Vertex Weighting-Based Tabu Search for *p*-Center Problem

张庆雲

Introduction

- ✓ The p-center problem consists of choosing p centers from a set of candidate centers to serve a set of clients, where each client is served by one of its closest centers.

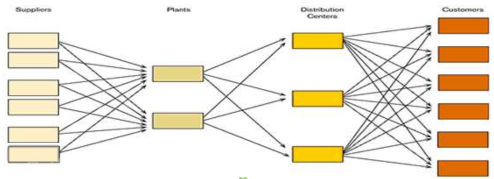
- ✓ The p-center problem is a classical combinatorial optimization problem

- ✓ P-center problem is a challenging NP-hard problem.

Introduction

✓ Real-world applications

city planning



supply-chain management

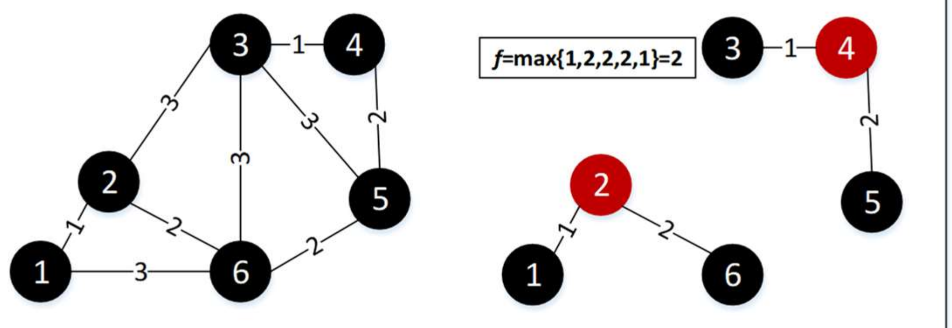
Problem description

✓已知：无向连通图$G(V,E)$，图中任意两点的距离，服务点数P

✓决策变量：从N（一般：候选中心=用户节点）个节点中选择P个节点作为服务节点（记为集合X），服务剩下的（$N-P$）个用户节点

✓约束条件：每个节点由离它最近的服务节点唯一提供服务，这条边称为服务边

✓优化目标：最小化所有用户服务边的最大值f ($f = \max(\min(d_{ij}))$)



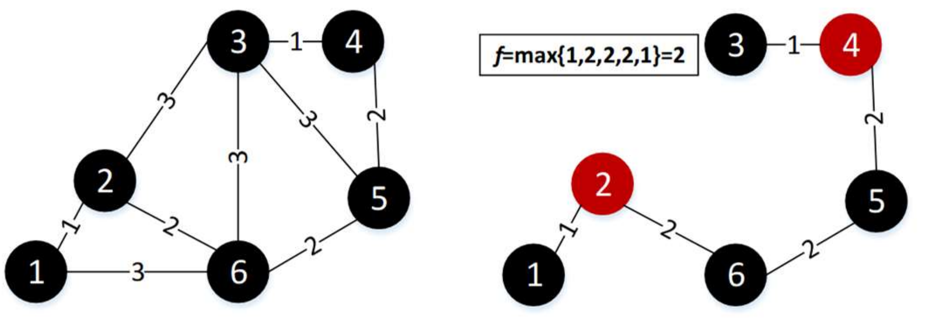
(a) Original graph G of *p*-center problem. (b) The best solution of G when p=2

Problem description

$$
\begin{aligned}
(PC) \quad & \min\ r, && (1)\\
\text{s.t.}\ & \sum_{j\in C} x_j \le p, && (2)\\
& \sum_{j\in C} y_{ij} = 1, \forall i \in V, && (3)\\
& y_{ij} \le x_j, \forall i \in V, \forall j \in C, && (4)\\
& \sum_{j\in C} d_{ij} y_{ij} \le r, \forall i \in V, && (5)\\
& x_j, y_{ij} \in \{0,1\}, r \in R^+, \forall i \in V, \forall j \in C. && (6)
\end{aligned}
$$

$x_j = 1$: Candidate facility *j* is opened as a center

$y_{ij} = 1$: Client *i* is served by facility *j*



(a) Original graph G of *p*-center problem. (b) The best solution of G when p=2

Problem transformation

P-center 问题优化转判定:

✓Given an ordered list $\Gamma = \{r_1, r_2, ..., r_k\}$ of distinct edge lengths.

Problem transformation

Set-covering problem

✓问题描述：给出一个集合集$S = \{S_1, S_2, \dots S_n\}$和一个元素集$V = \{1,2,3, \dots m\}$

✓决策条件：从N个集合中选择k个集合覆盖所有的M个元素

✓约束条件：每个元素只能由可以覆盖它的集合覆盖

✓优化目标：最小化集合数k。

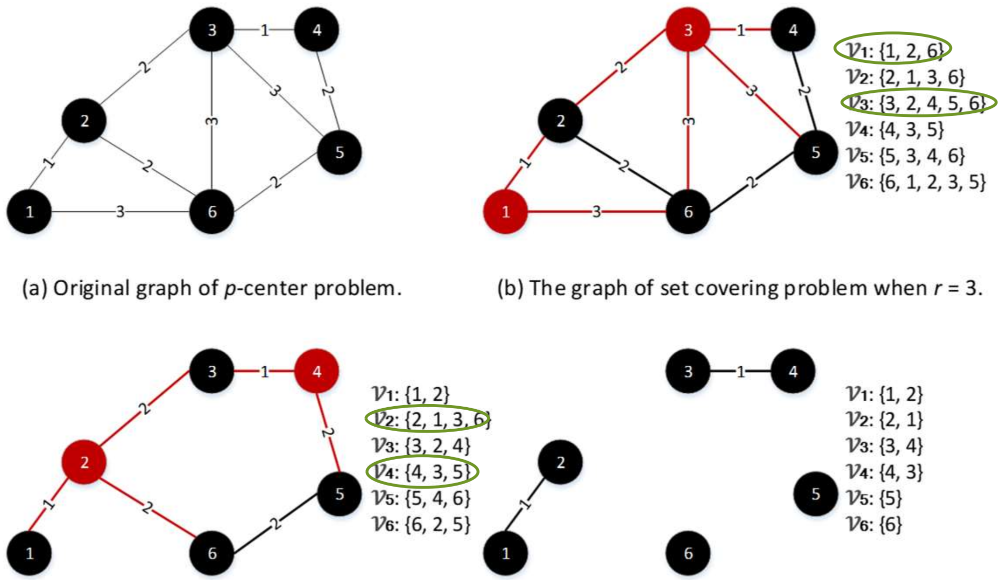
如果将集合覆盖问题的集合数确定：

目标：寻找k个集合，覆盖当前所有的元素

Problem transformation

? p-center问题和集合覆盖问题之间的联系

✓如何转化



(a) Original graph of p-center problem. (b) The graph of set covering problem when $r = 3$.

(c) The graph of set covering problem when $r = 2$. (d) The graph of set covering problem when $r = 1$.

p=2

$\Gamma \rightarrow \{3,2,1\}$

✓For covering radius r_q, we are asked whether there exists p centers such that all the clients can been covered within the covering radius r_q.

Problem transformation

原问题转换后的模型

$$(SC_q) \quad \min \sum_{i \in V} u_i, \tag{7}$$

$$\text{s.t.} \sum_{j \in C, d_{ij} \leq r_q} x_j \geq 1 - u_i, \forall i \in V, \tag{8}$$

Client is covered by at least one center

Covering radius

$$\sum_{j \in C} x_j = p, \tag{9}$$

$$x_j, u_i \in \{0,1\}, \forall i \in V, \forall j \in C. \tag{10}$$

$u_i = 1$: Client i is uncovered

Transforming *p*-center problem to a series of decision subproblem (set cover problem)

Solving Procedure

✓首先使用优化模型求解得到一个较好的初始半径r_0。

✓从r_0开始求解p −center问题，一旦半径r_0时得到可行解，我们将根据序列表$\Gamma = \{r_0, r_1, \ldots, r_k\}$ 继续缩小半径，然后继续求解，直到在时间限制内无法找到一个可行解 (可以覆盖所有节点的解）。

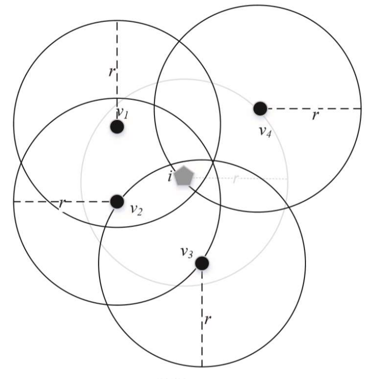
✓我们的重点将放在给定半径r_q的模型(SC_q)的解决方法上。

Local Search

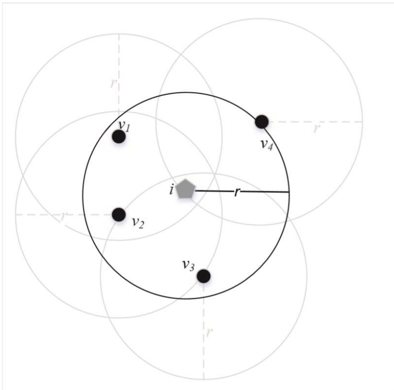
Symbol

C_i: gives the set of candidate centers which are able to serve client i, that is, $C_i = \{j \in C | i \in \mathcal{V}_j\}$.

$\mathcal{V}_j$: denote the set of clients that candidate center j can serve within the current covering radius.



C_i



$\mathcal{V}_j$

$C_i = \mathcal{V}_i$

Initial Solution--Greedy

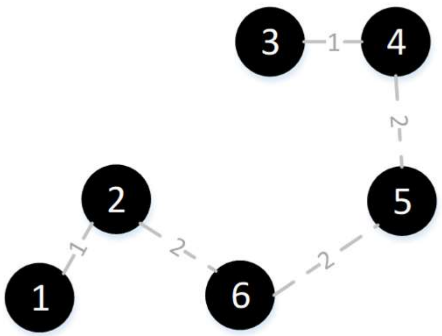
✓The constructive heuristic opens centers one by one under a maximal coverage principle

✓ It iteratively selects a candidate center *j* which covers most uncovered clients and inserts it into the current solution *X*

$$j = \arg max_{j \in C/X} |\mathcal{V}_j \cap \mathrm{U(X)}$$

The set of clients that are not served in solution X.

✓ If there are multiple candidate centers covering the same number of uncovered clients, ties are broken randomly.

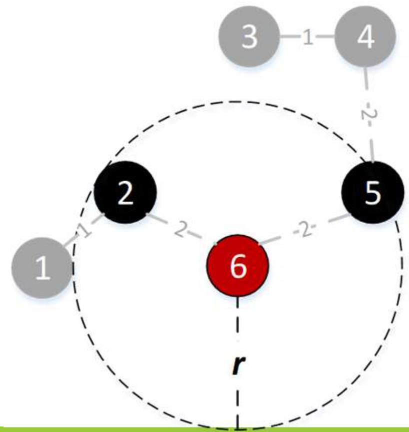


$\mathcal{V}_1$: {1, 2}
$\mathcal{V}_2$: {2, 1, 6}
$\mathcal{V}_3$: {3, 4}
$\mathcal{V}_4$: {4, 3, 5}
$\mathcal{V}_5$: {5, 4, 6}
$\mathcal{V}_6$: {6, 2, 5}

p=2

U(X)={1,2,3,4,5,6}

Random-6



$\mathcal{V}_1$: {1, 2}
$\mathcal{V}_2$: {2, 1, 6}
$\mathcal{V}_3$: {3, 4}
$\mathcal{V}_4$: {4, 3, 5}
$\mathcal{V}_5$: {5, 4, 6}
$\mathcal{V}_6$: {6, 2, 5}

U(X)={1,3,4}

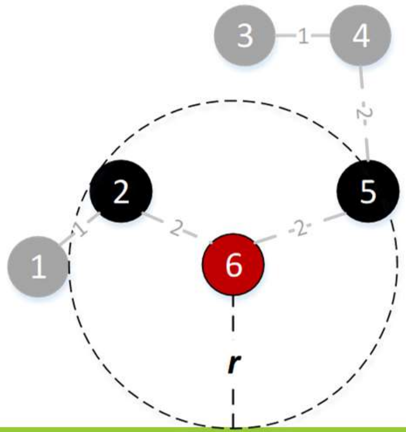
Initial Solution--Greedy

✓The constructive heuristic opens centers one by one under a maximal coverage principle

✓ It iteratively selects a candidate center *j* which covers most uncovered clients and inserts it into the current solution *X*

$$j = \arg max_{j\in C/X}|\mathcal{V}_j \cap \mathrm{U(X)}$$

✓ If there are multiple candidate centers covering the same number of uncovered clients, ties are broken randomly.

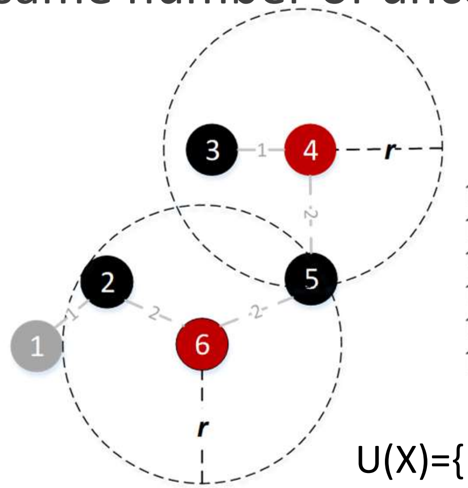


$\mathcal{V}_1$: {1, 2}
$\mathcal{V}_2$: {2, 1, 6}
$\mathcal{V}_3$: {3, 4}
$\mathcal{V}_4$: {4, 3, 5}
$\mathcal{V}_5$: {5, 4, 6}
$\mathcal{V}_6$: {6, 2, 5}

$\mathcal{V}_1 \cap U(X)$={1}
$\mathcal{V}_2 \cap U(X)$={1}
$\mathcal{V}_3 \cap U(X)$={3,4}
$\mathcal{V}_4 \cap U(X)$={3,4}
$\mathcal{V}_5 \cap U(X)$={4}

U(X)={1,3,4}

Random-4



$\mathcal{V}_1$: {1, 2}
$\mathcal{V}_2$: {2, 1, 6}
$\mathcal{V}_3$: {3, 4}
$\mathcal{V}_4$: {4, 3, 5}
$\mathcal{V}_5$: {5, 4, 6}
$\mathcal{V}_6$: {6, 2, 5}

U(X)={1}   X={4,6}

Evaluation function

✓The optimization objective is to minimize the number of uncovered vertices.

$$\min f(X) = \sum_{\forall i \in V} w_i u_i$$

$$u_i = \begin{cases} 0, & \text{Client } i \text{ can be covered by some centers,} \\ 1, & \text{Client } i \text{ can not be covered.} \end{cases}$$

w_i: represent the weight of vertex i, initial assignment is 1

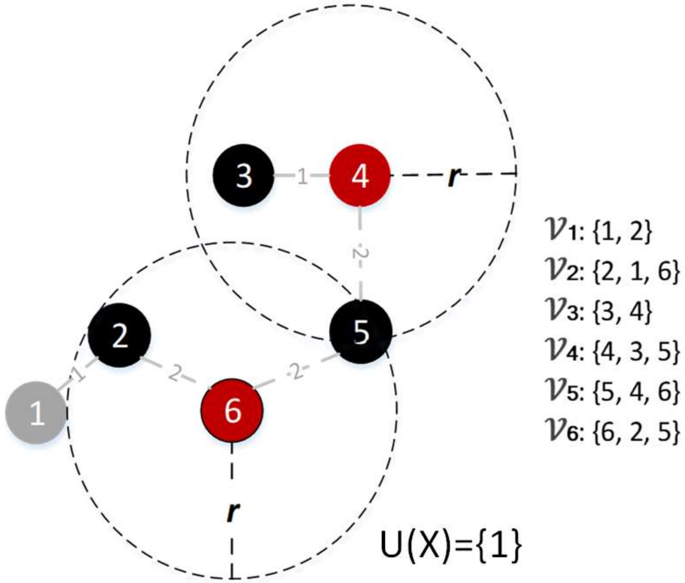
Neighborhood structure

✓Swap

- A swap move produces a neighboring solution: opening a candidate facility $i \in C \setminus X$ as a center (add i to the set of centers), and closing another center $j \in X$ (remove j from the center set);
- The neighborhood size is $p * (n - p)$;
- The objective value can only be improved by covering some uncovered vertices, so the VWTS algorithm will only evaluate a swap move $Swap(i, j)$ if i covers some uncovered vertices in $U(X)$.

Swap-add center

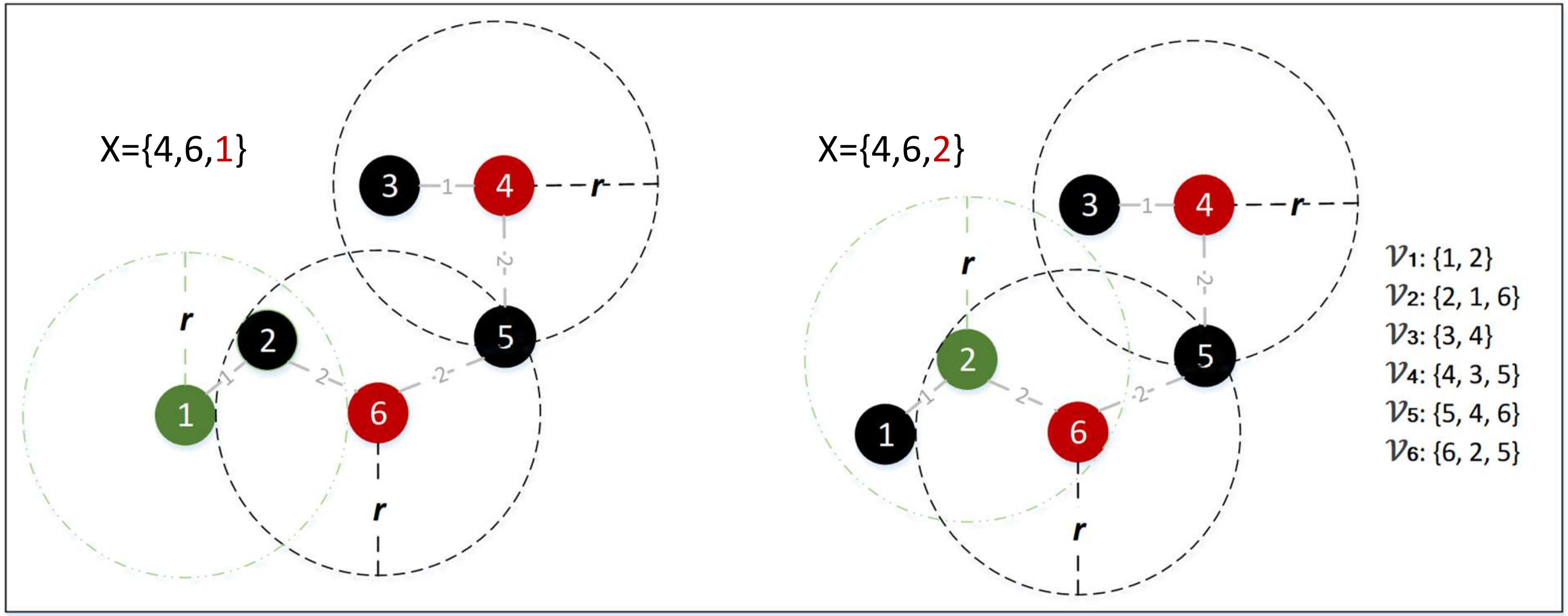
- ✓ 找到未覆盖节点集合$U(X) = 1$（如果有多个未覆盖节点时，随机选择一个）
- ✓ 能覆盖节点1的集合$\mathcal{C}_1 = \{1,2\}$
- ✓ 分别试探加入1,2中的一个节点，形成p+1个节点的中间解



$X = \{4,6\}$

Swap-add center

- ✓ 找到未覆盖节点集合$U(X) = 1$（如果有多个未覆盖节点时，随机选择一个）
- ✓ 能覆盖节点1的集合$\mathcal{C}_1 = \{1,2\}$
- ✓ 分别试探加入1,2中的一个节点，形成p+1个节点的中间解



Neighborhood Evaluation

- ✓ we use an incremental evaluation technique to accelerate the evaluation.

- ✓The effect of each move on the objective function can be quickly calculated by a special data structure.

- ✓Each time a move is carried out, only the move values affected by this move are updated accordingly.

Neighborhood Evaluation

	公式	含义	备注
$j \in X$	$\delta_j = \sum_{i \in \mathcal{V}_j \cap U(X \setminus \{j\})} w_i$	表示只能由中心j覆盖的节点的权重和	j在当前解中
$j \notin X$	$\delta_j = \sum_{i \in \mathcal{V}_j \cap U(X)} w_i$	表示能被j覆盖的所有未覆盖节点的权重和	j不在当前解中

✓加入节点i, $f(X \cup \{i\}) = f(X) - \delta_i$

✓删除中心j, $f(X \setminus \{j\}) = f(X) + \delta_j$

✓邻域评估,$swap(i,j)$： $f(X \oplus Swap(i,j)) = f(X) - \delta_i + \delta_j$

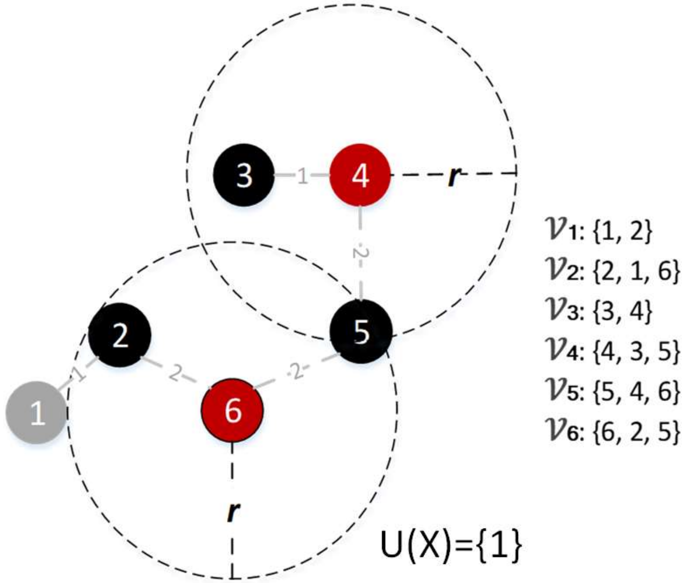
Neighborhood Evaluation

Algorithm 3 Open a center virtually

```
function TryToOpenCenter(i)
    for all v ∈ V_i do    /* |X ∩ C_v|: number of centers */
        if |X ∩ C_v| = 1 then          /* covering v in X */
            /* cancel penalty for making v uncovered */
            δ_l ← δ_l − w_v, for l ∈ X ∩ C_v          /* O(1) */
        end if   /* l was the only center covering v but */
    end for  /* it will not be the only one if i opens so */
end function  /* closing l does not make v uncovered */
```

时间复杂度**O(n)**

Find the best swap pair



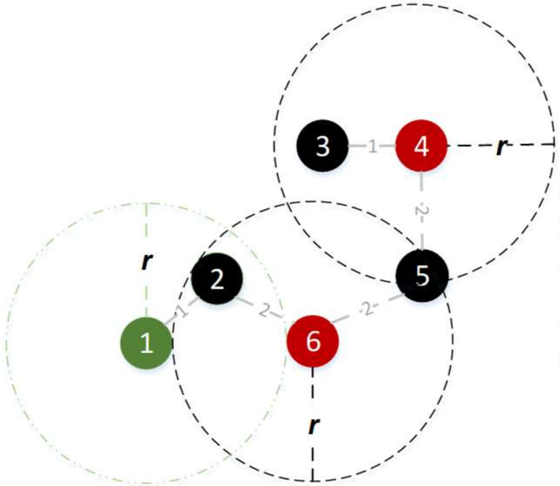
$X = \{4,6\}$

$f(X) = 1$

i	δ_i
1	1
2	1
3	0
4	2
5	0
6	2

Find the best swap pair

加入节点1



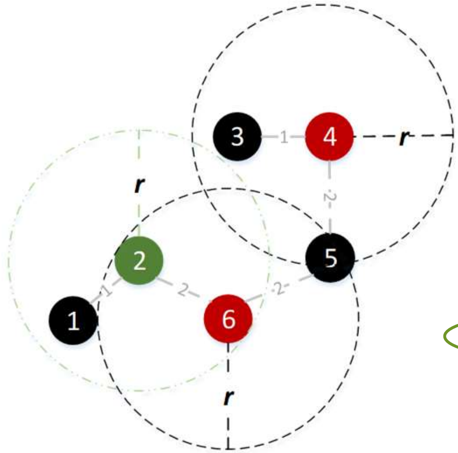
$\mathcal{C}_1=\mathcal{V}_1$: {1, 2}
$\mathcal{C}_2=\mathcal{V}_2$: {2, 1, 6}
$\mathcal{C}_3=\mathcal{V}_3$: {3, 4}
$\mathcal{C}_4=\mathcal{V}_4$: {4, 3, 5}
$\mathcal{C}_5=\mathcal{V}_5$: {5, 4, 6}
$\mathcal{C}_6=\mathcal{V}_6$: {6, 2, 5}

i	δ_i'
1	1
2	1
3	0
4	2
5	0
6	2-1

初始所有节点权重w=1

$$f(X) \cup \{1\} = 1 - 1 = 0$$

加入节点2



$\mathcal{C}_1=\mathcal{V}_1$: {1, 2}
$\mathcal{C}_2=\mathcal{V}_2$: {2, 1, 6}
$\mathcal{C}_3=\mathcal{V}_3$: {3, 4}
$\mathcal{C}_4=\mathcal{V}_4$: {4, 3, 5}
$\mathcal{C}_5=\mathcal{V}_5$: {5, 4, 6}
$\mathcal{C}_6=\mathcal{V}_6$: {6, 2, 5}

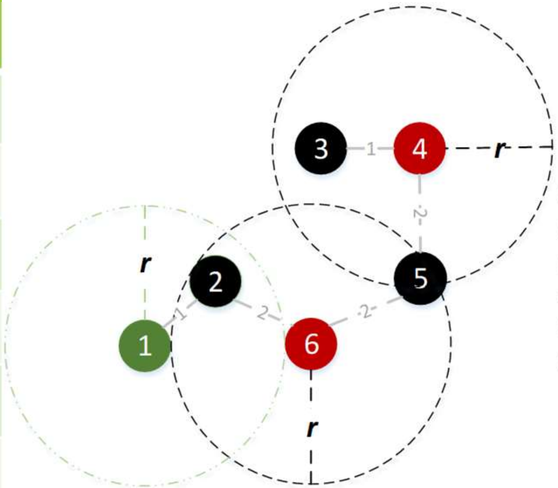
i	δ_i'
1	1
2	1
3	0
4	2
5	0
6	2-2

$$f(X) \cup \{2\} = 1 - 1 = 0$$

Find the best swap pair

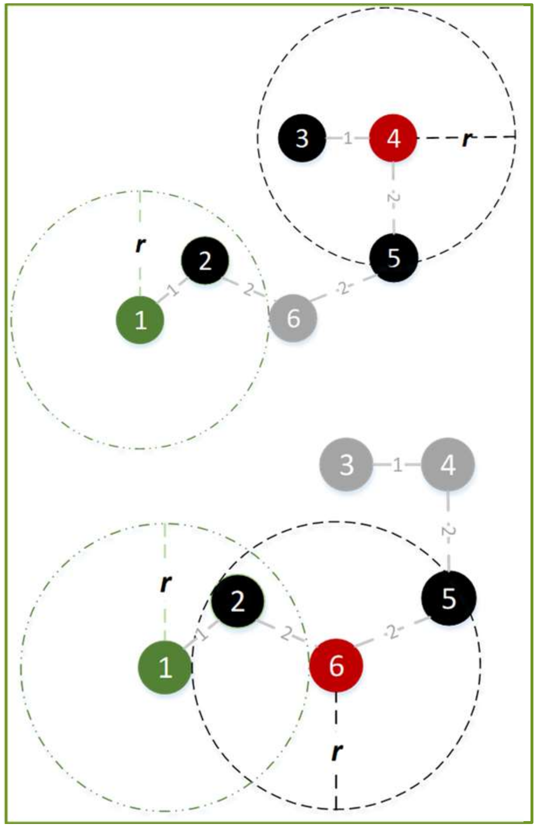
✓添加后的p+1个节点的中间解，需要一次试探删除原始的中心节点4,6，寻找最优的交换对。

i	δ_i'
1	1
2	1
3	0
4	2
5	0
6	1



$\mathcal{C}_1=\mathcal{V}_1$: {1, 2}
$\mathcal{C}_2=\mathcal{V}_2$: {2, 1, 6}
$\mathcal{C}_3=\mathcal{V}_3$: {3, 4}
$\mathcal{C}_4=\mathcal{V}_4$: {4, 3, 5}
$\mathcal{C}_5=\mathcal{V}_5$: {5, 4, 6}
$\mathcal{C}_6=\mathcal{V}_6$: {6, 2, 5}

$f(X) \cup \{1\} = 0$



Swap(1,6)

$$f(X \oplus Swap(1,6)) \\ = f(X \cup \{6\}) + \delta_6 \\ = 0 + 1 = 1$$

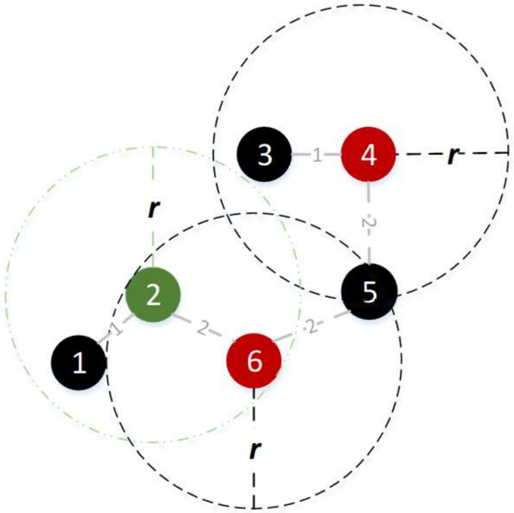
Swap(1,4)

$$f(X \oplus Swap(1,4)) \\ = f(X \cup \{4\}) + \delta_4 \\ = 0 + 2 = 2$$

Find the best swap pair

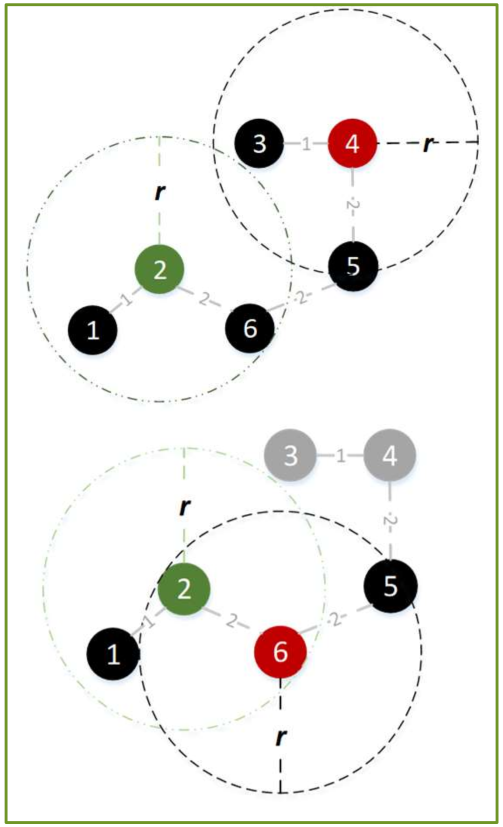
✓添加后的p+1个节点的中间解，需要一次试探删除原始的中心节点4,6，寻找最优的交换对。

i	δ_i'
1	1
2	1
3	0
4	2
5	0
6	0



$\mathcal{C}_1 = \mathcal{V}_1$: {1, 2}
$\mathcal{C}_2 = \mathcal{V}_2$: {2, 1, 6}
$\mathcal{C}_3 = \mathcal{V}_3$: {3, 4}
$\mathcal{C}_4 = \mathcal{V}_4$: {4, 3, 5}
$\mathcal{C}_5 = \mathcal{V}_5$: {5, 4, 6}
$\mathcal{C}_6 = \mathcal{V}_6$: {6, 2, 5}

$$f(X) \cup \{2\} = 0$$



Swap(2,6)

$$f(X \oplus Swap(2{,}6)) = f(X \cup \{6\}) + \delta_6 = 0 + 0 = 0$$

Swap(2,4)

$$f(X \oplus Swap(2{,}4)) = f(X \cup \{4\}) + \delta_4 = 0 + 2 = 2$$

Find the best swap pair

Algorithm 2 Find the best swap pair

function FINDPAIR($X, TL, iter$)
  The set of best swap moves $M \leftarrow \varnothing$
  The best objective value $obj \leftarrow +\infty$
  $v \leftarrow$ a randomly picked uncovered vertex in $U(X)$
  $\delta'_j \leftarrow \delta_j, \forall j \in C$ /* backup before trial moves */
  for all $i \in \mathcal{C}_v$ **do** /* $\mathcal{C}_v$: candidates covering v */
    TryToOpenCenter(i) /* (Algorithm 3) */
    for all $j \in X$ **do** /* evaluate closing center j */
      if $\{i, j\} \cap TL = \varnothing$ **then** /* not tabu move */
        if $f(X \oplus \text{Swap}(i, j)) < obj$ **then**
          $obj \leftarrow f(X \oplus \text{Swap}(i, j))$
          $M \leftarrow \{\text{Swap}(i, j)\}$
        else if $f(X \oplus \text{Swap}(i, j)) = obj$ **then**
          $M \leftarrow M \cup \{\text{Swap}(i, j)\}$
        end if
      end if
    end for
    $\delta_j \leftarrow \delta'_j, \forall j \in C$ /* restore after trial moves */
  end for /* $v \in U(X) \Leftrightarrow \mathcal{C}_v \cap X = \varnothing$ */
  return a randomly picked move in M
end function

最坏情况下时间复杂度为$O(n^2)$
在节点均匀分布时$|\mathcal{C}_v| \approx n/p$，所以
复杂度接近$O(n * p)$

Make a swap move

✓根据目标函数，取$f(X \oplus Swap(i,j))$最小的一对节点对进行交换，并更新所有候选中心的δ值。

Algorithm 4 Make a swap move

function MAKEMOVE(i, j)
  for all $v \in \mathcal{V}_i$ **do** /* consequences of opening i */
    if $|X \cap \mathcal{C}_v| = 1$ **then** /* (Algorithm 3) */
      $\delta_l \leftarrow \delta_l - w_v$, for $l \in X \cap \mathcal{C}_v$
    else if $|X \cap \mathcal{C}_v| = 0$ **then**
      $\delta_l \leftarrow \delta_l - w_v, \forall l \in \mathcal{C}_v \setminus \{i\}$
    end if /* cancel reward for covering v */
  end for
  $X \leftarrow X \cup \{i\} \setminus \{j\}$
  for all $v \in \mathcal{V}_j$ **do** /* consequences of closing j */
    if $|X \cap \mathcal{C}_v| = 0$ **then** /* add reward for */
      $\delta_l \leftarrow \delta_l + w_v, \forall l \in \mathcal{C}_v \setminus \{j\}$ /* covering v */
    else if $|X \cap \mathcal{C}_v| = 1$ **then**
      $\delta_l \leftarrow \delta_l + w_v$, for $l \in X \cap \mathcal{C}_v$
    end if /* add penalty for uncovering v */
  end for
end function

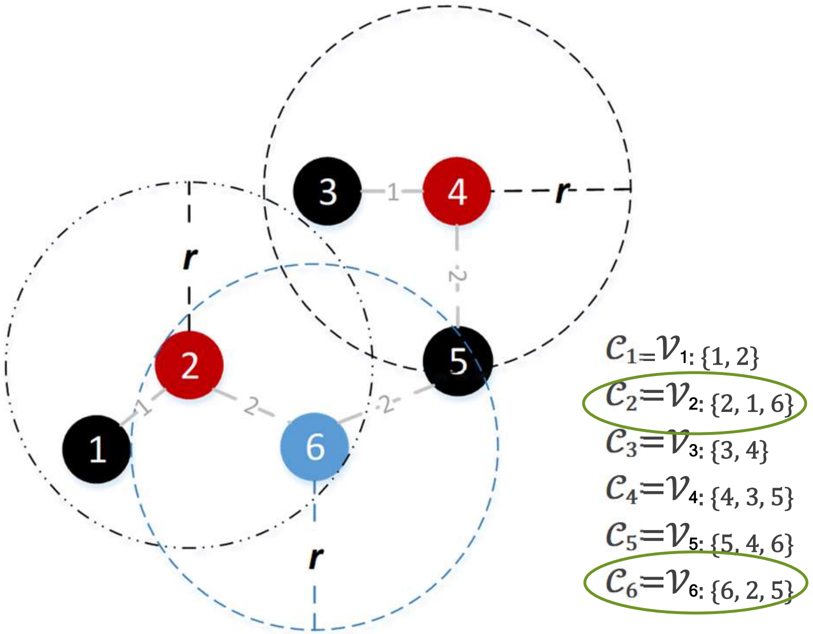
Make a swap move

✓根据目标函数，取$f(X \oplus Swap(i,j))$最小的一对节点对进行交换，并更新所有候选中心的δ值。

i	δ_i
1	1
2	1
3	0
4	2
5	0
6	2

Swap(2,6)

i	δ_i
1	1-1=0
2	1+2=3
3	0
4	2+1=3
5	0
6	2-2=0



Tabu Search

✓Tabu search usually incorporates a recency-based tabu list to prohibit revisiting recently visited solutions;

✓The tabu strategy prevents closing newly opened centers or reopening newly closed ones immediately

Tabu Tenure

✓For the tabu list, $Swap(i, j)$ at iteration $iter$ introduces two vertices $\{i, j\}$ in tabu list TL and neither i nor j can be involved at the next tt iterations.

✓Here, the tabu tenure tt is statically determined by

$$tt = 1$$

TabuTenure Table

$Vertex$	TL_i
1	0
2	0
3	0
4	0
5	0
6	0

$Vertex$	TL_i
1	0
2	0+1
3	0
4	0
5	0
6	0+1

Swap(2,6)

Fast Implementation

$Vertex$	TL_i
1	0
2	0
3	0
4	0
5	0
6	0

$Vertex$	TL_i
1	0
2	10+1
3	0
4	0
5	0
6	10+1

$$tt = Iter + 1$$

Swap(2,6)

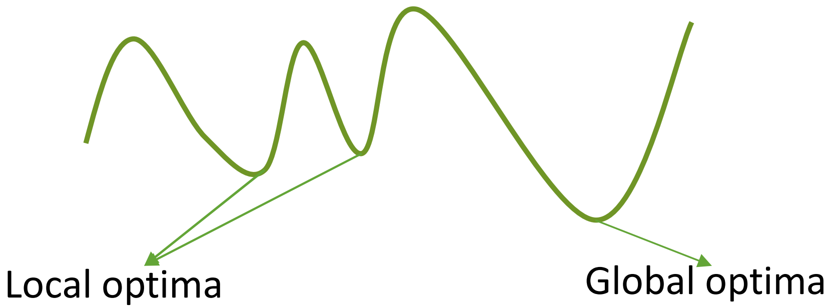
Vertex Weighting Technique

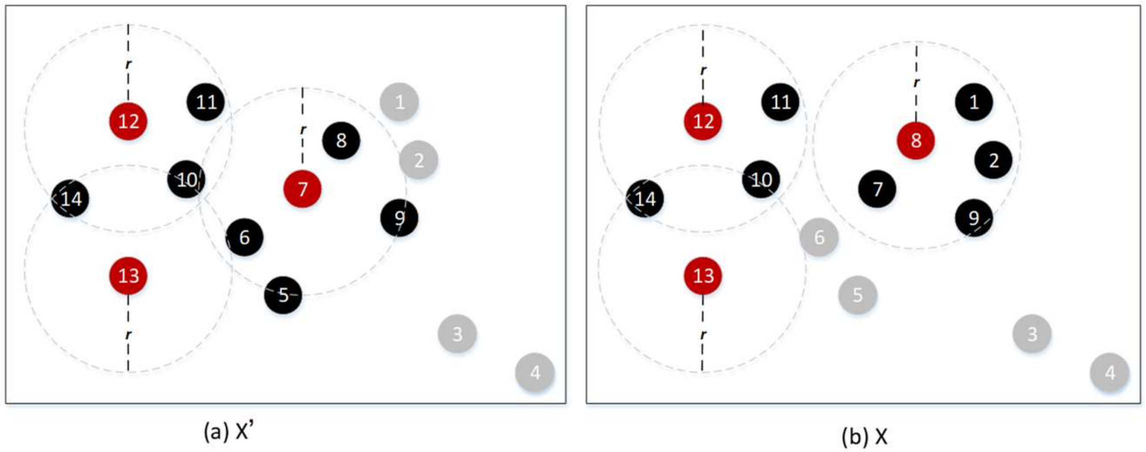
Vertex Weighting Technique

- ✓ If the algorithm keeps failing to cover a client, it implies that this vertex is hard to cover and we should treat it with higher priority;
- ✓ When the tabu search is trapped in local optimal solution X, the algorithm increases the weight w_i of each uncovered client $i \in U(X)$ by one unit.
- ✓ This process changes the landscape of the solution space such that X is no longer a local optimum, and the search will be able to continue to explore other search areas.

Vertex Weighting Technique

✓When the tabu search is trapped in local optimal solution X, the algorithm increases the weight w_i of each uncovered client $i \in U(X)$ by one unit.





$U(X) = U(X') = 4 \longrightarrow w_5 = w_5+1\ w_6 = w_6+1$

$w_3 = w_3+1\ w_4 = w_4+1$

Vertex Weighting Technique

Effectiveness

✓The vertex weighting technique is able to prevent vertices from being repeatedly uncovered and diversify the search in an adaptive manner;

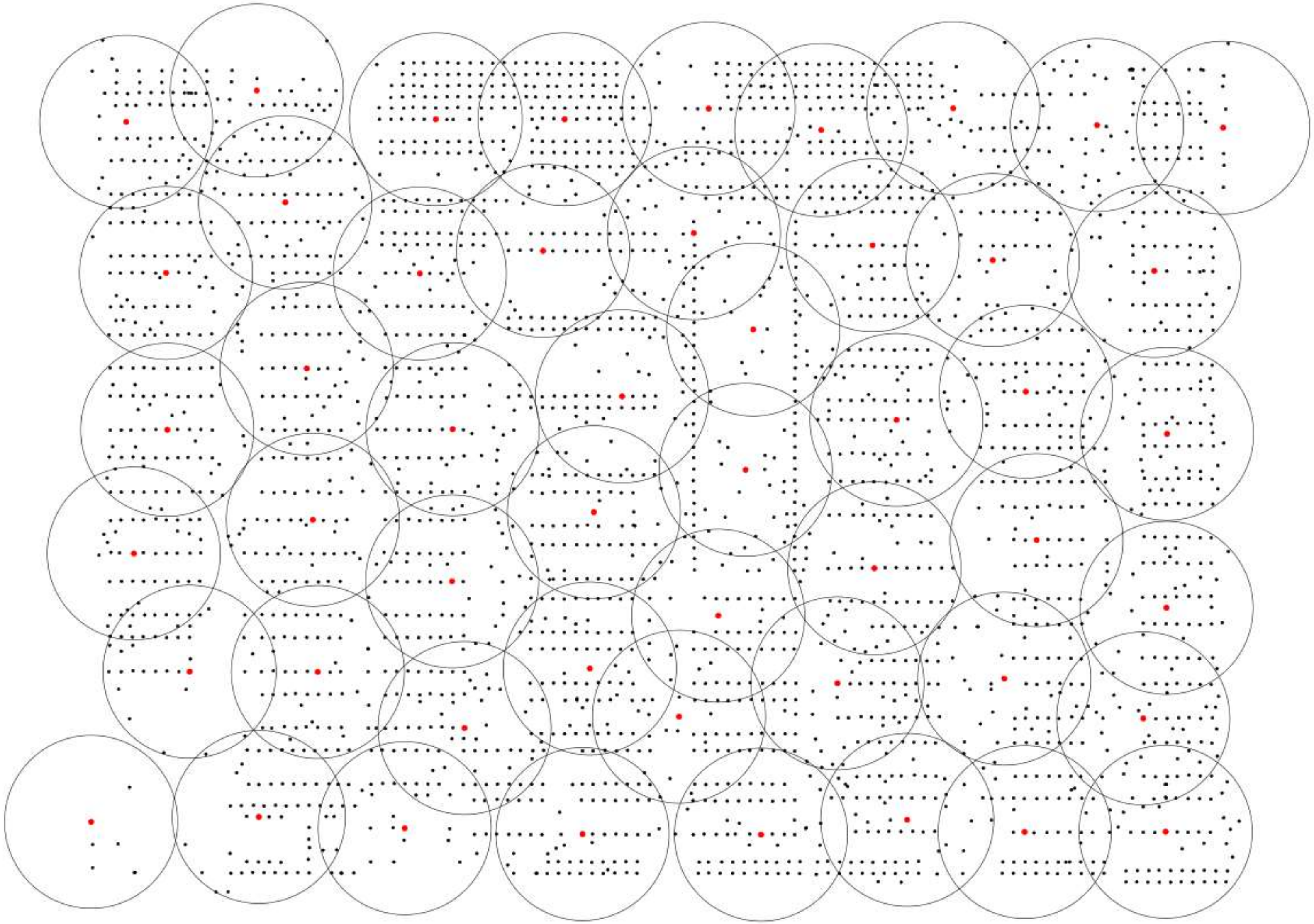
✓It modifies the solution space in a smooth way, which can guide the search to promising search regions.

The VWTS Algorithm

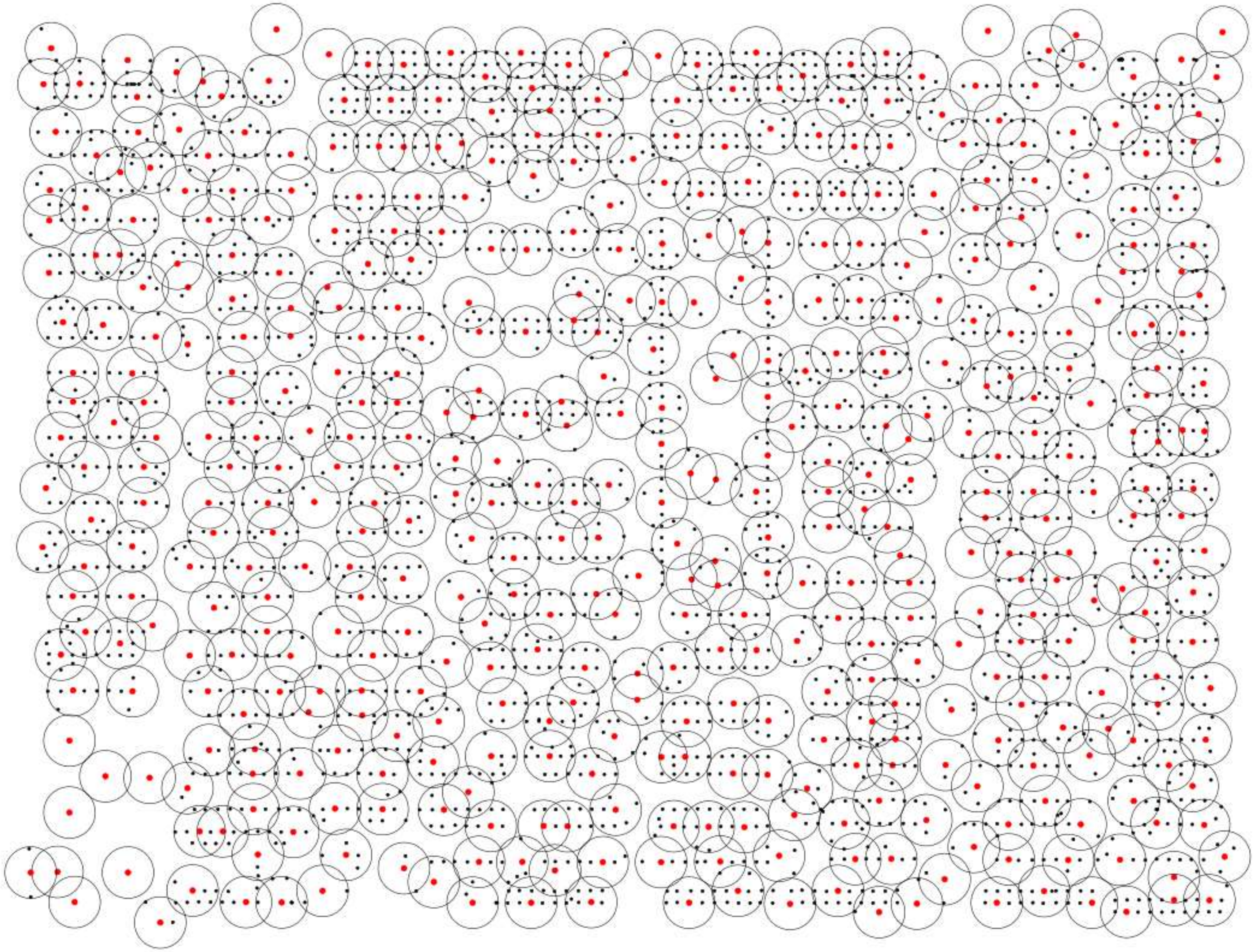
Algorithm 1 The main framework of the VWTS algorithm

Input: A graph G, a center number p, a covering radius r_q
Output: The best solution found so far X^*

A set of p centers $X \leftarrow \text{Init}(G, p, r_q)$ /* (Section 3.1) */
$X^* \leftarrow X$, $X' \leftarrow X$, tabu list $TL \leftarrow \varnothing$, $iter \leftarrow 1$
Vertex weights $w_i \leftarrow 1, \forall i \in V$ /* (Section 3.2) */
while termination condition is not met **do**
    $(i, j) \leftarrow \text{FindPair}(X', TL, iter)$ /* (Algorithm 2) */
    $\text{MakeMove}(i, j)$ /* (Algorithm 4) */
    if $|U(X)| < |U(X^*)|$ **then** /* $U(X)$ is the set of */
        $X^* \leftarrow X$ /* clients uncovered by X */
    else if $|U(X)| \geq |U(X')|$ **then**
        $w_v \leftarrow w_v + 1, \forall v \in U(X)$ /* (Section 3.2) */
    end if /* more uncovered clients than last solution */
    $TL \leftarrow \{i, j\}$ /* update tabu list (Section 3.4) */
    $X' \leftarrow X$, $iter \leftarrow iter + 1$
end while



pcb3038 p=50



pcb3038 p=500

Instances

5min内稳定求得可行解（所有节点全覆盖）

Instance	n	p	r_1	r_2	r_3
pcb3038p40	3038	40	336.42		
pcb3038p50	3038	50	298.10	298.04	297.83
pcb3038p100	3038	100	206.63	206.60	206.31
pcb3038p150	3038	150	164.77	164.55	164.40
pcb3038p200	3038	200	140.90	140.09	140.06

所有算例已根据当前半径删边处理为相应的集合覆盖算例

References

✓Qingyun Zhang , Zhipeng Lü, Zhouxing Su , Chunmin Li, Yuan Fang, Fuda Ma. (2020) Vertex weighting-based tabu search for p-center problem. Bessiere C, ed., Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, 1481-1487 (ijcai.org),

URL http://dx.doi.org/10.24963/ijcai.2020/206

图着色提交结果

【1】王宇轩1
夏媛1
张嘉洋1
王智远1
聂士锋
【2】张嘉洋2
【3】夏媛2
王宇轩2
谢汶泰1

【4】程丁丁
【5】鲁镇仪
陈泳帆1
【6】谢汶泰2
【7】陈泳帆2
【8】王智远2
何佳琪
王伟
【9】刘静蕾

提交建议

1. 按要求提交.

2. 按要求提交.

3. 按要求提交.

4. 提交的版本不要打印太多输出.

5. 提高程序的健壮性, 避免依赖算例中的冗余信息 (比如边数)

最终结果：按优度排名

Thanks!