城市空间建模与仿真

第十四讲 城市空间三维数据特征学习与识别-Graph-Cut点云分割

任课教师：汤圣君
建筑与城市规划学院 城市空间信息工程系

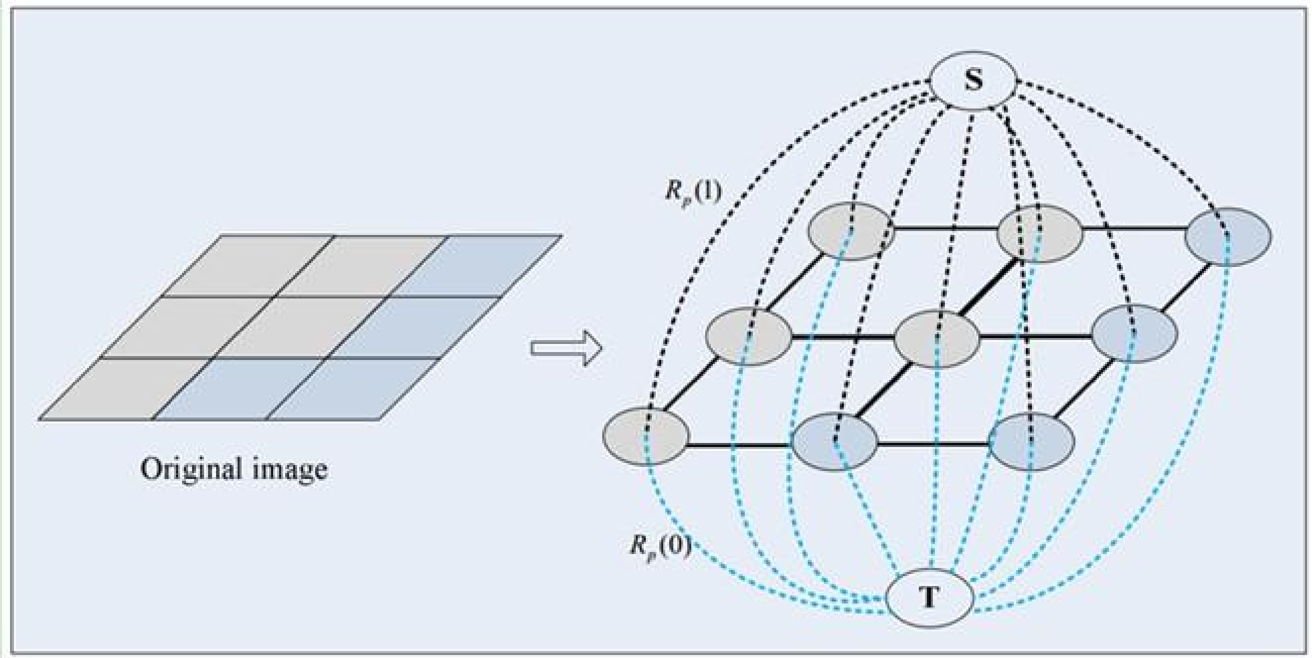
目录

CONTENTS

目录
CONTENTS

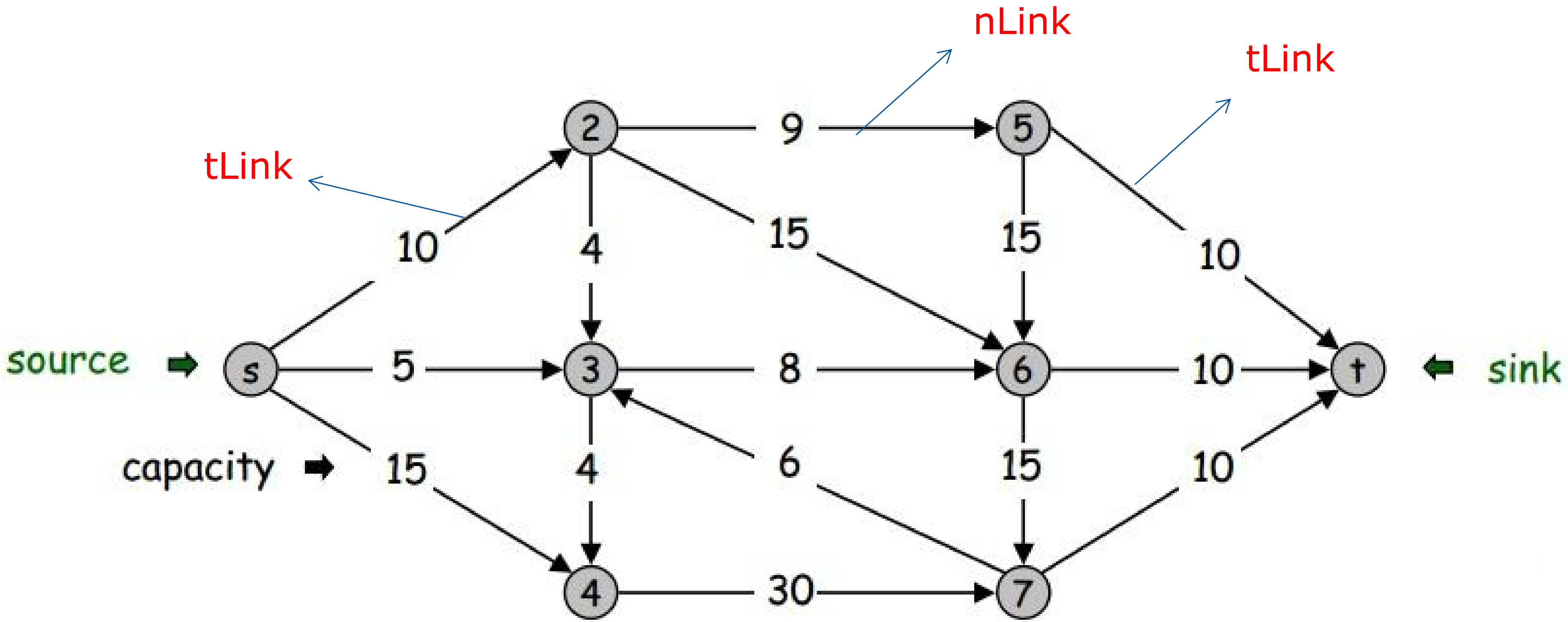
Graph-cut 原理

- **Graph cuts是一种十分有用和流行的能量优化算法，在图像处理领域普遍应用于点云分割、前后背景分割（Image segmentation）、立体视觉（stereo vision）、抠图（Image matting）。**



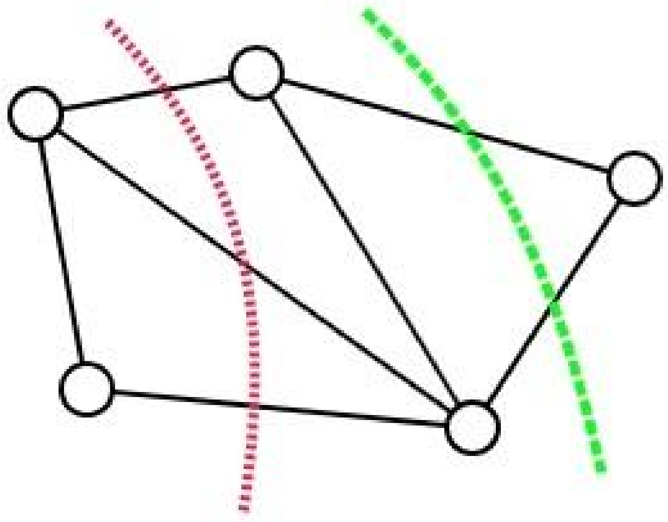
Graph-cut 原理

- 首先用一个无向图G=<V,E>表示要分割的图像，**V和E分别是顶点（vertex）和边（edge）的集合**。普通的图由顶点和边构成，如果边有方向的，这样的图被则称为有向图，否则为无向图，且边是有权值的，不同的边可以有不同的权值，分别代表不同的物理意义。而**Graph Cuts图是在普通图的基础上多了2个顶点，这2个顶点分别用符号”S”和”T”表示，统称为终端顶点**。其它所有的顶点都必须和这2个顶点相连形成边集合中的一部分。所以Graph Cuts中有两种顶点，也有两种边。



Graph-cut 原理-最小割

- 图的最小割可以分很多情况进行讨论，例如有向图、无向图，边的权重等。下图是一张无向无权重图和它的两个割，**红色的线格割掉了三条边，而绿色的线割掉了两条边，很明显绿色的线为该图的最小割**。



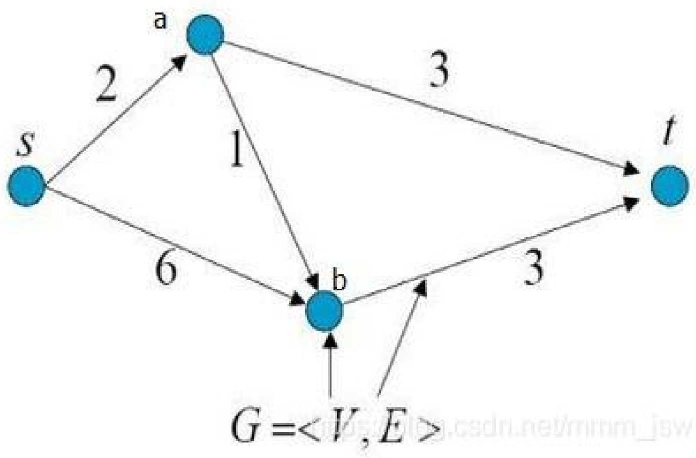
Graph-cut 原理-最小割

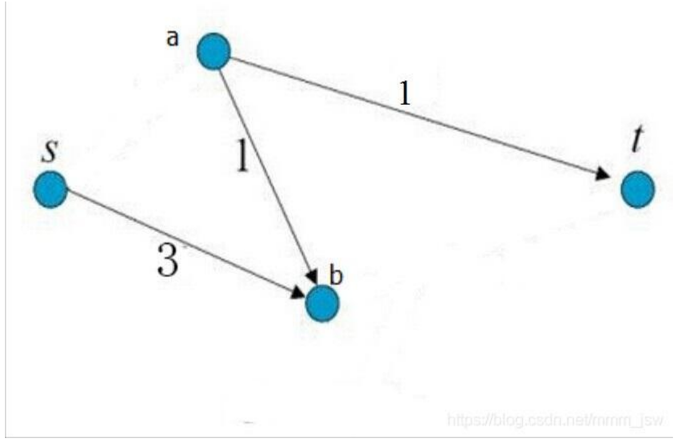
- 图的最小割可以分很多情况进行讨论，例如有向图、无向图，边的权重等。下图是一张无向无权重图和它的两个割，**红色的线格割掉了三条边，而绿色的线割掉了两条边，很明显绿色的线为该图的最小割**。

- s -> a -> t
- s -> b -> t
- s -> a -> b-> t

剪掉边：

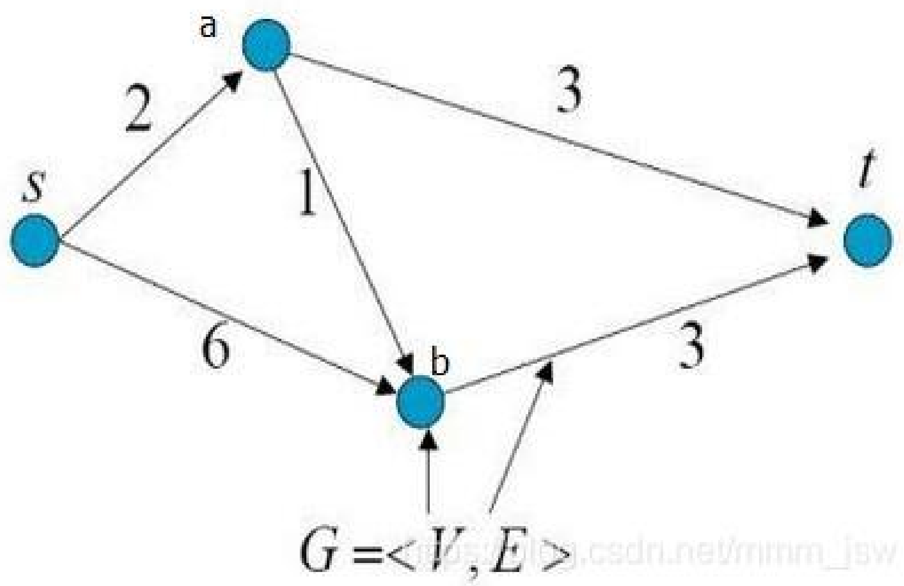
- s -> a
- b -> t





Graph-cut 原理-最大流

- 假如顶点s源源不断有水流出，边的权重代表该边允许通过的最大水流量，顶点t流入的水流量最大是多少?
- 顶点t能够流入的最大水流量为：2 + 3 = 5。
- 可以发现图中的最小割和最大流都为5，经过数学证明可以知道，图的最小割问题可以转换为最大流问题。所以，算法上在处理最小割问题时，往往先转换为最大流问题。



- s -> a -> t：流量被边”s -> a”限制，最大流量为2
- s -> b -> t：流量被边”b -> t”限制，最大流量为3
- s -> a -> b-> t：边”s -> a”的流量已经被其他路径占满，没有流量

Graph-cut 原理-最大流-最小割

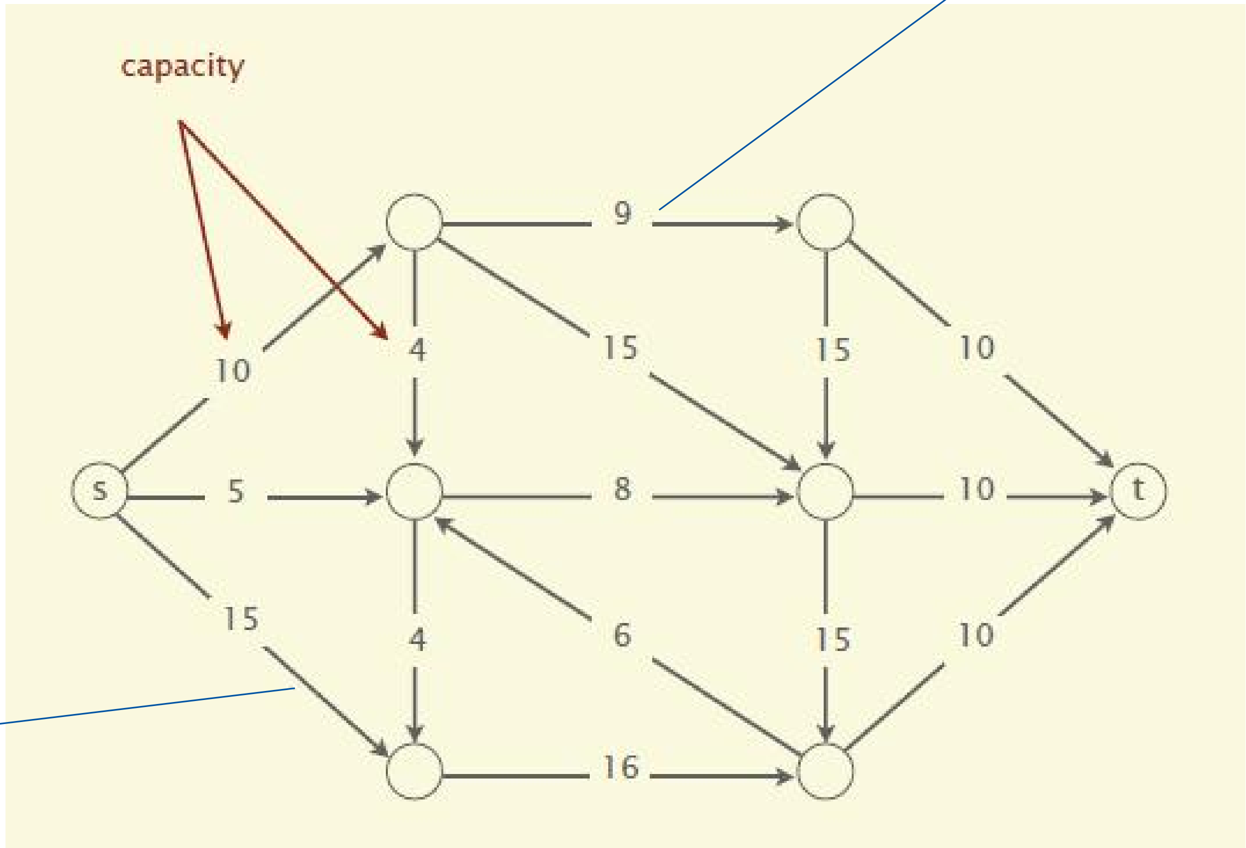
最大流和最小割的关系是什么?

- **最大流不可能大于最小割，**因为最大流所有的水流都一定经过最小割那些割边，流过的水流怎么可能比水管容量还大呢?
- **最大流不可能小于最小割**，如果小，那么说明水管容量没有物尽其用，可以继续加大水流。
- **由此可见，最大流和最小割的其实都是在求解同一个问题。**

Graph-cut 原理-最大流

- **maxflow问题。跟mincut问题类似，maxflow要处理的情况也是一个有向图，并有一个原顶点（source vertex）和目标（target vertex）。边的权值为正,又称之为容量（capacity）**

反向路径

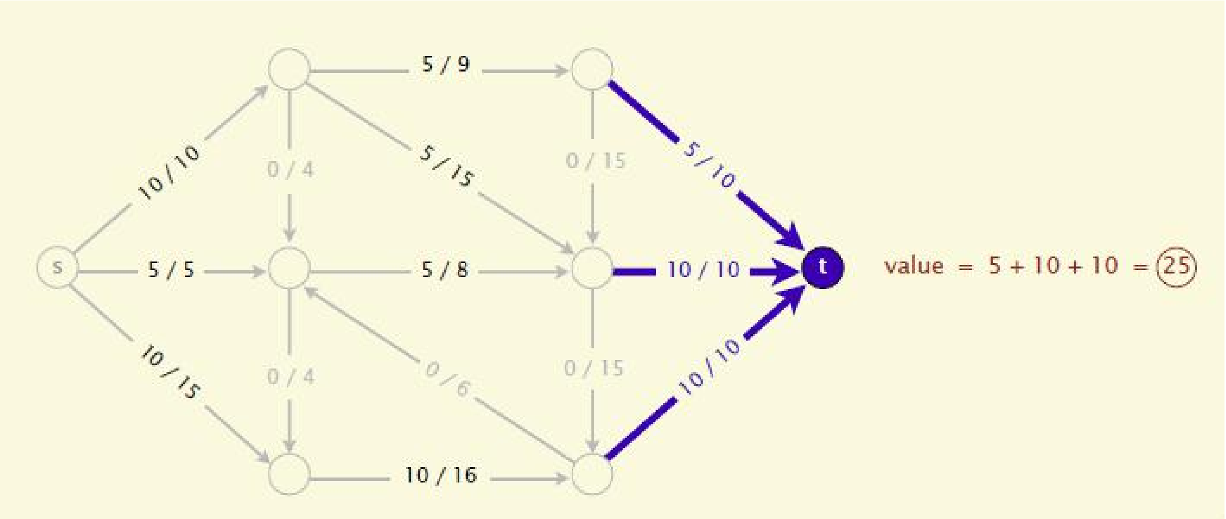


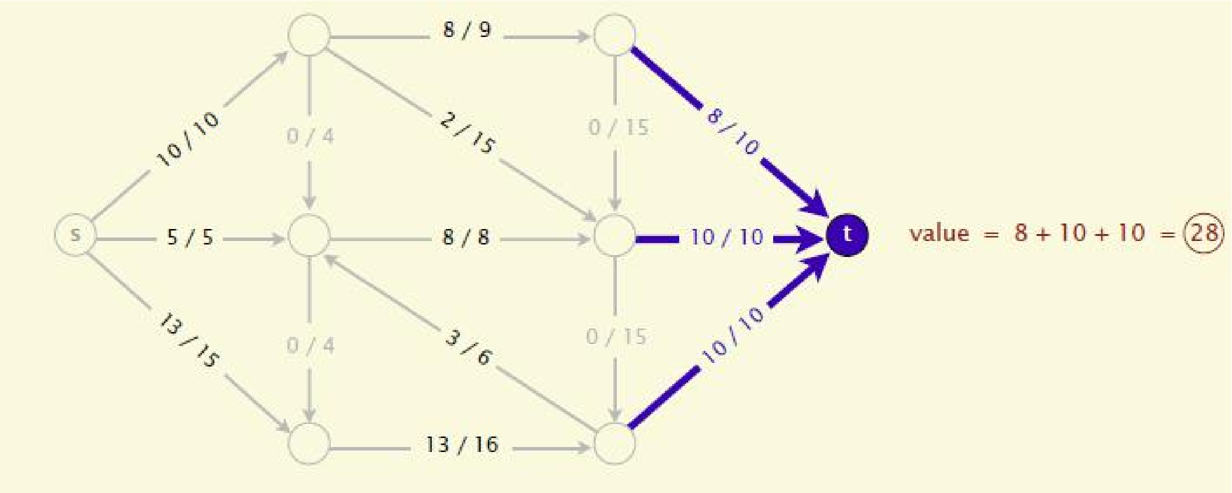
前向路径

Graph-cut 原理-最大流

- 一个st-flow(简称flow)是为每条边附一个值，这个值需要满足两个条件
  - 0<=边的flow <<边的capacity
  - 除了s和t外，每个顶点的inflow要等于outflow

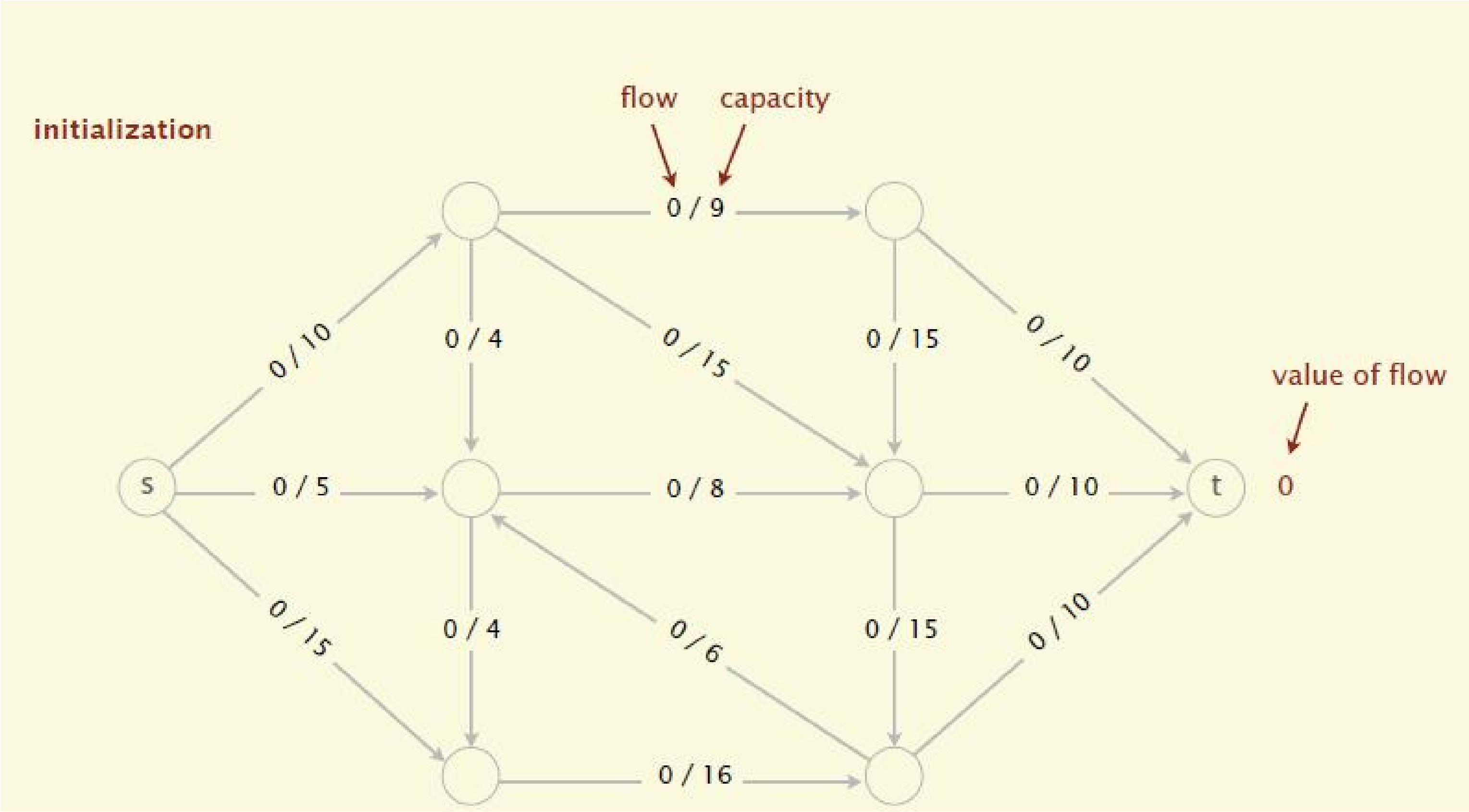
其实这个很好理解，可以想象成水管或者电流。





Graph-cut 原理-最大流

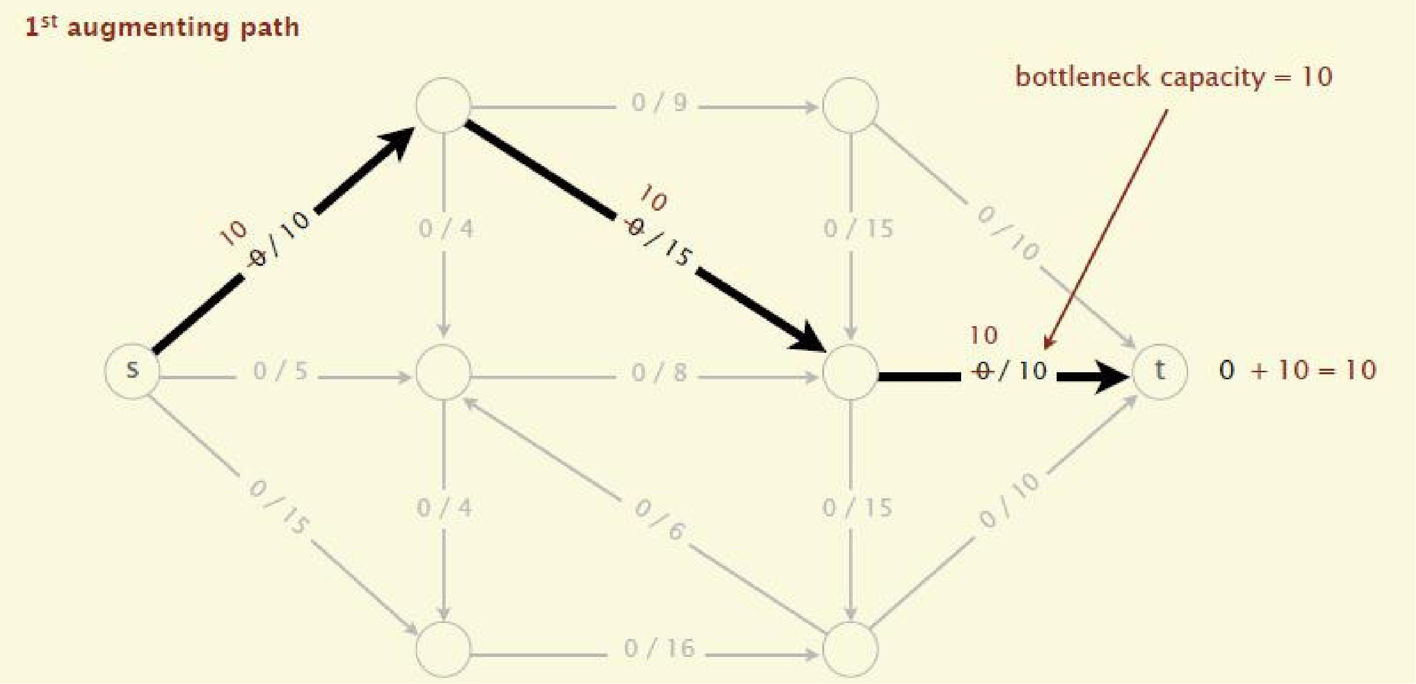
- 介绍一种解maxflow的算法Ford-Fulkerson,为了方便，简称FF算法

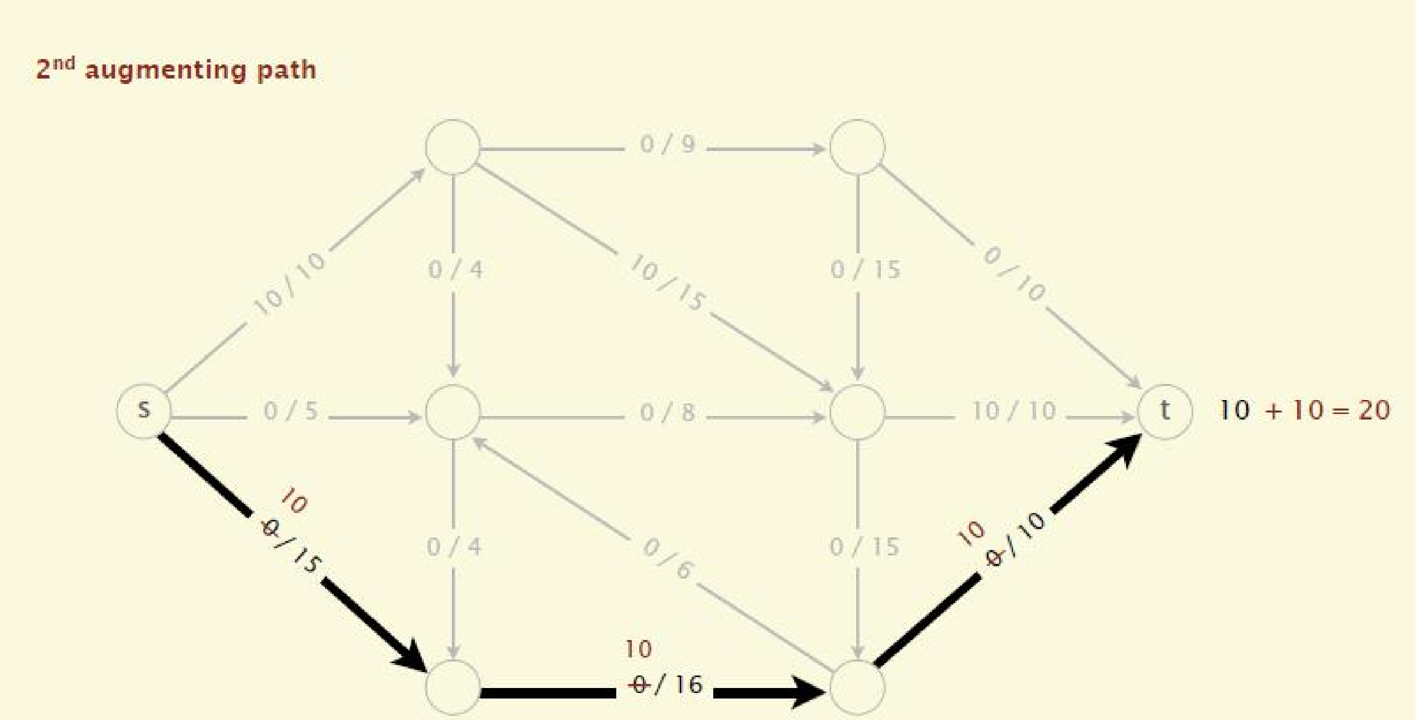


初始化，所有边的flow都初始化为0

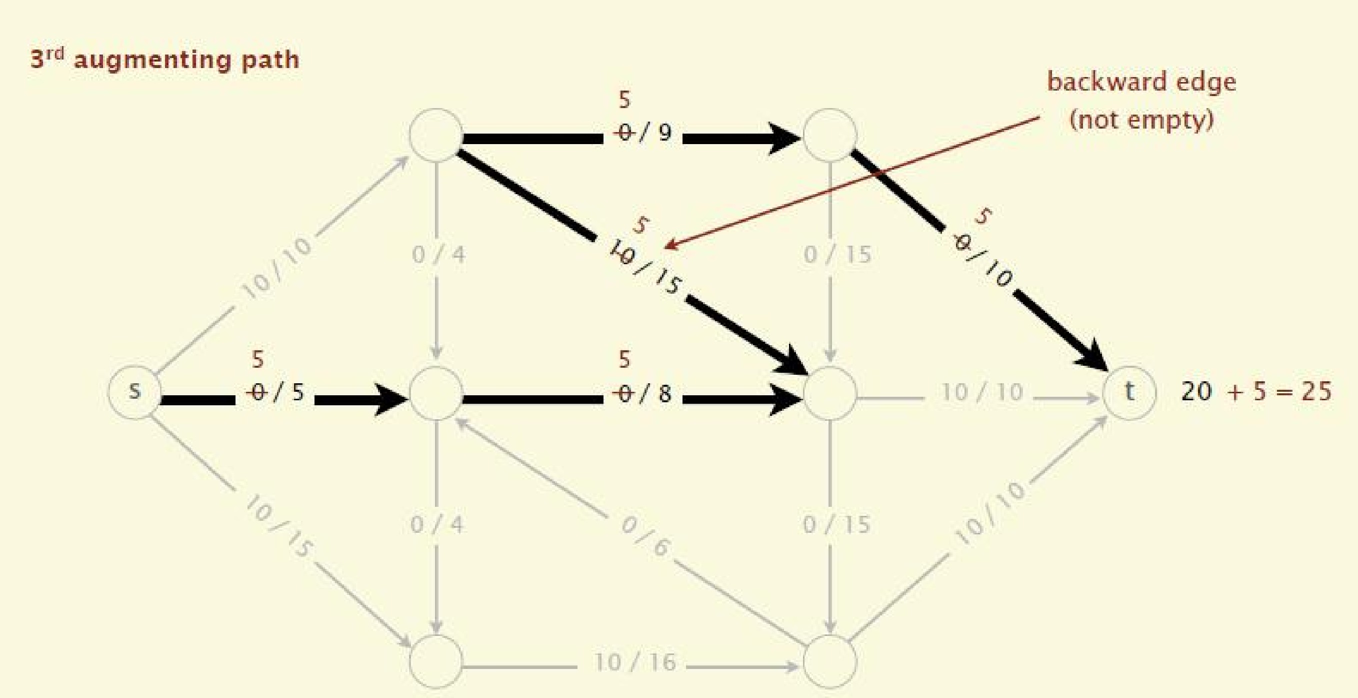
FF算法

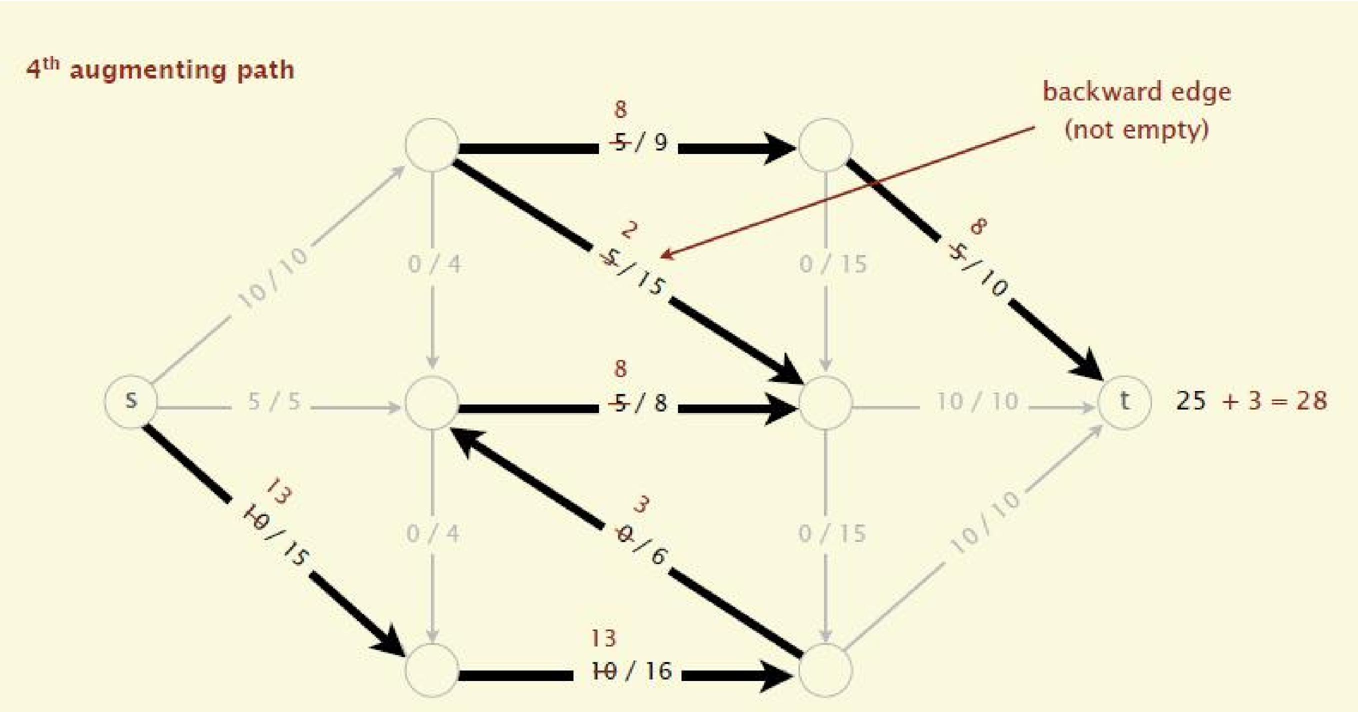
- 沿着增广路径增加flow。增广路径是一条从s到t的无向路径，但也有些条件，**可以经过没有满容量的前向路径（s到t）或者是不为空的反向路径(t->s)**





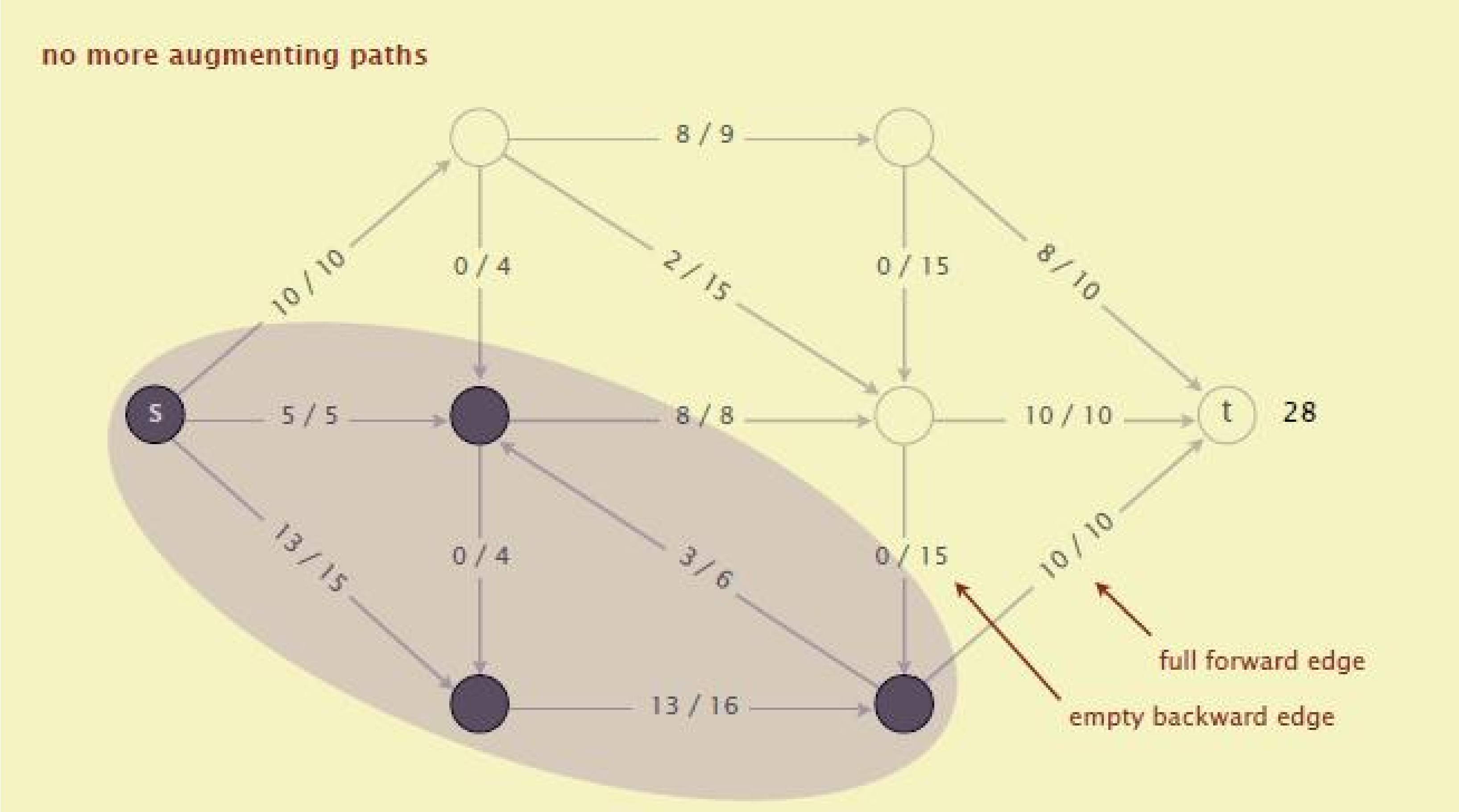
FF算法


3rd augmenting path
backward edge
(not empty)
5
0 / 9
10 / 10
0 / 4
5
10 / 15
0 / 15
5
0 / 10
s
5
0 / 5
5
0 / 8
10 / 10
t
20 + 5 = 25
10 / 15
0 / 4
0 / 6
0 / 15
10 / 10
10 / 16

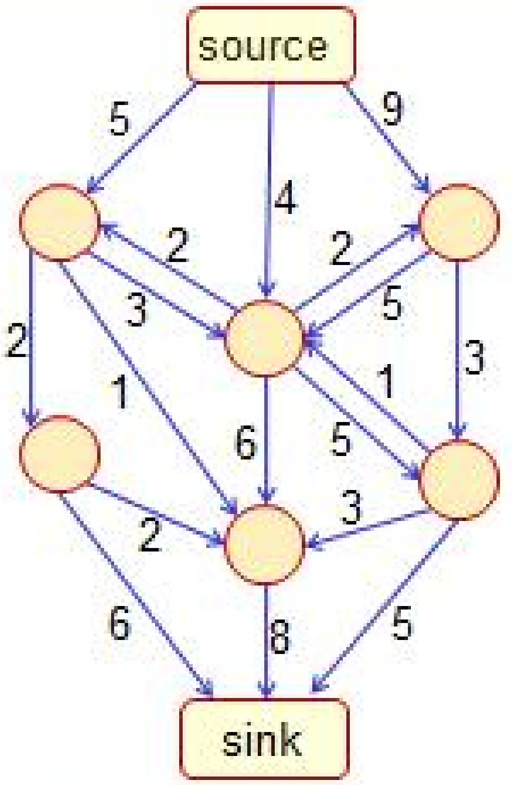

4th augmenting path
backward edge
(not empty)
8
5 / 9
10 / 10
0 / 4
2
5 / 15
0 / 15
8
5 / 10
s
5 / 5
8
5 / 8
10 / 10
t
25 + 3 = 28
13
10 / 15
0 / 4
3
0 / 6
0 / 15
10 / 10
13
10 / 16

FF算法

- s在A中，且A中的顶点通过一些无向的边连接而成，这些边**要么是不是满的前向边要么是非空的反向边**



Graph-cut 原理-最小割求解



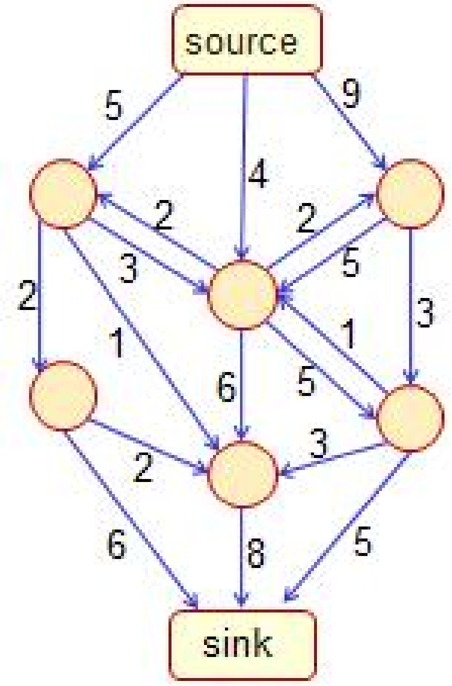
Task :

Minimize the cost of the cut

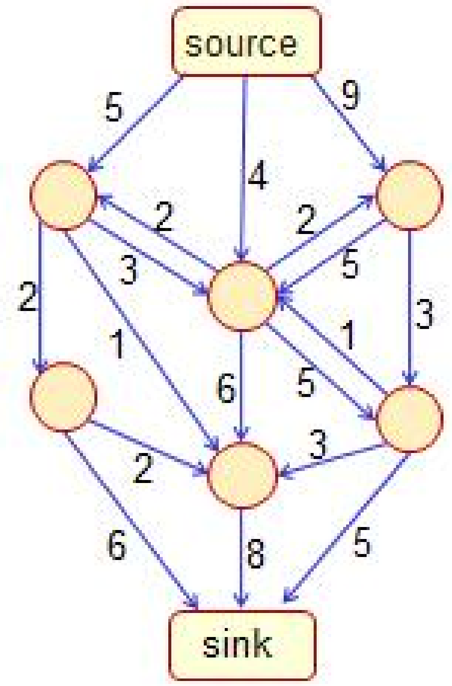
1) Each node is either assigned to the source S or sink T

2) The cost of the edge (i, j) is taken if ($i \in S$) and ($j \in T$)

Graph-cut 原理-最小割求解



$$\min_{S,T} \sum_{i\in S, j\in T} c_{ij}$$

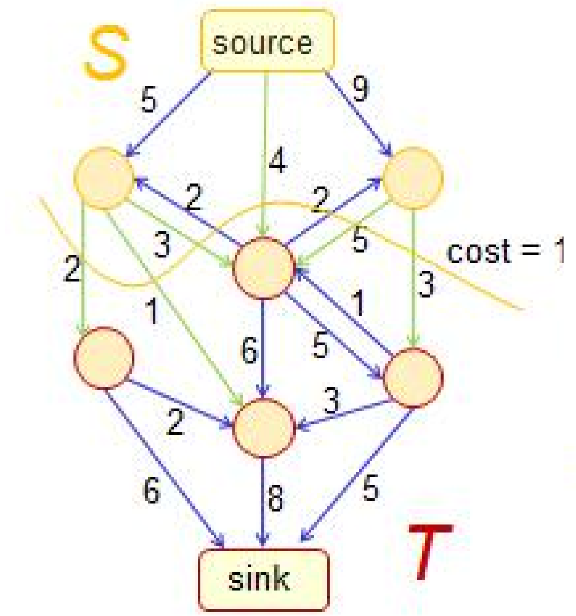


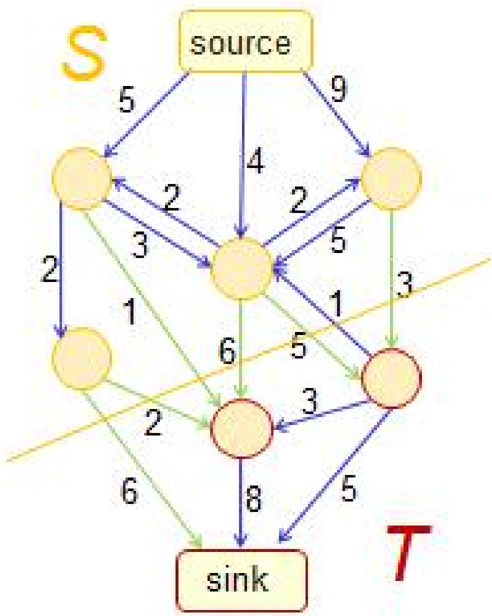
edge costs

$$\min_{S,T} \sum_{\boxed{i\in S}, \boxed{j\in T}} \boxed{c_{ij}} \qquad s.t. \quad s\in S, \quad t \in T$$

source set　　sink set

Graph-cut 原理-最小割求解

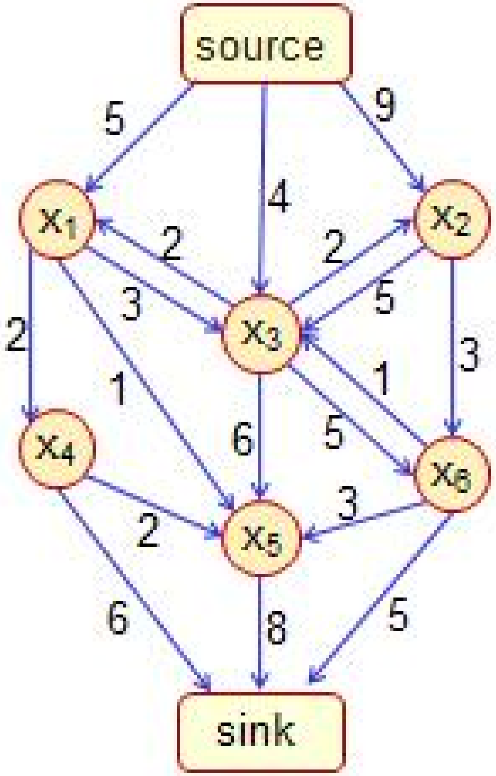




$$\min_{S,T} \sum_{\underbrace{i\in S}_{\text{source set}},\underbrace{j\in T}_{\text{sink set}}} \overbrace{c_{ij}}^{\text{edge costs}} \quad s.t. \quad s\in S, \quad t\in T$$

$$\min_{S,T} \sum_{\underbrace{i\in S}_{\text{source set}},\underbrace{j\in T}_{\text{sink set}}} \overbrace{c_{ij}}^{\text{edge costs}} \quad s.t. \quad s\in S, \quad t\in T$$

Graph-cut 原理-最小割求解

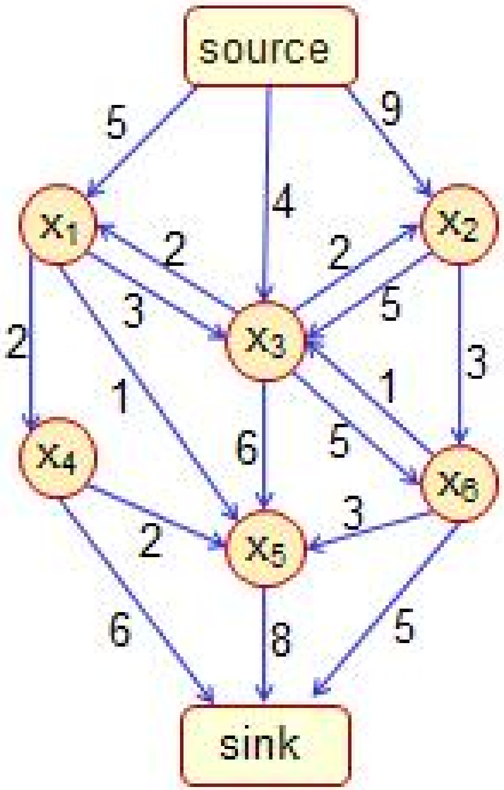


$$\min_{\mathbf{x}} \sum_{(i,j)\in E} c_{ij}(1-x_i)x_j$$

$$s.t. \quad x_s = 0, \quad x_t = 1$$

$$x_i = 0 \implies x_i \in S \qquad x_i = 1 \implies x_i \in T$$

Graph-cut 原理-最小割求解



$$\min_{\mathbf{x}} \sum_{(i,j)\in E} c_{ij}(1-x_i)x_j$$

$$s.t. \quad x_s = 0, \quad x_t = 1$$

transformable into Linear program (LP)（线性规划）

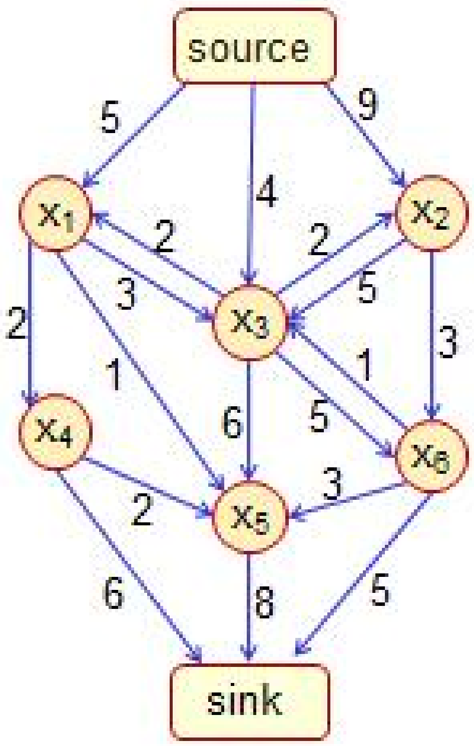
$$\min_{\mathbf{x},\mathbf{d}} \quad c_{ij} d_{ij}$$

$$s.t. \quad d_{ij} \geq x_j - x_i \qquad d_{ij} \geq 0$$

$$x_s = 0 \qquad\qquad x_t = 1$$

Dual to max-flow problem

Graph-cut 原理-最小割求解

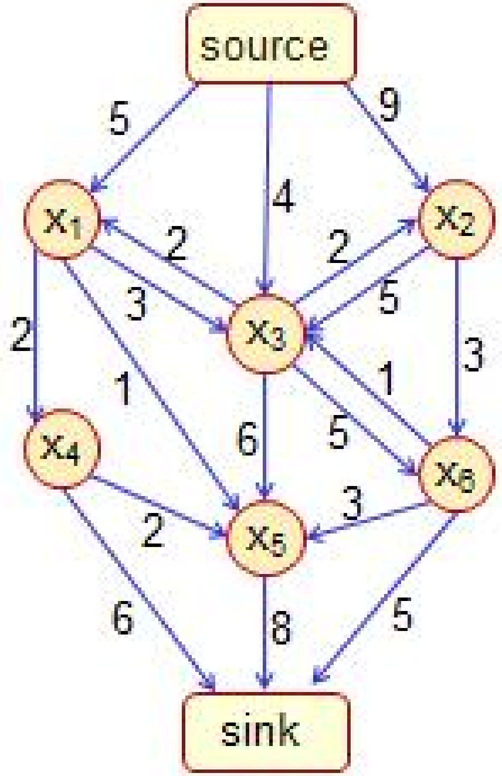


After the substitution of the constraints :

$$\min_{\mathbf{x}} \sum_{(s,i)\in E} c_{si}x_i + \sum_{(i,t)\in E} c_{it}(1-x_i)$$
$$+ \sum_{(i,j)\in E, i,j \notin \{s,t\}} c_{ij}(1-x_i)x_j$$

$$x_i = 0 \implies x_i \in S \qquad x_i = 1 \implies x_i \in T$$

Graph-cut 原理-最小割求解



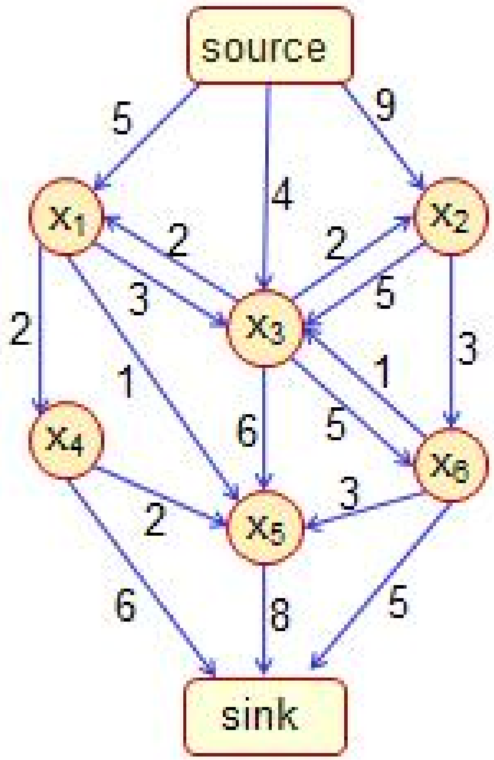
source edges   sink edges

$$\min_{\mathbf{x}} \sum_{(s,i)\in E} c_{si}x_i + \sum_{(i,t)\in E} c_{it}(1-x_i) + \sum_{(i,j)\in E, i,j\notin\{s,t\}} c_{ij}(1-x_i)x_j$$

$x_i = 0 \Rightarrow x_i \in S$ $x_i = 1 \Rightarrow x_i \in T$

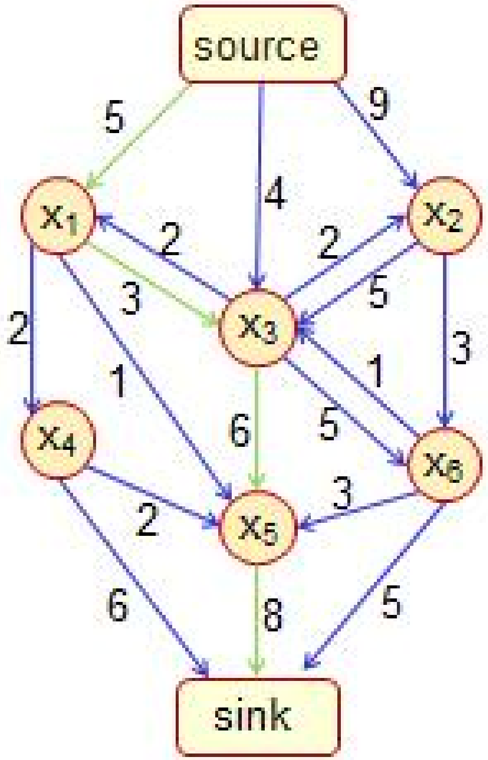
Edges between variables

Graph-cut 原理-最小割求解



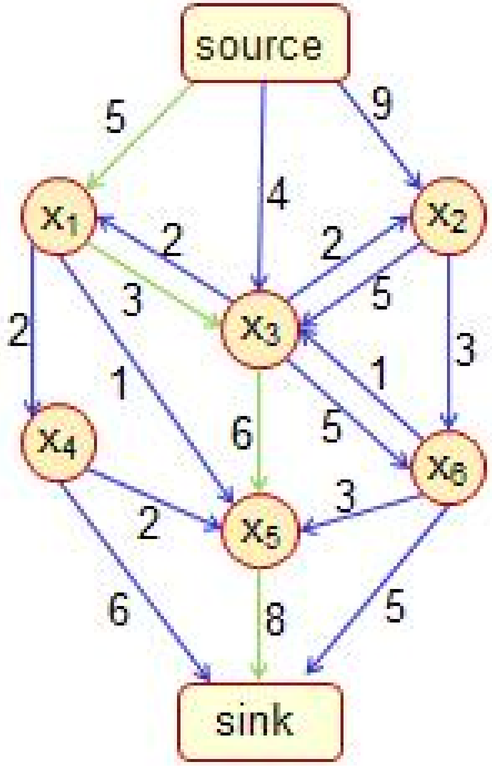
$$
\begin{aligned}
C(\mathbf{x}) = {} & 5x_1 + 9x_2 + 4x_3 + 3x_3(1-x_1) + 2x_1(1-x_3) \\
& + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \\
& + 1x_5(1-x_1) + 6x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \\
& + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) \\
& + 8(1-x_5) + 5(1-x_6)
\end{aligned}
$$

Graph-cut 原理-最小割求解



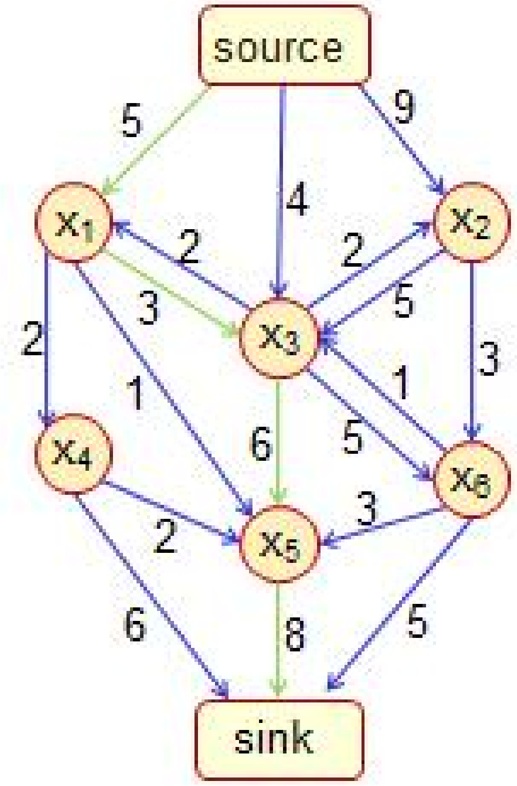
$$C(x) = 5x_1 + 9x_2 + 4x_3 + 3x_3(1-x_1) + 2x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 6x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4)$$
$$+ 8(1-x_5) + 5(1-x_6)$$

Graph-cut 原理-最小割求解



$$
\begin{aligned}
C(x) = {} & 2x_1 + 9x_2 + 4x_3 + 2x_1(1-x_3) \\
& + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \\
& + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \\
& + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) \\
& + 5(1-x_5) + 5(1-x_6) \\
& + 3x_1 + 3x_3(1-x_1) + 3x_5(1-x_3) + 3(1-x_5)
\end{aligned}
$$

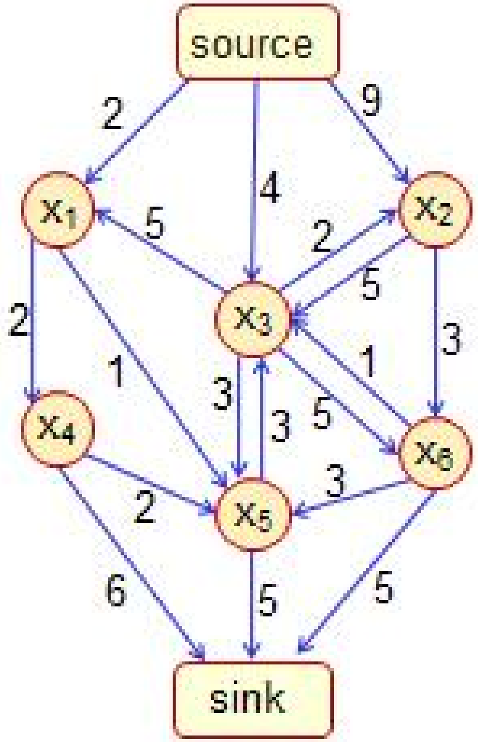
Graph-cut 原理-最小割求解



$$
\begin{aligned}
C(\mathbf{x}) = {} & 2x_1 + 9x_2 + 4x_3 + 2x_1(1-x_3) \\
& + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \\
& + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \\
& + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) \\
& + 5(1-x_5) + 5(1-x_6) \\
& + 3x_1 + 3x_3(1-x_1) + 3x_5(1-x_3) + 3(1-x_5)
\end{aligned}
$$

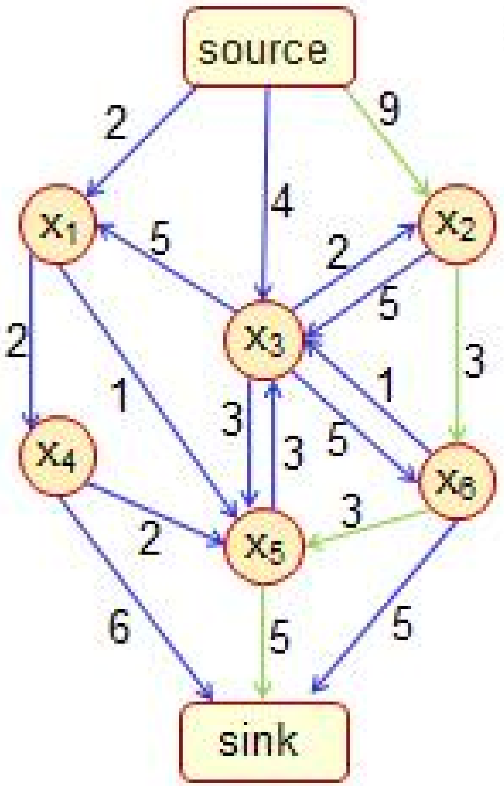
$$
\boxed{\begin{gathered} 3x_1 + 3x_3(1-x_1) + 3x_5(1-x_3) + 3(1-x_5) \\ = \\ 3 + 3x_1(1-x_3) + 3x_3(1-x_5) \end{gathered}}
$$

Graph-cut 原理-最小割求解



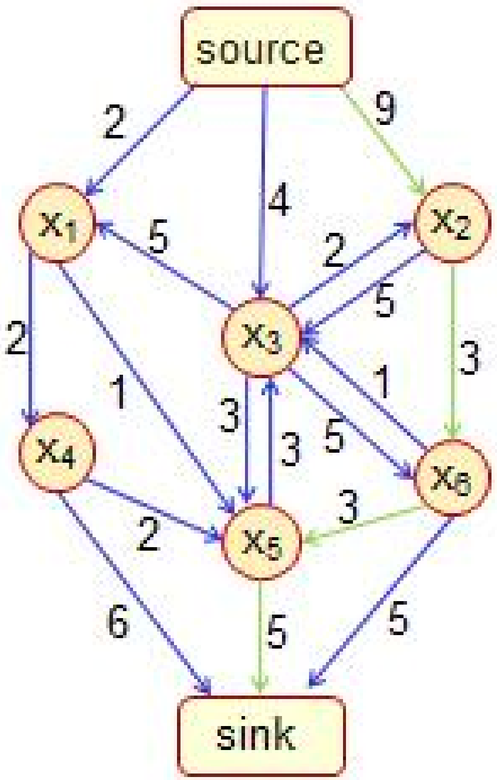
$$
\begin{aligned}
C(\mathbf{x}) = & \ 3 + 2x_1 + 9x_2 + 4x_3 + 5x_1(1-x_3) \\
& + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \\
& + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \\
& + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) \\
& + 5(1-x_5) + 5(1-x_6) + 3x_3(1-x_5)
\end{aligned}
$$

Graph-cut 原理-最小割求解



$$
\begin{aligned}
C(x) = {} & 3 + 2x_1 + 9x_2 + 4x_3 + 5x_1(1-x_3) \\
& + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \\
& + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \\
& + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) \\
& + 5(1-x_5) + 5(1-x_6) + 3x_3(1-x_5)
\end{aligned}
$$

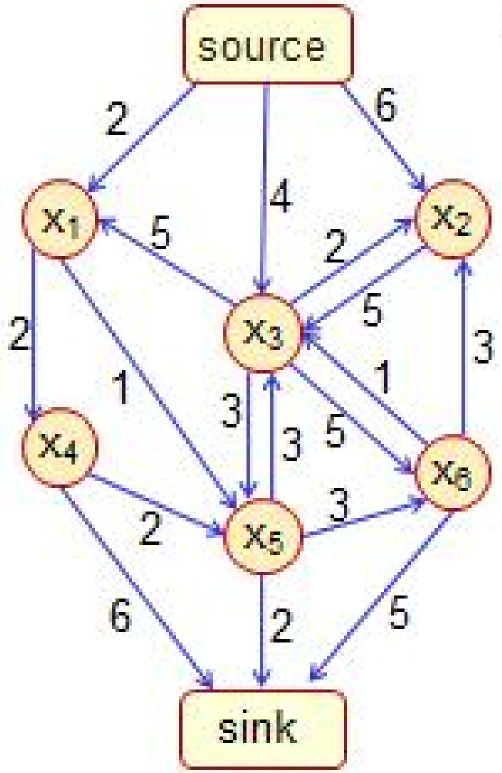
Graph-cut 原理-最小割求解



$$
\begin{aligned}
C(x) &= 3 + 2x_1 + 6x_2 + 4x_3 + 5x_1(1-x_3) \\
&+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \\
&+ 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \\
&+ 2x_5(1-x_4) + 6(1-x_4) \\
&+ 2(1-x_5) + 5(1-x_6) + 3x_3(1-x_5) \\
&+ 3x_2 + 3x_6(1-x_2) + 3x_5(1-x_6) + 3(1-x_5)
\end{aligned}
$$

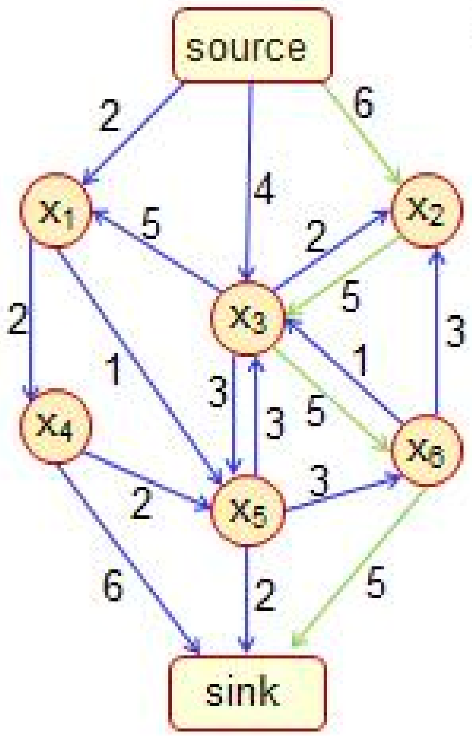
$$
\boxed{\begin{gathered} 3x_2 + 3x_6(1-x_2) + 3x_5(1-x_6) + 3(1-x_5) \\ = \\ 3 + 3x_2(1-x_6) + 3x_6(1-x_5) \end{gathered}}
$$

Graph-cut 原理-最小割求解



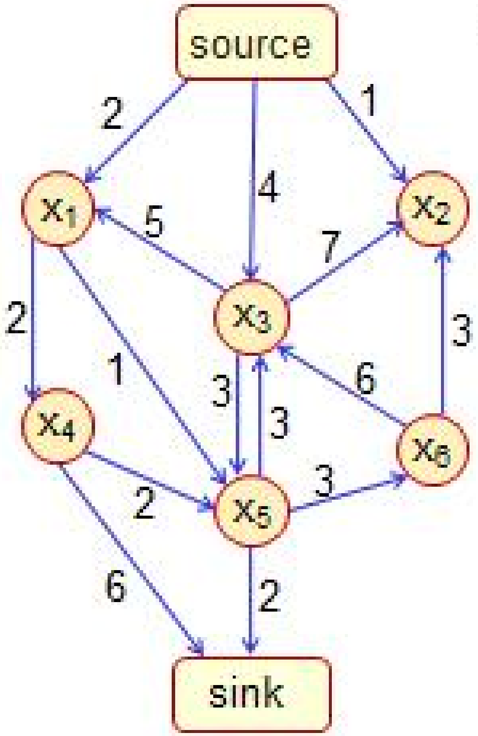
$$
\begin{aligned}
C(\mathbf{x}) = {} & 6 + 2x_1 + 6x_2 + 4x_3 + 5x_1(1-x_3) \\
& + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \\
& + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \\
& + 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \\
& + 2(1-x_5) + 5(1-x_6) + 3x_3(1-x_5)
\end{aligned}
$$

Graph-cut 原理-最小割求解



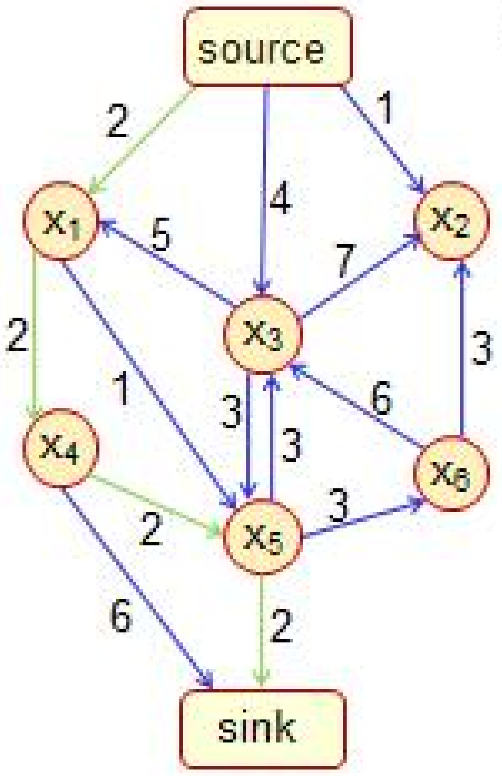
$$C(x) = 6 + 2x_1 + 6x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$
$$+ 2(1-x_5) + 5(1-x_6) + 3x_3(1-x_5)$$

Graph-cut 原理-最小割求解



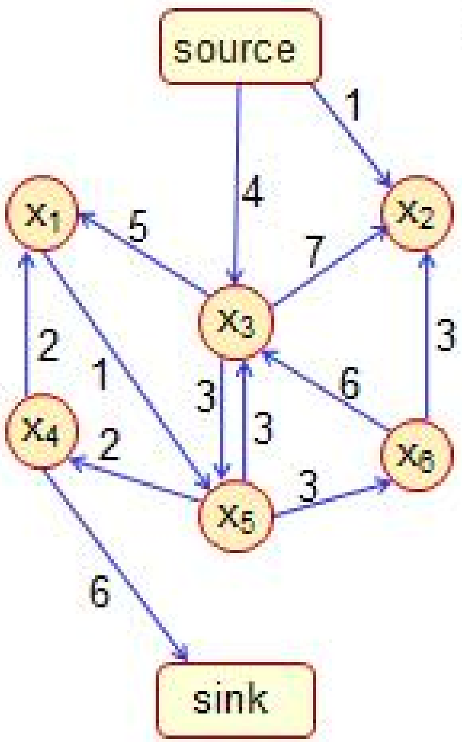
$$
\begin{aligned}
C(\mathbf{x}) = 11 &+ 2x_1 + 1x_2 + 4x_3 + 5x_1(1-x_3) \\
&+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_4(1-x_1) \\
&+ 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6) \\
&+ 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \\
&+ 2(1-x_5) + 3x_3(1-x_5)
\end{aligned}
$$

Graph-cut 原理-最小割求解



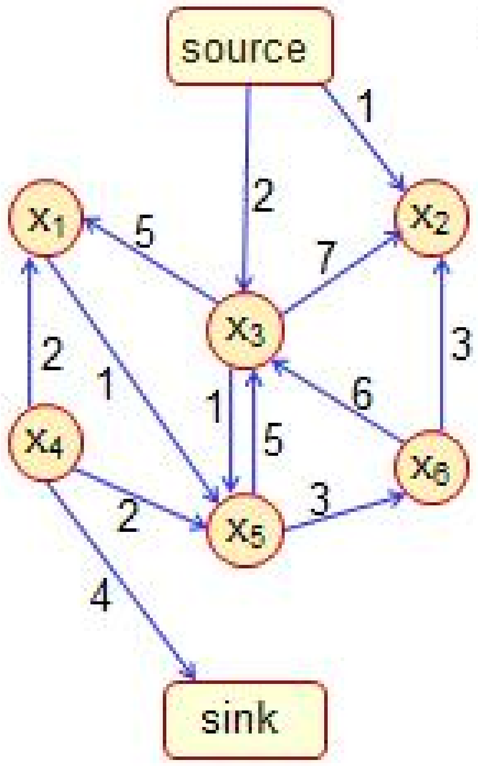
$$C(\mathbf{x}) = 11 + 2x_1 + 1x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6)$$
$$+ 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$
$$+ 2(1-x_5) + 3x_3(1-x_5)$$

Graph-cut 原理-最小割求解



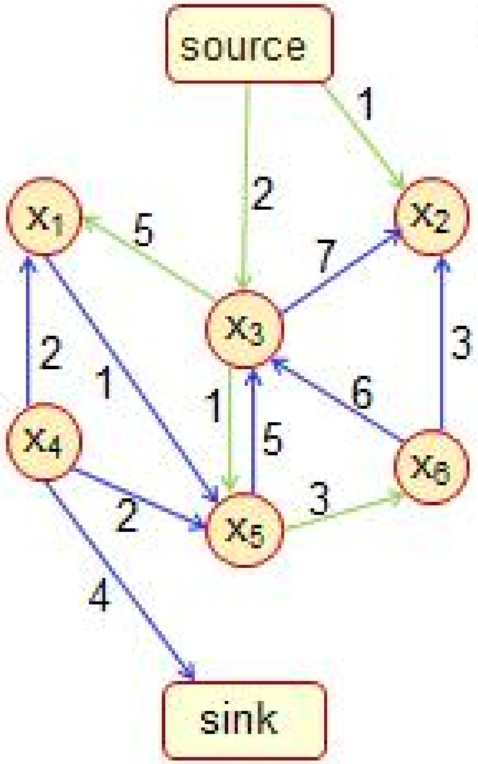
$$
\begin{aligned}
C(\boldsymbol{x}) = 13 &+ 1x_2 + 4x_3 + 5x_1(1-x_3) \\
&+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) \\
&+ 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6) \\
&+ 2x_4(1-x_5) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \\
&+ 3x_3(1-x_5)
\end{aligned}
$$

Graph-cut 原理-最小割求解



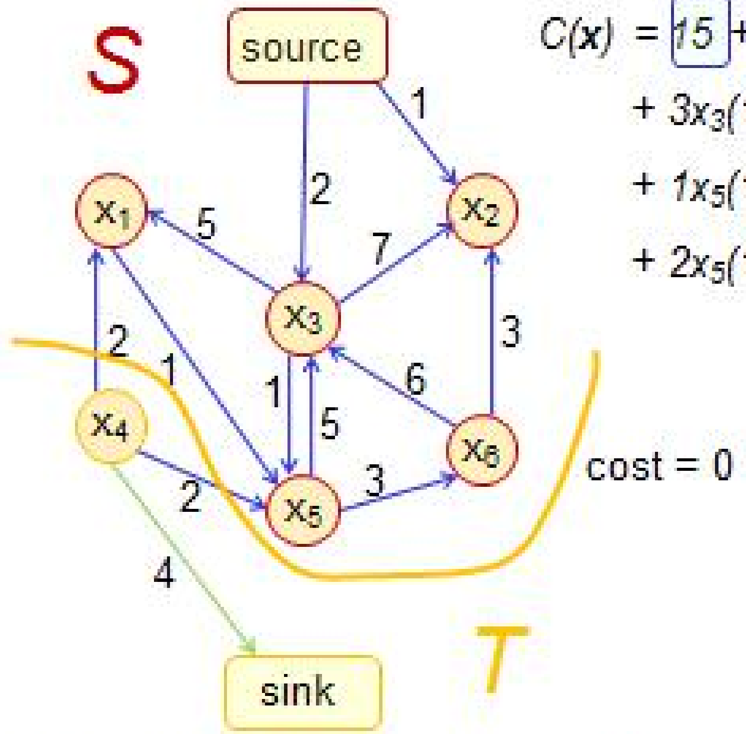
$$
\begin{aligned}
C(\mathbf{x}) = {} & 15 + 1x_2 + 4x_3 + 5x_1(1-x_3) \\
& + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) \\
& + 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5) \\
& + 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)
\end{aligned}
$$

Graph-cut 原理-最小割求解



$$
\begin{aligned}
C(\mathbf{x}) &= 15 + 1x_2 + 4x_3 + 5x_1(1-x_3) \\
&\quad + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) \\
&\quad + 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5) \\
&\quad + 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)
\end{aligned}
$$

Graph-cut 原理-最小割求解



$$
\begin{aligned}
C(x) = \overset{\text{min cut}}{\boxed{15}} &+ 1x_2 + 4x_3 + 5x_1(1-x_3) \\
&+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) \\
&+ 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5) \\
&+ 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)
\end{aligned}
$$

T – set of nodes that can reach t

(not necesarly the same)

- All coefficients positive
- Must be global minimum

目录

CONTENTS

点云最小图割

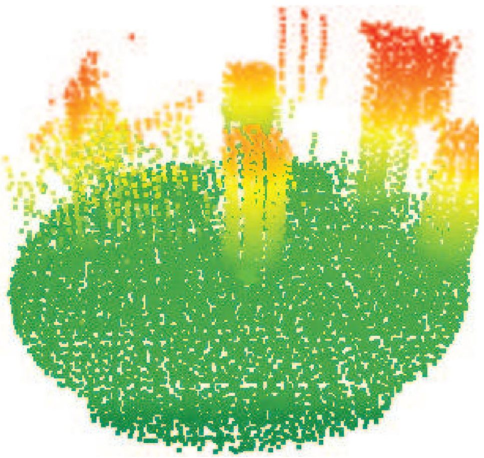
- **基于采样一致的点云分割算法显然是意识流的**，它只能割出大概的点云（可能是杯子的一部分，但杯把儿肯定没分割出来）。
- **基于欧式算法的点云分割面对有牵连的点云就无力了**（比如风筝和人，在不用三维形态学去掉中间的线之前，是无法分割风筝和人的）。
- **基于法线等信息的区域生长算法则对平面更有效**，没法靠它来分割桌上的碗和杯子。也就是说，上述算法更关注能不能分割

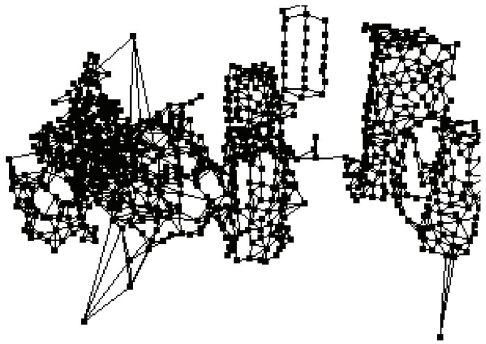
- 我们还需要一个方法来解决分割的“好不好”这个问题。也就是说，有没有哪种方法，可以在一个点不多，一个点不少的情况下，把目标和“其他”分开。

点云最小图割

- **切割有两个非常重要的因素：**
  - ✓ 第一个是获得点与点之间的拓扑关系，也就是生成一张“图”。
  - ✓ 第二个是给图中的连线赋予合适的权值。只要这两个要素合适，最小割算法就会办好剩下的事情。

- **连接算法如下**
  - ✓ 找到每个点最近的n个点
  - ✓ 将这n个点和父点连接
  - ✓ 找到距离最小的两个块（A块中某点与B块中某点距离最小），并连接
  - ✓ 重复3，直至只剩一个块





点云最小图割

- 根据连接点的类型的不同，边被分为以下三种情况，不同的情况，边被赋予不同的权值。
  - 连接输入点云各点之间的边，为上述边赋予的权值被称为平滑代价，由一下公式计算：

$$smoothCost = e^{-(\frac{dist}{\sigma})^2}$$

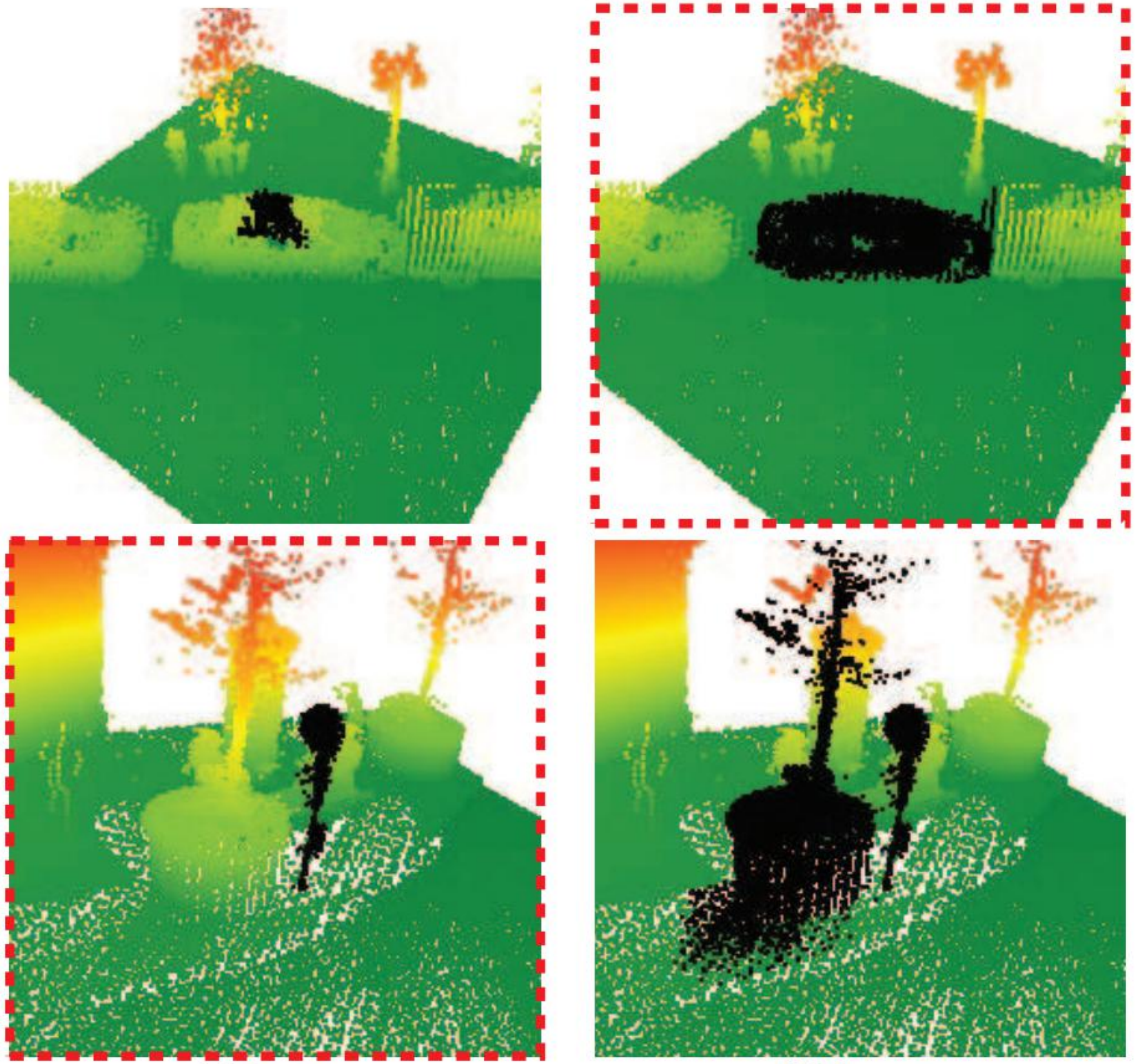
dist为各点之间的距离，距离越大，该边被切断的概率越大。

  - 连接输入点云中的点与原点的边，该边赋予的权值，被称为前景惩罚，该值由用户输入，作为该算法的输入参数。通过输入原点和半径实现。
  - 连接输入点云中的点与汇点的边，该边赋予的权值，被称为背景惩罚，由以下公式计算：

$$backgroundPenalty = (\frac{distanceToCenter}{radius})$$

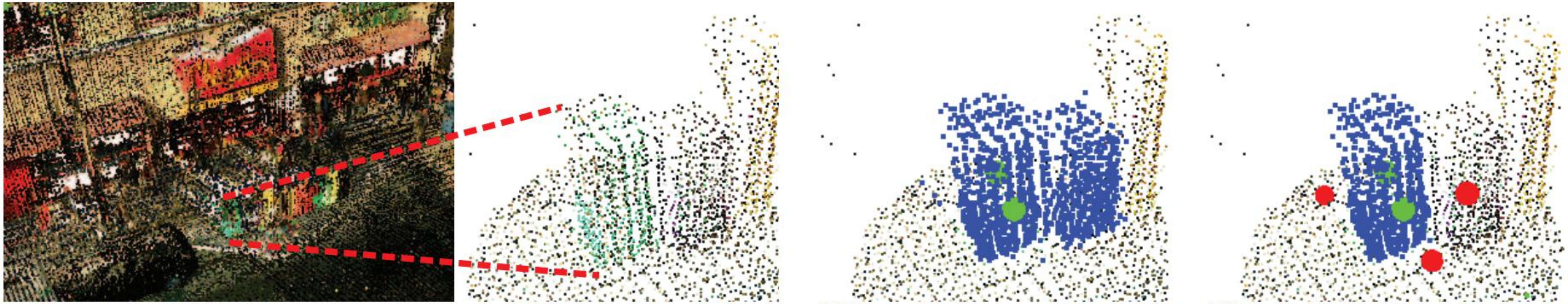
distanceToCenter是点到前景点云（目标点云）预期中心的水平距离

点云最小图割-自动化分割



(a) R = 2m (b) R = 4m

点云最小图割-交互式分割



(a) Input　(b) User chooses radius　(c) User adds foreground constraints　(d) Result after additional constraints

示例：https://blog.csdn.net/cbb8234078009/article/details/106875842

谢谢