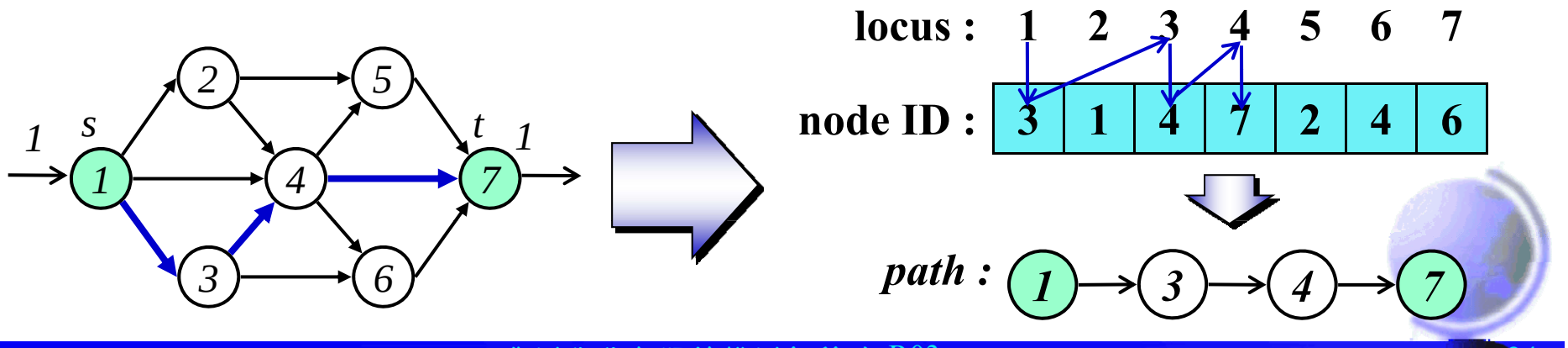
GA求解

- 可变长编码
  - 优点：
    - 染色体到解的映射是 1:1 的（唯一性）
    - 从理论上来讲，收敛性要好于基于优先级的编码方法
  - 缺点：
    - 通常，遗传操作会产生违反约束或带有循环路径的非法染色体。
    - 需要采用修复技术将非法染色体合法化。



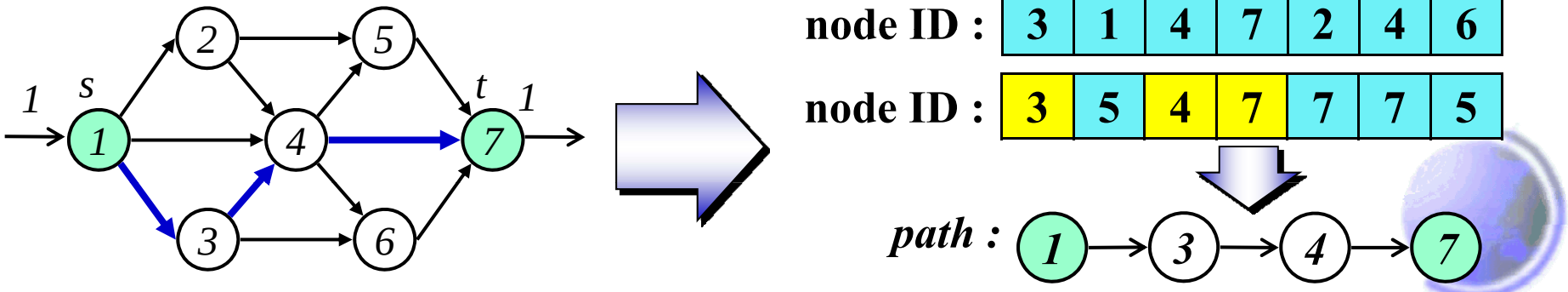
GA求解

- 定长编码
  - 由 **Inagaki et al. , 1999** 提出，用于在路由应用中确定多路由器。
  - 编码由整数序列组成：
    - 基因的位置表示节点，
    - 基因的值表示要访问的节点编号。



GA求解

- 定长编码
  - 优点：
    - 任何路径都有一个相应的编码 (完备性)，对遗传搜索来说，解空间中的任何一点都是可达的
    - 使用特定的遗传操作可以获得合法的路径
  - 缺点：
    - 编码到解的映射是 *m*:1 ，而且发生的概率要高于基于优先权的编码方法
    - 遗传操作过程中，可能产生一些在适值上类似于初始染色体的子代，因而延缓了进化的过程。

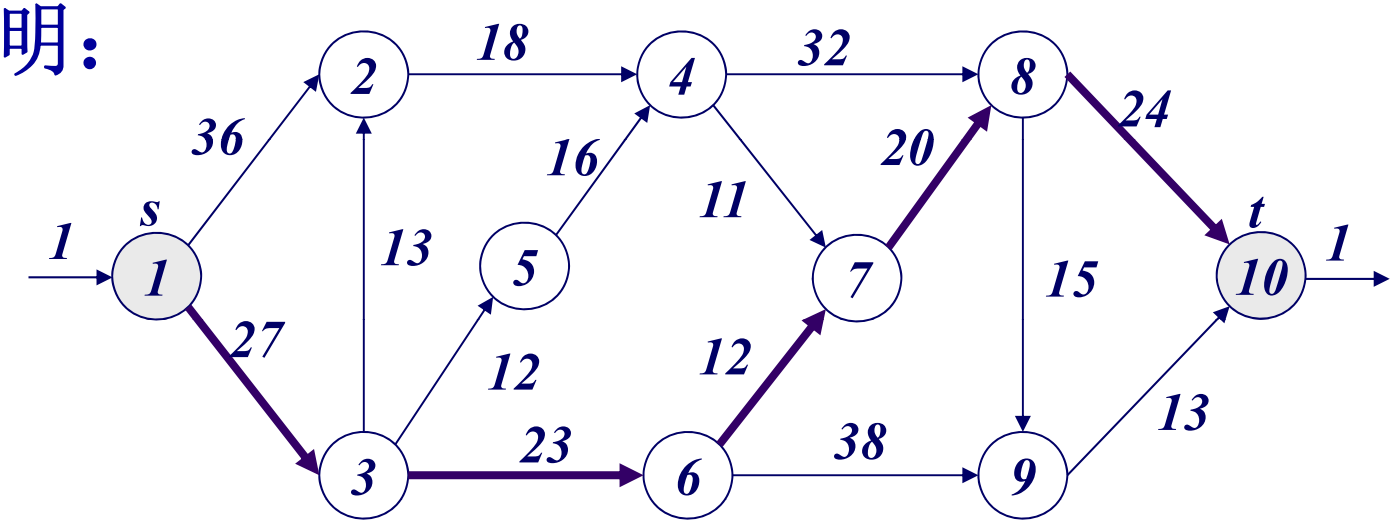


GA求解

- 不同编码方法的性能比较
  - 可变长编码方法：
    - 与其它方法相比，具有最好的收敛性能
    - 然而，遗传操作可能产生不可行解
    - 必须采用修复技术将非法染色体转换为合法的。在一些大型网络设计当中，计算时间较慢。
  - 定长编码方法：
    - 编码会产生 ***m*:1** 的映射
    - 必须采用特殊的遗传操作，因而产生的一些子代可能在适值上类似于初始的染色体。
  - 基于优先级的编码方式：
    - 虽然会产生 ***m*:1** 的映射，但概率较低，是一种相对较好的编码方式。

GA求解－基于优先级的遗传算法

- 种群初始化
  - 可采用随机的方法生成染色体，构成初始种群
- 解码说明：



与 i 相邻的节点集合 $S(i)$:	(2,3)	(4)	(2,5,6)	(7,8)	(4)	(7,9)	(8)	(9,10)	(10)	( )
节点ID: i	1	2	3	4	5	6	7	8	9	10
优先级: $v(i)$	7	3	4	6	2	5	8	10	1	9

路径 P_k: 1→3→6→7→8→10

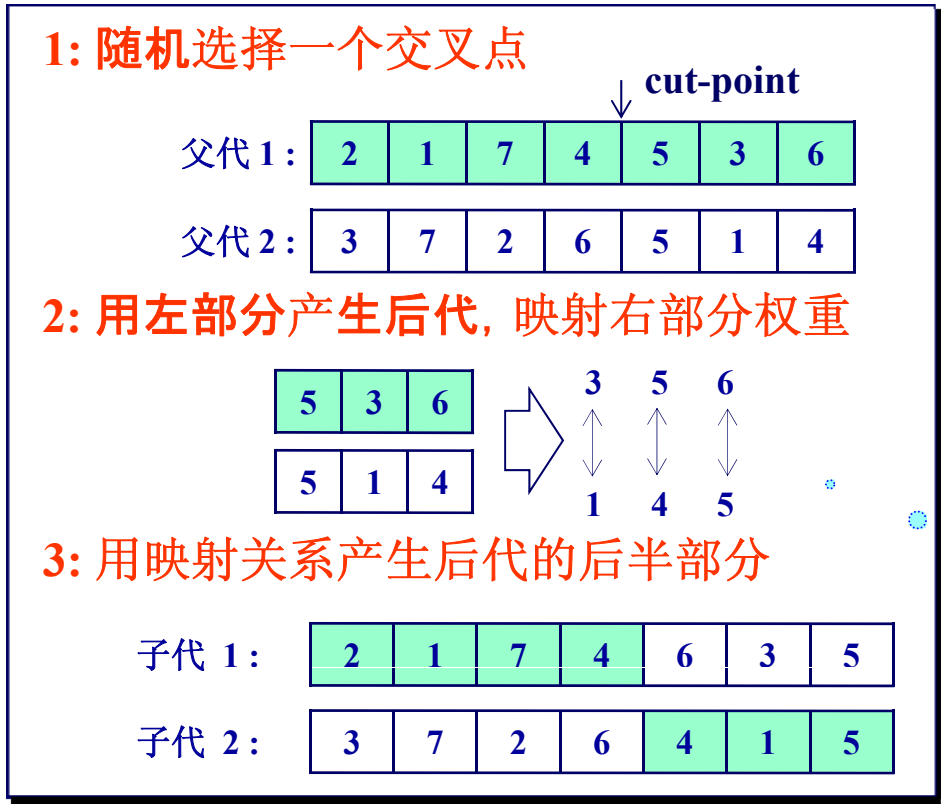
目标函数值: z = 27+23+12+20+24=106

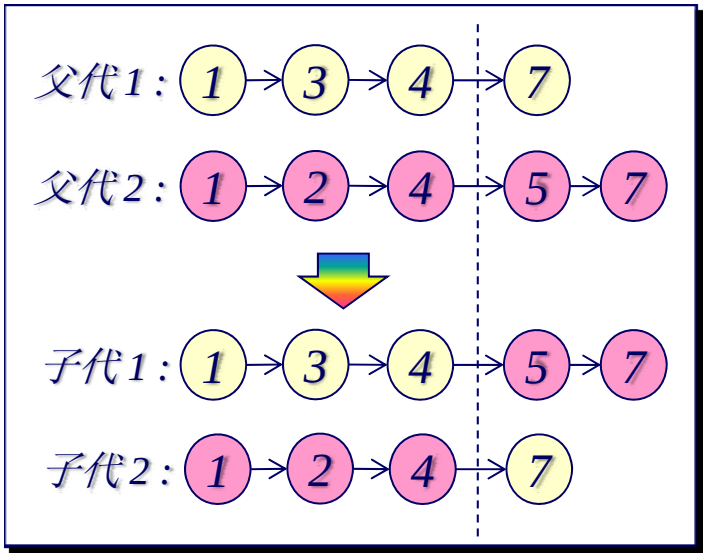
GA求解－基于优先级的遗传算法

- 交叉
  - 基于优先级的编码方法，其本质是一种换位表达方法
  - 因此，旅行商问题的针对换位表达的交叉算子，都可以应用到最短路问题的基于优先级的遗传算法当中。
    - **PMX、OX、CX、PBX 、OBX、……**
  - 然而这些交叉方法
    - 交叉机制不同于传统的单点交叉
    - 后代可能会产生一些不能继承父代特征的染色体
    - 因而延缓了进化过程。

GA求解－基于优先级的遗传算法

- 权重映射交叉（Weight Mapping Crossover (WMX)）
  - Cheng，Gen 等提出的一种新的交叉算子。
  - WMX 可以看作是针对换位表达的单点交叉的扩展
    - 因具有单点交叉的性质，所以能较好地继承父代的特征。
  - 交叉过程如下：





通过对右半部分权重进行排序，实现映射

GA求解－基于优先级的遗传算法

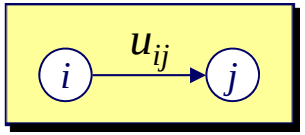
- 变异
  - 对于换位表达的染色体编码方式，变异运算相对比较容易，例如：
    - 反转变异 （**Inversion**）
    - 插入变异 （**Insertion**）
    - 移位变异 （**Displacement**）
    - 互换变异 （**Swap mutation**）
  - 参见：
    - 旅行商问题

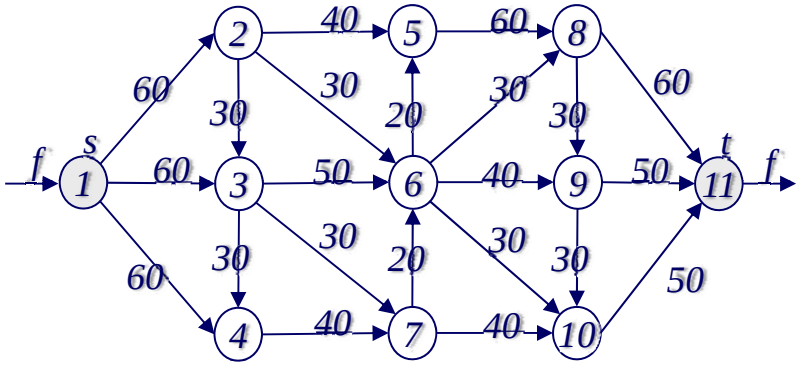
最大流问题

(Maximum Flow Problem)

最大流问题

- 许多系统包含有流量问题。例如：
  - 公路系统中的车辆流；
  - 控制系统中的信息流；
  - 供水系统中的水流；
  - 金融系统中的现金流等。

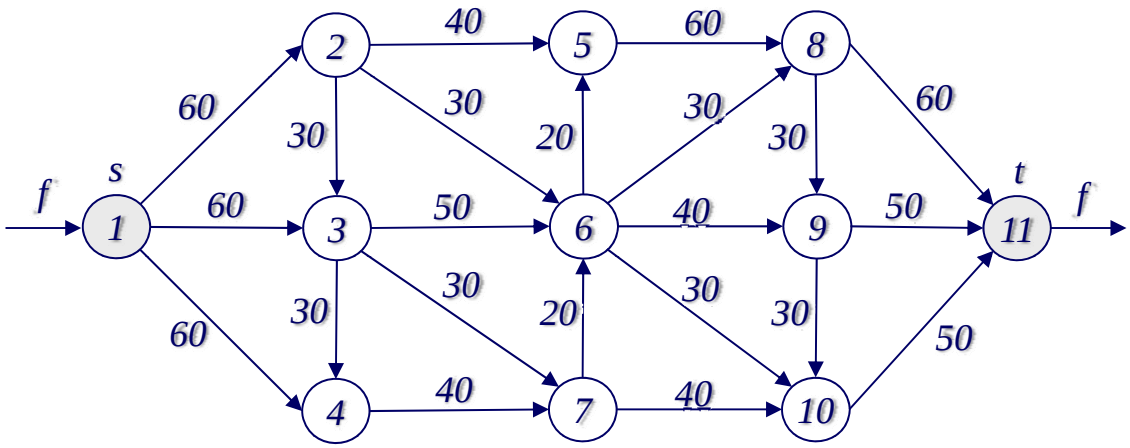


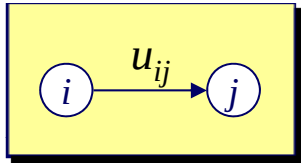


- 最大流问题就是寻找一个可行解，使得从指定的原点 s 到指定的汇点 t 传送的流量最大。
- 如果我们把 u_{ij} 解释为弧 (i, j) 的最大流率，则 MXF 可以看作是从节点 s 到节点 t 每单位时间的最大稳态流。

网络模型

- 网络模型可描述为：
  - 有向图：$G=(V, A)$，其中 V 是节点的集合， A 是连接的集合
  - u_{ij} 是和每个连接 (i, j) 相关的容量
  - 起点: 节点 **1**
  - 终点: 节点 ***m***





模型描述

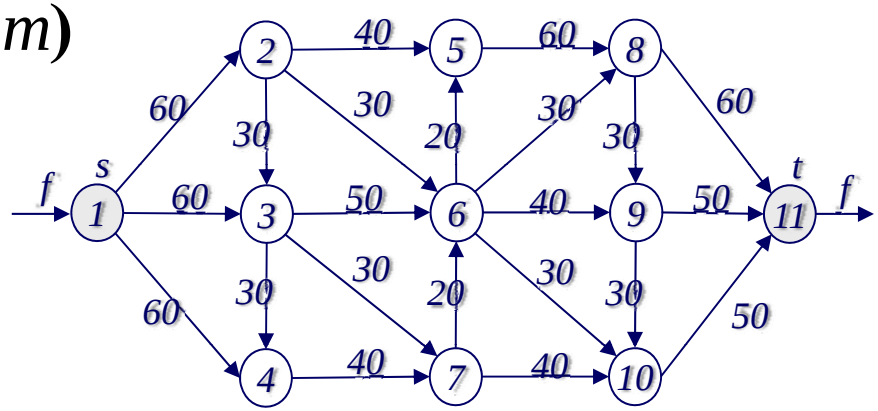
- 用数学公式可描述为：

$$\max \quad z = f$$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} f & (i=1) \\ 0 & (i=2,3,\cdots,m-1) \\ -f & (i=m) \end{cases}$$

$$0 \le x_{ij} \le u_{ij}, \quad (i,j) \in A$$

$$f \ge 0$$



where,

x_{ij}：*从点 i 到点 j 的流量；*

u_{ij}：*和每个连接 (i, j) 相关的容量 (最大流量)；*

- 约束条件的含义是：
  - 节点 1 只有流出的流；
  - 节点 *m* 只有流入的流；
  - 中间的流入、流出的流是相等的。

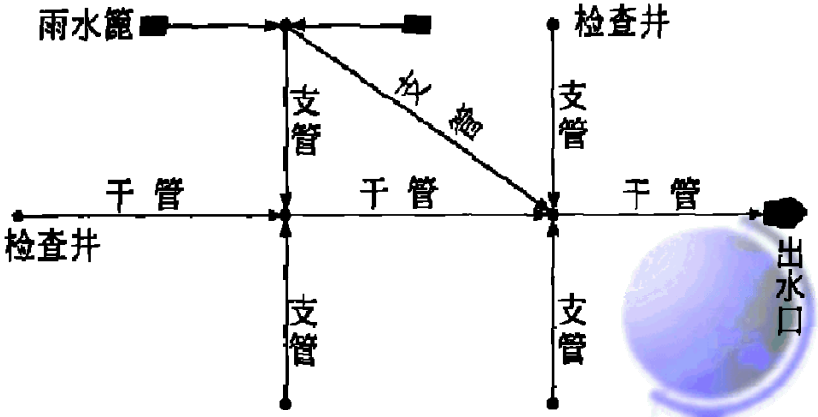
特点

- 一类特殊的线性规划问题
- 可行流和最大流的定义方法与线性规划中可行解和最优解的定义是一致的
- 与最短路问题不同，从起点到终点的流，可以流经多条路径

应用

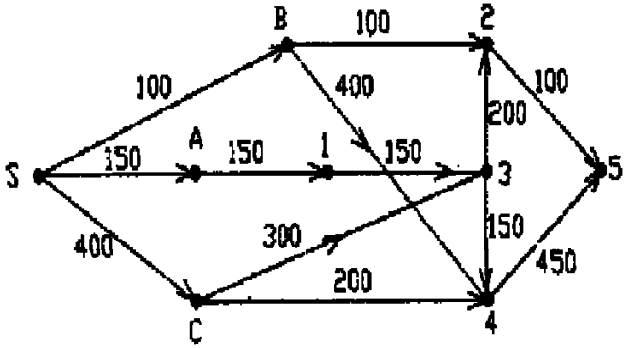
- 城市排水管网设计问题
  - 在排水管网设计时，必须考虑最大流量的问题，这涉及到一个城市应对暴雨时城市的排水能力。排水管网可以构成有向（管线水流有向）网络模型。管网设备包括雨水箅（源点）、检查井（节点）、雨水管（边）、出水口（汇点）等，每条管线都有设计的最大流量。
  - 设一个虚拟源点和所有的源点相连，设一个虚拟汇点和所有的汇点相连，则构成单起点单终点的最大流网络模型。
- 通风网络最大流计算问题
  - 和排水管网类似，入风口（源点）

和出风口（汇点）通常不止一个，
可将入风口连接成一个虚拟源点，
出风口链接成一个虚拟汇点，
则构成最大流问题的网络模型。

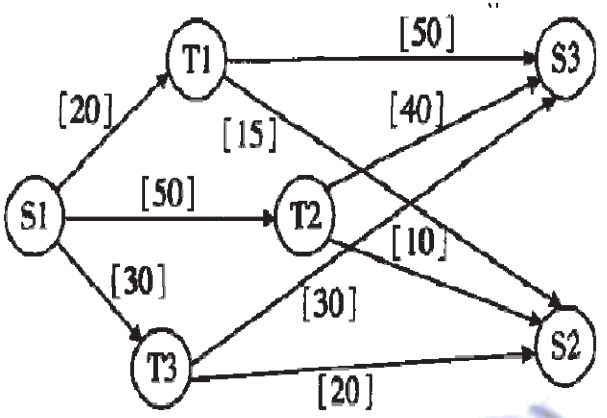


应用

- 输电网络最大流分析
  - 发电站是起点（A,B,C多个可能），变电所是中间节点(1,2,3,4)，城市是终点(5)，线路作为边有输电容量限制。最大流对应城市的最大供电能力。



- 突发事件的应对
  - 例如：地震、火灾、洪水等突发事件的紧急救援工作。急救中心作为起点（S1可以有多个），中间节点是一些治疗门诊(T1,T2,T3)，最终的接收医院作为终点（S2,S3可以有多个）。节点之间的边表示运输路线。边的容量表示一天之内最多能运送的病人数量。最大流问题对应着救援人数最多。



- ……

应用

- 均匀并行机调度 (Scheduling on Uniform Parallel Machines)
  - maximum lateness problem
  - (weighted) minimum completion time problem
  - (weighted) maximum utilization problem.
- 双处理机的分布式计算 (Distributed Computing on a Two-Processor Computer)
  - 这个问题涉及到指派程序的不同的模块(子程序)到两个处理器，在某种程度上最小化进程间通信和计算的集中成本。
- 油罐车调度问题 (Tanker Scheduling Problem)
  - 例如：轮船公司已经签订了在几个不同的始发点－目的地之间运送易腐烂货物的合同。由于货物是易腐烂的，所以客户指定了精确的到达目的地的日期（例如：交货期）。
- 运输方案确定问题
  - 由产地往销售地运送产品，如何确定运输方案能使运送的产品最多。

一般的求解方法

- **Ford-Fulkerson Algorithm**
  - 它以寻找图中流的可扩路 **(augmenting path)** 的方式进行。 通过增加流的可扩路到图中已经建立的流中，当找不到可扩路时，则达到最大流。
- **Maximum Flow Algorithm**
  - 最大流问题的一个增量算法，它试图寻找当网络中删除或增加一条边时，网络中的最大流。
  - 它也表明单位变化的其它情形可以看作是网络中边的插入和删除的一种特殊情形。
- **Push-relabel algorithm**
- **……**

GA求解

- 应用 GA 求解 MXF 问题，要比其它的网络问题更具有挑战性（例如：SPP，MST 等）。
- 问题独有的特点：
  - 每条边的流可以是 0 到容量限制之间的任意值，具有更多的选择自由度；
  - 而在许多其它问题当中，选择一条边可能意味着简单地增加一个固定距离。
- 必须满足两个条件：
  - 每条边的流必须在 0 到容量限制之间；
  - 在每个顶点，流入流出流必须平衡。
- 编码方式：
  - 基于优先级的编码
    - 与 SPP 的编码方式相同

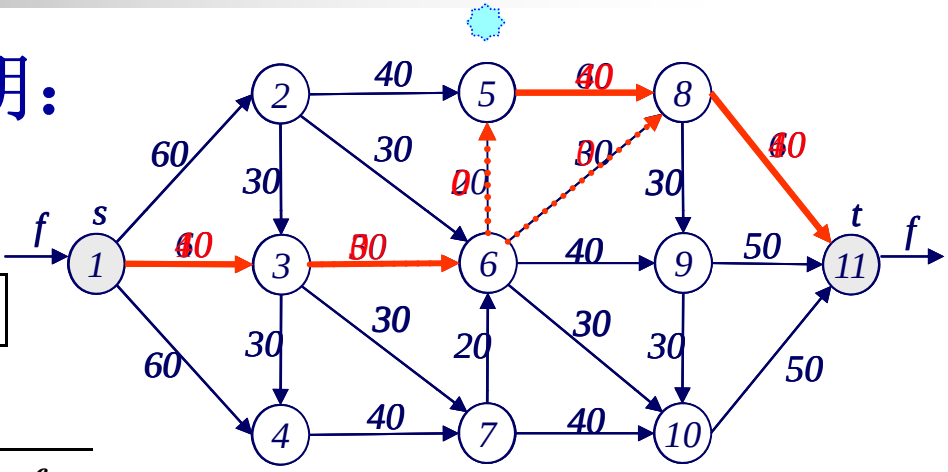
GA求解

- 解码过程分为两个阶段：
  - 第一阶段：采用单路径生长过程产生路径。即从节点 1 到 m，每一步都将具有最高优先权的节点加入路径（与SPP解码相同）。
  - 第二阶段：采用全路径生长过程产生全路径。
    - 对给定的一条路径，计算其流量 f_k
    - 通过从每条弧的 u_{ij} 中移除所使用的量，可以得到一个具有新的容量限制 $\bar{u}_{ij}$ 的网络
    - 通过单路径生长过程，可以获得第二条路径。
    - 通过重复这一过程，可以获得给定染色体的最大流，直到不能采用这种方法定义新的网络为止。

GA求解

- 基于优先级编码的图式说明：

node ID :	1	2	3	4	5	6	7	8	9	10	11
priority :	2	1	6	4	11	9	8	10	5	3	7



k	i	S_i	l	P_k	S_1	f_k
1	0			1		
	1	2, 3, 4	3	1, 3		
	3	4, 6, 7	6	1, 3, 6		
	6	5, 8, 9, 10	5	1, 3, 6, 5		
	5	8	8	1, 3, 6, 5, 8		
	8	9, 11	11	1, 3, 6, 5, 8, 11	2, 3, 4	20
2	0			1		
	1	2, 3, 4	3	1, 3		
	3	4, 6, 7	6	1, 3, 6		
	6	8, 9, 10	8	1, 3, 6, 8		
	8	9, 11	11	1, 3, 6, 8, 11	2, 3, 4	30

k : number of paths

i : start node

S_i : the set of nodes with all nodes adjacent to node i

l : sink node

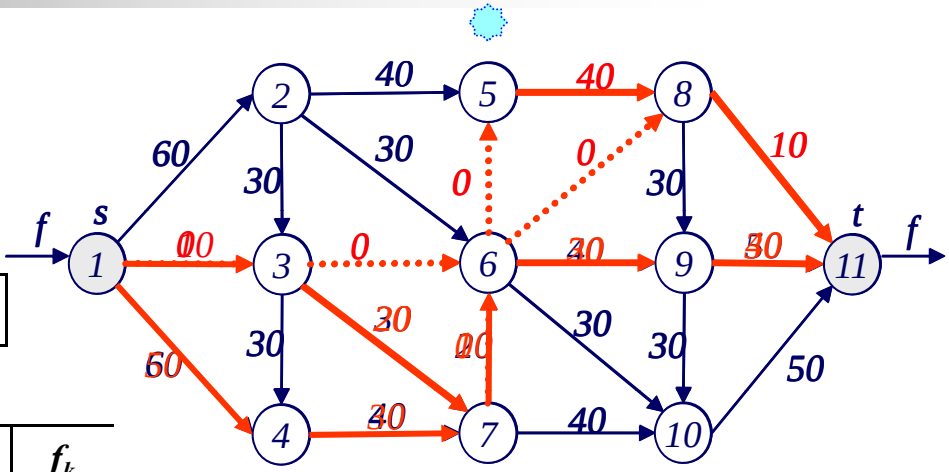
P_k : the kth path

f_k : the flow of kth path

GA求解

Chromosome:

node ID :	1	2	3	4	5	6	7	8	9	10	11
priority :	2	1	6	4	11	9	8	10	5	3	7



k	i	S_i	l	P_k	S_1	f_k
3	0			1		
	1	2, 3, 4	3	1, 3		
	3	4, 7	7	1, 3, 7		
	7	6, 10	6	1, 3, 7, 6		
	6	9, 10	9	1, 3, 7, 6, 9		
	9	10, 11	11	1, 3, 7, 6, 9, 11	2, 4	10
4	0			1		
	1	2, 4	4	1, 4		
	4	7	7	1, 4, 7		
	7	6, 10	6	1, 4, 7, 6		
	6	9, 10	9	1, 4, 7, 6, 9		
	9	10, 11	11	1, 4, 7, 6, 9, 11	2,4	10

***k* : number of paths**

***i* : start node**

S_i* : the set of nodes with all nodes adjacent to node *i

***l* : sink node**

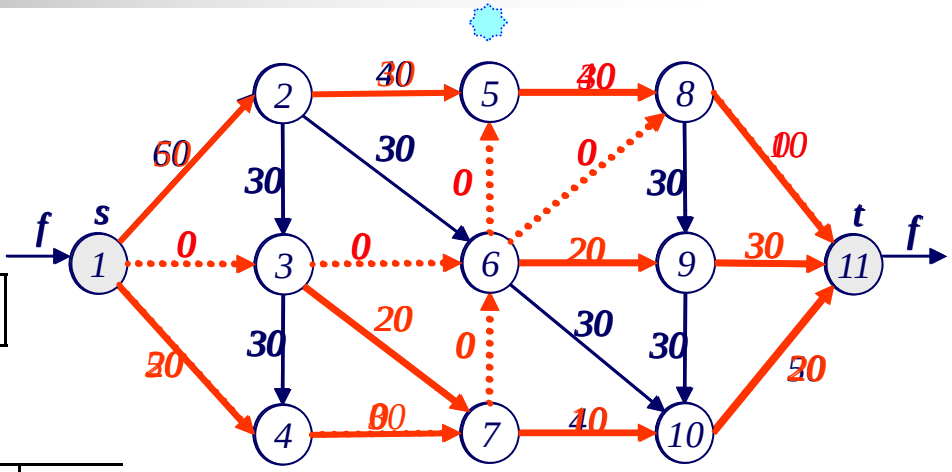
***P_k* : the kth path**

***f_k* : the flow of kth path**

GA求解

Chromosome:

node ID :	1	2	3	4	5	6	7	8	9	10	11
priority :	2	1	6	4	11	9	8	10	5	3	7



k	i	S_i	l	P_k	S_1	f_k
5	0			1		
	1	2, 4	4	1, 4		
	4	7	7	1, 4, 7		
	7	10	10	1, 4, 7, 10		
	10	11	11	1, 4, 7, 10, 11	2	30
6	0			1		
	1	2	2	1, 2		
	2	3, 5, 6	5	1, 2, 5		
	5	8	8	1, 2, 5, 8		
	8	9, 11	11	1, 2, 5, 8, 11	2	10

k : number of paths

i : start node

S_i : the set of nodes with all nodes adjacent to node i

l : sink node

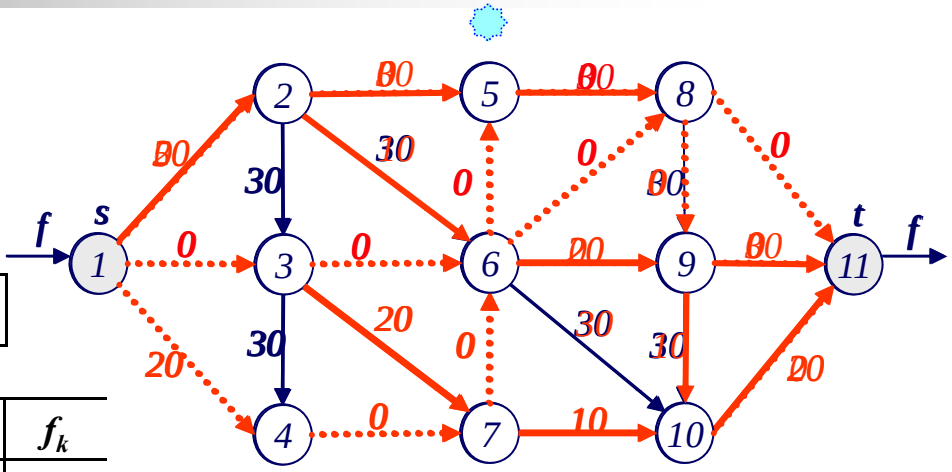
P_k : the kth path

f_k : the flow of kth path

GA求解

Chromosome:

node ID :	1	2	3	4	5	6	7	8	9	10	11
priority :	2	1	6	4	11	9	8	10	5	3	7



k	i	S_i	l	P_k	S_1	f_k
7	0			1		
	1	2	2	1, 2		
	2	3, 5, 6	5	1, 2, 5		
	5	8	8	1, 2, 5, 8		
	8	9	9	1, 2, 5, 8, 9		
	9	10, 11	11	1, 2, 5, 8, 9, 11	2	30
8	0			1		
	1	2	2	1, 2		
	2	3, 6	6	1, 2, 6		
	3	9, 10	9	1, 2, 6, 9		
	4	10	10	1, 2, 6, 9, 10		
	5	11	11	1, 2, 6, 9, 10, 11		20

k : number of paths

i : start node

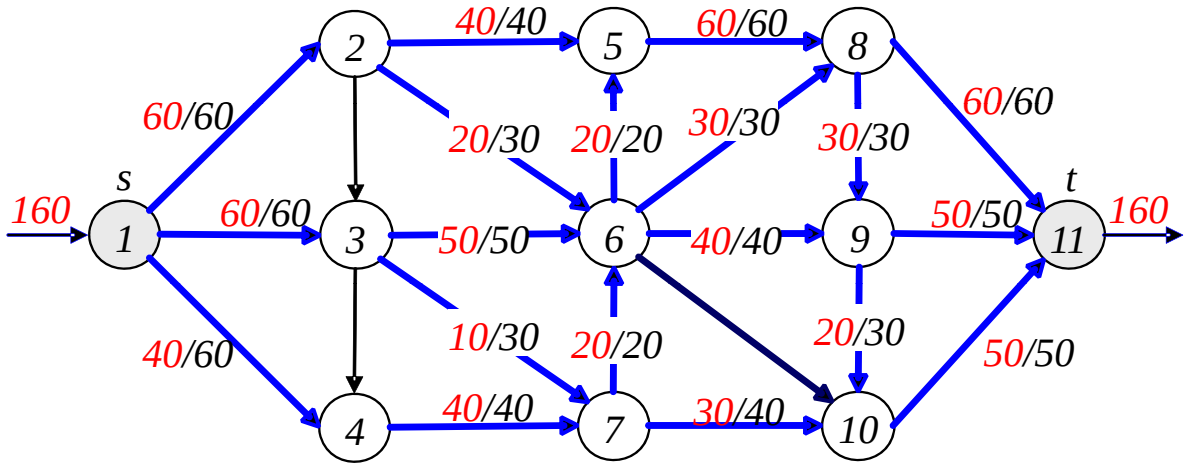
S_i : the set of nodes with all nodes adjacent to node i

l : sink node

P_k : the kth path

f_k : the flow of kth path

GA求解



Chromosome:

node ID :	1	2	3	4	5	6	7	8	9	10	11
priority :	2	1	6	4	11	9	8	10	5	3	7

Objective function value： $z=20+30+10+10+30+10+30+20=160$

GA求解

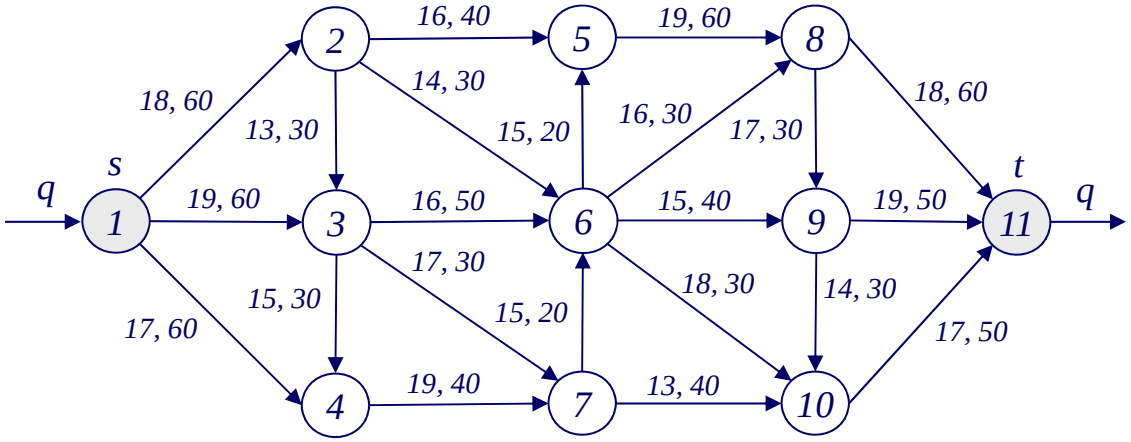
- 交叉
  - **WMX**
  - **PMX**
- 变异
  - 互换变异
- 选择
  - 轮盘赌

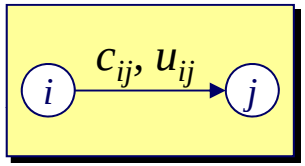
最小费用流问题

(Minimum Cost Flow Problem)

最小费用流问题

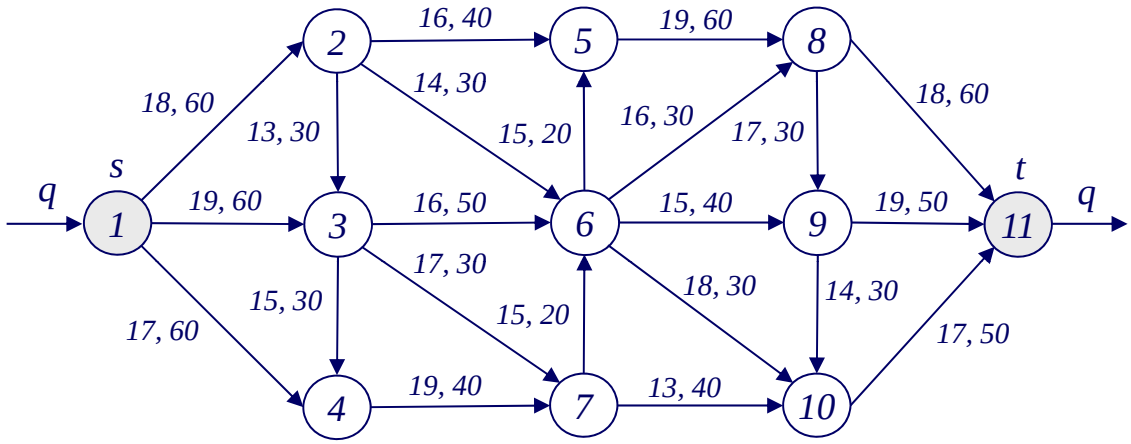
- 在涉及流的系统当中，人们有时不仅要考虑流量，还要考虑“费用”的因素。
- MCF 模型，是所有网络设计问题的最基本的模型。
- 通过网络发送流，以满足供应和需求的关系，问题是：选择什么样的路径（计划）才能使总费用最小。
- 每条弧 (i, j) 具有容量 u_{ij}、单位成本c_{ij}，每个节点具有一个固定的外部流。

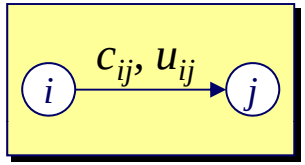




网络模型

- 网络模型可描述为：
  - 有向图：$G=(V, A)$，其中 V 是节点的集合， A 是连接的集合
  - u_{ij} 是和每个连接 (i, j) 相关的容量
  - c_{ij} 是和每个连接 (i, j) 相关的单位成本
  - 起点: 节点 1
  - 终点: 节点 m





模型描述

- 用数学公式可描述为：

$$\min \quad z = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} q & (i=1) \\ 0 & (i=2,3,\cdots,m-1) \\ -q & (i=m) \end{cases}$$

$$0 \le x_{ij} \le u_{ij}, \quad (i,j) \in A$$

where,

x_{ij}：从点 i 到点 j 的流量；
u_{ij}：和每个连接 (i, j) 相关的容量 (最大流量)；
c_{ij}：和每个连接 (i, j) 相关的单位成本；
q：全部流的值（供给、需求量）

- 约束条件的含义是：
  - 节点 1 只有流出的流；
  - 节点 *m* 只有流入的流；
  - 中间的流入、流出的流是相等的。

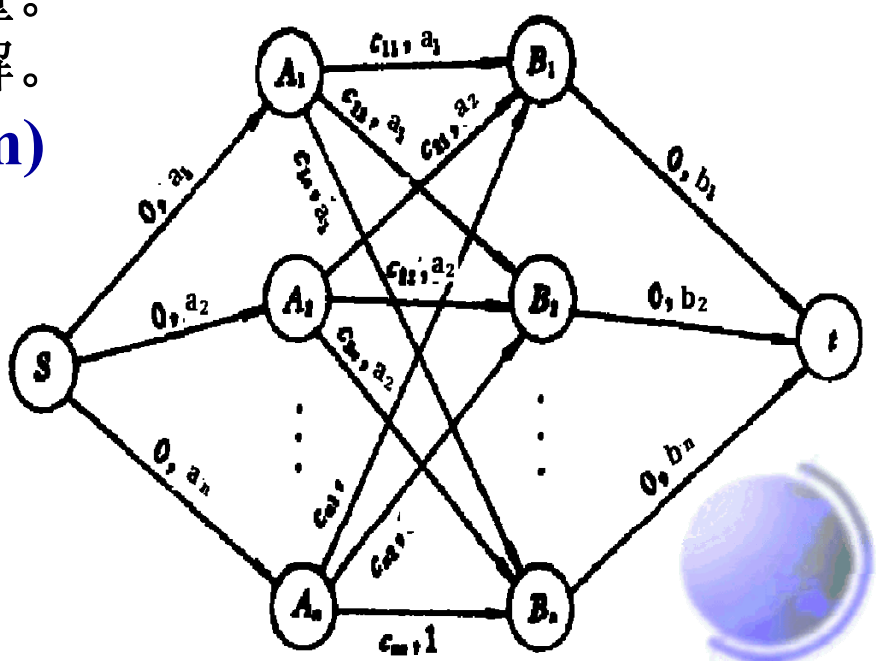
特点

- 一类特殊的线性规划问题
- 当 q=1 时，问题退化为 SPP 问题

应用

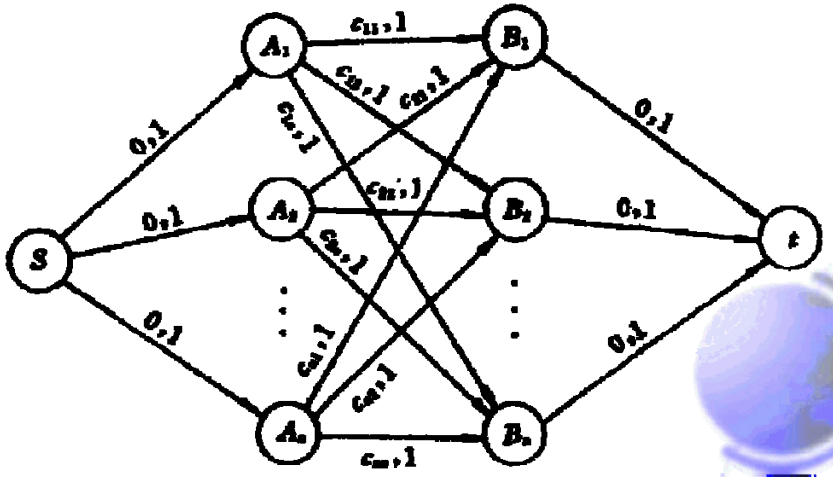
- 运输问题 (Transportation Problem)
  - 运输问题有多个起点和终点，有供给和需求，从不同的供给点到需求点运输货物，存在单位运输费用。
  - 设一个虚拟源点和所有的起点相连，设一个虚拟汇点和所有需求点相连。运输问题的目标函数就相当于最小费用流的目标函数，每条边有不同的运输成本，供给量相当于虚拟源点到供给点的边的容量，需求量相当于需求点到汇点的边的容量，运输边的容量可以设为供给点的容量。
  - 可应用最小费用流算法进行求解。
- 分配问题 (Distribution Problem)

  （和运输问题类似）
  - 从制造厂到仓库的产品分配，
  - 或从仓库到零售商的产品分配



应用

- 带有配送中心的运输问题（物流配送问题）
  - 在实际的物资运输中，对于运输量很大、运输路程很远的物资，一般都会在产地和销地之间设有配送中心。可以提高运输的效率，减少运输费用，节约运力、缓解交通紧张的矛盾等。
- 用于求解指派问题
  - 节点Ai表示资源，节点Bj表示工作，这两组节点之间的弧表示指派关系，而弧上第一个数字为指派费用第二个数字为弧的容量。
  - 节点S是一个虚拟发点，到每个Ai刚好有一条弧，单位费用为0，而容量均为1。节点t是虚拟收点，从Bj到t刚好有一条弧，单位费用均为0，而容量均为1。
  - 指派问题就等价于求从S到t的一个流量值为n的最小费用流问题。
- 在配电网网络重构中的应用
- 能源综合输送系统规划
- ……



一般的求解方法

- 连续最短路算法（Successive Shortest Path Algorithm ）
  - 连续的最短路径算法保持了每一步解决方案的最优化并且力争获得可行解。
- 原始-对偶算法（ Primal-dual Algorithm ）
  - 原始-对偶算法应用于最小费用流问题与连续的最短路径算法类似，也是通过维持一条伪流来满足缩减成本的最优化条件。通过在最短路径上增大流量的方法来逐渐使其转化为可行流。
- 瑕疵算法（ Out-of-Kilter Algorithm）
  - 瑕疵算法是只满足（其中）大多数平衡条件的方法。所以中间的处理过程可能既违反最优性条件也违反流的范围约束。
- 松弛算法（Relaxation Algorithm）
- 网络单纯形算法(Network Simplex Algorithm)

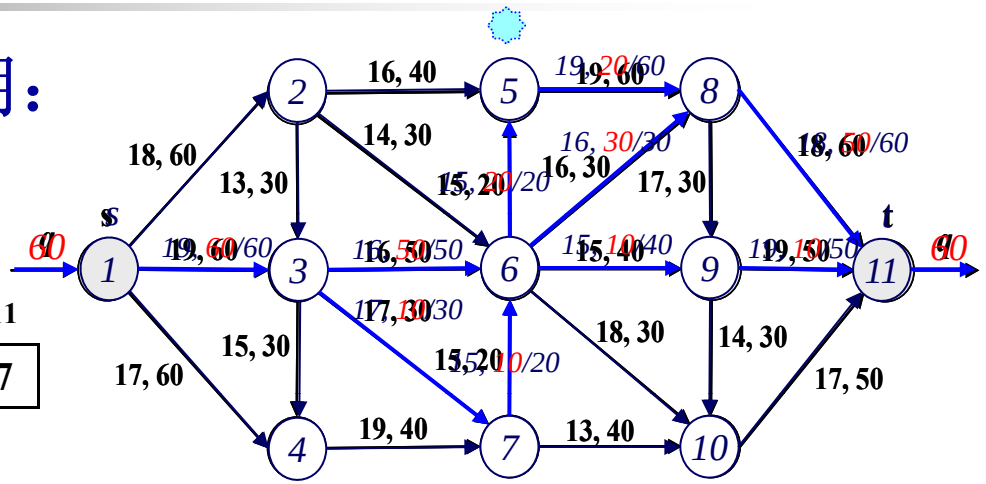
GA求解

- 基于优先级的编码
  - 与 SPP 的编码方式相同

- 解码过程分为两个阶段：
  - 第一阶段：采用单路径生长过程产生路径。从节点 1 到 m，每一步都将具有最高优先权的节点加入路径
  - 第二阶段：采用全路径生长过程产生全路径。
    - 对给定的一条路径，计算其流量 f_k 和 c_k
    - 通过从每条弧的 u_{ij} 中移除所使用的量，我们可以得到一个具有新的容量限制 $\bar{u}_{ij}$ 的网络
    - 通过单路径生长过程，我们可以获得第二条路径。
    - 通过重复这一过程，我们可以获得给定染色体的最大流，直到$\Sigma f_k \geq q$ 为止。

GA求解

- 基于优先级编码的图式说明：



Chromosome:

node ID :	1	2	3	4	5	6	7	8	9	10	11
priority :	2	1	6	4	11	9	8	10	5	3	7

流量要求： q = 60

k	*i*	S_i	*l*	P_k	S_1	f^k	c^k
1	0			1			
	1	2, 3, 4	3	1, 3			
	3	4, 6, 7	6	1, 3, 6			
	6	5, 8, 9, 10	5	1, 3, 6, 5			
	5	8	8	1, 3, 6, 5, 8			
	8	9, 11	11	1, 3, 6, 5, 8, 11	2, 3, 4	20	87
2				1, 3, 6, 8, 11	2, 3, 4	30	69
3				1, 3, 7, 6, 9, 11	2, 4	10	85

*目标函数值： z = 20*87+30*69+10*85*
= 4660

k : number of paths
i : start node
S_i : the set of nodes with all nodes adjacent to node i
l : sink node
P_k : the kth path
f^k : the flow of kth path
c^k : minimum possible cost

双准则网络设计问题

(Bicriteria Network Design Problem)

双准则网络设计问题

- 实际应用当中，网络设计常常需要同时优化多个目标。
- 当我们进行具体设计的时候，就可能发生这样的事情：
  - 在通信网络当中，为了提高网络性能，寻找既考虑低成本（或延迟）又考虑高吞吐量(或可靠性)的链接的集合；
  - 在制造系统当中，通常考虑的两个准则是最小化成本和最大化生产；
  - 在物流系统当中，改善物流生产率的主要驱动力是通过合同周期（交付周期）和物流成本的显著减少，改善客户服务和资产利用情况。
- 由于两个目标通常是相互冲突的，所以无法获得问题的最优解，因此对于决策者，更感兴趣去获得整个 **Pareto** 解的集合。
- 通常的 **BNPs** 问题被定义为极小化两个目标（不同的成本函数），并被扩展到许多多目标网络设计问题当中。

最小费用最大流问题

- 最小费用最大流问题比一般的双准则网络设计问题更复杂。
- 与一般的 **BNP** 的区别是：
  - 问题的有效路径集合可能非常大，随问题的规模可能是指数增长的；
  - 因而，在最坏的情况下，求解问题的计算时间可能会随着问题的规模按指数增长。
- 在具有流容量和费用的网络中，最小费用最大流问题就是同时确定从源点到汇点的最大流 z_1 和最小费用 z_2。

模型描述

- 用数学公式可描述为：

$$\max \quad z_1 = f$$

$$\min \quad z_2 = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij}$$

$$\text{s. t.} \quad \sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} f & (i=1) \\ 0 & (i=2,3,\cdots,m-1) \\ -f & (i=m) \end{cases}$$

$$0 \le x_{ij} \le u_{ij} \quad ,\forall (i,j) \in A$$

$$f \ge 0$$

where,

x_{ij}：*从点 i 到点 j 的流量；*

u_{ij}：*和每个连接 (i, j) 相关的容量 (最大流量)；*

c_{ij}：*和每个连接 (i, j) 相关的单位成本；*

- 约束条件的含义是：
  - 节点 1 只有流出的流；
  - 节点 *m* 只有流入的流；
  - 中间的流入、流出的流是相等的。

特点

- NP-难的问题
- 不是一个简单的单目标到多目标的扩展

GA求解

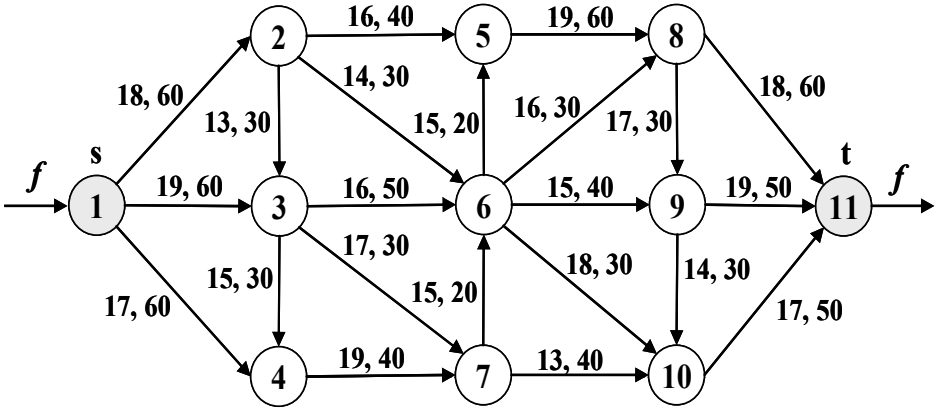
- 基于优先级的编码过程
  - 与 SPP 的编码方式相同

- 解码过程分为两个阶段：
  - 第一阶段：采用单路径生长过程产生路径。从节点 1 到 m，每一步都将具有最高优先权的节点加入路径
  - 第二阶段：采用全路径生长过程产生全路径。
    - 对给定的一条路径，计算其流量 f_k 和 c_k
    - 通过从每条弧的 u_{ij} 中移除所使用的量，我们可以得到一个具有新的容量限制 $\bar{u}_{ij}$ 的网络
    - 通过单路径生长过程，我们可以获得第二条路径。
    - 通过重复这一过程，我们可以获得给定染色体的最大流，直到不能采用这种方法定义新的网络为止。

GA求解

- 基于优先级编码的图式说明：

node ID :	1	2	3	4	5	6	7	8	9	10	11
priority :	2	1	6	4	11	9	8	10	5	3	7



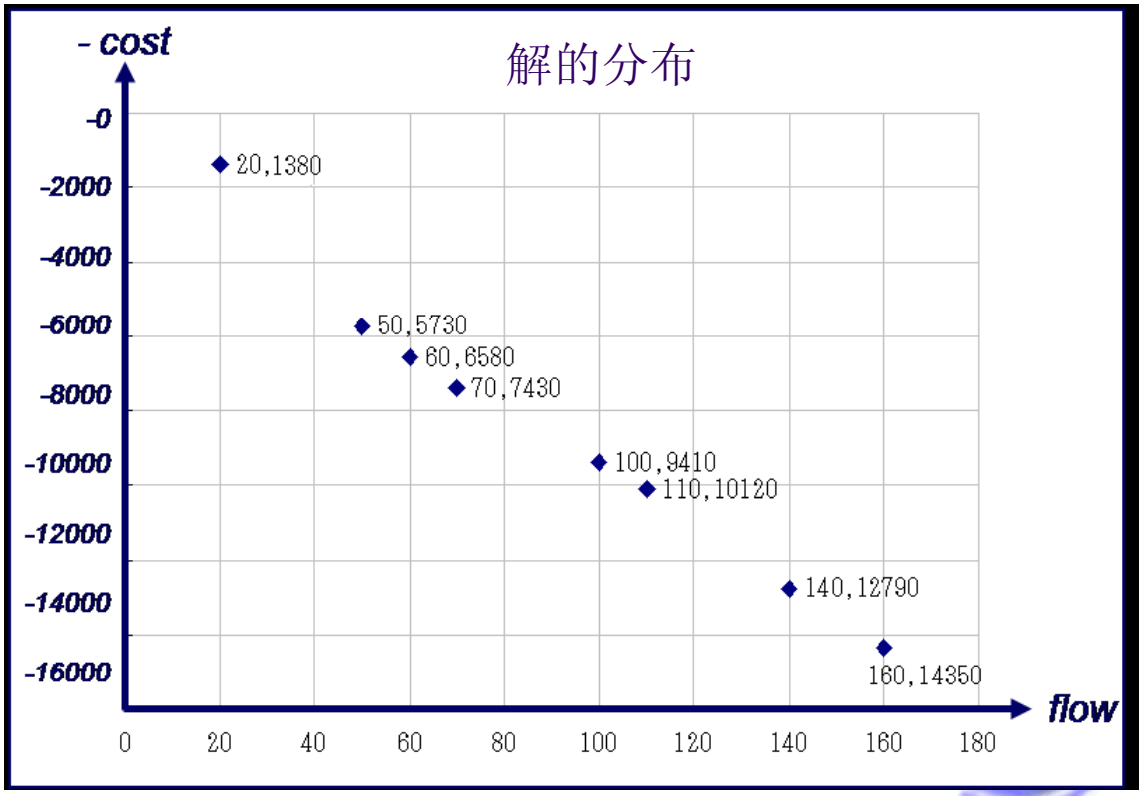
k	P_k	z_1^k	z_2^k
1	1, 3, 6, 5, 8, 11	20	1380
2	1, 3, 6, 8, 11	50	5730
3	1, 3, 7, 6, 9, 11	60	6580
4	1, 4, 7, 6, 9, 11	70	7430
5	1, 4, 7, 10, 11	100	9410
6	1, 2, 5, 8, 11	110	10120
7	1, 2, 5, 8, 9, 11	140	12790
8	1, 2, 6, 9, 10, 11	160	14350

k : *number of paths*

P_k : *the kth path*

z_1^k : *maximum possible flow*

z_2^k : *minimum possible cost*



GA求解

- 适值的确定
  - 基于AWA（Adaptive Weight Approach）的自适应评价函数（Cheng、Gen等提出）
  - 计算过程如下：
    - 令：染色体为: v_k , $k \in popSize$, 路径数为: L_k ,流为: f_i^k ， 成本为: c_i^k ，其中 $i \in L_k$ 为第 k 个染色体中的第 i 条路径。
    - 步骤1：定义两个特殊点，由当前种群中两个目标的最大、最小值构成

$$z^+ = \{z_1^{\max}, z_2^{\max}\} \qquad z^- = \{z_1^{\min}, z_2^{\min}\}$$

$$z_1^{\max} = \max\{ f_i^k \mid i \in L_k, k \in popSize \} \quad z_2^{\max} = \max\{ -c_i^k \mid i \in L_k, k \in popSize \}$$

$$z_1^{\min} = \min\{ f_i^k \mid i \in L_k, k \in popSize \} \quad z_2^{\min} = \min\{ -c_i^k \mid i \in L_k, k \in popSize \}$$

    - 步骤2：计算目标 1 和目标 2 的自适应权重

$$w_1 = \frac{1}{z_1^{\max} - z_1^{\min}}, \quad w_2 = \frac{1}{z_2^{\max} - z_2^{\min}}$$

    - 步骤3：计算染色体的适值函数

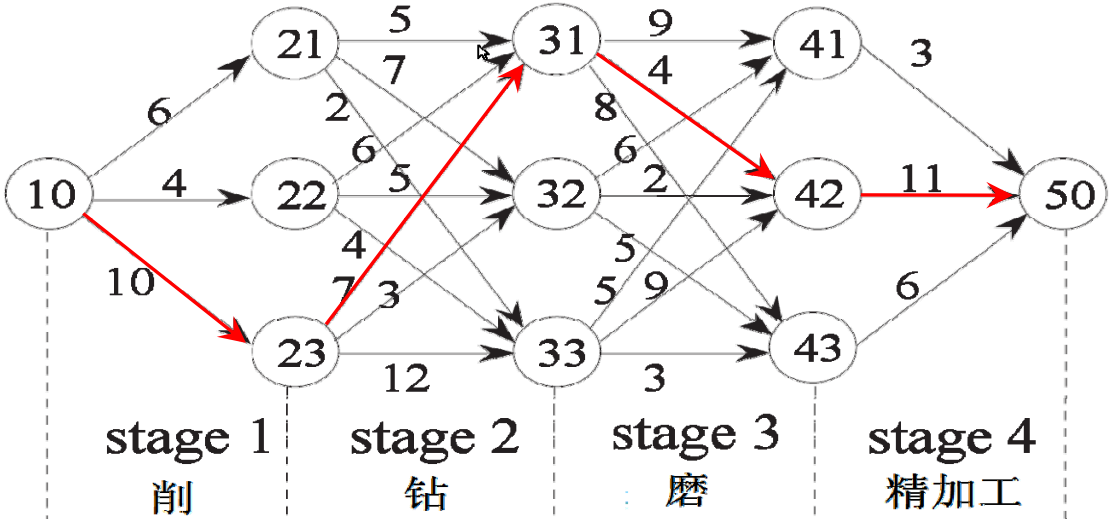
$$eval(v_k) = \frac{\sum_{i=1}^{L_k} \left(w_1 (f_i^k - z_1^{\min}) - w_2 (c_i^k - z_2^{\min}) \right)}{L_k}, \quad \forall k \in popSize$$

多阶段工序计划问题

(Multi-stage Process Planning Problem)

问题描述

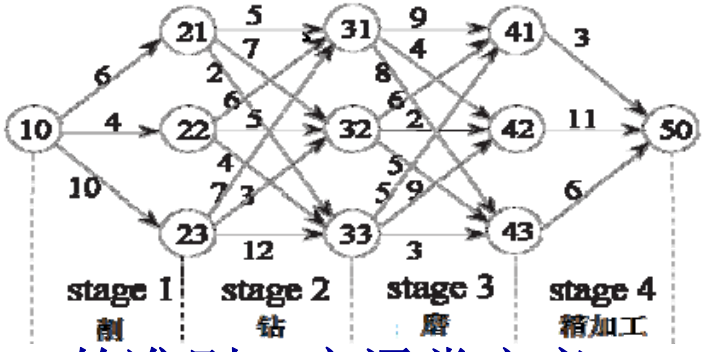
- 多阶段工序计划(MPP)问题在制造系统中很常见。
- MPP系统通常包含一系列的机械操作，诸如削、钻、磨、精加工等，将一零件加工成最终形态或产品。
- 整个过程可以分为几个阶段，在每个阶段。都有一套相似的加工操作。
- MPP问题就是要在给定的目标，如成本最小、时间最小、质量最好，或是其中的多个目标下，从所有可能的可选计划中，寻找最优的工序加工计划。如图所示。



模型描述

- 对于一个n阶段的MPP问题，令s_k是阶段k的一种状态，$D_k(s_k)$是在阶段k可能的状态集合，x_k是阶段k确定选择某种状态的决策变量，显然$x_k \in D_k(s_k)$，MPP问题可以构造为：

$$\min_{\substack{x_k \in D_k(s_k) \\ k=1,2,\ldots,n}} V(x_1, x_2, \ldots, x_n) = \sum_{k=1}^{n} v_k(s_k, x_k)$$



- 其中，$v_k(s_k, x_k)$代表在阶段k状态s_k中确定x_k的准则，它通常定义为实数，如成本、时间或距离。

- 问题可以改写为动态递归表达式(dynamic recurrence expression)，如果$f_k(x_k, s_k)$代表从阶段k到最终阶段(n)的最优工序加工计划，则问题的动态递归可表达如下：

$$f_k(x_k, s_k) = \min_{x_k \in D_k(s_k)} \{v_k(s_k, x_k) + f_{k+1}(x_{k+1}, s_{k+1})\},\ k = 1, 2, \ldots, n-1.$$

求解难度

- 尽管利用最短路径方法和动态规划可以解决这类问题，
- 但随着问题规模的增大，许多阶段和状态都必须考虑，这将大大影响最短路径法或动态规划求解最优解时的效率。
- 特别是，对于一个多目标的**MPP**问题。传统的技术无法使用，尽管有人运用了基于目标规划的技术，但这些技术只能处理小型网络。

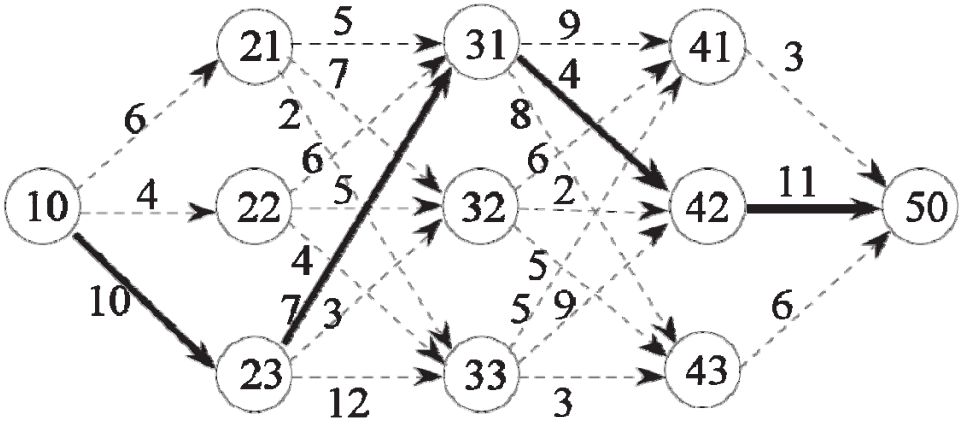
遗传算法求解

- 状态排列编码
  - 对于MPP问题，每个阶段的可选状态可以用一系列整数表示。在工序计划的某个阶段上选择某项作业的一个状态，相当于选择一个整数，该整数不大于该阶段可能的状态数。
  - MPP问题可以以状态排列表示的形式简明地编码。
  - 这种状态排列编码与MPP问题是1－1对应的，交叉变异、解码和评价也很容易。

状态排列编码

3	1	2

基因的取值范围取决于相应阶段的状态数



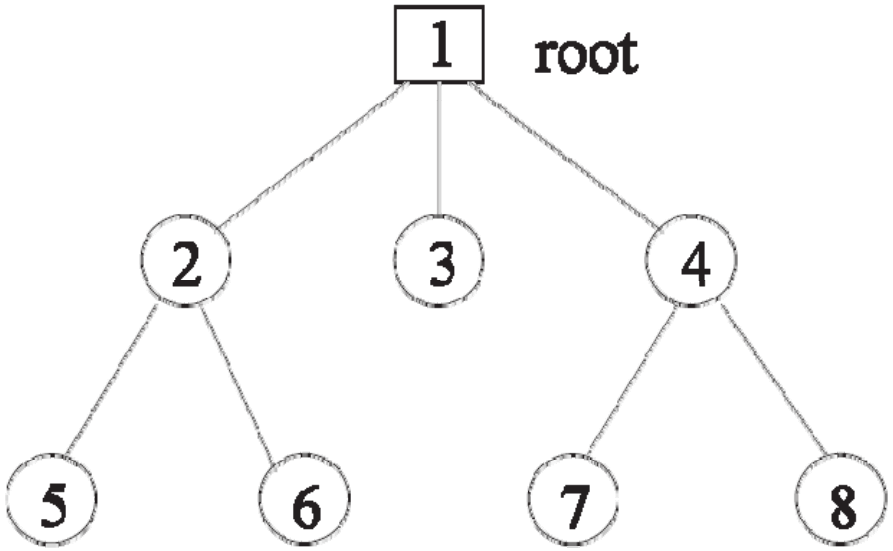
  - 状态排列编码具有两大优势：
    - 对于*n*个阶段的MPP问题，编码长度仅为：*n*-1，节省空间；
    - 通常的交叉变异总是产生可行的后代。

集中式网络设计问题

(Centralized Network Design Problem)

集中式网络的概念

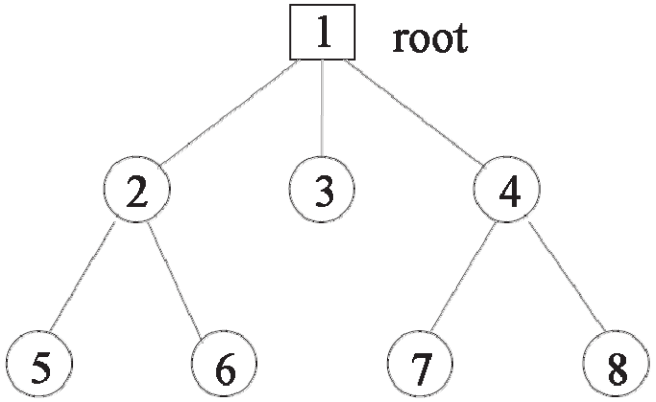
- 集中式网络是一个所有通信都来往于一个点的网络。
- 在这样的网络中，终端被直接连接到中央节点，或一组终端形成一颗树，通过共享一条链路（多点线路）连接到中央节点。
- 这意味着这一问题的最优拓扑结构对应图中的一颗树，除一个节点外，其它的节点都对应着一个终端，边对应着通信的线路。子树对应一条多点线路。
- 通常在通信中，中央节点最多可以处理给定固定量的信息。反过来，这也相应地限制了任何连接到中央节点的链路上的信息传输最大量。



问题的描述

- 考虑完全无向图 $G=(V,E)$，
- 令$V=\{1,2,\ldots,n\}$代表终端节点集合，1表示中央节点或“根”节点。
- 令$E=\{(i,j)|i,j\in V\}$代表所有可能远程通信线路的边的集合
- 对于节点子集$S(\subset V)$，定义$E(S)=\{(i,j)|i,j\in S\}$为两端点都在S中的边
- 对于所有的边$(i,j)\in E$，定义下面的二元决策变量：

$$x_{ij} - \begin{cases} 1, & \text{if edge } (i,j) \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$



- 令 c_{ij}是解中边 (i,j) 的固定成本。
- 假设 d_i 代表节点 i 的需求量，其中约定根节点的需求量 $d_1=0$.
- $d(S)$ 代表 S 中节点需求量之和
- 子树的容量用 k 表示

问题的描述

- 集中式网络设计问题可以描述如下：

$$\min \ z = \sum_{i=1}^{n-1}\sum_{j=2}^{n} c_{ij}x_{ij}$$

$$\text{s. t.} \quad \sum_{i=1}^{n-1}\sum_{j=2}^{n} x_{ij} = n-1$$

$$\sum_{\substack{i\in S \\ j>1}}\sum_{j\in S} x_{ij} \le |S| - \lambda(S), \quad S \subseteq V\backslash\{1\}, \quad |S| \ge 2$$

$$\sum_{\substack{i\in U \\ j>1}}\sum_{j\in U} x_{ij} \le |U| - 1, \quad U \subset V, \quad |U| \ge 2, \quad \{1\} \in U$$

$$x_{ij} = 0 \text{ or } 1, \quad i = 1, 2, \ldots, n-1, \quad j = 2, 3, \ldots, n.$$

- 对于所有的生成树问题，约束1恒成立，一棵 n 个节点的树，必定有n-1条边。
- 约束3对于生成树是一个标准不等式。如果多于$|U|$-1条边连接子集U中的节点，则子集U中必然包含圈。
- 不等式2中的$\lambda(S)$是指集合S的装箱数，即用大小为 k 的箱子装项目大小为d_i的节点所需要的箱子数目，对所有的 $i \in S$。
- 不等式2除了反映容量限制以外，这些约束类似于不等式3。
- 如果集合S不包含根节点，则S的节点必须包含在至少$\lambda(S)$棵不同的根的子树中

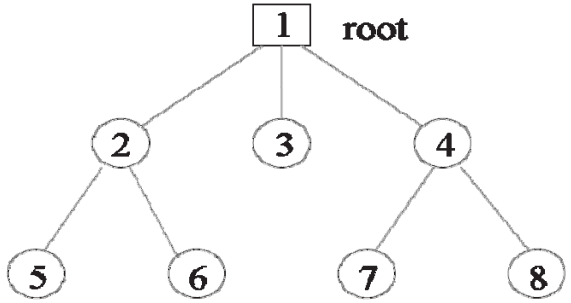
问题的求解难度

- **Papadimitriou**认为这是**NP-**难题。
- 许多早期的工作利用启发式方法寻求良好的可行解。
- 目前，所有这类问题的启发式算法仅集中于如何处理约束，以使问题更易于求解。
- 在割平面算法和分支定界算法中，问题的拓扑结构往往被忽略。因此，这会导致约束个数的指数膨胀。

遗传算法求解

- 基于树的排列
  - 为了利用遗传算法解决集中式网络设计问题，候选解的编码可采用基于“树”的排列编码，如图所示，一个编码由两部分组成：
    - 顶点编码部分，基因值不重复地在1~*n*之间取值
    - 度值编码部分，基因值在1~*b*之间取值，*b*是所有顶点的度约束
  - 其编码过程与最小生成树问题中的基于“度”的排列编码类似，
  - 根据集中式网络的特点：
    - 顶点编码部分的第一个基因恒定为1。
    - 特别地，对于根节点，它的度不应小于$\lceil |V|/k \rceil$（当非根节点的需求量为1时），这反映了连接到根节点的子树的数量满足容量限制。

顶点编码	1	2	5	6	3	4	7	8
度值编码	3	3	1	1	1	3	1	1



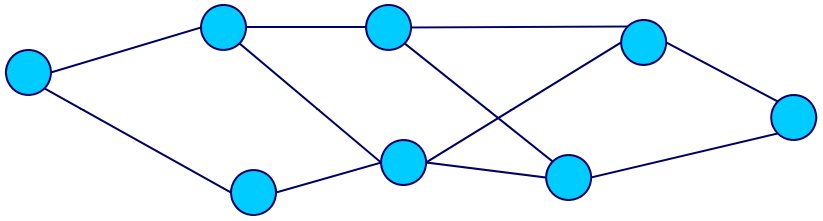
  - 其它操作（解码、交叉、变异等），可参见求解度约束最小生成树问题的基于“度”的排列编码，对进化过程中产生的超出根节点容量限制的染色体进行修正是必须的。

计算机网络扩展问题

(Computer Network Expansion Problem)

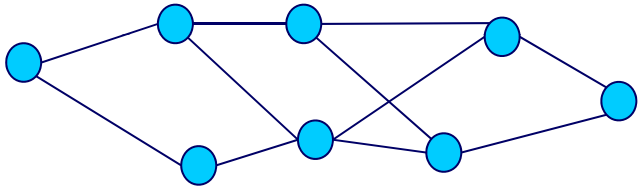
计算机网络扩展的概念

- 低价计算设备的出现导致了计算机网络的爆炸性发展，为网络的分布式计算带来了不少好处。
- 计算机网络较之集中式系统的主要优势之一就是提高系统的可靠性
  - 系统的可靠性不仅依靠它的节点和通信线路的可靠性，还依赖于节点是如何通过通信线路被连接的(例如网络的拓扑结构)。
  - 网络拓扑结构可由无向图$G=(V,E)$及其网络的可靠性、信息延迟和网络容量来刻画，这些性能特性依赖于图的许多属性，例如：直径、平均距离、节点的端口数(节点的度)和边的数量
  - 可靠性随直径和平均距离的减少而增加，随边数的减少而降低。
  - 直径越大，网络中的延迟越长。
- 网络扩展涉及更多跨网络计算而带来的不断增长的需求。
  - 为满足这一需求，网络的大小随着用户需求的增加不断地膨胀。在这种情况下，通过适当地增加新的通信线路和计算机节点实现扩展，在特定的限制条件下，优化网络的可靠性度量。



问题的描述

- **Kumar, Pathak and Gupta** 开发了基于遗传算法的计算机网络扩展方法，用于在给定的一组网络约束下，优化特定的目标函数（可靠性度量）
- 对于网络扩展问题，定义如下假设：
  - 网络的拓扑结构由无向图G描述，
  - 节点代表处理单元，边代表通信线路，
  - 图G中不存在任何自回路，
  - 任何边的故障统计意义上是相互独立的，
  - 边有两个状态：运行和故障
  - 网络是统计相关的。



- 在直径和度约束下，最大化计算机网络可靠性的网扩扩展问题可以构造如下：

$$\max\ R(\boldsymbol{x})$$

$$\text{s. t.}\ \sum_{j=1}^{n} x_{ij} \le k_i, i = 1, \cdots, n$$

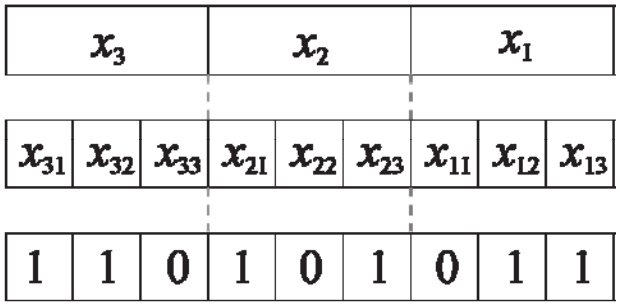
$$\max\{d_{ij} | i = 1, \cdots, n; j = 1, \cdots, n\} \le D$$

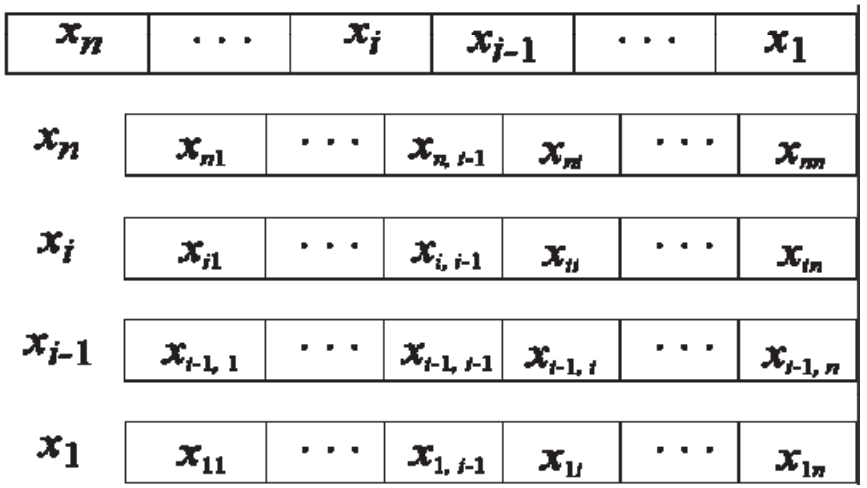
$$x_{ij} = \begin{cases} 1, & \text{if edge exists between i,j} \\ 0, & \text{otherwise} \end{cases}$$

  - 其中n是节点数，k_i表示节点i的度。d_{ij} 是节点i和j间跳跃的最小数，D是网络直径的上界。

Kumar，Pathak和Gupta的方法

- 染色体编码
  - 为了表示节点的连通性，可以使用一种二进制串表示的编码方法。
  - 网络连通性的完整染色体被分成节点域。节点域的多少等于新的节点加入现有网络后网络中的节点数。
  - 图中给出了三个节点网络连通性的一种排列，其中，一个新节点x'_3加入到现有的两节点(x_1,x_2)网络。

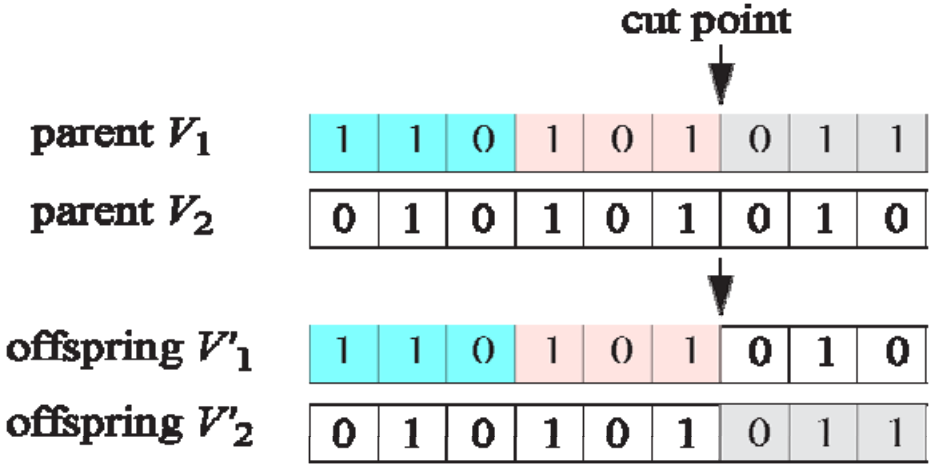




  - 一般来说，如果在扩展的网络中有n个节点，开始的i-1个节点表示原先网络中的节点，则剩下的节点是新增入网络的。
  - 扩展网络的染色体如图所示。通常x'_{ii}和x_{ii}为0，因为图中不允许存在自回路。
- 初始种群的产生
  - 初始种群完全是随机产生的，在初始化中不使用任何准则产生有偏的或有适应能力的种群。

Kumar，Pathak和Gupta的方法

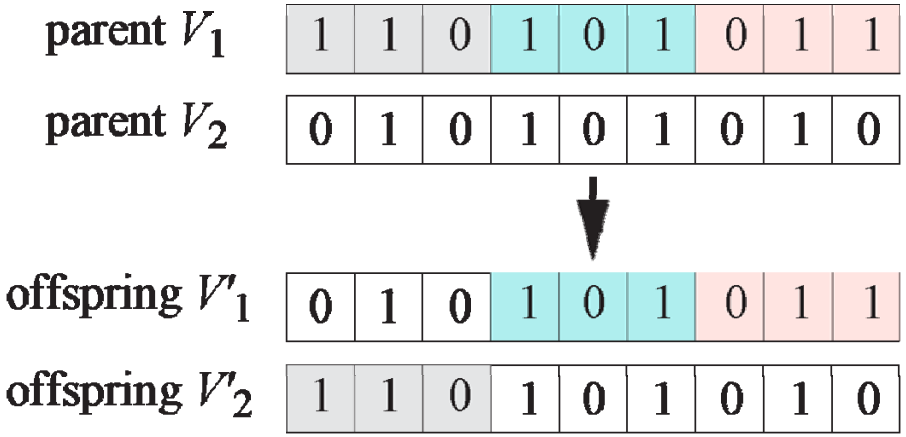
- 交叉运算分为两类：
  - SNFSO(简单节点域交换运算)
  - RNFSO(随机节点域交换运算)
- SNFSO(简单节点域交换运算)
  - 杂交发生在随机选出的两个染色体的节点域边界上。
  - 然后，利用随机数发生器在(0,*n*-1)上产生一整数作为杂交点。



  - 杂交运算会产生网络连通异常的染色体。如子代1中的节点x_3域显示了节点3连接节点1和2，但x_1域表明节点1仅连接节点2。
  - 这种交叉的异常可以通过调整运算改进

Kumar，Pathak和Gupta的方法

- **RNFSO(随机节点域交换运算)**
  - 每次随机交换染色体中的节点域。
  - 为交换一特定标号的节点域，随机数将在[0,*n*-1]中产生。
  - 假设要交换的是节点x_3域，其交叉过程如下：



① 令j=1
② 如果x_{ij}=1则转③，否则转④
③ 如果x_{ji}=0，则令x_{ji}=1
④ 如果x_{ji}=1，则令x_{ji}=0
⑤ j=j+1，
⑥ 如果j>n则停止，否则转②

  - 与SNFSO方法一样，RNFSO也会产生异常的染色体，同样可以通过调整运算改进。
- 异常染色体的调整运算
  - 由于交叉运算生产了染色体连通性上的矛盾，需要进行调整以便使之可行，步骤如图所示。

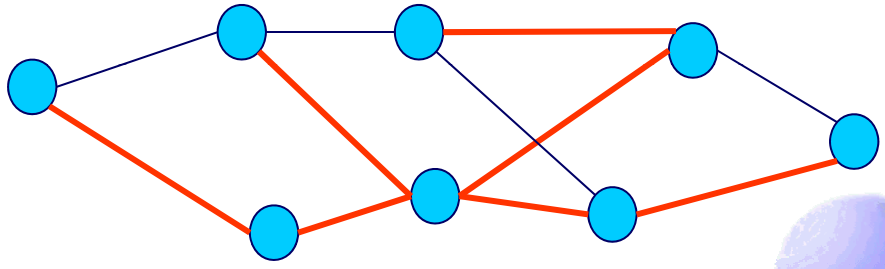
Kumar，Pathak和Gupta的方法

- 变异
  - 如果说杂交运算稍稍降低了节点域的连通性，则变异运算将提高节点域的连通性。
  - 变异步骤如下：
    ① 从那些加入现有网络的，且度数小于上界的点中，随机选择节点域x_i。
    ② 随机选择另一个不与x_i相连的且度数小于上界的节点x_k。
    ③ 如果不存在这样的点，则执行第1步；如果尝试次数超过了指定的次数，则停止。
    ④ 如果找到了合适的节点x_i和x_k，通过令节点域x_i中的x_{ik}位为1，令节点域x_k中的x_{ki}位为1，产生一条新的边。

Kumar，Pathak和Gupta的方法

- 适应值和可靠性的计算
  - 既然目标函数是可靠性，则适应值可由可靠性的计算决定
  - 计算机网络的可靠性（**Compter network reliability**）定义为在给定网络中，每个节点与其它所有节点通信的可能性。
  - 网络可靠性的计算：
    - 计算网络中的所有生成树T，任何T_i通过n-1条边连接G中所有的点，因此是满足节点间通信的最小链接。
    - 利用预先给出的边的可靠性（概率 p），计算被操作的生成树的可靠性，进而用于网络可靠性的计算。

$$\sum_{T_i \in T}\left(\prod_{e \in Ti} p_e\right)\left(\prod_{e \in (E \setminus Ti)}(1-p_e)\right)$$



End