运筹学通论I

胡晓东

应用数学研究所

中国科学院数学与系统科学研究院

http://www.amt.ac.cn/member/huxiaodong/index.html

5. 组合优化-算法设计技巧

精确算法

分而治之 (搜索、排大小序、旅行商)
动态规划 (最短路、三角剖分、背包)
分支定界 (整数线性规划、旅行商、工件排序)

近似算法或启发式算法

贪婪策略 (最小生成树、最大可满足、背包、
顶点覆盖、独立集、旅行商)
局部搜索(最大匹配、旅行商、最大割)
序贯法 (工件排序、装箱、顶点着色)
整数规划法 (顶点覆盖、最大可满足、最大割)
随机方法 (最小割、最大可满足、顶点覆盖)
在线算法 (页面调度、k-服务器、工件排序、装箱)
不可近似 (最大团、背包、旅行商、装箱、连通控制集等)

5.5 局部搜索

基于局部搜索的算法的主要思想是：从一个初始可行解开始，在一定范围内寻找一个更好的解；然后再从这个可行解开始，不断地重复这个过程，直到无法找到更好的解为止。一般来讲，任给一个可行解 x，我们称 z 是它的一个相邻可行解如果 z 与 x 的几何距离很近，或者它们具有非常相似的离散结构。

给定一个可行解 x，局部搜索算法试图在它的所有相邻可行解中找到一个具有更好的度量函数（比如优化问题的目标函数）的解。当算法找到了一个可行解，与它相邻的可行解的度量函数值都不比它的度量函数值好，算法就终止，并输出这个可行解。然而，尽管输出的可行解无法再改进（没有比它更好的相邻可行解），但是通常它只是一个局部最优解，而非全局最优解。

5.5 局部搜索（续一）

针对一个优化问题，要设计一个局部搜索算法，我们需要考虑以下因素：

- 如何找到一个初始可行解，从而可以开始进行局部搜索。对尽管对许多优化问题来讲，很容易就可以构造一个初始可行解，但是还有一些优化问题的初始可行解并不是很容易就可以构造出来，此时，就需要求助于一些技巧，比如贪婪策略。

- 如何定义一个可行解的邻域结构。注意，我们不必在所有相邻的可行解中找到一个最好的可行解，通常只需要找到一个比当前的可行解要好的可行解即可。

5.5 局部搜索（续二）

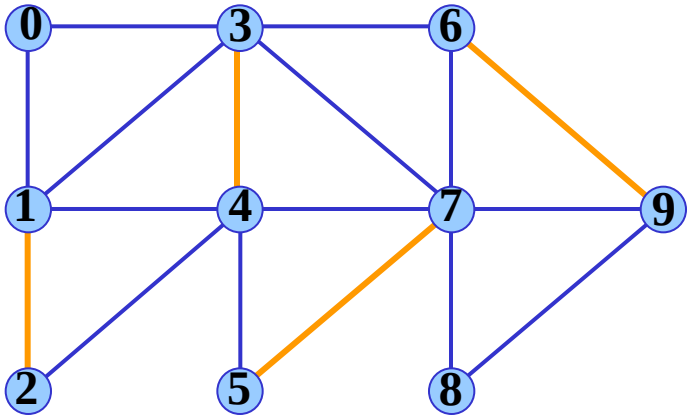
另外，还需要注意的是，对于某一个具体的优化问题来讲，如果我们设计的邻域结构具有这样的性质：可以在多项式时间内从一个可行解出发搜索到一个更好的相邻可行解，且可以经过多项式时间的搜索迭代找到一个全局最优解，那么这样的局部搜索算法就可以在多项式时间内找到问题的一个最优解。

然而，遗憾地是，对于一个 **NP-**难问题来讲，我们不能期望可以定义具有这样性质的邻域结构除非 **P =NP**。求解这些问题时， 我们只能希望设计一个局部搜索算法，它可以找到问题的一个局部最优解，并同时能证明它是一个很好的近似（全局）最优解。

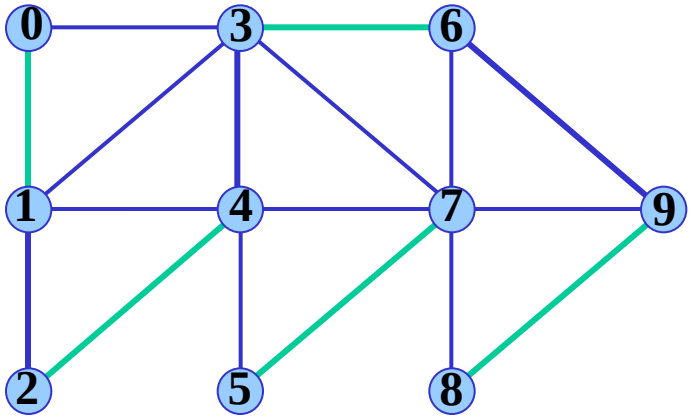
5.5 最大匹配

首先，考虑著名的**最大匹配问题**（匹配亦称对集）。它是我们在讨论线性规划的应用时介绍的指派问题（亦称婚姻问题）的一个一般情形。

问　题: 最大匹配
实　例: 图 $G=(V, E)$
可行解: 图G 的匹配 $M \subset E$（其中任意两条边都没有公共端点）
目　标: 最大化匹配 M 所含有边的条数



极大匹配

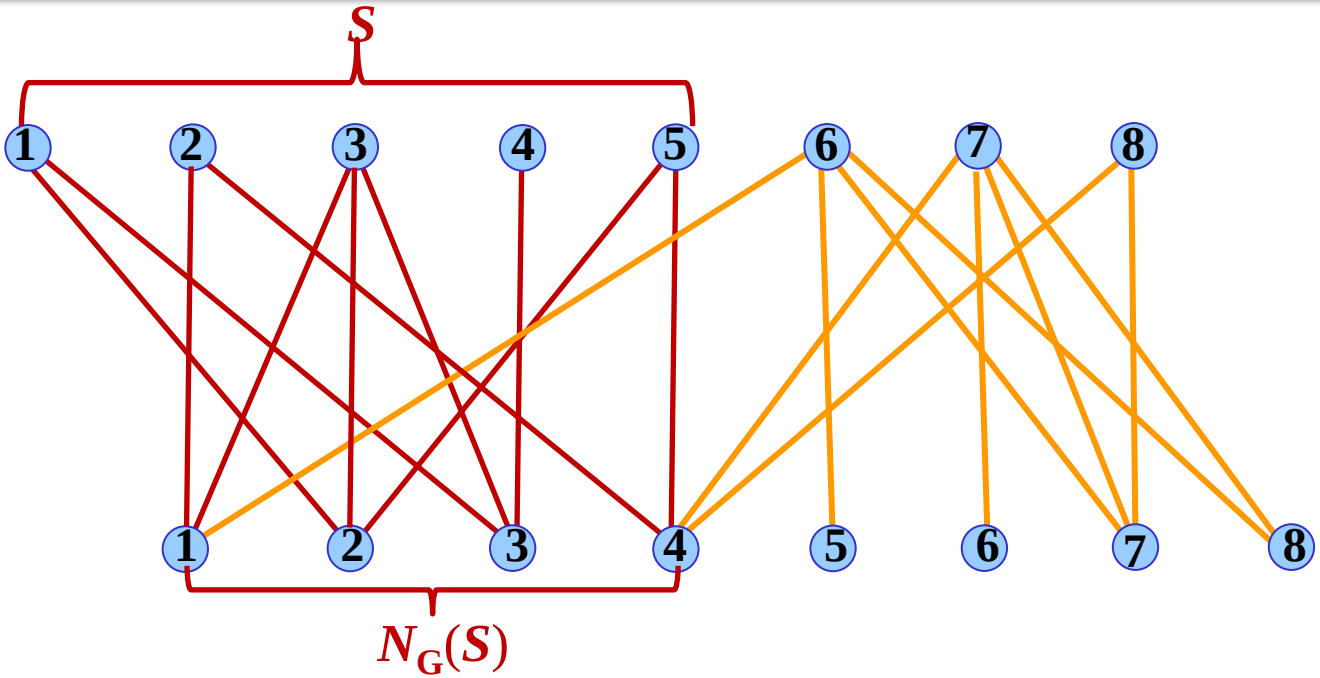


最大匹配

5.5 最大匹配（续一）

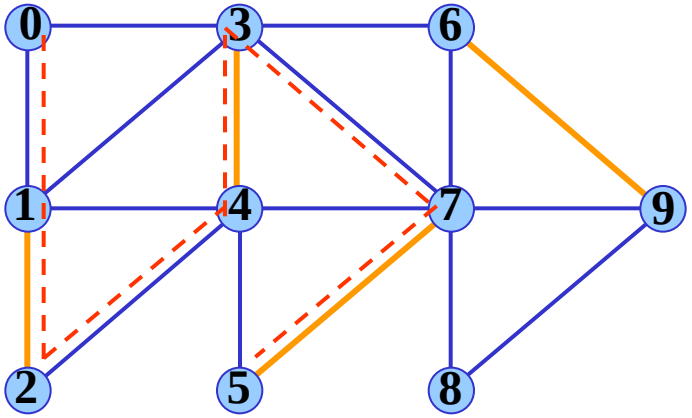
给定图 G 的一个匹配 M，若G 中所有顶点都与匹配M中莫条边关联，则称M为完美匹配。完美匹配一定是最大匹配，但反之不然。

Hall 定理 设 $G=(X\cup Y, E)$ 为一个二部图，则G有完美匹配的充要条件是$|X|=|Y|$，且对任何 $S\subseteq X$ 或 $S\subseteq Y$，具有 $|S|\leq|N_G(S)|$，其中$N_G(S)$表示 G 中所有与 S 中的顶点相邻的顶点组成的集合。

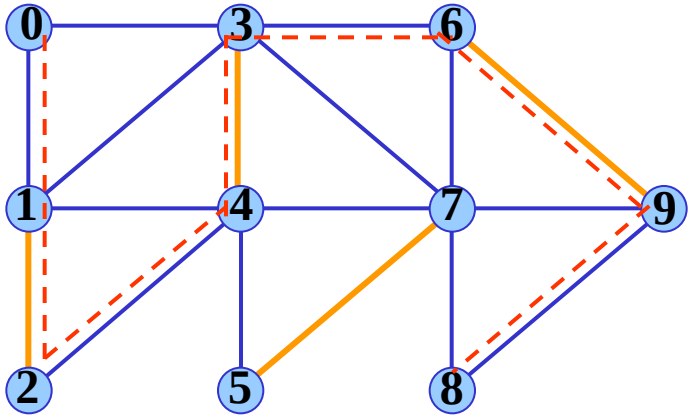


5.5 最大匹配（续一）

给定图 G 的一个匹配 M，称其中的边为匹配边，其它的边是自由边。另称 M 中的一条边 $e \in M$ 的两个端点是匹配点，称顶点 $v \in V$ 是 **M-饱和的**如果它是 M 中某条边的一个端点；否则称其为 **M-非饱和的**。称图 G 中的一条路是 **M-交错路**如果它的边交替地出现在集合 $E \setminus M$ 和 M 中。最后，称一条 M-交错路是一条 **M-增长路**如果其始点和终点都是 M-非饱和的。



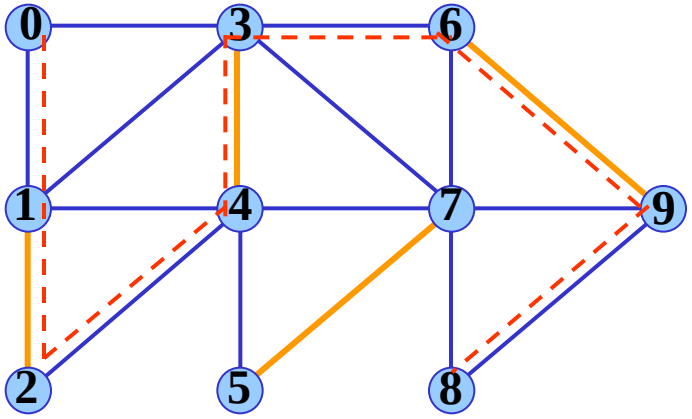
一条 M-交错路



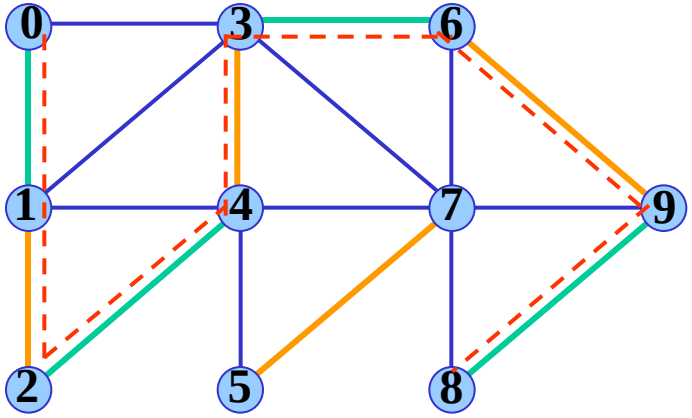
一条 M-增长路

5.5 最大匹配（续二）

引理 1 设 M 是图 G 的一个匹配。若 P 是一条 M-增长路上的边的集合，则 M 和 P 的对称差 $M \oplus P$ 是一个匹配，其含有的边数为 $|M|+1$，其中 $M \oplus P = M \setminus P \cup P \setminus M$。



一条 M-增长路



一个更大的匹配

证明 (练习) 首先用反证法证明对称差 $M \oplus P$ 是一个匹配。 然后假设 P 含有 $2k-1$ 条边， $k \geq 1$，且其中的 k 条边不是 M-饱和的而其他 $k-1$ 条边属于 M。因此有 $M \oplus P$ 包含 $k+1 = |M|+1$ 条边。

5.5 最大匹配（续三）

定理 1 图G 的一个匹配 M 是最大匹配当且仅当图 G 中不存在任何 M-增长路。

证明. ‘必要性’可以由**定理 1** 直接推出。下面去我们考虑‘充分性’。用反证法，假设 M 不是一个最大匹配，而且图 G 中也不存在任何 M-增长路。则此时一定存在图 G 的一个匹配 M' 使得 $|M'| > |M|$。

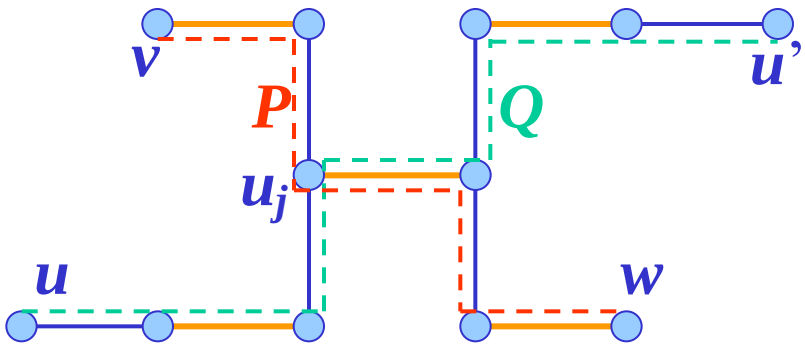
我们现在考虑对称差 $M \oplus M'$ 中的边，它们构成了图 G 的一个子图 G'。注意，这个子图 G' 不一定是连通的，它的连通分支或者是一条路或者是是含有偶数条边的一个圈。又因为 $|M'| > |M|$，所以至少存在一条路它含有 M' 中的边比 M 中的边要多一条。易知这是一条 M-增长路。从而产生矛盾。

5.5 最大匹配（续四）

引理 2 设 M 是图 $G=(V, E)$ 的一个匹配。若不存在一条以 $u \in V$ 为始点的 M-增长路，但是存在一条从 v 到 w 的 M-增长路 P。则也不存在一条以 $u \in V$ 为始点的 $M \oplus P$-增长路。

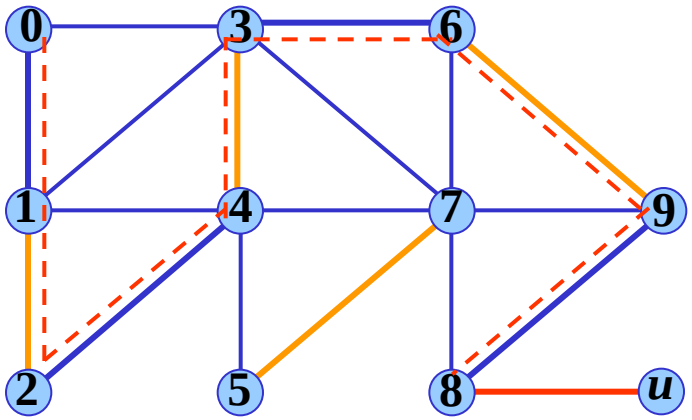
证明 用反证法，假设存在一条以 u 为始点的 $M \oplus P$-增长路 Q。如果 Q 与 P 没有公共的顶点，那么 P 并不能使得路增长，因而 Q 是一条 M-增长路。产生矛盾！

因此设$Q=<u_1=u, u_2, \ldots, u_k=u'>$ 并设u_j 是P中第一个在 Q 中出现的顶点。注意， u_j 将 P 分成两部分，其中一部分一定以u_j 为终点且含有 M 中的一条边。这一部分，再加上 Q 中到 u_j 的那部分构成了一条以u 为始点的 M-增长路。又产生矛盾!

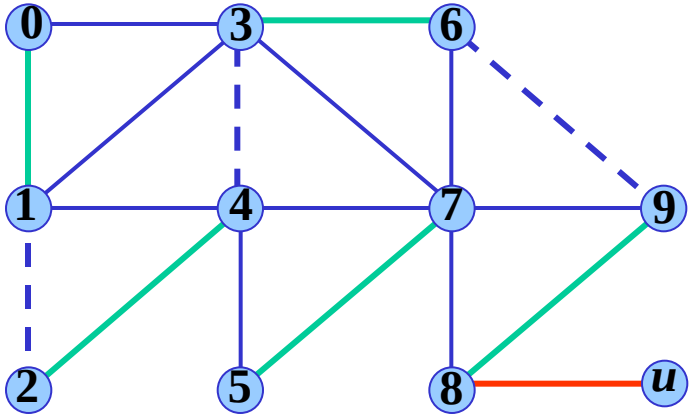


5.5 最大匹配（续五）

推论 1 设 $M_1, M_2, \ldots, M_k$ 是图G 的一组匹配，其中 M_i 是通过 M_{i-1} 的某条 M_i-增长路得到的，$2 \le i \le k$。若 u 是一个 M_1-非饱和顶点，且不存在以u 为始点的一条 M_1-增长路。则也不存在一条以u 为始点的 M_i-增长路，$i \ge 2$。



一条M_{i-1}-增长路



没有以u为始点的M_i-增长路

证明 (练习*) 应用对 k 的数学归纳法和**定理2**。

5.5 最大匹配（续六）

任给一个图***G***，求其最大匹配的局部搜索算法的主要步骤如下：

1. 从任意一个匹配 ***M*** 开始，比如一条边或者是一个极大匹配

2. 选定一个 ***M*-非饱和顶点 *u***

2.1 如果存在一条以***u*** 为始点的***M*-增长路**，那么由此路可以得到一个新匹配 ***M'*** 其含有的边数 $|M'| = |M| + 1$

2.2 如果不存在一条以 ***u*** 为始点的 ***M*-增长路**，
那么选取另外一个还没有考虑过的 ***M*-非饱和顶点**

3. 重复**步骤 2** 直到所有顶点都是 ***M*-饱和**的
或者所有的 ***M*-非饱和**顶点都已经被考虑过了

5.5 最大匹配（续七）

ALGORITHM 8.1 *Local Search Algorithm*

Step 0 Initializing.
find an initial matching M (e.g., a maximal matching).
$c[v] := 0$ for all vertex $v \in V$ (all vertices are not considered).

Step 1 Determining if the current matching is a maximum matching.
if every vertex $v \in V$ is M-saturated
or every M-unsaturated vertex has been considered.
then stop and **return** M.

Step 2 Finding an M-augmenting path.
while there is an M-unsaturated vertex u with $c[u] = 0$ **do**
choose an M-unsaturated vertex u with $c[u] = 0$.
if there is no M-augmenting path starting from u
then $c[u] := 1$ (u has been considered.)
else go to Step 3.
end-while

Step 3 Producing a larger matching.
$M := M \oplus P$.
go to Step 1.

5.5 最大匹配（续八）

定理 2. 任给最大匹配问题的一个实例，局部搜索算法可以找到其最优解。

证明 根据算法的规则，当对所有的非饱和顶点都无法找到图***G***的一条增广路时，算法终止运行。此时，根据**推论1**，在每一次寻找以一个非饱和顶点为始点的增广路失败时，我们可以知道在原图 ***G*** 中也不存在这样一条路。因而根据**引理 2** 可知，图中不存在当前匹配的一条增长路，因此由 **定理1** 可知，当前的匹配即是一个最大的匹配。

我们称上述求解最大匹配问题的算法为局部搜索算法，这是因为我们可以将一个匹配 ***M*** 的邻域定义为所有可以经过某条***M*-增长路**得到的匹配集合。

5.5 最大 k-割

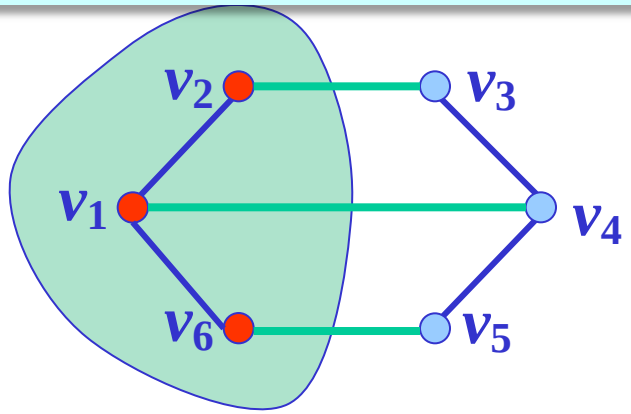
我们现在考虑最大(权) **k-割问题**。该问题的一种简单的特殊情形，所有边的权值都是一样的， 仍然是 **NP-难**的。

问　题: 最大 k-割问题

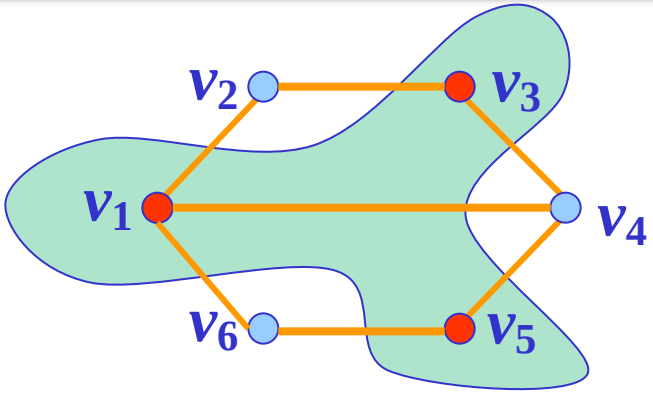
实　例: 图 $G(V, E)$，及每条边 $(u, v)\in E$ 上的赋权 $w(u, v)$

可行解: 顶点集V 的一个**k-划分**，$V_1\cup V_2\cup\ldots\cup V_k = V$, $V_i\cap V_j = \varnothing$

目　标: 最大化 k-割的权值 $w(V_1,\ldots,V_k) = \sum_{u\in V_i,\, v\in V_j} w(u, v)$



一个 2-割

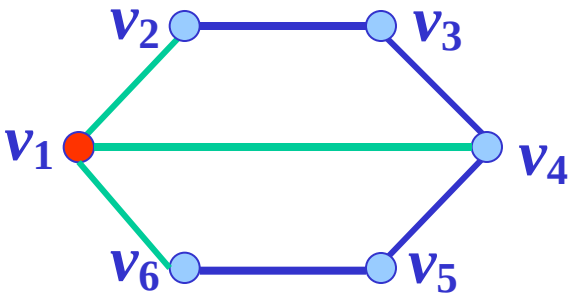


一个最大 2-割

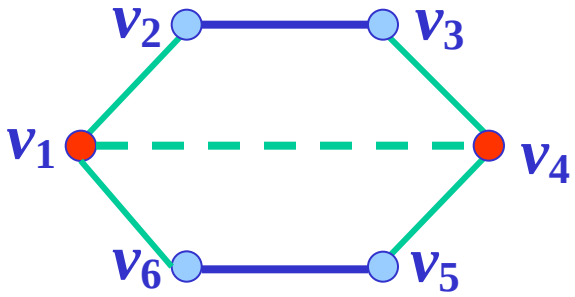
5.5 最大k-割（续一）

求解最大k-割问题的局部搜索算法的主要步骤如下：

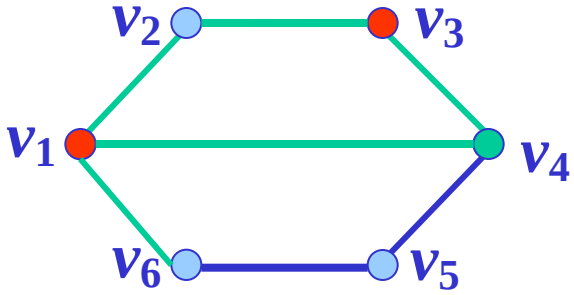
1. 任意选取一个顶点集的 k-划分
2. 将一个顶点从划分的一个集合中移动到另一个集合中
   只要能增加 k-割的权值
3. 重复步骤 2 直到无法再增加当前 k-割的权值



初始 2-割



改进后的 2-割



更好改进的2-割

5.5 最大*k*-割（续二）

ALGORITHM 8.4 *Vertex Moving Algorithm*

generate an initial partition $V_1, \cdots, V_k$;
$i := 1$,
$j := 0$.
while $j \leq |V|$ **do**
    $V_i' := V_i$.
    while $V_i' \neq \emptyset$ **do**
        pick a vertex $u \in V_i'$,
        $V_i' := V_i' \setminus \{u\}$.
        if there exists V_j such that $\sum_{v \in V_j} w((u,v)) > \sum_{v \in V_i} w((u,v))$
        then $V_j := V_j \cup \{u\}$, $V_i := V_i \setminus \{u\}$, $j := 0$.
        else $j := j + 1$.
    end-while
    if $i + 1 > k$ **then** $k := 1$
    else $i := i + 1$
end-while
return $V_1, \cdots, V_k$.

5.5 最大k-割（续三）

注意，如果我们将一个 k-划分的邻域定义为所有可以通过将其中一个集合中的某个顶点移动到另外一个集合可得到所有 k-划分，那么我们可以说上述算法就是一个局部搜索算法。

定理 3 任给最大 k-割问题的一个实例 I，设$c_{VM}(I)$ 为局部搜索算法输出的 k-割的权值，$c_{opt}(I)$ 是最优 k-割 的权值。则

$$c_{VM}(I)/c_{opt}(I) \geq 1/2,\ k = 2,$$

$$c_{VM}(I)/c_{opt}(I) \geq 1 - 1/(k-1),\ k \geq 2。$$

证明. 我们这里仅考虑 $k > 2$ 的情形。用 $w(S)$ 表示集合$S \subset E(G)$ 中所有边的权重之和，用 $V'(v)$ 表示所有与顶点 $u \in V' \subset V(G)$ 相关联的边 $(v, u) \in E$ 的集合。

5.5 最大k-割（续四）

假设集合组 $V_1, V_2, \ldots, V_k$ 构成算法输出的 k-划分，其权值为 $w(V_1,V_2, \ldots, V_k)$。我们用一个指标函数 $i(v)$ 表示当算法终止时，顶点 v 所在的集合的标号。例如，若 $i(v) = j$，顶点 v 属于集合 V_j。 因此我们有

$$w(V_1,V_2, \ldots, V_k) = \frac{1}{2}\sum_{v\in V}\left(\sum_{i\neq i(v)} w(V_i(v))\right)$$

求和号前面有一个分数 1/2 是因为每一条边的权值都计算了两次。此外注意

$$\sum_{i\neq i(v)} w(V_i(v)) \geq \sum_{i\neq j} w(V_i(v)),\ \text{对每个}\ j \neq i(v),$$

上面的不等式成立是因为如果对某个指标 $j \neq i(v)$ 这个不等式不成立，那么应该将顶点 v 从集合$V_{i(v)}$ 移至集合V_j 中。将上面的不等式对所有 $j \neq i(v)$ 相加，即得

5.5 最大k-割（续五）

$$(k-1)\sum_{i\neq i(v)} w(V_i(v)) \geq w\left(V_{i(v)}(v)\right) + (k-2)\sum_{i=1}^{k} w\left(V_i(v)\right)$$
$$\geq (k-2)\sum_{i=1}^{k} w(V_i(v)).$$

将以上两个不等式相加，即得如下不等式

$$w(V_1, V_2, \ldots, V_k) \geq \frac{1}{2}\sum_{v\in V}\left(\frac{k-2}{k-1}\sum_{i=1}^{k} w(V_i(v))\right)$$
$$= \frac{k-2}{k-1}\sum_{(u,v)\in E} w(V_i(v))$$
$$= \left(1-\frac{1}{k-1}\right) w(E)$$

由这个不等式即可推出定理的结论, 这是因为最大 k-割 的权值不会超过 $w(E)$。

5.5 最大k-割（续六）

定理 4. 对于非赋权最大 k-割问题，顶点移动算法的时间复杂度为 $O(|V|\,|E|)$。

证明 (练习)。

需要指出的是，对于赋权最大 k-割问题，顶点移动算法的运行时间为指数阶的！ 这是因为，相对于算法输出的局部最优 k-割的权值，算法在运行过程中每一次将一个顶点从一个集合移动至另一个集合中所增加的权值有可能非常的小。

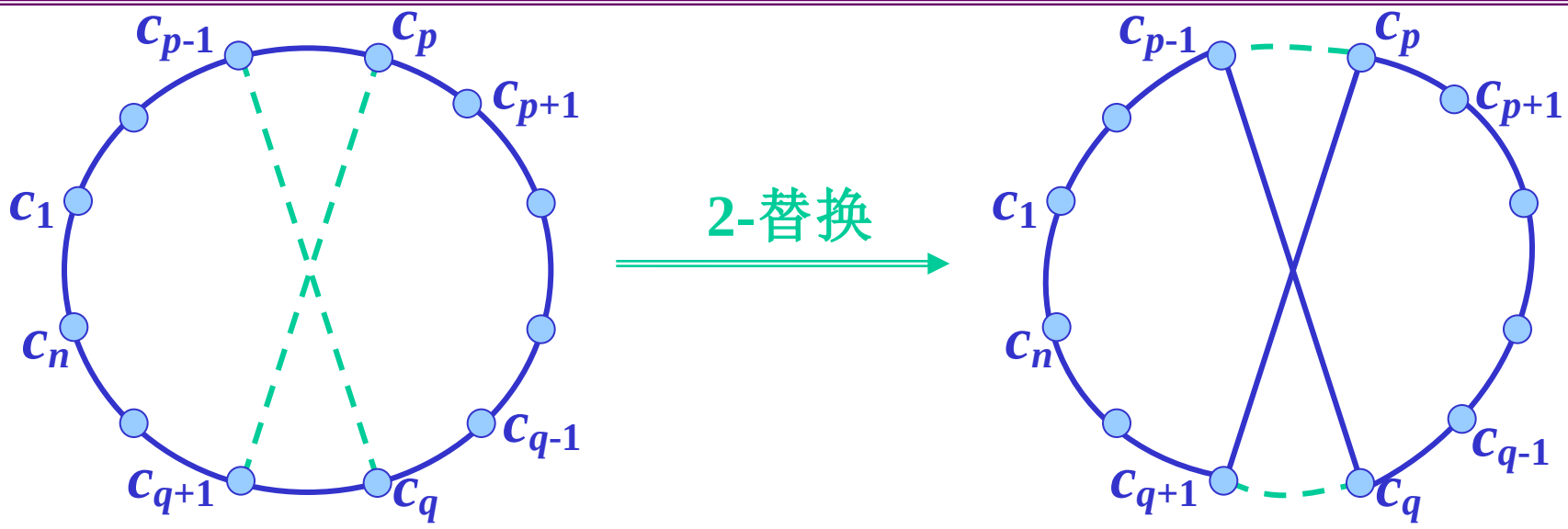
5.5 一般旅行商

我们再回来考虑一般旅行商问题。这一次我们要说明如何使用局部搜索技巧求解该问题。

给定一般旅行商问题的一个实例，要找到它的一个初始可行解是很容易，n 个城市的任意一个循环排列都对应一个可行解。或者我们可以应用一个贪婪算法，比如，最邻近城市方法，找到一个初始可行解。

对于对称的旅行商问题，即从城市 c_i 到城市 c_j 的距离与从城市 c_j 到城市 c_i 的距离是一样的，我们可以定义一个非常简单的邻域结果，称为 **2-最优**。它是基于以下一个基本性质：给定一个可行解 $T = (c_1\, c_2 \ldots c_n\, c_1)$，将其中的两条边 (c_{p-1}, c_p) 和 (c_q, c_{q+1}) 用另外两条边 (c_{p-1}, c_q) 和 (c_p, c_{q+1}) 替换，可以得到另外一个可行解 T'。 我们称这个替换为一个 **2-替换**。

5.5 一般旅行商（续一）

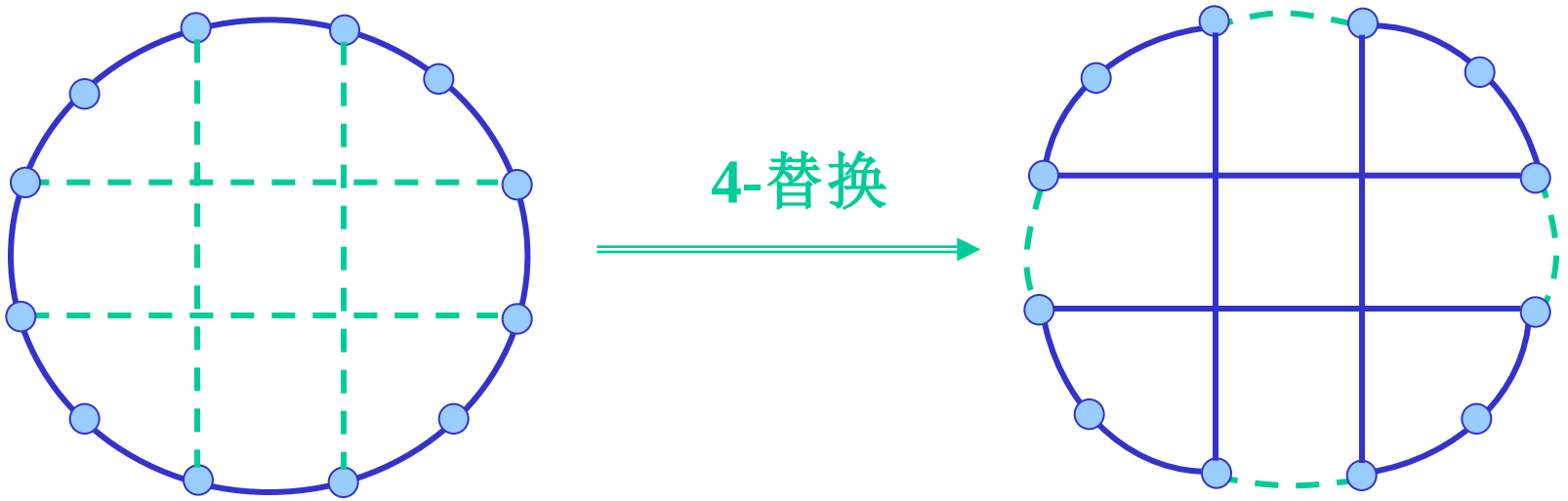


ALGORITHM 8.5 *The 2-Edge Interchange Algorithm*

$T := \{c_1, c_2, \cdots, c_n\}$ (an initial tour);
$N := \{(i, j) \mid i \neq j, i, j = 1, 2, \cdots, n\}$.
while $N \neq \emptyset$ **do begin**
    choose $(p, q) \in N$;
    $N := N \setminus \{(p, q)\}$.
    if $d(c_{p-1}, c_q) + d(c_p, c_{q+1}) < d(c_{p-1}, c_p) + d(c_q, i_{q+1})$ **then**
        $T := (c_1, c_2, \cdots, c_{p-1}, c_q, c_{q-1}, \cdots, c_p, c_{q+1}, \cdots, c_n)$;
end-while
return T.

5.5 一般旅行商（续二）

一个可行解 T 的 2-最优邻域结构包含所有可以经过2-替换所得到的可行解。根据这个定义可知，一个可行解的邻域内有 $n(n-1)/2$ 个相邻的可行解。因而，在一个可行解的邻域内找到一个更好的可行解最多需要 $O(n^2)$ 次 2-替换运算。



思考题: 如果一个局部搜索算法采用 k-替换， 那么其计算复杂度是多少呢？通过对任给的一个可行解进行一系列k-替换是否就可以得到最优解？

5.5 一般旅行商（续三）

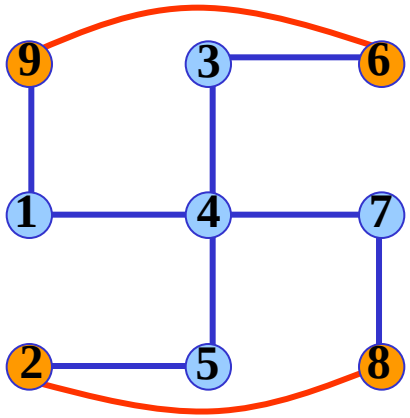
大量的数值模拟研究表明，采用 **3-替换**的算法得到的解比采用 ***k*-替换**得到解要好，但是运算时间也要长。然而，采用 ***k*-替换**的算法， $k>3$ ，运行时间显著增加，但是并没有显著的改善解的性能。因此，在实际应用中，很少采用 ***k*-替换**算法。

此外，数值模拟研究还表明，采用 ***k*-替换**的算法的运行时间不仅依赖邻域的大小（即 k 的数值大小），而且还依赖于初始解的选取。

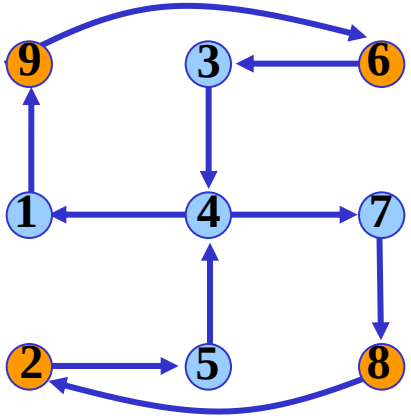
对于一般旅行商问题，我们无法在理论上确定采用 ***k*-替换**的算法得到解的目标函数值与最优值之比的一个上界。不过，对于欧氏旅行商问题，可以得到这样一个上界。

5.5 一般旅行商（续四）

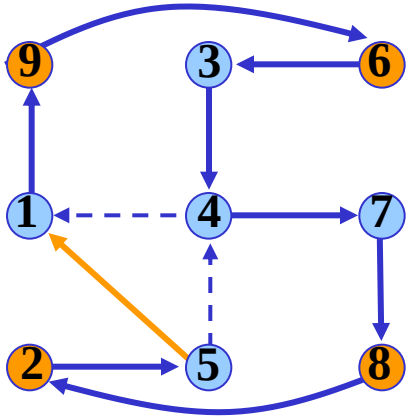
最后，我们给出求解度量空间上的旅行商问题的一个非常著名的**3/2-**近似算法， 它是 **N. Christofides** 于1976年的一篇研究报告中提出来的（从未正式发表，但被引用近千次）。下图给出了它的求解步骤。注意，它并不是一个局部搜索算法。



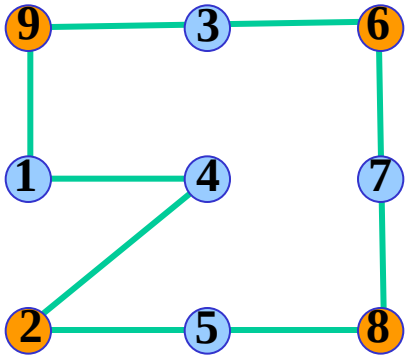
构造一棵最小生成树和其奇度点的完美匹配



构造一个欧拉回路



应用抄近路技巧得到一个哈密顿圈



一个最优解

5.5 一般旅行商（续五）

Christofides 算法的主要思想如下：

首先，找到一棵最小生成树 T_{MST}。用 W 表示 T_{MST} 中奇度数的顶点集合。在诱导子图 $G[W]$ 中构造一个完美匹配 M。然后，构造一个辅助图 $G'=(V', E')$，其边集合为 $E' = E(T_{MST}) \cup M$。显然，图 G' 是连通的，且其中的每一个顶点的度数都是偶数。

若 V' 中所有顶点的度数皆为 **2**，则图 G' 是一个可行解。否则，假设 $v \in V'$ 的度数至少是 **4**。则存在边 (u, v) 和 (v, w) 使得从图 G' 中去掉它们，并将边 (u, w) 添加进图 G' 中，所得到新图仍然是一个连通图。而且，它的所有顶点的度数仍然是偶数。这是因为，图 G' 存在一个欧拉回路，我们可以选取这个回路上相邻的两条边 (u, v) 和 (v, w)。再用抄近路技巧，并重复此过程直到 V' 中所有顶点的度数皆为**2**（即得一个可行解）。

5.5 一般旅行商（续六）

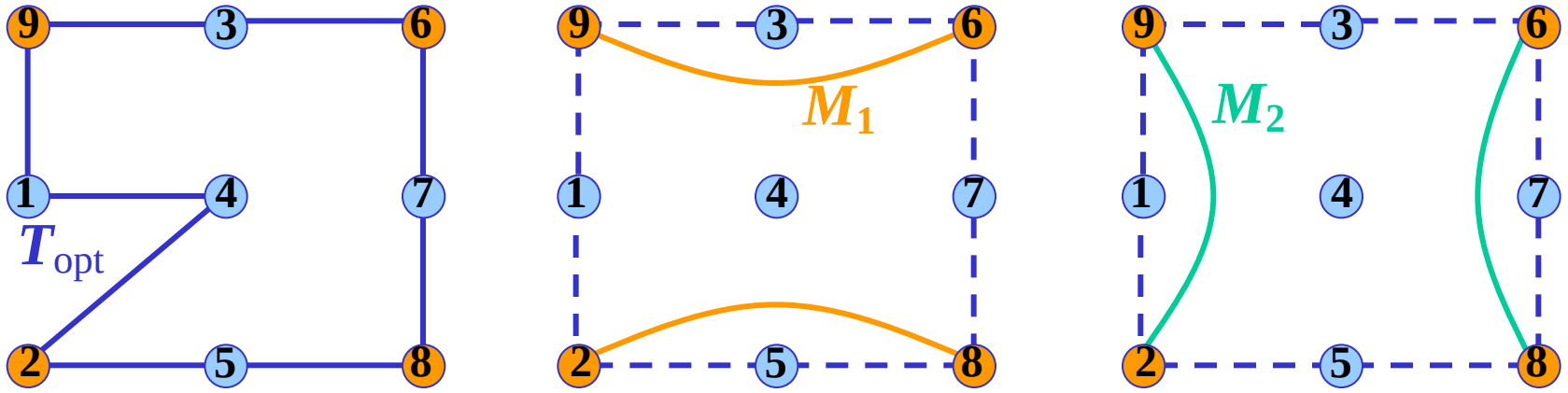
定理 5 任给一般旅行商问题的一个实例 **I**，**Christofides** 算法在时间 $O(|V|^4)$ 内输出一个可行解其长度$c(T_C)$ 不超过最优解长度 $c(T_{opt})$ 的**3/2** 倍。

证明 注意图 G' 是由最小生成树 T_{MST} 和完美匹配 M 组成。因而有 $c(T_c) \le c(E') = c(T_{MST}) + c(M)$。 现假设 T_{opt} 是一个最优解，且 $i_1, i_2, \ldots, i_{2m}$ 是T_{MST} 中奇度数的顶点集合 W', 它们也是以此顺序出现在最优解 T_{opt} 中。考虑 W'的两个匹配，

$$M_1 = \{(i_1, i_2), (i_3, i_4), \ldots, (i_{2m-1}, i_{2m})\} \text{ 和}$$
$$M_2 = \{(i_2, i_3), (i_4, i_5), \ldots, (i_{2m}, i_1)\}。$$

再根据三角不等式，可得 $c(T_{opt}) \ge c(M_1) + c(M_2)$。故有$c(T_{opt}) \ge 2c(M)$ 这是因为 M 是权值最小的一个匹配。此外，还有 $c(T_{MST}) \le c(T_{opt})$。 因此，最后可得 $c(T_C) \le 3c(T_{opt})/2$。

5.5 一般旅行商（续七）



实际上，我们可以构造一般旅行商问题的一系列实例，当用 **Christofides 算法**求解这些实例时，所得近似解的长度与最优解的长度之比可以趋于 **3/2**。

练习 借助下图实例说明**Christofides 算法**的近似比是**3/2**。

5.5 总结

对于一个具体的优化问题，我们可以定义很多种可能的邻域结构，它们会使得相应的局部搜索算法的（近似）解的性能不一样，运行时间也会不一样。在我们设计和选取邻域结构的时候，需要考虑的主要因素包括：

- 局部最优解与全局最优解是否很接近
- 如何在一个邻域中搜索更好的一个解
- 要验证在一个邻域中不存在更好的一个解是否很困难
- 需要生成多少个解才能找到一个局部最优解

可以看出以上四个因素是相互关联的。实际上，如果邻域太大的，那么搜索到一个更好的解，或者证明不存在更好的一个解，会变得非常复杂和费时。