



TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

SỬ DỤNG GENETIC ALGORITHM ĐỂ GIẢI BÀI TOÁN CLUSTERING

Người hướng dẫn:

PGS.TS. Huỳnh Thị Thanh Bình

ThS. Đỗ Tuấn Anh

Nhóm:

01

Mã học phần:

IT4906

Mã lớp:

141325

ONE LOVE. ONE FUTURE.

Thành viên trong nhóm

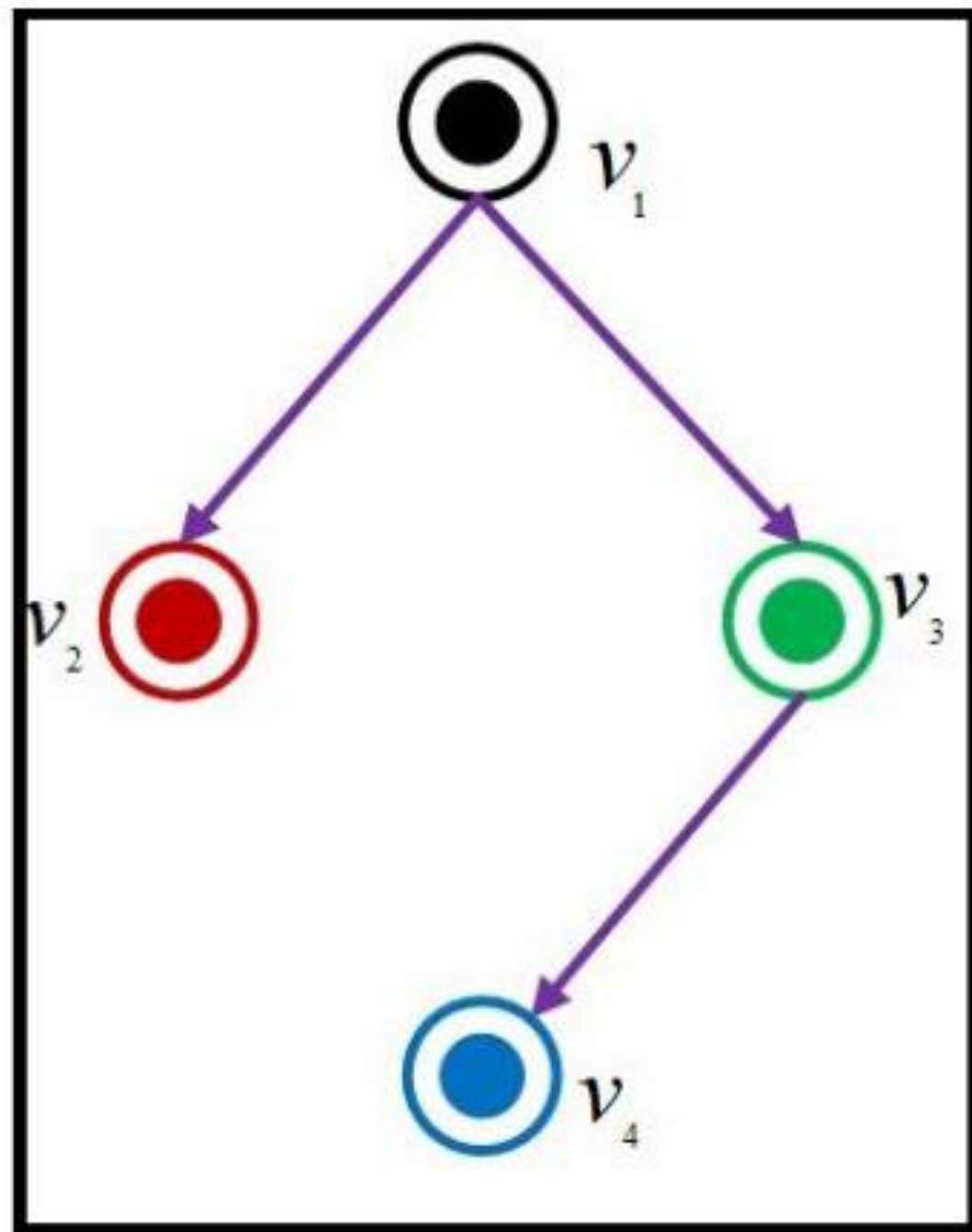
1. Nghiêm Ngọc Phong - 20204676

2. Nguyễn Quốc Nhật Minh - 20200408

3. Bùi Lâm Thanh - 20204606

CLUSTERINER

CHỦ ĐỀ



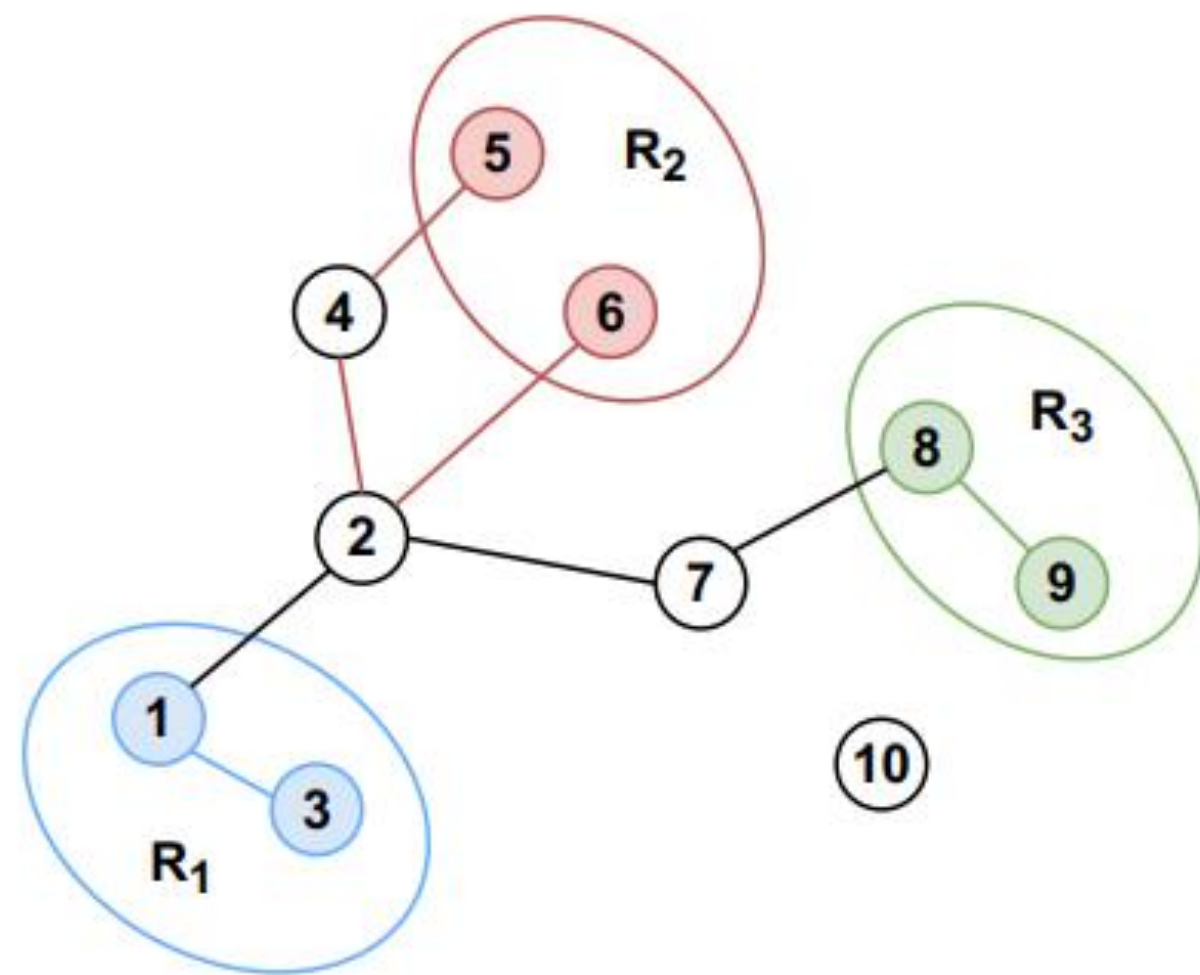
1. GIỚI THIỆU

2. CÁC NGHIÊN CỨU TRƯỚC ĐÂY

3. THUẬT TOÁN ĐỀ XUẤT

4. SO SÁNH CÁC THUẬT TOÁN

1. Giới thiệu



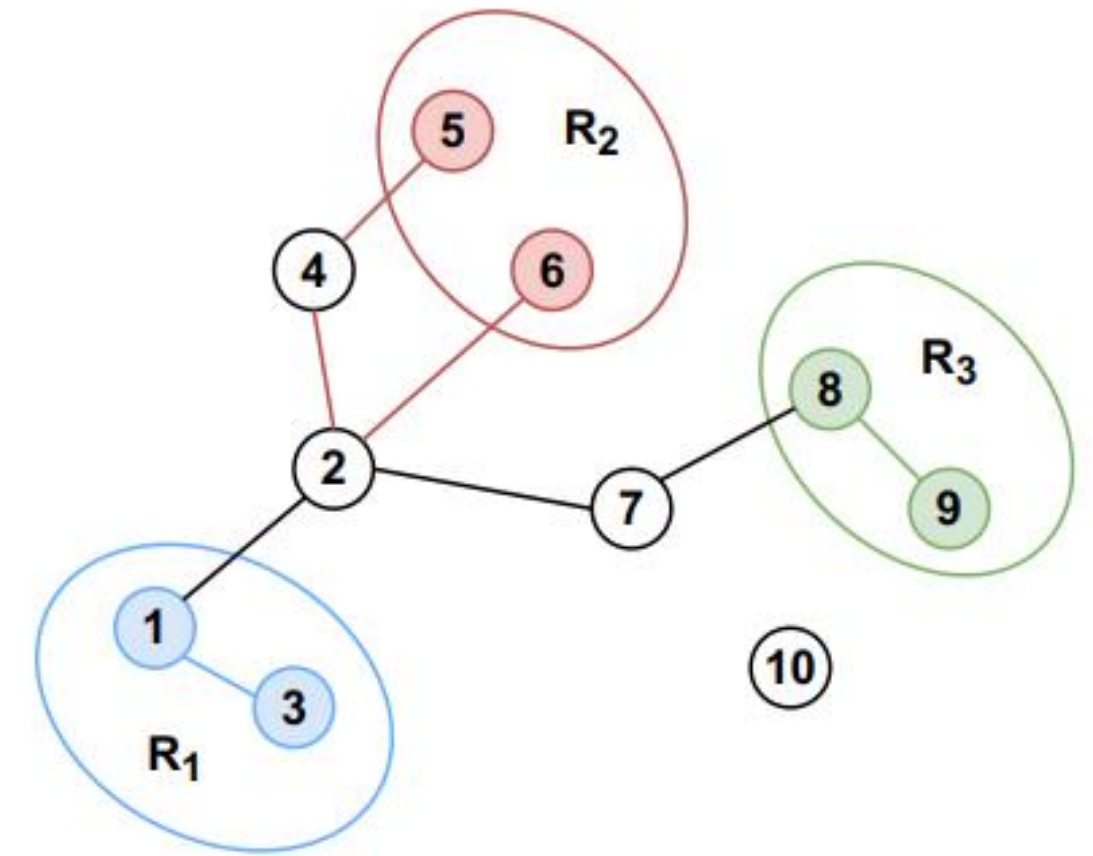
- **Steiner Tree Problem** được biết đến là một bài toán NP-khó trong lĩnh vực tối ưu hóa và đồ thị.
- Bài toán quan tâm đến tìm kiếm một cây có chi phí nhỏ nhất T - một đồ thị con không chu trình liên thông của một đồ thị $G = (V, E, w)$ sao cho kéo dài tất cả các đỉnh trong một tập hợp các đỉnh $R \subset V$.
- Kết quả đại diện cho truyền thông đa điểm trong mạng, rất hữu ích cho nhiều các ứng dụng vì nó có thể làm giảm chi phí duy trì nhiều kết nối 1 - 1.
- Nhiều biến thể của nó đã được phát triển và mở rộng nghiên cứu, **CluSteiner** là một trong số đó.

1. Giới thiệu

CluSteiner là một biến thể của bài toán cây Steiner cổ điển trong đồ thị.

Trong bài toán này, một đồ thị vô hướng có trọng số $G = (V, E, w)$ và các đỉnh required R .

Mục tiêu của bài toán là tìm một chu kỳ có trọng số tối thiểu đồ thị con liên thông, tức là một cây G bao gồm tất cả các đỉnh trong R . Các đỉnh không bắt buộc $(V \setminus R)$ được sử dụng làm điểm trung gian trong cây được gọi là các đỉnh Steiner. Một cây steiner T là một cây phân cụm nếu tất cả các cây local đôi một không có đỉnh chung nhau



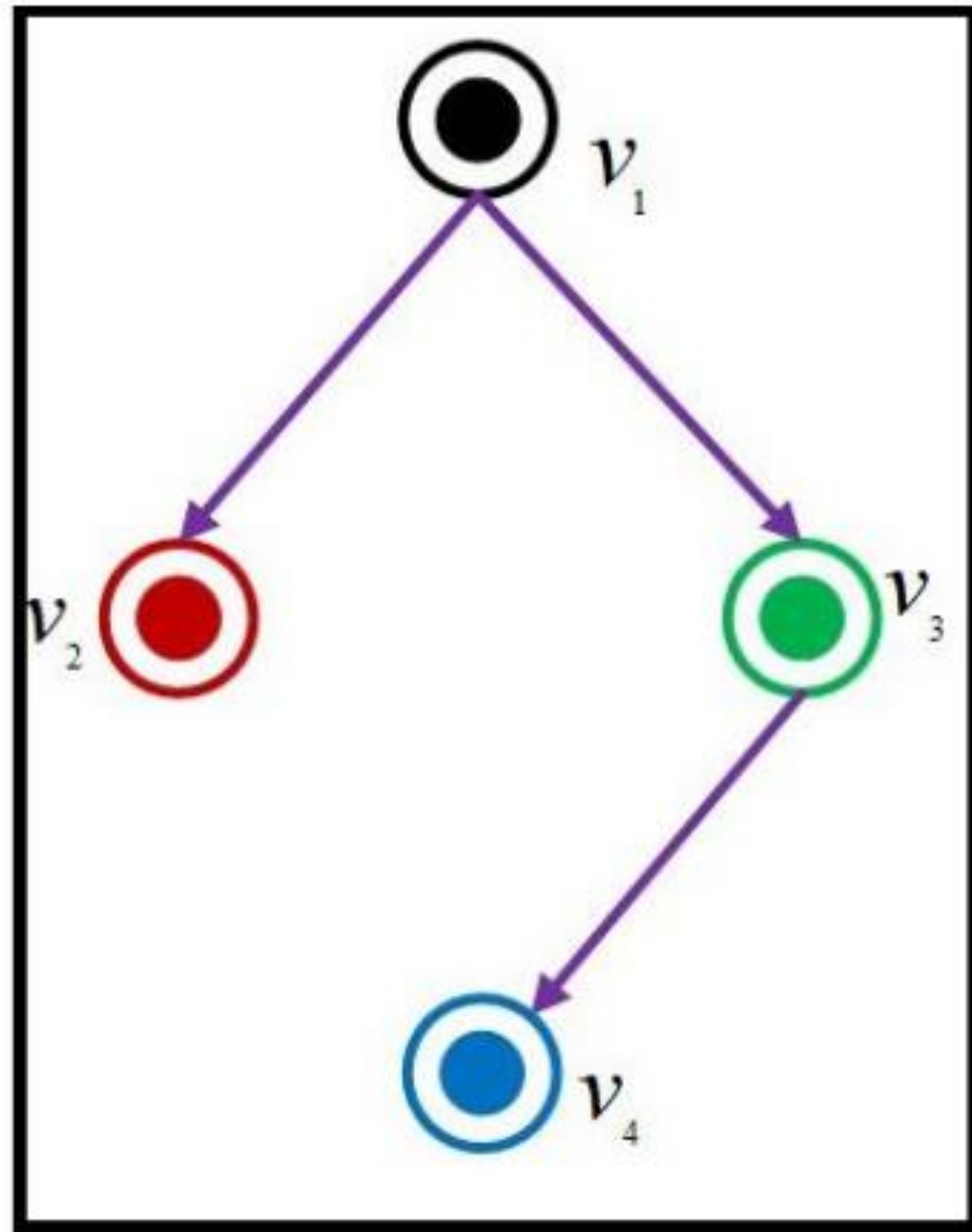
Một phương án chấp nhận được của bài toán

1 Giới thiệu

Input	<ul style="list-style-type: none">- A weighted undirected graph $G = (V; E; w)$- A set of required vertices $R \subset V$- A partition $V' = \{V_1, V_2, \dots, V_k\}$ of V. $R_i = R \cap V_i$
Output	A Steiner tree $T = (V_T, E_T)$
Objective	Minimize $\sum_{e \in E_T} w(e)$
Constraints	<ul style="list-style-type: none">- T is a Steiner tree: $R \subset V_T$- A subtree $T_i = (V_{T_i}, E_{T_i})$ is a Steiner tree of cluster V_i. $R_i \subset V_{T_i}$, $V_{T_i} \subset V_i$

CLUSTERINER

CHỦ ĐỀ



1. PHẦN GIỚI THIỆU

2. CÁC NGHIÊN CỨU TRƯỚC ĐÂY

3. THUẬT TOÁN ĐỀ XUẤT

4. SO SÁNH CÁC THUẬT TOÁN

2. Các nghiên cứu trước đây

(Genetic Algorithm based Approach to Solve the Clustered Steiner Tree Problem)

2.1. Ý tưởng của thuật toán

Ý tưởng chính của thuật toán đó là chia bài toán CluSteiner thành hai bài toán con:

- Tìm các local trees của k cụm đỉnh. (1)
- Tìm các cạnh để kết nối k cụm vào thành một cây. (2)

Đối với vấn đề (1), thuật toán **SPH** có thể được sử dụng cho tất cả các đồ thị con để tìm được local tree cho từng cụm. Trong các đỉnh tự do, mỗi đỉnh chỉ được thuộc trong một local tree duy nhất.

Vậy khi xây dựng cây cục bộ, đỉnh nào được đưa vào cây sẽ bị loại khỏi tập đỉnh tự do, có thể thấy thứ tự xây dựng cây cục bộ cho các cụm thay đổi sẽ gây ra thay đổi lớn đến kết quả cuối cùng. Vì thế, tác giả sử dụng GA cho việc chọn ra thứ tự xây dựng local tree cho các cụm được thể hiện bởi thuật toán **SPGA**.

2. Các nghiên cứu trước đây

2.1. Ý tưởng của thuật toán

Thuật toán SPH được sử dụng trong việc xây dựng cây cục bộ cho các cụm.

Input: $G = (V, E, w)$, $R \subset V$

Output: A Steiner tree $T = (V_T, E_T)$

- 1: $T \leftarrow \text{random } r \in R$
- 2: **while** $\exists r \in R, r \notin V_T$ **do**
- 3: $r \leftarrow$ the nearest vertex in R but not in
- 4: $T = T +$ the shortest path to r
- 5: **end while**
- 6: **return** T

SPH Algorithm

Thuật toán SPGA được sử dụng trong việc hoán thứ tự các cụm để tìm ra thứ tự xây dựng các cây cục bộ tốt nhất sao cho chi phí đạt được là nhỏ nhất.

Input: $G = (V, E, w)$, $R \subset V$, $R = \{R_1, R_2, \dots, R_k\}$

Output: A clustered Steiner tree $T = (V_T, E_T)$

- 1: $P_0 \leftarrow$ Initialize N individuals
- 2: **Evaluate**(P_0)
- 3: $T \leftarrow$ Best solution in P_0
- 4: $i \leftarrow 0$
- 5: **while** (Stopping conditions not satisfied) **do**
- 6: $C_i \leftarrow \text{Crossover}(P_i) + \text{Mutation}(P_i)$
- 7: **Evaluate**(C_i)
- 8: $P_{i+1} \leftarrow \text{Selection}(P_i + C_i)$
- 9: $T \leftarrow$ Best solution in P_{i+1}
- 10: $i \leftarrow i + 1$
- 11: **end while**
- 12: **return** T

SPGA Algorithm

2 Các nghiên cứu trước đây

2.1. Phương pháp mã hoá cho mỗi bộ NST

Trong thuật toán SGPA, mỗi một nhiễm sắc thể sẽ được mã hóa bằng một mảng k phần tử, đại diện cho độ ưu tiên của các cụm. Cụm có độ ưu tiên cao hơn sẽ được ưu tiên xây dựng cây cục bộ trước

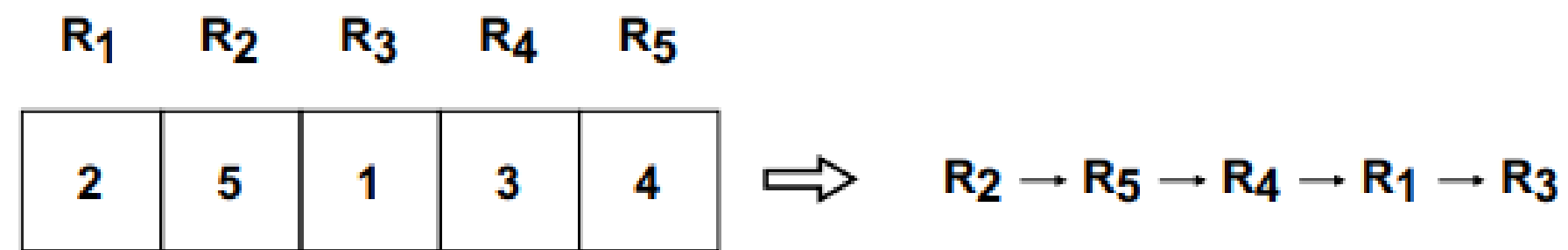


Fig. 2: An example chromosome representation

Theo như hình ảnh **Fig. 2**, ta có độ ưu tiên của cụm $R_2 = 5$, $R_5 = 4$, $R_4 = 3$, $R_1 = 2$, $R_3 = 1$, theo đó, thứ tự xây dựng cây cục bộ sẽ là $R_2 - R_5 - R_4 - R_1 - R_3$.
Quần thể đầu tiên N_0 sẽ được tạo ra bởi N **hoán vị ngẫu nhiên**, đại diện cho N cá thể của quần thể.

2 Các nghiên cứu trước đây

2.2. Đánh giá nhiệm sắc thể

Để đánh giá một NST, ta cần xây dựng cây Steiner và tính toán chi phí của nó. Quy trình gồm 2 bước đánh giá tương ứng với 2 bài toán con.

- **[1]** Sử dụng thuật toán SPH để xây dựng cây cục bộ theo thứ tự ưu tiên. Chi phí của cây là tổng trọng số tất cả các cạnh của cây.
- **[2]** Thu nhỏ các cụm thành các đỉnh đại diện, sau đó lại sử dụng thuật toán SPH để liên kết các đỉnh đại diện cho các cụm đó.

Gọi $A(T)$ là tổng trọng số của tất cả các cây cục bộ của các cụm, $B(T)$ là tổng chi phí của các cạnh nối các cụm, thì chi phí của cây Steiner T được xác định bằng tổng 2 chi phí trên:

$$c(T) = \alpha(T) + \beta(T)$$

2 Các nghiên cứu trước đây

2.3. Xây dựng cây cục bộ

Dựa vào thứ tự dựng cây cục bộ theo NST, ta loại bỏ các cạnh không cần thiết khỏi đồ thị G để tạo đồ thị con $G_i = (V_i, E_i)$.

- $V_i = R_i \cup F_i$ với F_i là tập các đỉnh tự do (chưa được thêm vào cây cục bộ nào). $F = V \setminus R$, là tập hợp tất cả các đỉnh tự do. Khi đó, ta có:

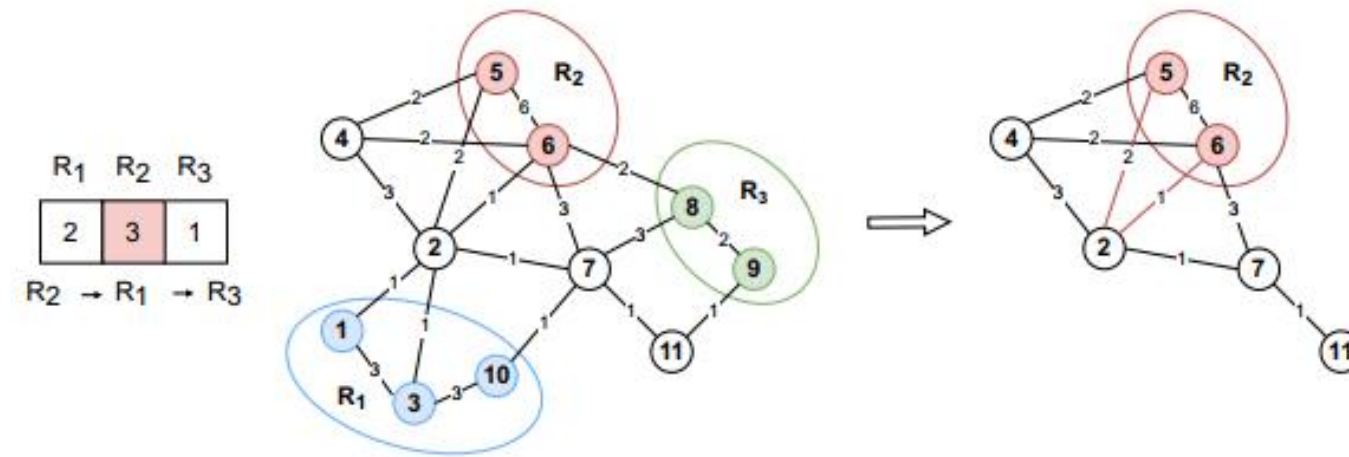
$$F_i = F \setminus (F \cap (V_{T_1} \cup V_{T_2} \cup \dots \cup V_{T_{i-1}}))$$

- $E_i \subset E$ là tập hợp các cạnh có 1 trong 2 điểm nằm trong V_i :

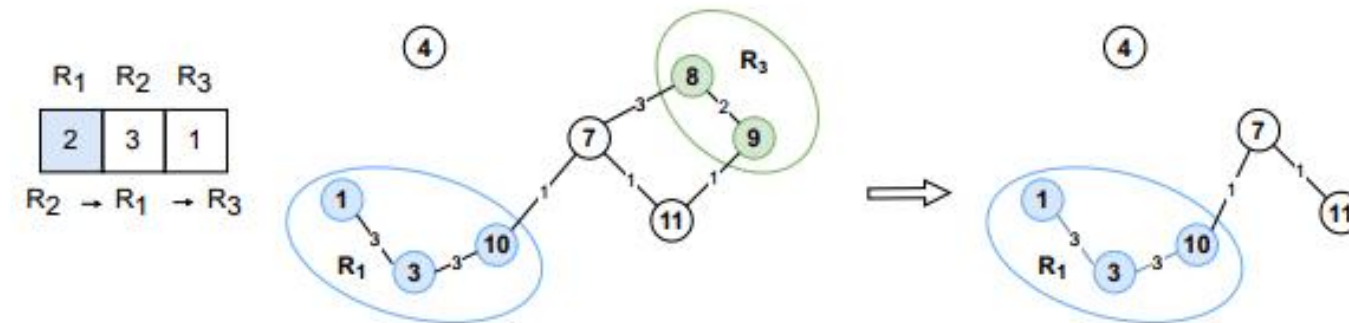
$$E_i = \{(u, v) \in E \mid u \in V_i, v \in V_i\}$$

2 Các nghiên cứu trước đây

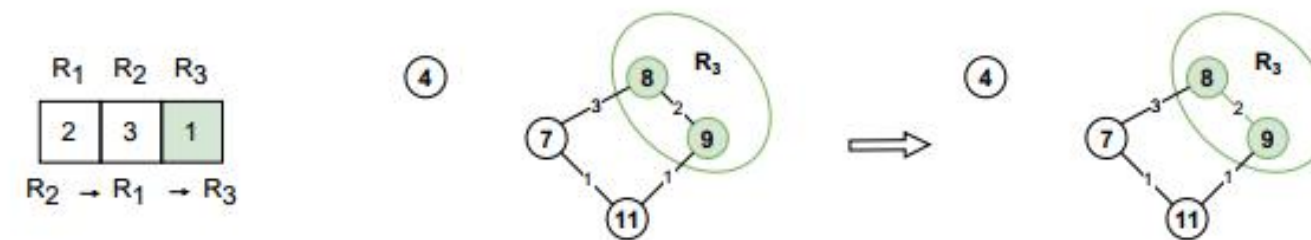
2.3. Xây dựng cây cục bộ



(a) Constructing G_1 of cluster 2



(b) Constructing G_2 of cluster 1



(c) Constructing G_3 of cluster 3

Fig. 3: Construction of local trees example

2 Các nghiên cứu trước đây

2.4. Xây dựng cạnh nối giữa các cụm

Ta có một đồ thị $G = (V, E, w)$. "Contraction" của một cạnh (u, v) có nghĩa là thay thế u, v bằng một đỉnh mới s , với s được coi như là một **đỉnh thu gọn**. Bất kì một đỉnh t nào khác có connect với u hoặc v sẽ được thay thế thành cạnh mới (s, t) và sẽ có trọng số **$w(s, t) = \min(w(u, t), w(v, t))$** . Sau khi thu gọn, ta có thể tìm cây khung nhỏ nhất cho đồ thị mới bằng chính thuật toán SPH.

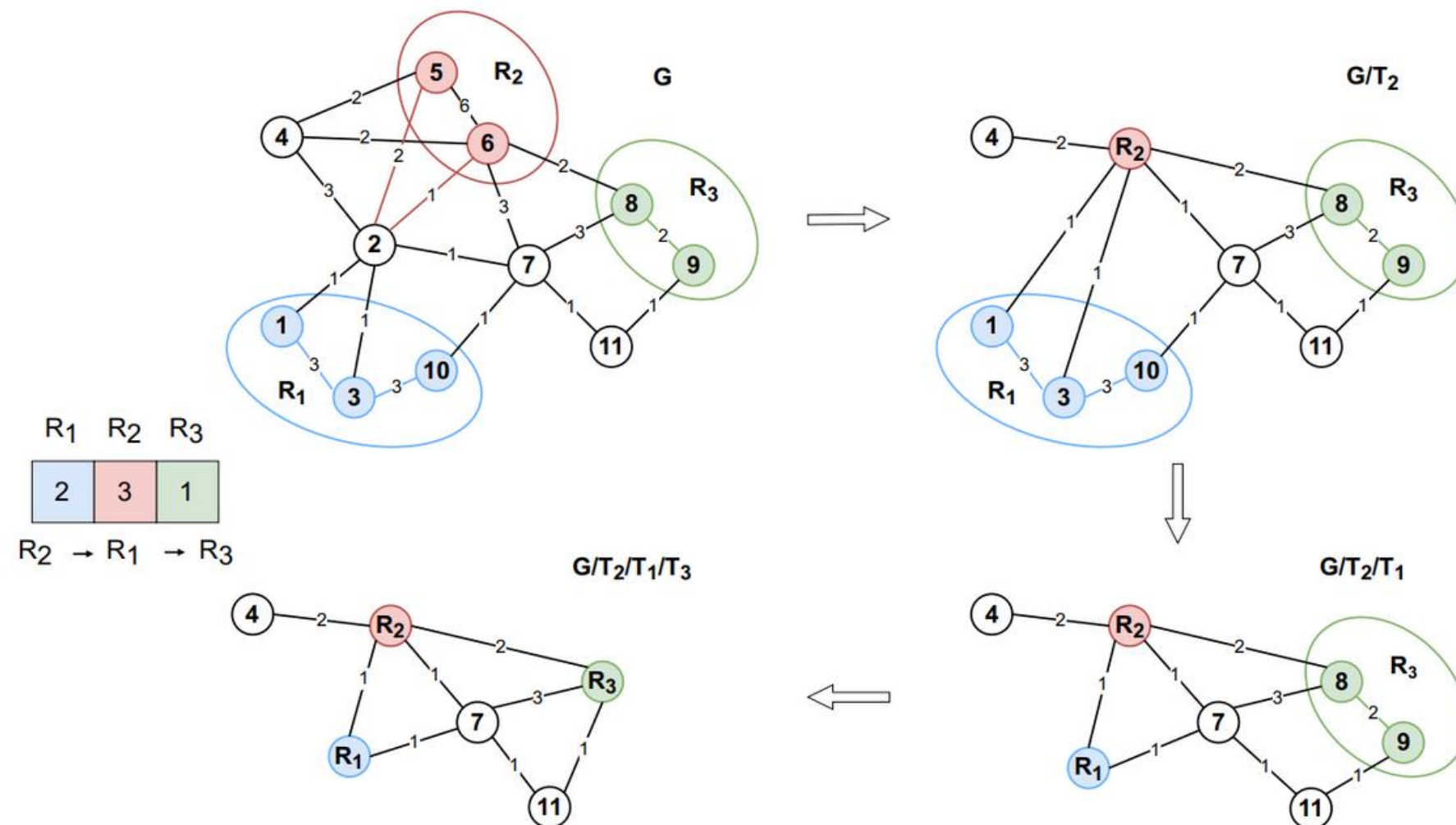


Fig. 4: Contraction of graph G

2 Các nghiên cứu trước đây

2.5. Xây dựng cạnh nối giữa các cụm

Thuật toán SPH đã được trình bày ở phần trước nhưng nó có một độ phức tạp về thời gian khá lớn. Điều này phần chính là do sử dụng nhiều lần Dijkstra để tìm đỉnh gần nhất với cây local đang được xây dựng.

Vì vậy, thuật toán SPH ở được tinh chỉnh chỉ sử dụng Dijkstra một lần nhưng vẫn có thể đem lại kết quả tương tự. Thay vì liên tục sử dụng Dijkstra để tìm đường đi ngắn nhất đến required vertex, các đỉnh trên đường đi đến required vertex được add ngược lại vào unvisited set với khoảng cách dự kiến được gán lại bằng 0.

Input: $G = (V, E, w)$, $R \subset V$

Output: A Steiner tree $T = (V_T, E_T)$ spanning R

```
1:  $V_T \leftarrow \emptyset, E_T \leftarrow \emptyset$ 
2:  $S \leftarrow V$ 
3:  $S_R \leftarrow R$ 
4:  $\forall v \in V : d(v) \leftarrow \infty$ 
5:  $r \leftarrow \text{random in } R$ 
6:  $d(r) \leftarrow 0$ 
7:  $S \leftarrow S \setminus \{r\}$ 
8: while  $S_R$  is not empty do
9:    $u \leftarrow \operatorname{argmin}_{u \in S} d(u)$ 
10:   $S \leftarrow S \setminus \{u\}$ 
11:  if  $u \in S_R$  then
12:     $S_R \leftarrow S_R \setminus u$ 
13:    while  $u \notin V_T$  do
14:       $V_T \leftarrow V_T \cup u, E_T \leftarrow E_T \cup (p(u), u)$ 
15:       $d(u) \leftarrow 0, S \leftarrow S \cup u$ 
16:       $u \leftarrow p(u)$ 
17:    end while
18:  end if
19:  for  $\forall (u, v) \in E$  do
20:    if  $d(u) + w(u, v) < d(v)$  then
21:       $d(v) \leftarrow d(u) + w(u, v)$ 
22:       $p(v) \leftarrow u$ 
23:    end if
24:  end for
25: end while
26: return  $T$ 
```

2 Các nghiên cứu trước đây

2.6. Mã hoá nhiễm sắc thể

2.6.1. Toán tử lai ghép

Các bước thực hiện toán tử lai ghép trong thuật toán SPGA để tạo ra 2 NST con từ 2 NST cha mẹ.

- Tạo 2 điểm lai ghép ngẫu nhiên
- Sao chép đoạn gen giữa hai điểm lai ghép của cha mẹ vào 2 con.
- Từ điểm lai ghép thứ 2, sao chép các phần tử phần còn lại của cha mẹ mà không xuất hiện ở trong con vào 2 nhiễm sắc thể con.

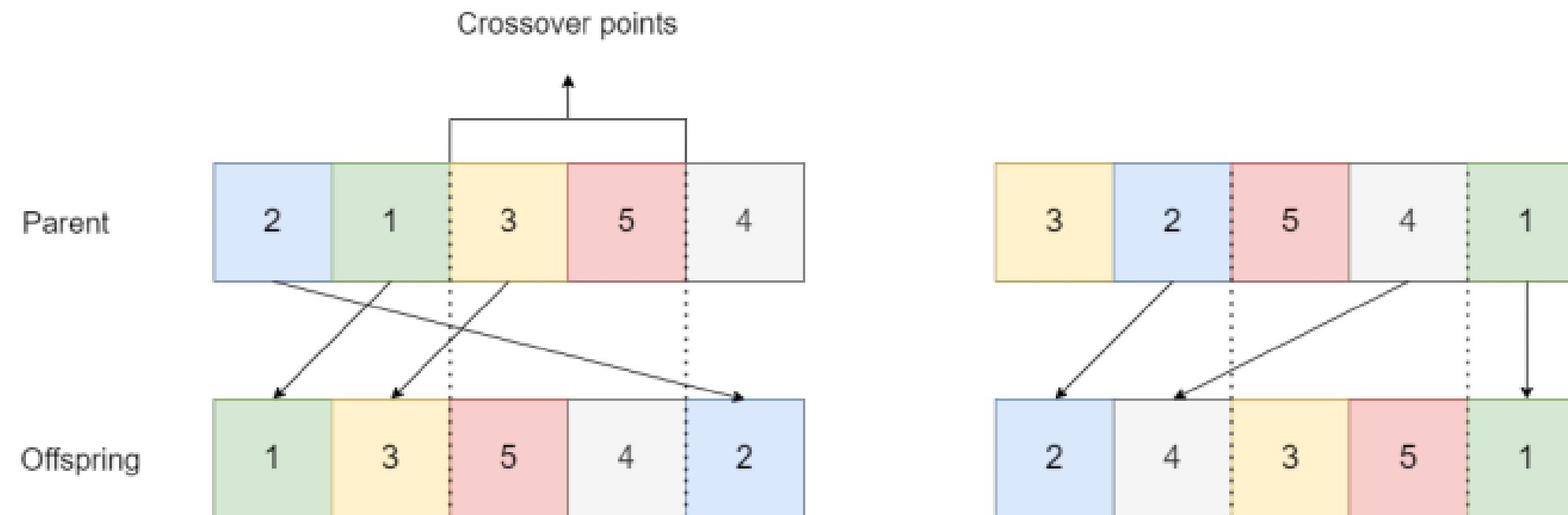


Fig. 5: An example Order Crossover (OX1)

2 Các nghiên cứu trước đây

2.6. Mã hoá nhiễm sắc thể

2.6.2. Toán tử lai ghép

Trong thuật toán SGPA, toán tử đột biến là "Swapping Mutation". Điều này đơn giản chỉ là hoán đổi vị trí của 2 gen ngẫu nhiên trong nhiễm sắc thể. Dưới đây là hình ảnh minh họa cho toán tử đột biến.

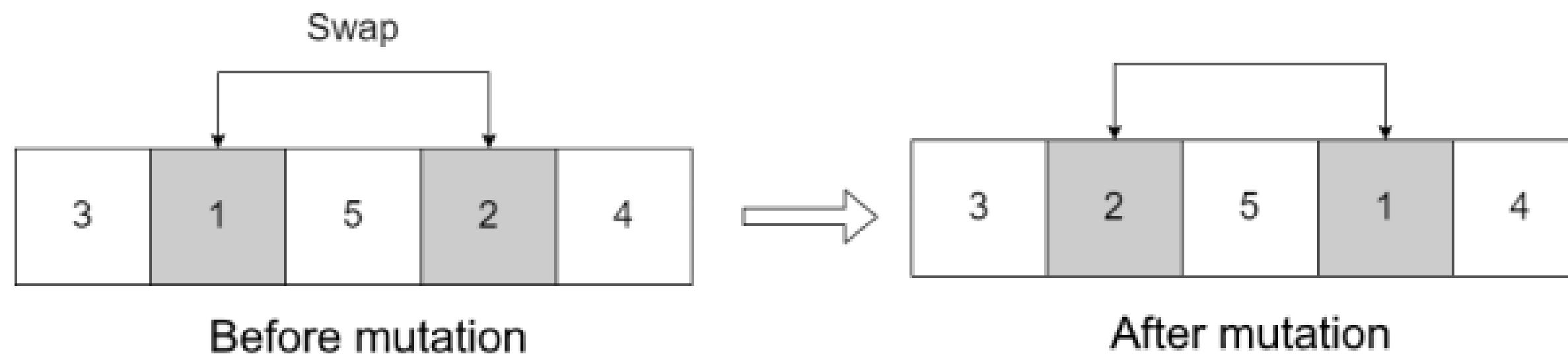


Fig. 6: Swap Mutation

2 Các nghiên cứu trước đây

2.7. Độ phức tạp thời gian tính toán

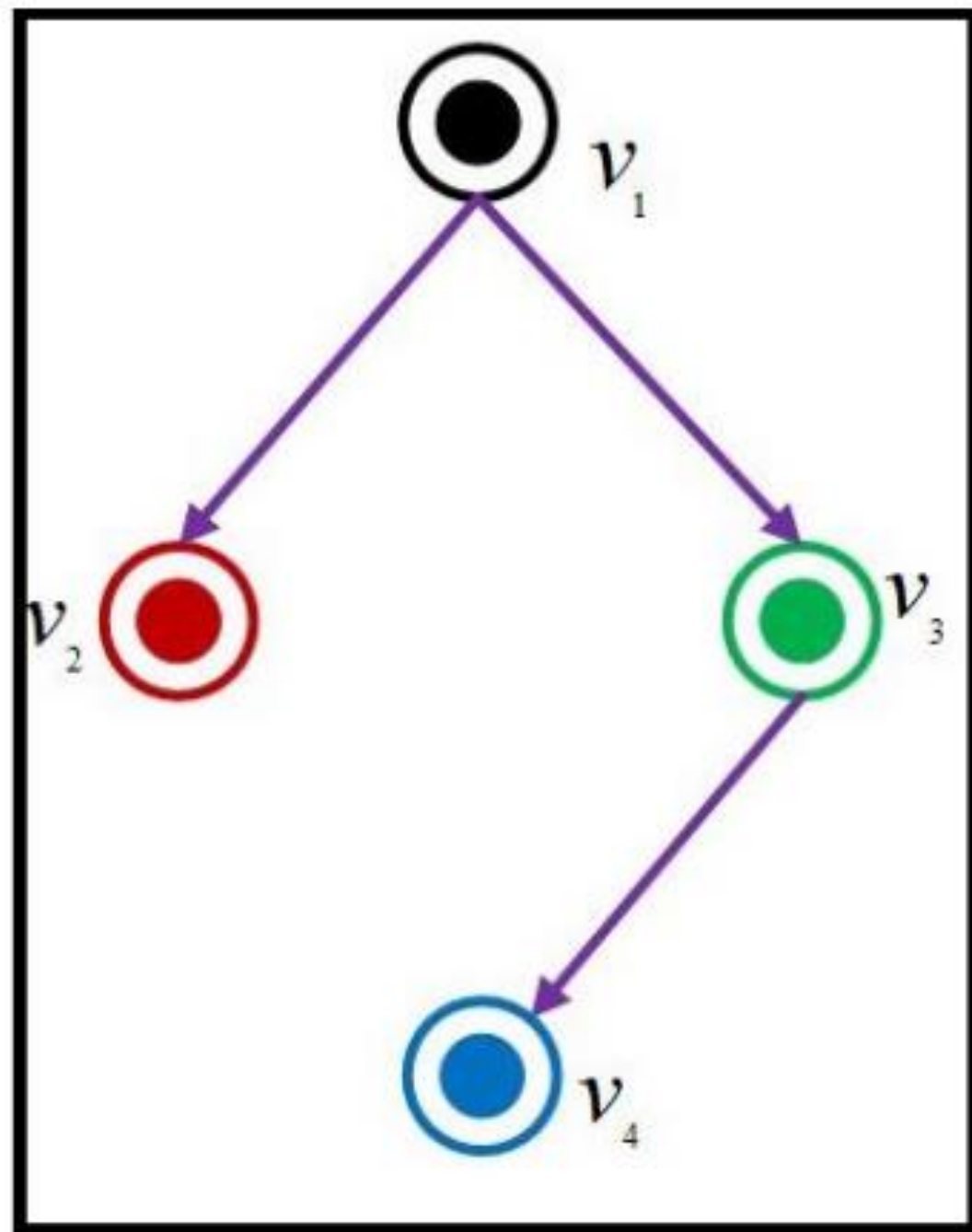
$$O(SPGA) = O(evaluation) = MAX_EVALUATIONS \times k \times (n \times \log_2 n + m)$$

Trong đó:

- + k là số cụm
- + MAX_EVALUATIONS = Số lượng đánh giá lớn nhất
- + n là số lượng đỉnh của đồ thị
- + m là số lượng cạnh

CLUSTERINER

CHỦ ĐỀ



1. PHẦN GIỚI THIỆU

2. CÁC NGHIÊN CỨU TRƯỚC ĐÂY

3. THUẬT TOÁN ĐỀ XUẤT

4. SO SÁNH CÁC THUẬT TOÁN

3. Thuật toán đề xuất

3.1 Nhược điểm của thuật toán SPH

Thuật toán SPH có một nhược điểm nhỏ là đối với các đỉnh required chưa được thêm vào cây local đang xây dựng, mà có **khoảng cách dự kiến** đến cây local đó bằng nhau, thuật toán sẽ không thể xác định chính xác được đỉnh nào sẽ được thêm vào cây con trước để cho xây được cây con tối ưu.

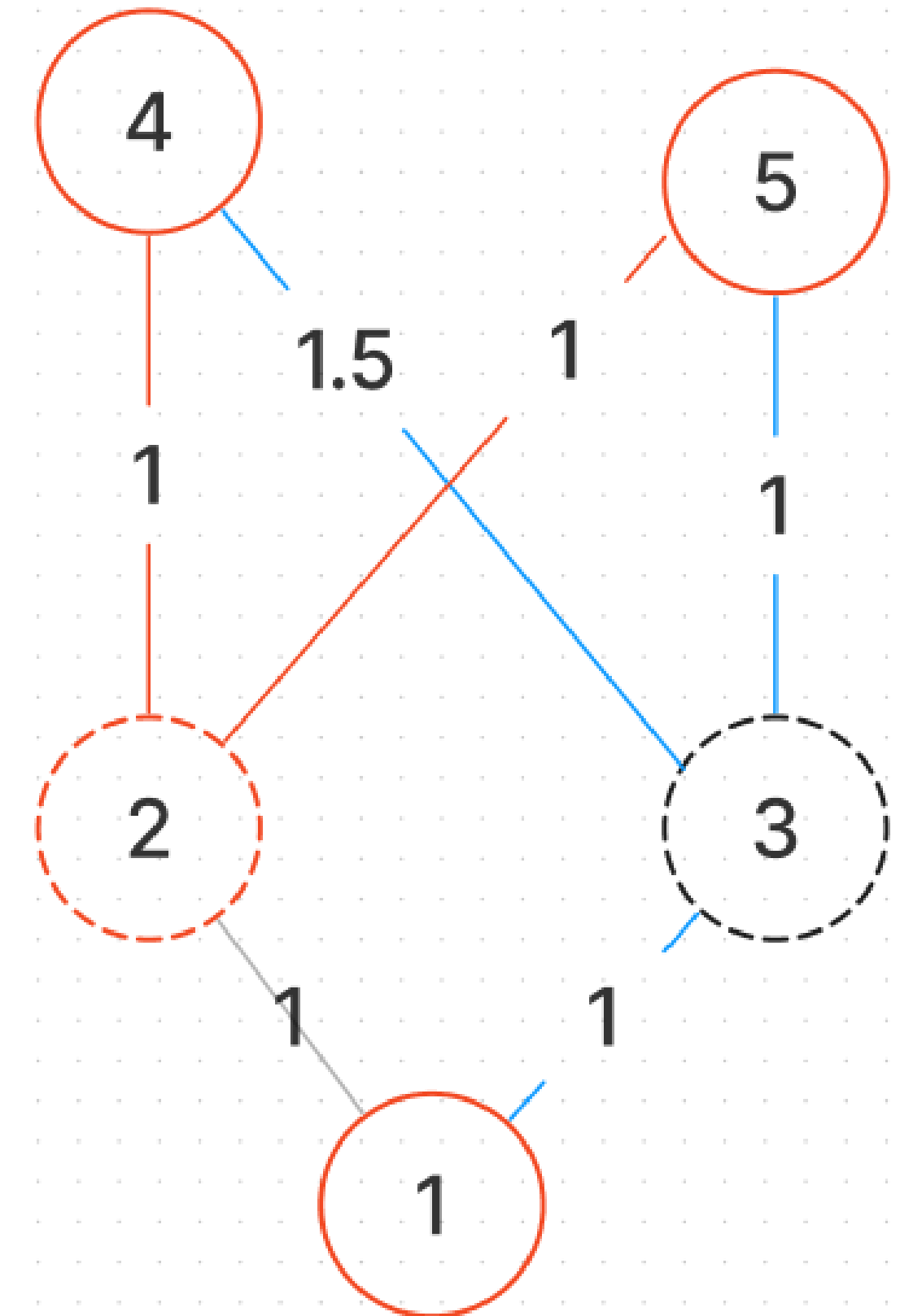
Với mã hoá hoán vị, quá trình xây dựng các cây local tương ứng với các cụm sẽ bị giới hạn số lượng các đỉnh non_required do một số đỉnh đã được thêm vào các cây local của cụm trước đó. Điều này thông thường giúp cho quá trình xây dựng các cây local có thể thoát khỏi trường hợp các đỉnh required của cụm nhận được giá trị khoảng cách dự kiến bằng nhau trong khi thực hiện thuật toán. Tuy nhiên, với cây local có thứ tự được xây dựng đầu tiên, điều này lại không xảy ra.

3. Thuật toán đề xuất

3.1 Nhược điểm của thuật toán SPH

Ví dụ: Cho đồ thị với 3 đỉnh required là 1, 4, 5. Cây local optimal của đồ thị trên được bôi màu đỏ. Gọi $d[i]$ là khoảng cách dự kiến của đỉnh i đến cây local hiện tại.

Khi thuật toán tính ra $d[4]=d[5]=2$, đỉnh 5 được chọn ra để đưa vào cây local trước thay vì đỉnh 4, đỉnh 3 sẽ được gán $d[3] = 0$. Do đó, đỉnh 4 được tính lại $d[4] = 1.5$. Cây local mà thuật toán tìm được sẽ là cây gồm các cạnh được bôi màu xanh như hình vẽ.



3. Thuật toán đề xuất

3.2 Phương pháp mã hoá đề xuất

Nhóm đề xuất phương pháp mã hoá các đỉnh tự do (non-required) bằng phương pháp mã hoá số nguyên thay vì mã hoá hoán vị thứ tự ưu tiên tính toán cho cây con các cụm.

Mỗi phần tử trong một nhiệm sắc thể được ánh xạ tới một đỉnh non_required trong đồ thị, và nhận các giá trị từ 0 đến $k-1$ (k là số cụm bài toán cho).

Với giá trị i mà một đỉnh non_required x nhận được, thuật toán sẽ lấy các đỉnh non_required x để thực hiện tính cây local của cụm thứ i (điều này có nghĩa là một số đỉnh non_required x có thể không được chọn vào cây local).

Việc giới hạn số lượng đỉnh tự do để tính toán cho từng cụm giúp giảm khả năng thuật toán SPH rơi vào trường hợp như đã đề cập ở trên. Tuy nhiên, cách mã hoá này làm tăng không gian tìm kiếm lên n^k khiến cho việc tìm ra nghiệm tối ưu trở nên khó khăn hơn.

1	2	5	4	0	2	1	2	0	4
---	---	---	---	---	---	---	---	---	---

Giả sử $k=6$, khi đó các phần tử sẽ nhận giá trị từ 0 đến 5 tương ứng với số thứ tự cụm

3. Thuật toán đề xuất

3.2 Phương pháp mã hoá đề xuất

3.2.1. Toán tử lai ghép

- Tạo 2 điểm lai ghép ngẫu nhiên
- Sao chép đoạn gen giữa hai điểm lai ghép của cha mẹ vào 2 nhiễm sắc thể con.
- Các vị trí gen còn lại của cha mẹ (p_1, p_2) , thực hiện sinh gen vào vị trí trống còn lại trong 2 nhiễm sắc con (c_1, c_2) như sau:
 - Lấy ngẫu nhiên 2 số nguyên $e_1, e_2 \in [0, k - 1]$
 - Sinh gen c_1, c_2 theo công thức

$$\begin{cases} c_1 = (p_1 + p_2 + e_1) \bmod k \\ c_2 = (p_1 - p_2 + e_2) \bmod k \end{cases}$$

3. Thuật toán đề xuất

3.2.1. Toán tử lai ghép

Ví dụ: $n=5$, $k = 3$, $e1 = 1$, $e2 = 2$

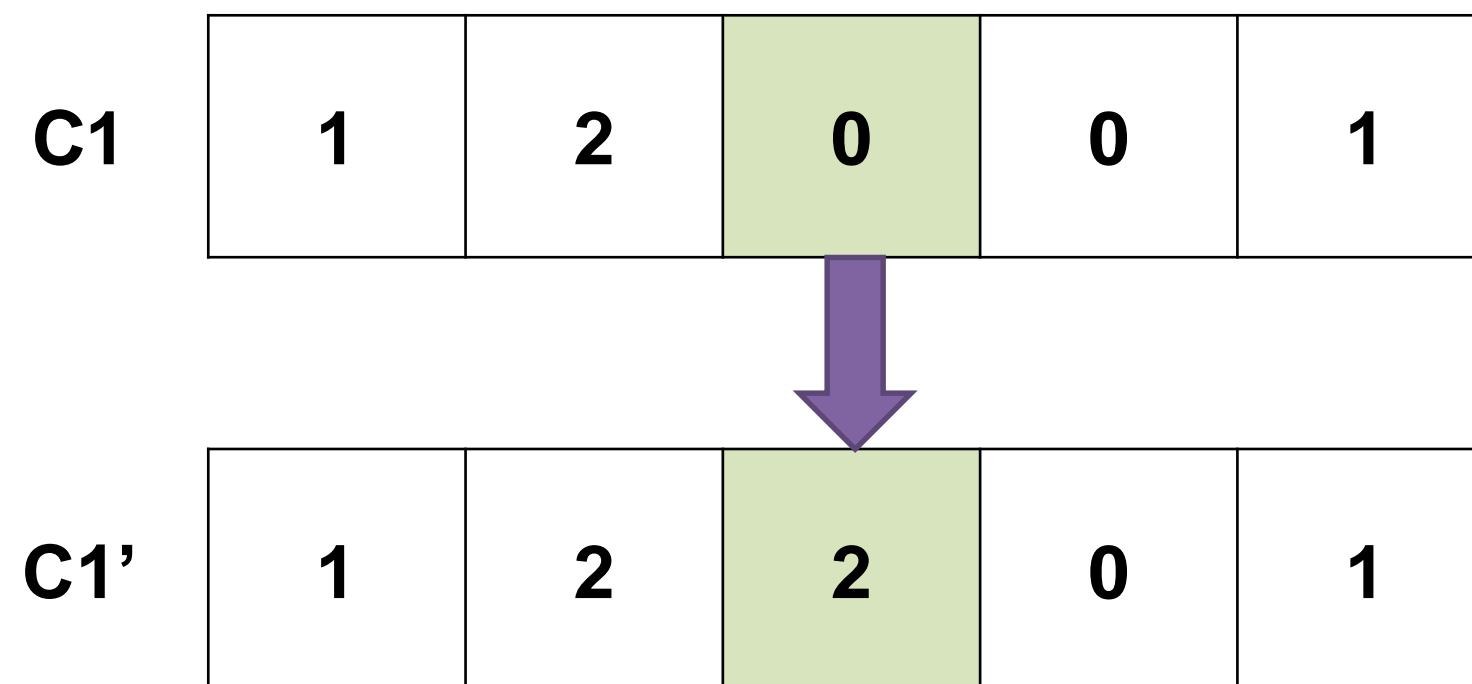
P1	1	1	0	0	2
	2	0	0	2	1
C1	1	2	0	0	1
	1	0	0	2	0

3. Thuật toán đề xuất

3.2 Phương pháp mã hoá đề xuất

3.2.2. Toán tử đột biến

- Chọn ngẫu nhiên 1 phần tử và thay đổi thành 1 giá trị bất kì trong khoảng $[0, k-1]$



3. Thuật toán đề xuất

3.3 Độ phức tạp

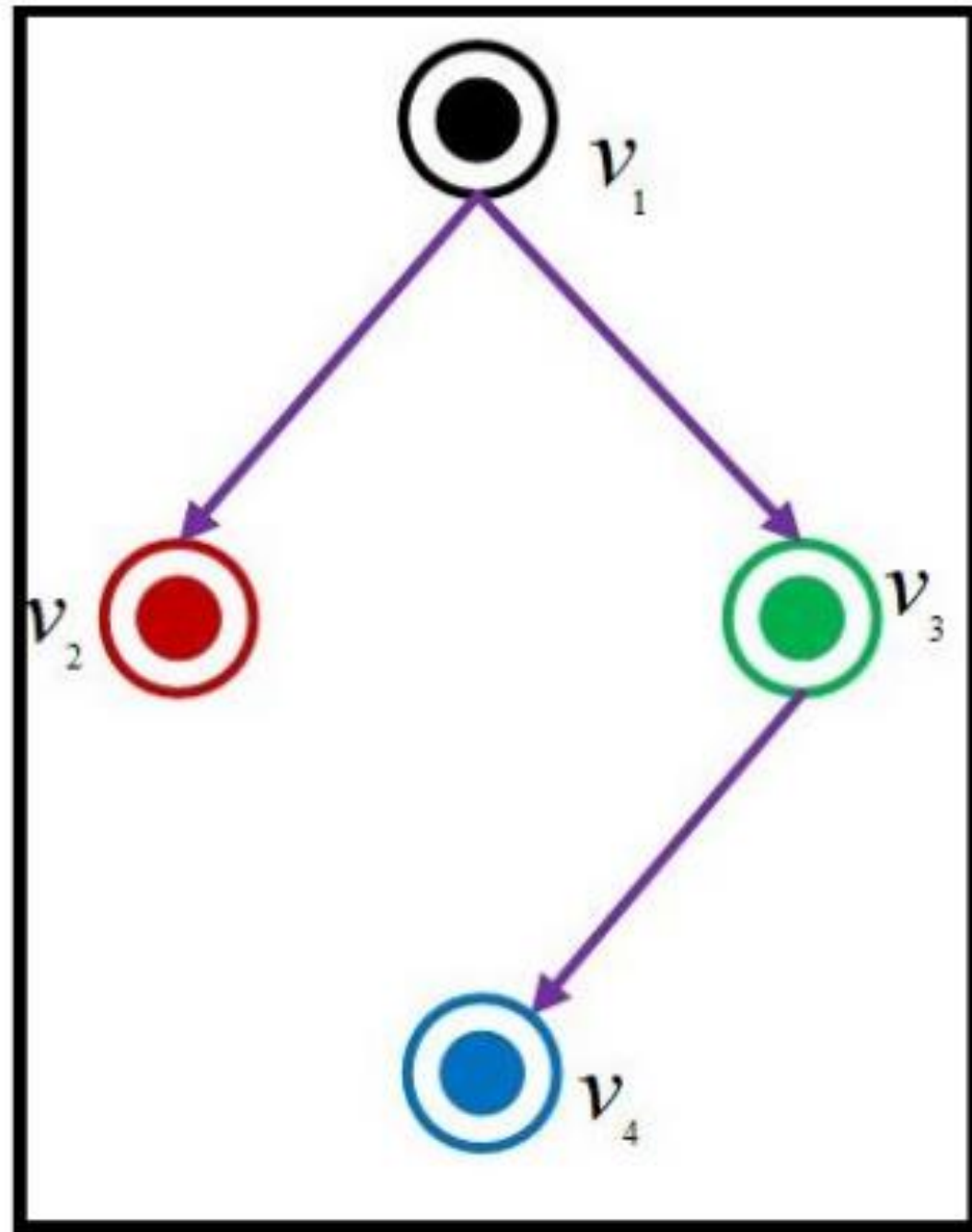
Sự khác biệt giữa thuật toán trước đây và thuật toán đề xuất là phương pháp mã hoá, do đó, độ phức tạp của thuật toán không có quá nhiều sự khác biệt:

$$O(SPGA) = O(evaluation) = MAX_EVALUATION \times (n \log_2 n + m)$$

Mặc dù trong mỗi evaluation, thuật toán SPH được sử dụng k lần để tìm các local tree và thêm 1 lần để liên kết các cụm, độ phức tạp của k lần chạy SPH là $n \log_2 n + m$ do mỗi đỉnh chỉ được đưa vào SPH để tính toán đúng 1 lần.

CLUSTERINER

CHỦ ĐỀ



- PHẦN GIỚI THIỆU
- 2. CÁC NGHIÊN CỨU TRƯỚC ĐÂY
- 3. THUẬT TOÁN ĐỀ XUẤT
- 4. SO SÁNH CÁC THUẬT TOÁN**

4. So sánh các thuật toán

4.1 Bộ dữ liệu

- Pham Dinh, T.: Clutsp instance, Mendeley data v3 (2019):
<https://data.mendeley.com/datasets/b4gcgybvt6/3>
- Các bộ dữ liệu được dùng trong thực nghiệm được lấy một phần trong tập Non-Eulidean Type1_Small:
 - Số lượng: 15 tập.
 - Mỗi bộ dữ liệu bao gồm thông tin về số đỉnh, số cụm, ma trận khoảng cách giữa các điểm và các đỉnh trong mỗi cụm.
 - Các bộ dữ liệu không có thông tin về kết quả tối ưu.

4. So sánh các thuật toán

4.2 So sánh kết quả trên các bộ dữ liệu

- Phương pháp lựa chọn cha mẹ theo thứ hạng: Chọn ra tập hợp gồm $n \times 0.1$ số lượng cha mẹ ngẫu nhiên trong quần thể hiện tại rồi lấy 2 cha mẹ tốt nhất trong tập hợp.
- Phương pháp đấu tranh sinh tồn: Giữ lại các cá thể ưu tú.
- Các tham số được dùng trong thuật toán SPGA ở cả 2 thuật toán:

Parameters	Definition	Value
<i>POP SIZE</i>	Number of individuals in population	100
<i>MAX GENERATIONS</i>	Maximum number of GA generations	50
<i>MAX EVALUATIONS</i>	Maximum number of evaluations	5000
<i>Pc</i>	Crossover probability	1.0
<i>Pm</i>	Mutation probability	0.2
Seeds	Number of runs	30

4. So sánh các thuật toán

4.2 So sánh kết quả trên các bộ dữ liệu

	Original algorithm			Proposal algorithm		
Dataset	avg	best	time(s)	avg	best	time(s)
75lin105.txt	668	668	8551.413	926.2	835	1568.062
5st70.txt	409	409	1922.643	543.367	452	893.928
5pr76.txt	664	664	2702.053	775.433	684	1357.634
5eil76.txt	475	475	2511.95	499.733	473	1185.566
5eil51.txt	691	691	1261.639	757.633	690	654.71
5berlin52.txt	920	920	1804.733	932.6	920	499.823
50rat99.txt	746	746	8659.092	1457.433	1110	1398.26
50lin105.txt	976	976	10727.86	1836.267	1489	2057.471
10st70.txt	772	772	2760.239	905.533	805	692.648
10rat99.txt	897	897	5306.593	1476.267	1239	1804.591
10pr76.txt	795	795	3452.729	841.567	741	992.709
10kroB100.txt	626	626	4884.982	1109.567	841	1671.503
10eil76.txt	806	806	3392.782	910	818	947.837
10eil51.txt	887	887	1838.096	901.167	887	567.332
10berlin52.txt	567	567	1629.63	635	567	388.814

4. So sánh các thuật toán

4.2 So sánh kết quả trên các bộ dữ liệu

- Từ kết quả trên, ta có thể thấy trong phần lớn trường hợp, thuật toán trong bài báo cho ra kết quả ổn định và tốt hơn rất nhiều so với thuật toán đề xuất. Điều này có thể dễ dàng giải thích như trên khi số lượng cụm của các bộ dữ liệu là rất nhỏ, việc mã hoá với số cụm có thể thu được không gian tìm kiếm không quá lớn giúp cho việc tìm ra kết quả tối ưu nhanh hơn.
- Mặc dù với tham số về số lượng quần thể, thể hệ được sử dụng là như nhau, thuật toán trong bài báo lại có thời gian chạy gấp nhiều lần so với thuật toán đề xuất. Điều này là dễ hiểu do độ phức tạp của thuật toán nhóm đề xuất nhỏ hơn k lần so với thuật toán của bài báo.
- Ngoài ra, trong một vài trường hợp đặc biệt, thuật toán của nhóm đề xuất có kết quả best tốt hơn do phương pháp của nhóm có thể khắc phục một vài nhược điểm của SPH như đã đề xuất ở phần trước.

4.2 So sánh kết quả trên các bộ dữ liệu

Kết quả thống kê sử dụng SPSS

Average statistic

Wilcoxon Signed Ranks Test

		Ranks		
		N	Mean Rank	Sum of Ranks
avg2 - avg1	Negative Ranks	0 ^a	.00	.00
	Positive Ranks	15 ^b	8.00	120.00
	Ties	0 ^c		
	Total	15		

- a. avg2 < avg1
- b. avg2 > avg1
- c. avg2 = avg1

Test Statistics^a

	avg2 - avg1
Z	-3.408 ^b
Asymp. Sig. (2-tailed)	.001

- a. Wilcoxon Signed Ranks Test
- b. Based on negative ranks.

Best statistic

Wilcoxon Signed Ranks Test

		Ranks		
		N	Mean Rank	Sum of Ranks
best2 - best1	Negative Ranks	3 ^a	3.33	10.00
	Positive Ranks	9 ^b	7.56	68.00
	Ties	3 ^c		
	Total	15		

- a. best2 < best1
- b. best2 > best1
- c. best2 = best1

Test Statistics^a

	best2 - best1
Z	-2.275 ^b
Asymp. Sig. (2-tailed)	.023

- a. Wilcoxon Signed Ranks Test
- b. Based on negative ranks.

4.2 So sánh kết quả trên các bộ dữ liệu

Kết quả thống kê sử dụng SPSS

Average statistic

Sign Test

Frequencies		N
avg2 - avg1	Negative Differences ^a	0
	Positive Differences ^b	15
	Ties ^c	0
	Total	15

- a. avg2 < avg1
- b. avg2 > avg1
- c. avg2 = avg1

Test Statistics ^a	
	avg2 - avg1
Exact Sig. (2-tailed)	.000 ^b

- a. Sign Test
- b. Binomial distribution used.

Marginal Homogeneity Test	
	avg1 & avg2
Distinct Values	30
Off-Diagonal Cases	15
Observed MH Statistic	10899.000
Mean MH Statistic	12703.384
Std. Deviation of MH Statistic	698.851
Std. MH Statistic	-2.582
Asymp. Sig. (2-tailed)	.010

Best statistic

Sign Test

Frequencies		N
best2 - best1	Negative Differences ^a	3
	Positive Differences ^b	9
	Ties ^c	3
	Total	15

- a. best2 < best1
- b. best2 > best1
- c. best2 = best1

Test Statistics ^a	
	best2 - best1
Exact Sig. (2-tailed)	.146 ^b

- a. Sign Test
- b. Binomial distribution used.

Marginal Homogeneity Test	
	best1 & best2
Distinct Values	27
Off-Diagonal Cases	12
Observed MH Statistic	8525.000
Mean MH Statistic	9351.000
Std. Deviation of MH Statistic	385.080
Std. MH Statistic	-2.145
Asymp. Sig. (2-tailed)	.032

4. So sánh các thuật toán


4.3 Điều chỉnh các tham số trên thuật toán đề xuất đề tìm nghiệm tốt hơn

Parameters	Definition	Value
<i>POP SIZE</i>	Number of individuals in population	100
<i>MAX GENERATIONS</i>	Maximum number of GA generations	200
<i>MAX EVALUATIONS</i>	Maximum number of evaluations	20000
<i>Pc</i>	Crossover probability	1.0
<i>Pm</i>	Mutation probability	0.2
Seeds	Number of runs	30

4. So sánh các thuật toán

4.3 Điều chỉnh các tham số trên thuật toán đề xuất để tìm nghiệm tốt hơn

	Original algorithm			Proposal algorithm		
Dataset	avg	best	time(s)	avg	best	time(s)
75lin105.txt	668	668	8551.413	868	759	6305.332
5st70.txt	409	409	1922.643	485.7	409	3432.079
5pr76.txt	664	664	2702.053	740.167	684	5410.55
5eil76.txt	475	475	2511.95	483.033	470	5564.448
5eil51.txt	691	691	1261.639	751.433	690	2298.591
5berlin52.txt	920	920	1804.733	922.633	920	1917.252
50rat99.txt	746	746	8659.092	1266.533	1083	5365.606
50lin105.txt	976	976	10727.86	1683.967	1292	7621.875
10st70.txt	772	772	2760.239	850.467	781	2899.914
10rat99.txt	897	897	5306.593	1380.667	1192	6892.955
10pr76.txt	795	795	3452.729	779.967	705	3709.608
10kroB100.txt	626	626	4884.982	974.333	763	6445.717
10eil76.txt	806	806	3392.782	867.5	818	5135.804
10eil51.txt	887	887	1838.096	890.267	887	2779.842
10berlin52.txt	567	567	1629.63	592.733	567	1790.026



THANK YOU !

ONE LOVE. ONE FUTURE.

References

1. PGS.TS Huỳnh Thị Thanh Bình - Giáo trình Tính toán tiến hoá
2. Tuan Anh Do, Ha-Bang Ban, Thi Thanh Binh Huynh, Minh Tu Le and Binh Long Nguyen – “Genetic Algorithm based Approach to Solve the Clustered Steiner Tree Problem”