Phạm Đình Thành (2021)
(22): 93 - 103

CẢI TIẾN TOÁN TỬ ĐỘT BIẾN TRONG THUẬT TOÁN TIẾN HÓA ĐA NHÂN TỐ GIẢI BÀI TOÁN CÂY KHUNG PHÂN CỤM ĐƯỜNG ĐI NGẮN NHẤT

Phạm Đình Thành
Trường Đại học Tây Bắc

TÓM TẮT

Bài toán cây khung phân cụm đường đi ngắn nhất được ứng dụng nhiều trong tối ưu hệ thống tưới tiêu nông nghiệp, hệ thống cáp mạng và mạng lưới phân phối hàng hóa, dịch vụ. Do bài toán cây khung phân cụm đường đi ngắn nhất thuộc lớp bài toán NP-Khó nên các hướng tiếp cận gần đây thường sử dụng các thuật toán xấp xỉ để tìm lời giải, trong đó, hướng tiếp cận sử dụng kết hợp giữa thuật toán tiến hóa đa nhân tố và thuật toán tham lam ngẫu nhiên tìm được kết quả tối ưu trên nhiều bộ dữ liệu. Tuy nhiên, toán tử đột biến trong hướng tiếp cận này vẫn còn hạn chế khi luôn cố định số lần thay thế cạnh mới trên cá thể. Để khắc phục hạn chế trên, nghiên cứu đề xuất toán tử đột biến có khả năng thay đổi số lần thay thế cạnh mới trên cá thể trong mỗi lần thực hiện, cũng như có khả năng thay thế nhiều cạnh mới trên cá thể. Để chứng minh hiệu quả của đề xuất, nghiên cứu đã tiến hành thực nghiệm các thuật toán trên nhiều bộ dữ liệu khác nhau. Kết quả thực nghiệm đã chỉ ra tính hiệu quả của toán tử được đề xuất.

Từ khóa: *Thuật toán tiến hóa đa nhân tố, cây khung phân cụm đường đi ngắn nhất, tối ưu tổ hợp.*

1. Giới thiệu

Bài toán tìm cây khung có chi phí nhỏ nhất (*Minimal-Cost Spanning Tree - MCST*) trên đồ thị có trọng số là một trong các bài toán nổi tiếng trong lĩnh vực tối ưu rời rạc cũng như trong khoa học máy tính. Bài toán MCST được ứng dụng trong nhiều lĩnh vực thực tế như: tối ưu hệ thống truyền thông, tối ưu hệ thống giao vận, v.v. [16].

Trong nhiều ứng dụng mạng, nhằm đảm bảo tính hiệu quả và bảo mật, các thiết bị đầu cuối có thể được chia vào các nhóm sao cho việc kết nối giữa các thiết bị đầu cuối trong cùng một nhóm có tính "cục bộ". Khi đó, việc đảm bảo liên kết giữa các thiết bị đầu cuối, tương ứng với việc cần phải tìm cây khung của đồ thị con với các đỉnh thuộc cùng một nhóm. Với những yêu cầu thực tiễn đó, một lớp các bài toán cây khung, trong đó tập đỉnh được phân chia thành các tập con đã được quan tâm nghiên cứu. Trong đó, bài toán cây phân cụm đường đi ngắn nhất (*Clustered Shortest-Path Tree Problem - CluSPT*) [5] là bài toán có vai trò quan trọng trong thực tiễn và nhận được nhiều sự quan tâm nghiên cứu.

Do CluSPT thuộc lớp bài toán NP-Khó [5] nên các hướng tiếp cận thường sử dụng các thuật toán xấp xỉ. Trong những năm gần đây, các thuật toán có ý tưởng bắt nguồn từ tự nhiên được sử dụng rộng rãi để giải các bài toán có mức độ phi tuyến cao hoặc các bài toán tối ưu rất khó [19]. Trong các thuật toán lấy ý tưởng từ quá trình tối ưu hóa trong tự nhiên, thuật toán tiến hóa đa nhân tố (*Multi-Factorial Evolutionary Algorithm - MFEA*) là một trong các thuật toán được quan tâm nghiên cứu nhiều trong thời gian gần đây [6]. Do thuật toán MFEA được kế thừa các ưu điểm của quá trình trao đổi tri thức tiềm ẩn (*implicit knowledge transfer*) giữa các bài toán nên quá trình tìm kiếm lời giải của thuật toán MFEA được cải thiện về cả tốc độ và chất lượng so với lời giải tìm được khi sử dụng thuật toán tiến hóa (*Evolutionary Algorithm - EA*) cơ bản.

Trong các nghiên cứu về áp dụng thuật toán MFEA để giải bài toán CluSPT,

nghiên cứu kết hợp giữa thuật toán MFEA và thuật toán tham lam ngẫu nhiên (*Randomized Greedy Algorithm - RGA*) (ký hiệu là G-MFEA) tìm được lời giải trội hơn các thuật toán khác, với một số bộ dữ liệu thuật toán G-MFEA tìm được lời giải tối ưu. Tuy nhiên, do kết hợp với thuật toán RGA nên sau một số thế hệ, độ đa dạng quần thể có xu hướng giảm nhanh. Bên cạnh đó, khi số lượng cụm của đồ thị đầu vào lớn, toán tử đột biến luôn thực hiện một lần thay thế cạnh của cá thể nên khả năng khai thác (*exploitation*) không gian tìm kiếm của thuật toán G-MFEA bị hạn chế.

Do đó, nghiên cứu này muốn cải thiện hiệu quả của thuật toán G-MFEA thông qua cải tiến toán tử đột biến. Toán tử đột biến được cải tiến có khả năng tạo ra nhiều sự thay đổi trên cá thể thông qua khả năng thay thế nhiều lần cạnh của cá thể và số lần thay mới cạnh trên cá thể cũng được thay đổi linh hoạt trong mỗi lần thực hiện đột biến.

Các phần còn lại của nghiên cứu được tổ chức như sau: phần 2 trình bày về phát biểu bài toán và các khái niệm liên quan; phần 3 trình bày về các nghiên cứu liên quan; phần 4 giới thiệu về toán tử đột biến cải tiến; phần 5 phân tích kết quả thực nghiệm của thuật toán đề xuất; phần kết luận của nghiên cứu được trình bày trong phần 6.

2. Phát biểu bài toán

Cho $G = (V, E, w)$ là một đơn đồ thị vô hướng, liên thông, có trọng số cạnh không âm; trong đó V và E lần lượt là tập đỉnh và tập cạnh của đồ thị; w là ma trận trọng số cạnh của đồ thị. Cho trước tập các đỉnh $S \subseteq V$, ký hiệu $G[S]$ là đồ thị con của G được cảm sinh bởi tập S; $T[S]$ là đồ thị con của cây khung T của đồ thị G được cảm sinh bởi tập S.

Định nghĩa 2.1 (Phân hoạch của tập đỉnh của đồ thị [7]). *Cho $G = (V, E, w)$ là một đơn đồ thị vô hướng, liên thông, các cạnh có trọng số không âm. Tập $C = \{C_1, C_2, \ldots, C_h\}$ được gọi là phân hoạch của V nếu $C_1 \cup C_2 \cup \ldots \cup C_h = V$ và $C_i \cap C_j = \emptyset, \forall i, j \in [1, h], i \neq j$.*

Định nghĩa 2.2 (Đồ thị phân cụm). *Cho $G = (V,\ E,\ w)$ là một đơn đồ thị vô hướng, liên thông, các cạnh có trọng số không âm. Nếu tồn tại tập phân hoạch $C = \{C_1, C_2, \ldots, C_h\}$ của V thì G được gọi là đồ thị phân cụm, tập $C_1, C_2, \ldots, C_h$ được gọi là các cụm (cluster) của đồ thị. Đồ thị phân cụm của G với tập phân hoạch C được ký hiệu là $G = (V,\ E,\ w,\ C)$.*

Định nghĩa 2.3 (Đồ thị G-Graph). *Cho $G = (V,\ E,\ w,\ C)$ là một đồ thị phân cụm. G-Graph là đồ thị được suy ra từ đồ thị G trong đó mỗi đỉnh của G-Graph tương ứng với một cụm trong đồ thị G, giữa hai đỉnh của đồ thị G-Graph có cạnh nối khi có ít nhất một cạnh nối giữa các đỉnh của các cụm tương ứng trong đồ thị G.*

Định nghĩa 2.4 (Chi phí định tuyến giữa hai đỉnh [5]). *Cho $G = (V, E, w)$ là một đơn đồ thị vô hướng, liên thông, các cạnh có trọng số không âm. Chi phí định tuyến giữa hai đỉnh $u, v \in V$ trên cây khung T (ký hiệu $d_T(u,v)$) của đồ thị G được tính bằng chi phí đường đi nối giữa hai đỉnh đó trên cây khung T.*

Bài toán CluSPT được phát biểu như sau [5]:

Cho đồ thị phân cụm $G = (V, E, w, C)$ với $C = \{C_1, C_2, \ldots, C_h\}$ và đỉnh nguồn $s \in V$. Mục tiêu của bài toán CluSPT là tìm một cây khung T của đồ thị G sao cho:

- Với mỗi cụm $C_i (i = 1, \ldots, h)$, đồ thị $T[C_i]$ là một đồ thị liên thông.
- Tổng chi phí định tuyến giữa đỉnh nguồn s và các đỉnh còn lại trên cây khung T là nhỏ nhất, hay nói cách khác:

$$f(T) = \sum_{v \in V(T)} d_T(s, v) \rightarrow \min \quad (2.1)$$

3. Các nghiên cứu liên quan

Các bài toán liên quan đến tập đỉnh được phân vào các cụm đã được biết đến từ những năm 70 của thế kỷ trước [8]. Ngày nay, xuất phát từ yêu cầu cần tối ưu hệ thống mạng, bài toán cây phân cụm nhận được nhiều sự quan tâm trong cộng đồng các nhà nghiên cứu [4, 17].

Tác giả Y.S. Myung và nhóm nghiên cứu [9] đã nghiên cứu bài toán cây khung nhỏ nhất tổng quát (*Generalized Minimum Spanning Tree Problem - GMSTP*) với các đỉnh được chia vào các nhóm. Lời giải của bài toán GMSTP là cây có chi phí nhỏ nhất và mỗi nhóm chỉ chứa một đỉnh duy nhất. Trong nghiên cứu này, sau khi chứng minh GMSTP là bài toán thuộc lớp NP–Khó, các tác giả đã đề xuất hai mô hình quy hoạch nguyên tuyến tính và so sánh chúng trong việc giải mô hình nới lỏng quy hoạch nguyên khi áp dụng vào bài toán GMSTP. Kết quả thực nghiệm cho thấy khoảng cách giữa cận trên và cận dưới tăng khi số lượng nhóm và số đỉnh trong mỗi nhóm tăng.

Một trong các bài toán cây phân cụm khác nhận được nhiều sự quan tâm là bài toán cây Steiner phân cụm (*Clustered Steiner Tree Problem - CluSteinerTP*) [18]. Bài toán CluSteinerTP cũng chia các đỉnh vào các cụm, bài toán cây Steiner (*Steiner tree problem - STP*) là CluSteinerTP nếu các cụm không có phần tử chung [17]. Trong [18], dựa trên kết quả thực nghiệm, các tác giả cũng đã chỉ ra rằng nếu tỉ lệ Steiner thuộc khoảng (3, 4) thì kết quả là tốt nhất và từ đó đề xuất một thuật toán gần đúng giả bài toán CluSteinerTP bằng cách chuyển CluSteinerTP về bài toán cây Steiner.

Một biến thể khác của bài toán cây phân cụm, bài toán cây khung phân cụm có chi phí định tuyến nhỏ nhất (*Minimum Routing Cost Clustered Tree Problem - CluMRCT*) [7]. Trong nghiên cứu của mình các tác giả đã chỉ ra rằng bài toán CluMRCT thuộc lớp NP-Khó nếu bài toán có ít nhất hai cụm. Các tác giả cũng đã đề xuất một thuật toán xấp xỉ cận tỉ lệ 2 để giải bài toán CluMRCT bằng cách tạo đồ thị gồm hai mức dựa trên cây khung R-star và dựa trên hai đặc trưng của cây khung R-star.

Tác giả D'Emidio và các cộng sự [5] đã nghiên cứu một dạng khác của bài toán cây khung phân cụm, bài toán CluSPT. Tác giả cũng đã đề xuất thuật toán xấp xỉ (ký hiệu là AAL) để giải bài toán CluSPT. Thuật toán AAL chia bài toán CluSPT thành 2 bài toán con và tìm lời giải mỗi bài toán con trong mỗi giai đoạn khác nhau, trong đó, thuật toán Prim [11] được sử dụng để tìm lời giải của cả bài toán con thứ nhất (là cây khung nhỏ nhất của đồ thị nối giữa các cụm) và lời giải bài toán con thứ 2 (là cây khung nhỏ nhất của đồ thị con trong mỗi cụm). Chất lượng lời giải tìm được bằng thuật toán AAL còn hạn chế do thuật toán AAL có gắng cực tiểu tổng trọng số của cây khung của các đồ thị con mà không xem xét tới khoảng cách từ các đỉnh tới đỉnh nguồn.

Gần đây, một số thuật toán đã được đề xuất để giải bài toán CluSPT như: thuật toán tham lam, thuật toán tiến hóa, thuật toán tiến hóa đa nhân tố, v.v.. Trong nghiên cứu [3], các tác giả đã đề xuất áp dụng thuật toán MFEA (ký hiệu là E-MFEA) với các toán tử tiến hóa mới để giải bài toán CluSPT. Ý tưởng chính của các toán tử tiến hóa được đề xuất là xây dựng cây khung cho đồ thị con nhỏ nhất trước tiến, sau đó lần lượt tạo cây khung cho các đồ thị con lớn hơn dựa trên các cây khung của các đồ thị bé hơn.

Trong nghiên cứu [15], các tác giả đã sử dụng các thế mạnh của mã Cayley [10] để mã hóa lời giải và đề xuất các toán tử tiến hóa giải bài toán CluSPT. Cơ chế hoạt động của các toán tử tiến hóa được xây dựng dựa trên ý tưởng của các toán tử tiến hóa sử dụng mã hóa nhị phân và mã hóa hoán vị [1].

Trong nghiên cứu [2], các tác giả đề xuất thuật toán (ký hiệu là N-EA) dựa trên kết hợp giữa thuật toán EA và thuật toán Dijkstra. Trong cách tiếp cận này, bài toán CluSPT được phân rã thành hai bài toán con: bài toán thứ nhất sẽ xác định cây khung nối giữa các cụm; bài toán thứ hai sẽ tìm cây khung tốt nhất trong mỗi cụm. Mặc dù thuật toán N-EA giúp giảm hao phí tài nguyên cũng như thời gian thực hiện, tuy nhiên, thuật toán N-EA vẫn còn một số hạn chế như: mỗi cụm chỉ nối với các cụm khác thông qua một số đỉnh nên lời giải tìm được có thể chưa tối ưu.

Trong nghiên cứu [13], các tác giả đã đề xuất thuật toán xấp xỉ (ký hiệu là HB-RGA) để giải bài toán CluSPT. Thuật toán HB-RGA kết hợp giữa RGA và thuật toán Dijkstra. Trong thuật toán HB-RGA, cây đường đi ngắn nhất của mỗi cụm được xây dựng thông qua sử dụng thuật toán Dijkstra, trong khi, các cạnh nối giữa giữa các cụm được xây dựng bởi thuật toán RGA. Ưu điểm chính của thuật toán HB-RGA là khả năng khai thác không gian tìm kiếm để tạo ra lời giải tốt hơn từ lời giải ban đầu. Tuy nhiên, thuật toán HB-RGA có thể bị rơi vào bẫy cục bộ khi số chiều của bài toán tăng do thuật toán dựa trên chiến lược tham lam thường có khả năng khai phá (*exploration*) không gian tìm kiếm không tốt.

Trong nghiên cứu [14], các tác giả đề xuất thuật toán dựa trên thuật toán MFEA (ký hiệu là G-MFEA) gồm có hai tác vụ: nhiệm vụ của tác vụ thứ nhất là xác định lời giải hợp lệ của bài toán CluSPT, trong khi nhiệm vụ của tác vụ thứ hai là cải thiện chất lượng lời giải tìm được trong tác vụ thứ nhất thông qua cơ chế trao đổi vật chất di truyền tiềm ẩn (*implicit genetic transfer*) giữa các tác vụ. Tác vụ thứ hai sẽ tìm lời giải tốt nhất dựa trên tối ưu các cạnh nối giữa các cụm của mỗi lời giải bài toán CluSPT tìm được ở tác vụ thứ nhất.

Mặc dù thuật toán G-MFEA tìm được lời giải gần lời giải tối ưu, tuy nhiên, khi số lượng cụm của đồ thị đầu vào lớn, toán tử đột biến trong thuật toán G-MFEA chưa giúp cải thiện nhiều chất lượng lời giải tìm được nên nghiên cứu này sẽ cải tiến toán tử đột biến trong thuật toán G-MFEA.

4. Thuật toán đề xuất

Do toán tử đột biến trong thuật toán G-MFEA (ký hiệu là OMO) chỉ thay đổi một cạnh nối giữa các cụm, các đỉnh nên khi số lượng cụm và số lượng đỉnh của đồ thị đầu vào tăng, toán tử đột biến càng có xu hướng tạo ít sự thay đổi hơn đối với cá thể. Bên cạnh đó, toán tử OMO luôn thực hiện một sự thay đổi trên cá thể nên khi số lượng cụm và số lượng đỉnh của đồ thị đầu vào thay đổi, toán tử OMO có thể không hiệu quả.

Từ các phân tích trên, nghiên cứu đề xuất toán tử đột biến mới (ký hiệu là IMO) sử dụng trong thuật toán G-MFEA. Ý tưởng chính của toán tử IMO là có thể thực hiện thay đổi nhiều hơn một cạnh nối giữa các cụm trên cá thể và số cạnh nối giữa các cụm bị thay đổi trong mỗi lần thực hiện đột biến không cố định.

Các bước của toán tử đột biến đề xuất được trình bày trong thuật toán bên dưới, trong đó, tham số *numMutation* sẽ xác định số lần tối đa thực hiện đột biến, hay nói cách khác, *numMutation* là số tối đa cạnh nối giữa các cụm của cá thể có thể bị thay đổi. Bên cạnh đó, khác với thuật toán của toán tử OMO, thuật toán cải tiến sẽ thực hiện thay đổi ngẫu nhiên r lần (dòng lệnh 3) cạnh nối giữa các cụm.

Ví dụ về các bước chính của toán tử IMO được minh họa trong hình 1. Xét đồ thị đầu vào được minh họa như trong hình 1a). Đồ thị G-Graph được xây dựng từ đồ thị đầu vào được minh họa trong hình 1b), cá thể trong không gian không gian tìm kiếm chung (*Unified Search Space - USS*) được minh họa trong hình 1c). Giả sử IMO chỉ

Thuật toán: Toán tử đột biến trong thuật toán G-MFEA

Input: Đồ thị phân cụm $G = (V, E, w, C)$; Cá thể cha mẹ $I = (ES, IE, LR)$;
Số lần đột biến tối đa $numMutation$;
Output: Cá thể con $I^* = (ES^*, IE^*, LR^*)$;

begin
  Tạo đồ thị G-Graph $G' = (V', E')$ từ đồ thị G;
  $r \leftarrow$ Số ngẫu nhiên trong nửa đoạn $[0, numMutation)$ ▷ Số lần thay thế cạnh mới;
  for $ii \leftarrow 1$ **to** r **do**
    $e' = (v'_i, v'_j) \leftarrow$ Chọn ngẫu nhiên một cạnh từ tập $E' \backslash ES$;
    $ES^* \leftarrow ES$;
    Thêm cạnh e' vào E^* để tạo thành chu trình;
    Xác định chu trình Δ trong E^*;
    Chọn ngẫu nhiên cạnh $e^* = (v'_m, v'_n) \in \Delta$ sao cho $e^* \neq e'$;
    Xóa cạnh e^* từ tập E^*;
    `/* Cập nhật đỉnh gốc cục bộ và cạnh nối giữa các đỉnh của các cụm */`
    $u = (u_h, u_p) \leftarrow$ Chọn ngẫu nhiên từ tập E một cạnh nối giữa đỉnh $u_h \in C_i$ và đỉnh $u_p \in C_j$;
    `/* Cập nhật thuộc tính IE của cá thể con đối với cạnh bị xóa */`
    $IE^*[m][n] \leftarrow null$;
    $IE^*[n][m] \leftarrow null$;
    `/* Cập nhật thuộc tính IE của cá thể con đối với cạnh mới thêm */`
    $IE^*[i][j] \leftarrow u_h$;
    $IE^*[j][i] \leftarrow u_p$;
    `/* Cập nhật đỉnh gốc cục bộ của các cụm` C_m `và` C_n `*/`
    if *(đỉnh gốc cục bộ của cụm* $C_t (t = m, n)$ *thuộc cạnh bị xóa* e^**) và (đỉnh gốc cục bộ của cụm* C_t *không còn cạnh nào nối tới một cụm khác)* **then**
      $LR^*[t] \leftarrow$ Chọn ngẫu nhiên từ cụm C_t một đỉnh có cạnh nối với một cụm khác;
  return (ES^*, IE^*, LR^*);

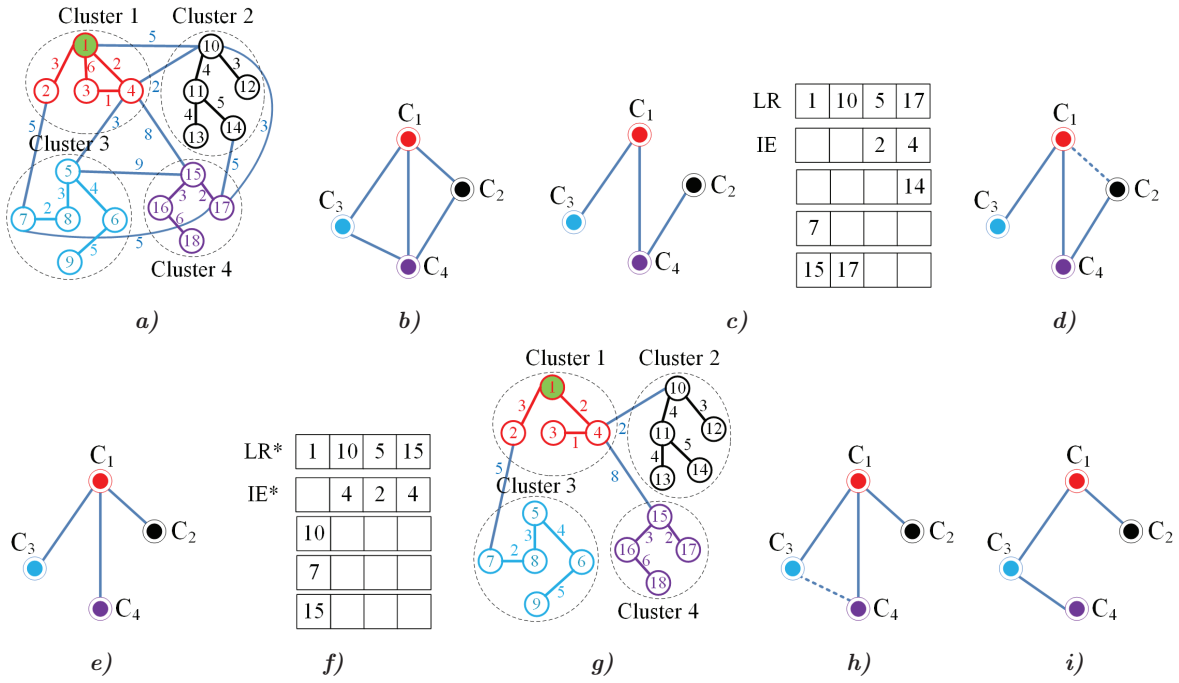
thực hiện một lần thay thế cạnh mới và hình 1d) là cá thể con nhận được sau khi thêm cạnh mới (C_1, C_2) vào cá thể để tạo thành chu trình (C_1, C_2, C_4, C_1); còn (C_2, C_4) là cạnh được xóa bỏ từ chu trình (C_1, C_2, C_4, C_1) để tạo thành cá thể mới. Trong bước tiếp theo, toán tử đột biến sẽ cập nhật các thuộc tính IE và LR của cá thể con mới tạo ra trong bước thứ nhất. Do cạnh (4, 10) được chọn để nối giữa hai cụm C_1 và C_2 nên tập IE* của cá thể con được cập nhật như trong hình 1f). Sau khi xóa cạnh (14, 17) nối giữa hai cụm C_2 và C_4 ở trong bước thứ nhất, đỉnh 17 trong cụm 4 không thuộc bất cứ cạnh nào nối cụm 4 tới các cụm khác. Do đó, một đỉnh khác có cạnh nối ra cụm khác được chọn làm đỉnh gốc của cụm 4. Trong cụm 4 chỉ có duy nhất đỉnh 15 thỏa mãn điều kiện nên đỉnh này được làm đỉnh gốc của cụm. Hình 1g) minh họa lời giải bài toán CluSPT được xây dựng từ các thể con mới. Hình 1h) và hình 1i) minh họa trường hợp toán tử IMO thực hiện thêm một lần thay thế cạnh, trong đó, hình 1h) minh họa cá thể nhận được sau khi thêm cạnh mới (C_3, C_4) vào để tạo thành chu trình (C_1, C_3, C_4, C_1); hình 1i) minh họa cá thể sau khi xóa cạnh (C_1, C_4) khỏi chu trình (C_1, C_3, C_4, C_1). Khi đó, đồ thị trong hình 1i) chính là đồ thị G-Graph của cá thể đầu ra của toán tử IMO.

5. Kết quả và đánh giá

5.1. Dữ liệu thực nghiệm

Do tính đa dạng về kiểu dữ liệu và nhận được đánh giá cao từ nhiều nhà nghiên cứu khi giải các bài toán trên đồ thị phân cụm nên tập dữ liệu MOM-Lib [8] được chọn để xây dựng tập dữ liệu đánh giá bài toán CluSPT (gọi tắt là CluSPT-Lib).

Do đầu vào của bài toán CluSPT cần thêm thông tin về đỉnh nguồn nên mỗi bộ



Hình 1: *Ví dụ minh họa các bước của toán tử đột biến*

dữ liệu trong tập MOM-Lib được chọn ngẫu nhiên một đỉnh làm đỉnh nguồn. Thông tin về các bộ dữ liệu được lưu tại [12].

Do có tới 248 bộ dữ liệu thuộc 6 tập dữ liệu (có tên là Type 1, ..., Type 6) nên nghiên cứu chỉ tiến hành thực nghiệm với các bộ dữ liệu thuộc tập dữ liệu nhỏ.

5.2. Tiêu chí đánh giá

Nghiên cứu tập trung phân tích các tiêu chí chất lượng lời giải tìm được của các thuật toán theo giá trị trung bình cộng chi phí của hàm mục tiêu (Avg) và giá trị tốt nhất tìm được trong các lần thực hiện (BF).

5.3. Môi trường, tham số thực nghiệm

Để đánh giá hiệu quả của toán tử IMO, nghiên cứu tiến hành hai nhóm thực nghiệm chính:

- Nghiên cứu phân tích hiệu quả của các thuật toán MFEA trong nghiên cứu [14] khi không sử dụng (ký hiệu là G-MFEA và khi sử dụng toán tử đột biến cải tiến IMO (ký hiệu IG-MFEA).
- Nghiên cứu phân tích ảnh hưởng của một số tham số của đồ thị đầu vào tới hiệu quả của thuật toán đề xuất.

Với mỗi bộ dữ liệu, thuật toán được thực nghiệm 30 lần trên máy tính cài đặt hệ điều hành Microsoft Windows 10 với cấu hình: CPU - Intel Xeon E5620, RAM - 8GB. Thuật toán HB-RGA được thực nghiệm với các tham số $\gamma = 50$, thuật toán tiến hóa đa nhân tố được thực nghiệm với tham số: số lần đánh giá là 50000 lần; kích thước quần thể $P = 100$; tỉ lệ lai ghép $p_c = 0,3$; xác suất ghép cặp ngẫu nhiên $rmp = 0,5$; số lần đột biến tối đa $numMutation = 2$.

5.4. Kết quả thực nghiệm

a) Phân tích hiệu quả của thuật toán sử dụng toán tử đề xuất

Các bảng 1, 2 và 3 trình bày kết quả so sánh giữa hai thuật toán G-MFEA và IG-MFEA. Trong các bảng này, tại mỗi dòng, giá trị tương ứng với lời giải tốt hơn sẽ được in nghiêng và có mầu đỏ.

Kết quả tại các bảng 1, 2 và 3 cho thấy thuật toán IG-MFEA tìm được lời giải tốt hơn thuật toán G-MFEA trên đa số bộ dữ liệu. Đối với mỗi tập dữ liệu, số bộ dữ liệu mà thuật toán IG-MFEA có kết quả tốt hơn thuật toán G-MFEA cũng nhiều hơn số bộ

dữ liệu mà thuật toán G-MFEA có kết quả tốt hơn thuật toán IG-MFEA. Tuy nhiên, kết quả so sánh giữa hai thuật toán trên các tập dữ liệu vẫn có điểm khác nhau, cụ thể:

- Tập dữ liệu Type 1: thuật toán IG-MFEA tốt hơn trên 19 bộ dữ liệu, trong khi thuật toán G-MFEA tốt hơn trên 3 bộ dữ liệu.
- Đối với tập dữ liệu Type 5, kết quả so sánh giữa hai thuật toán chênh lệch ít hơn so với tập dữ liệu Type 1 và Type 6 khi thuật toán IG-MFEA tốt hơn trên 8 bộ dữ liệu, trong khi thuật toán G-MFEA tốt hơn trên 4 bộ dữ liệu.
- Tập dữ liệu Type 6: thuật toán IG-MFEA tốt hơn trên 22 bộ dữ liệu, trong khi thuật toán G-MFEA tốt hơn trên 2 bộ dữ liệu.

b) Phân tích ảnh hưởng của tham số tới toán tử đề xuất

Do cá thể trong thuật toán G-MFEA [14] mã hóa cây khung nối giữa các cụm của đồ thị đầu vào nên nghiên cứu phân tích sự ảnh hưởng của số cụm của đồ thị đầu vào tới kết quả so sánh của thuật toán G-MFEA và IG-MFEA.

Hình 2 minh họa mối quan hệ giữa kết quả so sánh của các thuật toán và số cụm của đồ thị đầu vào. Trong các hình này, ký hiệu ">>" ("<<") nghĩa là "tốt hơn" ("kém hơn"). Ví dụ, "G-MFEA >> IG-MFEA" nghĩa là thuật toán G-MFEA tìm được lời giải tốt hơn thuật toán IG-MFEA.

Kết quả so sánh giữa hai thuật toán trong hình 2a) và hình 2c) có điểm tương đồng khi thuật toán IG-MFEA tìm được lời giải tốt hơn thuật toán G-MFEA khi số cụm lớn hơn 10 (Type 1), lớn hơn hoặc bằng 16 (Type 6). Đối với tập Type 1, thuật toán G-MFEA trội hơn thuật toán IG-MFEA trong ba trường hợp có số cụm bằng 10; hai thuật toán tìm được cùng kết quả trên các bộ dữ liệu có số cụm bằng 5. Đối tới tập Type 6, thuật toán G-MFEA không tốt hơn thuật toán G-MFEA trên các bộ dữ liệu có số cụm nhỏ hơn 12.

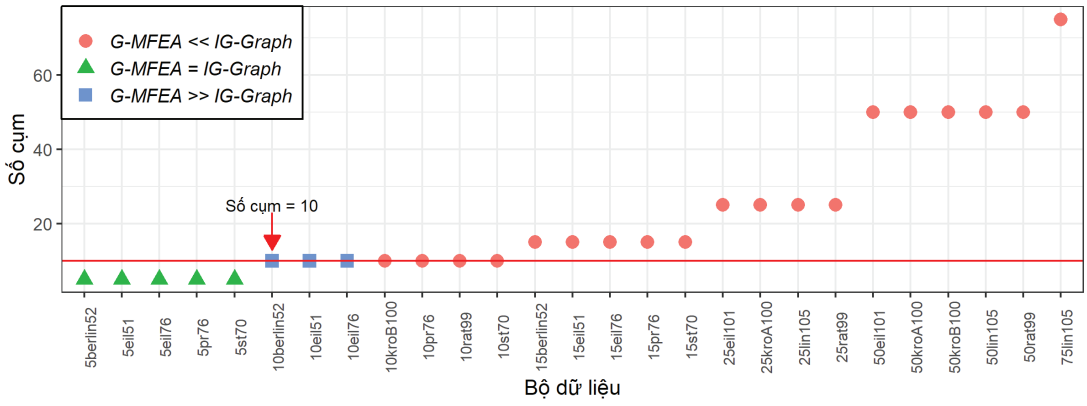
Bảng 1: Kết quả thực nghiệm của các thuật toán trên bộ dữ liệu thuộc Type 1

	IG-MFEA		G-MFEA	
Bộ dữ liệu	BF	Avg	BF	Avg
10berlin52	43724,1	43729,9	43724,1	*43724,1*
10eil51	1713,2	1713,4	1713,2	*1713,2*
10eil76	2203,3	2203,9	2203,3	*2203,3*
10kroB100	140522,2	*140528,6*	140551,2	140597,9
10pr76	522213,8	*522249,7*	522213,8	522340,4
10rat99	7520,2	*7523*	7520,2	7524
10st70	3095,2	*3095,3*	3095,2	3095,7
15berlin52	26314,2	*26343,8*	26315,5	26351,7
15eil51	1306,4	*1308,3*	1306,8	1309,1
15eil76	2909,1	*2910,3*	2909,1	2913,1
15pr76	704615,8	*705355,3*	705226,1	706505,5
15st70	4120,2	*4125,3*	4126,7	4135,5
25eil101	4687,6	*4710,6*	4700,4	4727,9
25kroA100	147405,7	*148533,1*	148767,9	149708,1
25lin105	98092,1	*99174,2*	98941,4	100585,3
25rat99	6861,3	*6931,8*	6930,9	7022,3
50eil101	3942,6	*4173*	4034,7	4178,1
50kroA100	167267,2	*175235,5*	173113,3	179506,1
50kroB100	145095,6	*153633,3*	149465,6	157831,1
50lin105	149623,0	*153169,8*	151901,5	154680,7
50rat99	8468,7	*8811,1*	8728,0	9002,1
5berlin52	22746,4	22746,4	22746,4	22746,4
5eil51	1769,4	1769,4	1769,4	1769,4
5eil76	2630,8	2630,8	2630,8	2630,8
5pr76	585008,0	585008	585008,0	585008
5st70	4520,1	4520,1	4520,1	4520,1
75lin105	167241,2	*177585,7*	169739,8	177610,8

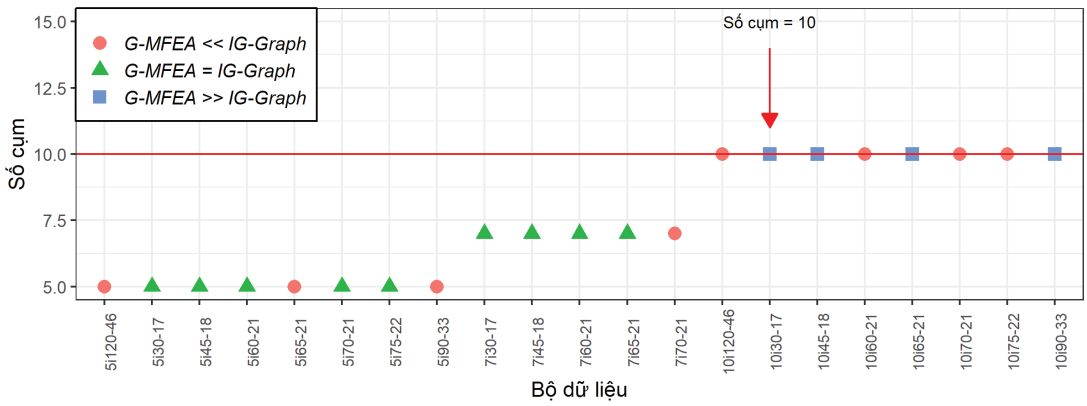
Kết quả so sánh giữa hai thuật toán trong hình 2b) khác với kết quả so sánh trong hai hình 2a) và 2c) khi thuật toán IG-MFEA chiếm ưu thế hơn so với thuật toán G-MFEA trên các bộ dữ liệu có số cụm nhỏ hơn 10.

6. Kết luận

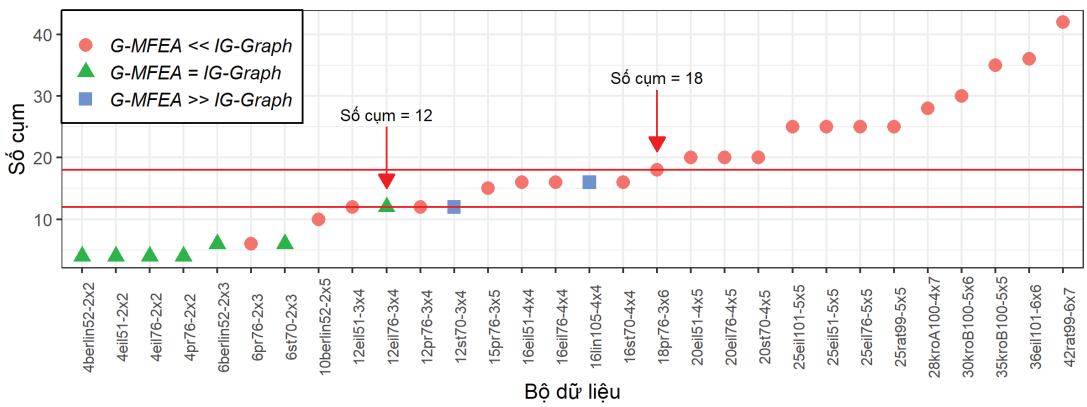
Mặc dù thuật toán G-MFEA tìm được lời giải tối ưu trong nhiều trường hợp, tuy nhiên toán tử đột biến sử dụng trong thuật toán này vẫn còn một số hạn chế. Nghiên cứu này tập trung vào mô tả cải tiến của toán tử đột biến trong thuật toán G-MFEA. Toán tử đột biến được đề xuất có thể thực hiện nhiều lần sự thay đổi trên trên cá thể



a) Type 1



b) Type 5



c) Type 6

Hình 2: *Biểu đồ phân tán của kết quả so sánh các thuật toán và số cụm của đồ thị.*

cũng như không cố định số lần tác động lên cá thể. Kết quả thực nghiệm trên các tập dữ liệu khác nhau cho thấy, thuật toán tiến hóa đa nhân tố sử dụng toán tử đột biến cải tiến tìm được kết quả tốt hơn thuật toán tiến hóa đa nhân tố khi sử dụng toán tử đột biến ban đầu trên đa số bộ dữ liệu thuộc tất cả các tập dữ liệu được thực nghiệm. Trong thời gian tới, nhóm tác giả sẽ tiếp tục thử nghiệm các trường hợp của toán tử đột biến được đề xuất, cũng như phân tích sâu thêm sự ảnh hưởng của các tham số tới hiệu quả của toán tử được đề xuất.

TÀI LIỆU THAM KHẢO

[1] Bäck, T., Fogel, D. B., and Michalewicz, Z. (2018). *Evolutionary computation 1: Basic algorithms and operators*. CRC press.

[2] Binh, H. T. T., Thanh, P. D., and Thang, T. B. (2019). New approach to solving the

Bảng 2: Kết quả thực nghiệm của các thuật toán trên bộ dữ liệu thuộc Type 5

	IG-MFEA		G-MFEA	
Bộ dữ liệu	BF	Avg	BF	Avg
10i120-46	93927,3	*93979,9*	93956,9	94034,3
10i30-17	13276,6	13276,7	13276,6	*13276,6*
10i45-18	22890,4	22897,1	22890,4	*22892,2*
10i60-21	33694,8	*33695,1*	33694,8	33702,8
10i65-21	37353,1	37357,1	37353,1	*37353,6*
10i70-21	38059,5	*38083,5*	38066,7	38187,3
10i75-22	65361,9	*65381,6*	65362,0	65397,3
10i90-33	51934,6	51990	51943,2	*51975,6*
5i120-46	61451,5	*61475,1*	61451,5	61495,3
5i30-17	14399,9	14399,9	14399,9	14399,9
5i45-18	14884,3	14884,3	14884,3	14884,3
5i60-21	28422,7	28422,7	28422,7	28422,7
5i65-21	30907,8	*30907,8*	30907,8	30911,7
5i70-21	35052,8	35052,8	35052,8	35052,8
5i75-22	34692,5	34692,5	34692,5	34692,5
5i90-33	51977,0	*51977*	51977,0	51977,3
7i30-17	20438,9	20438,9	20438,9	20438,9
7i45-18	20512,0	20512	20512,0	20512
7i60-21	36263,9	36263,9	36263,9	36263,9
7i65-21	34847,6	34847,6	34847,6	34847,6
7i70-21	39487,6	*39487,9*	39487,6	39491,1

Bảng 3: Kết quả thực nghiệm của các thuật toán trên bộ dữ liệu thuộc Type 6

	IG-MFEA		G-MFEA	
Bộ dữ liệu	BF	Avg	BF	Avg
10berlin52-2x5	27471,4	*27472,2*	27471,4	27473
12eil51-3x4	1699,0	*1699*	1699,0	1699,1
12eil76-3x4	2650,8	2650,8	2650,8	2650,8
12pr76-3x4	600023,5	*600532*	600430,9	600818,7
12st70-3x4	4106,5	4110,7	4106,5	*4110,1*
15pr76-3x5	524442,8	*525743,5*	525170,3	526166,2
16eil51-4x4	1301,4	*1303,5*	1302,7	1305,6
16eil76-4x4	2036,0	*2041,4*	2040,0	2052,2
16lin105-4x4	125058,5	125332,9	125052,2	*125289,8*
16st70-4x4	2933,0	*2939,7*	2935,4	2949,2
18pr76-3x6	638510,5	*640667,5*	639723,3	641700,1
20eil51-4x5	2284,4	*2290,8*	2288,7	2295,2
20eil76-4x5	2386,6	*2395,2*	2390,5	2402,2
20st70-4x5	2939,0	*2954,5*	2942,8	2967
25eil101-5x5	3612,1	*3635,8*	3649,2	3670,5
25eil51-5x5	1480,0	*1501,4*	1487,4	1507,3
25eil76-5x5	2194,8	*2218,1*	2219,1	2245,2
25rat99-5x5	11411,7	*11461,8*	11434,9	11485,9
28kroA100-4x7	133614,5	*136013,2*	136501,1	138342,8
30kroB100-5x6	198967,0	*200676,3*	200596,8	202209,8
35kroB100-5x5	129597,3	*131165,5*	130935,1	132840,4
36eil101-6x6	3892,3	*3964,4*	3929,2	3981,6
42rat99-6x7	9055,1	*9249*	9187,0	9393,5
4berlin52-2x2	23287,9	23287,9	23287,9	23287,9
4eil51-2x2	1898,5	1898,5	1898,5	1898,5
4eil76-2x2	2948,7	2948,7	2948,7	2948,7
4pr76-2x2	442693,0	442693	442693,0	442693
6berlin52-2x3	32128,6	32128,6	32128,6	32128,6
6pr76-2x3	648275,7	*648279,3*	648275,7	648507,9
6st70-2x3	3476,7	3476,7	3476,7	3476,7

clustered shortest-path tree problem based on reducing the search space of evolutionary algorithm. *Knowledge-Based Systems*, 180:12–25.

[3] Binh, H. T. T., Thanh, P. D., Trung, T. B., and Thao, L. P. (2018). Effective multifactorial evolutionary algorithm for solving the cluster shortest path tree problem. In *Evolutionary Computation (CEC), 2018 IEEE Congress on*, pages 819–826. IEEE.

[4] Chen, Y. H. (2017). The clustered and bottleneck clustered selected-internal steiner tree problems. In *The Second Malta Conference in Graph Theory and Combinatorics*.

[5] D'Emidio, M., Forlizzi, L., Frigioni, D., Leucci, S., and Proietti, G. (2019). Hardness, approximability, and fixed-parameter tractability of the clustered shortest-path tree problem. *Journal of Combinatorial Optimization*, pages 1–20.

[6] Gupta, A., Ong, Y.-S., and Feng, L. (2016). Multifactorial evolution: toward evolutionary multitasking. *IEEE Transactions on Evolutionary Computation*, 20(3):343–357.

[7] Lin, C.-W. and Wu, B. Y. (2016). On the minimum routing cost clustered tree problem. *Journal of Combinatorial Optimization*, pages 1–16.

[8] Mestria, M., Ochi, L. S., and de Lima Martins, S. (2013). GRASP with path relinking for the symmetric euclidean clustered traveling salesman problem. *Computers & Operations Research*, 40(12):3218–3229.

[9] Myung, Y.-S., Lee, C.-H., and Tcha, D.-W. (1995). On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241.

[10] Paulden, T. and Smith, D. (2006). Recent Advances in the Study of the Dandelion Code, Happy Code, and Blob Code Spanning Tree Representations. pages 2111–2118, Vancouver, BC, Canada. IEEE.

[11] Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell Labs Technical Journal*, 36(6):1389–1401.

[12] Thanh, P. D. (2018). CluSPT instances. Mendeley Data v3, http://dx.doi.org/10.17632/b4gcgybvt6.3.

[13] Thanh, P. D., Binh, H. T. T., Long, N. B., et al. (2019). A heuristic based on randomized greedy algorithms for the clustered shortest-path tree problem. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2915–2922. IEEE.

[14] Thanh, P. D., Binh, H. T. T., and Trung, T. B. (2020). An efficient strategy for using multifactorial optimization to solve the clustered shortest path tree problem. *Applied Intelligence*, 50(4):1233–1258.

[15] Thanh, P. D., Dung, D. A., Tien, T. N., and Binh, H. T. T. (2018). An effective representation scheme in multifactorial evolutionary algorithm for solving cluster shortest-path tree problem. In *Evolutionary Computation (CEC), 2018 IEEE Congress on*, pages 811–818. IEEE.

[16] Wu, B. Y. and Chao, K.-M. (2004). *Spanning trees and optimization problems*. Discrete mathematics and its applications. Chapman & Hall/CRC, Boca Raton, FL.

[17] Wu, B. Y. and Lin, C.-W. (2014). Clustered trees with minimum inter-cluster distance. In *Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on*, pages 1138–1141. IEEE.

[18] Wu, B. Y. and Lin, C.-W. (2015). On the clustered Steiner tree problem. *Journal of Combinatorial Optimization*, 30(2):370–386.

[19] Yang, X.-S. (2017). *Nature-Inspired Algorithms and Applied Optimization*. Springer.

IMPROVED MUTATION OPERATOR FOR MULTIFACTORIAL EVOLUTIONARY ALGORITHM TO SOLVE THE CLUSTERED SHORTEST-PATH TREE PROBLEM

Pham Dinh Thanh
Tay Bac University

ABSTRACT

Clustered Shortest-Path Tree Problem (CluSPT) has a wide range of applications in network design, agricultural irrigation and product distribution. Because the CluSPT problem is NP-hard, approximate approaches often use to solve the CluSPT problem in which an approach based on a combination between multifactorial evolution algorithm and randomized greedy algorithm can reach the optimal solution on instances. However, the mutation operator in the approach has a restriction when the number of new edge substitutions is constant. To overcome the limitation, this study proposed a mutation operator which is capable of changing the number of new edge replacements in each execution, as well as the ability to replace many new edges of the individual. To prove the effectiveness of the proposed operator, the algorithm is conducted on various types of instances. Experimental results have shown the efficiency of the proposed operator.

Keywords: *Multifactorial evolution algorithm, Clustered shortest-path tree problem, Combinatorial optimization.*

Ngày nhận bài: 18/10/2020; Ngày nhận đăng: 15/11/2020

Liên hệ: Email-thanhpd@utb.edu.vn