Tạp chí Tin học và Điều khiển học, T.29, S.3 (2013), 285–297

THUẬT TOÁN DI TRUYỀN LAI GHÉP THUẬT TOÁN ĐÀN KIẾN GIẢI BÀI TOÁN CỰC TIỂU HÓA ĐỘ TRỄ

BAN HÀ BẰNG, NGUYỄN ĐỨC NGHĨA

Viện Công nghệ Thông tin và Truyền thông - Đại học Bách khoa Hà Nội
Email: (BangBH, NghiaND) soict.hut.edu.vn

Tóm tắt. Bài toán cực tiểu hóa độ trễ thuộc lớp bài toán tối ưu hóa tổ hợp có nhiều ứng dụng trong thực tiễn. Trong trường hợp tổng quát, bài toán được chứng minh thuộc lớp NP-khó và ngoại trừ P = NP, không tồn tại lược đồ xấp xỉ với thời gian đa thức để giải nó. Bởi vậy, việc phát triển các thuật toán gần đúng là hướng tiếp cận phù hợp để giải bài toán. Trong bài báo này sẽ trình bày một thuật toán meta-heuristic (ACO-GA) lai ghép giữa thuật toán di truyền (GA) và thuật toán đàn kiến (ACO). Thuật toán đàn kiến đóng vai trò khởi tạo quần thể cho thuật toán di truyền. Trong khi đó, thông tin di truyền từ thuật toán di truyền giúp định hướng cá thể kiến chọn đường đi tốt hơn ở lần khởi tạo quần thể kế tiếp. Thêm vào đó, để duy trì tính đa dạng trong quần thể kiến, thuật toán sử dụng ba loại kiến với các đặc tính tìm kiếm đường đi khác nhau. Thực nghiệm thuật toán đã được thực hiện trên các bộ dữ liệu chuẩn. Kết quả thực nghiệm cho thấy thuật toán đưa ra lời giải với cận tỷ lệ tốt hơn so với các thuật toán meta-heuristic hiện biết trên nhiều bộ dữ liệu.

Từ khóa. Bài toán cực tiểu hóa độ trễ-MLP, meta-heuristic, ACO, GA.

Abstract. Minimum Latency Problem is a class of combinatorial optimization problems that have many practical applications. In the general case, the problem is proven to be NP-hard. Therefore, using a meta-heuristic algorithm is a suitable approach for solving this problem. In this paper, we propose a meta-heuristic algorithm which combines Ant Colony (ACO) and Genetic Algorithm (GA). In our algorithm, ACO generates a population for GA. Meanwhile, the genetic information of GA helps ants to create a better population in the next step. In addition, to maintain the diversity of population, our algorithm uses three types of the ants which have different characteristics.We evaluate the algorithm on five benchmark datasets. The experimental results show that our algorithm gives a better solution than the state-of-the-art meta-heuristic algorithms on several instances of datasets.

Key words. Minimum Latency Problem-MLP, meta-heuristic, ACO, GA.

1. GIỚI THIỆU

Bài toán cực tiểu hóa độ trễ (MLP) là một dạng khác của bài toán thợ sửa chữa lưu động (Traveling Repairman Problem). Bài toán MLP có nhiều ứng dụng trong thực tế, chẳng hạn khi một máy chủ hay một người thợ phải lập lịch thực thi một tập các yêu cầu sao cho tổng (trung bình) thời gian chờ đợi của các yêu cầu là cực tiểu [5, 11]. Do có thể quy bài toán MLP tổng quát về bài toán MLP trong trường hợp metric nhờ sử dụng một phép biến đổi đơn giản

(xem [15]), nên trong bài báo này, sẽ chỉ xét bài toán trong trường hợp metric. Bài toán MLP có thể phát biểu như sau: Cho đồ thị đầy đủ K_n với tập đỉnh $V = \{1, 2, ..., n\}$ và ma trận chi phí đối xứng không âm $C = \{c_{ij}|i,\ j = 1, 2, ..., n\}$, với c_{ij} là khoảng cách giữa hai đỉnh i và j. Giả sử $T = v_1, v_2, ..., v_n$ là một hành trình xuất phát từ v_1 (đường đi xuất phát từ v_1 đi qua mỗi đỉnh của đồ thị đúng một lần) trên K_n. Ký hiệu $Path(v_1, v_k)$ là đoạn đường đi từ v_1 đến v_k trên hành trình T. Ta gọi độ trễ của đỉnh v_k trên hành trình T, ký hiệu bởi $lat(v_k)$, là độ dài của đường đi $Path(v_1, v_k)$

$$lat(v_k) = \sum_{i=1}^{k-1} c(v_i, v_{i+1}),\ k = 2, 3, ..., n.$$

Độ trễ của hành trình T, ký hiệu là $L(T)$, được định nghĩa như tổng độ trễ của tất cả các đỉnh thuộc nó

$$L(T) = \sum_{k=2}^{n} lat(v_k).$$

Giả sử v_1 là đỉnh cho trước, bài toán MLP yêu cầu tìm hành trình xuất phát từ v_1 với độ trễ là nhỏ nhất.

Trong trường hợp tổng quát, bài toán được chứng minh thuộc lớp NP-khó, và ngoại trừ $P = NP$, không tồn tại lược đồ xấp xỉ với thời gian đa thức để giải nó [11]. Hiện nay, hàng loạt thuật toán đã được đề xuất để giải bài toán MLP. Trong hướng tiếp cận giải đúng, các thuật toán trong [4, 13] có thể áp dụng giải bài toán MLP với kích thước nhỏ (đồ thị với ít hơn 40 đỉnh). Trong không gian metric, nhiều thuật toán gần đúng cận tỷ lệ đã được đề xuất. Đầu tiên, Blum et al. [5] đưa ra thuật toán với cận tỷ lệ là 144, tiếp đến thuật toán của Goemans et al. [8] giảm cận tỷ lệ xuống còn 21.55. Công trình [2] của Arora et al. đề xuất thuật toán gần đúng với cận tỷ lệ 17.24. Thuật toán gần đúng của Archer et al. [1] có cận tỷ lệ 7.18 trong trường hợp metric và 3.01 trong bộ dữ liệu thực nghiệm TSPLIB. Gần đây, K.Chaudhuri et al. [6] đưa ra thuật toán gần đúng với cận tỷ lệ nhỏ nhất là 3.59. Trong hướng tiếp cận meta-heuristic, chúng tôi đề xuất một thuật toán dựa trên sơ đồ của thuật toán di truyền [3]. Kết quả thực nghiệm chỉ ra rằng chất lượng lời giải đạt được bởi thuật toán là tốt hơn so với các thuật toán gần đúng cận tỷ lệ hiện biết trên bộ dữ liệu thực nghiệm TSPLIB [14]. Sau đó, các thuật toán meta-heuristic được xem xét trong các công trình của A. Salehipour et al. [10], M. Silva et al. [12]. Các kết quả thực nghiệm trong [3, 10, 12] cho thấy meta-heuristic là hướng tiếp cận tiềm năng cho bài toán MLP.

Bài báo đề xuất thuật toán di truyền lai ghép với thuật toán đàn kiến (ACO-GA) để giải bài toán MLP. Trong thuật toán này, thuật toán đàn kiến đóng vai trò khởi tạo quần thể cho thuật toán di truyền. Ngược lại, thuật toán di truyền đóng vai trò định hướng cho đàn kiến ở lần khởi tạo quần thể kế tiếp. Thuật toán sử dụng ba loại kiến. Loại kiến thứ nhất sẽ sử dụng vết mùi và thông tin di truyền để tìm đường đi. Trong khi đó, loại kiến thứ hai chỉ sử dụng vết mùi, còn loại kiến thứ ba thực hiện một cách ngẫu nhiên việc lựa chọn đường đi. Tiến hành thực nghiệm thuật toán ACO-GA trên các bộ dữ liệu chuẩn đã được nhiều tác giả sử dụng. Kết quả thực nghiệm chỉ ra rằng, thuật toán ACO-GA đưa ra được lời giải với chất lượng tốt hơn so với các thuật toán hiện biết trên nhiều bộ dữ liệu.

Cấu trúc của bài báo gồm: Mục 2 trình bày thuật toán đề xuất. Thảo luận về thuật toán đề xuất được đề cập trong Mục 3 và kết quả thực nghiệm được trình bày ở Mục 4. Cuối cùng đưa ra các kết luận.

2. THUẬT TOÁN ĐỀ XUẤT

Thuật toán di truyền áp dụng rất hiệu quả cho lớp bài toán tối ưu hóa tổ hợp. Tuy nhiên, một vấn đề mà các thuật toán di truyền thường gặp là thuật toán hội tụ sớm sau một số vòng lặp, khi quần thể mất đi tính đa dạng. Trong [3], chúng tôi áp dụng kỹ thuật hủy diệt (Social Disaster Technique) nhằm duy trì tính đa dạng của quần thể. Tuy nhiên, quần thể mới cũng không đảm bảo duy trì được tính đa dạng và thuật toán lại rơi vào cực trị địa phương sau một số thế hệ tiếp theo.

Thuật toán đàn kiến [7] mô phỏng hành vi của các đàn kiến trong thực tế. Khi di chuyển các con kiến để lại vết mùi trên đường đi giúp cho các con kiến theo sau lần theo vết mùi đó. Dựa vào nồng độ mùi do các con kiến đi trước để lại, các con kiến theo sau lựa chọn đi theo đường đi có nồng độ mùi cao hơn. Tuy nhiên, một nghiên cứu về hành vi kiến [9] chỉ ra rằng có khoảng 20% số lượng kiến không có khả năng sử dụng vết mùi để tìm đường đi. Mặc dù vậy, chúng lại đóng một vai trò nhất định trong quần thể kiến. Các nhà nghiên cứu đã tiến hành thực nghiệm và thấy rằng loại kiến dựa theo vết mùi có vai trò mang thức ăn từ nguồn thức ăn tìm được về tổ. Tuy nhiên, chúng khó có khả năng tìm được nguồn thức ăn mới. Trong khi đó, loại kiến không có khả năng sử dụng vết mùi lại đóng vai trò như cá thể tìm nguồn thức ăn mới. Thực nghiệm về hành vi đàn kiến cho thấy, quần thể kiến chứa cả hai loại kiến có khả năng tìm kiếm nguồn thức ăn hiệu quả hơn so với quần thể kiến chỉ chứa một loại kiến duy nhất. Trong [13], S. Shimomura cũng đã đề xuất thuật toán đàn kiến với hai loại kiến. Kết quả thực nghiệm cũng chỉ ra rằng, quần thể kiến chứa cả hai loại kiến hiệu quả hơn so với quần thể kiến chỉ chứa một loại kiến.

Bài báo đề xuất thuật toán dựa trên cơ sở lai ghép thuật toán di truyền với thuật toán đàn kiến. Thuật toán đàn kiến được sử dụng để khởi tạo quần thể ban đầu cho thuật toán di truyền. Trong khi đó, thông tin từ thuật toán di truyền đóng vai trò định hướng đường đi cho đàn kiến ở quần thể kế tiếp. Để có được quần thể đa dạng, thuật toán luôn duy trì ba loại kiến. Loại kiến thứ nhất sử dụng vết mùi và thông tin di truyền để tìm đường đi. Trong khi đó, loại kiến thứ hai chỉ sử dụng vết mùi, còn loại kiến thứ ba lựa chọn tìm đường đi một cách ngẫu nhiên. Như vậy, có thể xem loại kiến thứ ba giống như một toán tử đột biến giúp thuật toán thoát khỏi cực trị địa phương. Sơ đồ của thuật toán được trình bày ở Thuật toán 1.

Ta chuyển sang mô tả chi tiết việc thực hiện các thao tác chính của thuật toán ACO-GA.

- Di chuyển của quần thể kiến: Quần thể kiến sẽ bao gồm S_p cá thể kiến, được đánh số bởi 1, 2, ..., S_p. Bắt đầu từ cá thể kiến 1, trong quá trình thực hiện thuật toán các cá thể kiến lần lượt thực hiện di chuyển, cá thể kiến này di chuyển xong mới đến lượt cá thể kiến tiếp theo.

- Mã hóa: Mỗi đường đi mà cá thể kiến tạo ra được mã hóa bởi một danh sách có thứ tự gồm n đỉnh $T = (v_1, v_2, ..., v_i, ..., v_n)$, trong đó v_i là đỉnh được đi qua thứ i trong đường đi. Ta cũng sẽ sử dụng ký hiệu $T[i]$ để chỉ đỉnh thứ i trong danh sách T.

- Khởi tạo vết mùi và thông tin di truyền: Gọi τ_{k-1ij} và g_{ij} là lượng mùi và thông tin di truyền có trên cạnh (v_i, v_j) mà kiến thứ k sử dụng để lựa chọn cạnh di chuyển ($k = 1, 2, ..., S_p$). Đối với quần thể kiến đầu tiên của ACO-GA, τ_{0ij} và g_{ij} được khởi tạo là τ_0 và g_0 tương ứng, còn đối với những quần thể kiến tiếp theo, τ_{0ij} được khởi tạo là tổng lượng mùi mà các cá thể kiến của quần thể kiến ở bước trước để lại, còn g_{ij} là tổng thông tin di truyền từ các cá thể kiến tốt nhất tìm được bởi thuật toán GA ở các bước trước của thuật toán ACO-GA.

- Chọn đỉnh tiếp theo: Giả sử, cá thể kiến thứ k đang ở đỉnh v_i và cần chọn đỉnh kế tiếp từ tập các đỉnh chưa thăm của cá thể kiến đó (ký hiệu tập đỉnh này là N_k) để di chuyển.

Algorithm 1. ACO-GA (K_n, C_{ij}, Sp, τ_0, g_0)

Input: K_n, C_{ij}, Sp, τ_0, g_0 tương ứng là đồ thị, ma trận chi phí, kích thước quần thể, giá trị khởi tạo vết mùi, thông tin di truyền.
Output: Cá thể tốt nhất tìm được G_k.
//Khởi tạo thông tin vết mùi và di truyền trên các cạnh
P=∅;//khởi tạo quần thể rỗng
for (i = 1; $i \le n$; i++)
  for (j = 1; $j \le n$; j++)
    $\tau[i][j] = \tau_0$; $g[i][j] = g_0$;
while (điều kiện dừng của ACO-GA chưa thỏa){k=0;
  while ($k < Sp$)
  { rd = **random**(100);//*rd là số ngẫu nhiên* ∈ (1, 100)
    $T_k = \{v_1\}$; //khởi tạoT_k gồm đỉnh xuất phát v_1
    $v_{next} = v_1$; //khởi tạo đỉnh xuất phát v_{next} cho kiến
    while ($|T_k| \le n$)
    { *Đặt* $N_k = \{v_j \mid v_j \notin T_k\}$
      //Chọn đỉnh kế tiếp nhờ **roulette_Wheel**
      v_{next} =**roulette_Wheel**(v_{next}, rd, N_k);
      $T_k = T_k \cup \{v_1\}$; //bổ sung đỉnh mới vào T_k
    }//end of while ($|T_k| \le n$)
    //Cập nhập thông tin vết mùi theo (2.4), (2.5)
    for (i = 1; $i < n$; i++) {
      $delta\_\tau[T_k[i],T_k[i+1]] = \sigma / L(T_k)$;
      $\tau[T_k[i],T_k[i+1]] = (1-p)*\tau[T_k[i],T_k[i+1]] + delta\_\tau[T_k[i],T_k[i+1]]$;
    }//end of for (i = 1; $i < n$; i++)
    k++;
    $P=P\cup\{T_k\}$;*//bổ sung cá thể kiến T_k vào quần thể P*
  }//end of while ($k < Sp$)
  *//**Các bước của thuật toán di truyền***
  //Quần thể ban đầu P bao gồm các hành trình
  //tạo được bởi các cá thể kiến
  G = cá thể tốt nhất trong quần thể P*;
  while (điều kiện dừng của thuật toán di truyền chưa thỏa)
  { $P_{ad} = \emptyset$; //khởi tạo tập các cá thể con
    for (i = 1; $i \le Sp$; i++) {
      //Lựa chọn cá thể cha TP và cá thể mẹ TM
      (*TP*, *TM*) = **selectionOperator**(*P*);
      //hàm rand(1) *trả lại số ngẫu nhiên* ∈ (0, 1)
      if (**rand**(1) ≤ *Pc*) { //lai ghép cá thể cha mẹ
        TC = **crossoverOperator**(*TP*, *TM*);
        if (**random**(1) ≤ *Pm*) *//đột biến cá thể con*
          TC = **mutationOperator**(*TC*);
        TC=**localSearch**(*TC*); *//tìm kiếm địa phương*
        //cập nhập cá thể tốt nhất:
        if (*L*(*TC*) < *L*(*G**)) *G** = *TC*;
        Bổ sung *TC* vào P_{ad};
      }//end of if (rand(1) ≤ *Pc*)
    }//end of for (i = 1; $i \le Sp$; i++)
    P = tập gồm *Sp* cá thể tốt nhất trong $P \cup P_{ad}$;
    //Cập nhập thông tin di truyền theo (2.6), (2.7)
    for (i = 1; $i < n$; i++) {
      $delta\_g[G^*[i],G^*[i+1]] = \sigma / L(G^*)$;
      $g[G^*[i],G^*[i+1]] = g[G^*[i],G^*[i+1]] + delta\_g[G^*[i],G^*[i+1]]$;
    } //end of for (i = 1; $i < n$; i++)
  }//end of while GA
}//end of while ACO-GA
return (cá thể tốt nhất tìm được bởi ACO-GA)

Algorithm 2. Function roulette_Wheel (v_i, rd, N_k)

Input: v_i, rd, N_k tương ứng là đỉnh hiện tại, giá trị số trong khoảng 1 đến 100, tập các đỉnh chưa thăm trong T_k.
Output: Đỉnh v được chọn đi kế tiếp v_i.
sum_Prob = 0;
if ($rd \le 50$) {
  //loại kiến sử dụng vết mùi và thông tin di truyền
  for $v_j \in N_k$
    $sum\_Prob = sum\_Prob + pow(\tau[i,j], \beta_\tau) * pow(\eta[i,j], \beta_\mu) * pow(g[i,j], \beta_g)$;
  //lựa chọn đỉnh kế tiếp theo (2.1)
  for $v_j \in N_k$
    $p[i,j] = pow(\tau[i,j], \beta_\tau) * pow(\eta[i,j], \beta_\mu) * pow(g[i,j], \beta_g) / sum\_Prob$;
  rd = **random**(*sum_Prob*); //0 < *rd* < *sum_Prob*
  for $v_j \in N_k$ {
    sum_ wheel = *sum_wheel* + *p*[*i*,*j*];
    if (*sum_wheel* > *rd*) **break**;
  }//end for $v_j \in N_k$
  $v = v_j$;
}//end if ($rd \le 50$)
if (50 < $rd \le 80$) {
  //loại kiến chỉ sử dụng vết mùi
  for $v_j \in N_k$
    $sum\_Prob = sum\_Prob + pow(\tau[i,j], \beta_\tau) * pow(\eta[i,j], \beta_\mu)$;
  //chọn đỉnh kế tiếp theo (2.2)
  for $v_j \in N_k$
    $p[i,j] = pow(\tau[i,j],\beta_\tau) * pow(\eta[i,j], \beta_\mu) / sum\_Prob$;
  rd = **random**(*sum_Prob*);
  for $v_j \in N_k$ {
    sum_wheel = *sum_wheel* + *p*[*i*,*j*];
    if (*sum_wheel* > *rd*) **break**;
  }//end for $v_j \in N_k$
  $v = v_j$;
}//end if (50 < $rd \le 80$)
if (rd > 80)*//loại kiến ngẫu nhiên*
  //chọn đỉnh kế tiếp theo (2.3)
  chọn ngẫu nhiên $v \in N_k$;
return *v*;

Algorithm 3. Function selectionOperator(*P*)

Input: Quần thể $P = \{p[1], p[2], \ldots, p[Sp]\}$
Output: Cá thể cha *TP* và cá thể mẹ *TM*.
//chọn một nhóm cá thể ngẫu nhiên từ quần thể *P*
IG=∅; // Khởi tạo
for (i = 0; i < *NG*; i++) {
  i = **random**(*Sp*);
  IG= *IG*∪{*p*[*i*]};
}//end of for (i = 0; i < *NG*; i++)
Sắp xếp các cá thể trong nhóm *IG* tăng dần theo độ trễ;
//Chọn hai cá thể với độ trễ nhỏ nhất làm cá thể cha và mẹ
TP = *IG*[0];
TM = *IG*[1];
return *TP*, *TM*;

Thuật toán ACO-GA duy trì ba loại kiến. Loại kiến đầu tiên sử dụng thông tin về vết mùi và thông tin về di truyền để lựa chọn đỉnh kế tiếp. Nếu cá thể kiến thứ k thuộc loại này, nó sẽ

di chuyển từ đỉnh v_i đến đỉnh v_j ($v_j \in N_k$) với xác suất:

$$p^1_{kij} = \frac{[\tau_{k-1ij}]^{\beta_\tau}[\eta_{kij}]^{\beta_\eta}[g_{ij}]^{\beta_g}}{\sum\limits_{v_j \in N_k}[\tau_{k-1ij}]^{\beta_\tau}[\eta_{kij}]^{\beta_\eta}[g_{ij}]^{\beta_g}}. \tag{2.1}$$

Loại kiến thứ hai chỉ sử dụng thông tin về vết mùi để lựa chọn đỉnh kế tiếp. Nếu cá thể kiến thứ k thuộc loại hai, thì nó sẽ di chuyển từ đỉnh v_i đến đỉnh v_j với xác suất là:

$$p^2_{kij} = \frac{[\tau_{k-1ij}]^{\beta_\tau}[\eta_{kij}]^{\beta_\eta}}{\sum\limits_{v_j \in N_k}[\tau_{k-1ij}]^{\beta_\tau}[\eta_{kij}]^{\beta_\eta}}. \tag{2.2}$$

Cuối cùng, loại kiến thứ ba không sử dụng bất cứ thông tin nào về mùi, cũng như thông tin di truyền. Nếu cá thể kiến thứ k thuộc loại này thì xác suất để nó di chuyển từ đỉnh v_i đến đỉnh v_j là:

$$p^3_{kij} = \frac{1}{|N_k|}. \tag{2.3}$$

Ở đây β_τ, β_μ và β_g là các tham số, $\eta_{kij} = 1/(\rho c_{ij})$ (ρ là vị trí của cạnh (v_i, v_j) trong đường đi của cá thể kiến thứ k). Công thức (2.1) và (2.2) được cải biên từ công thức có trong thuật toán ACO chuẩn của Marco Dorigo [7].

Số lượng kiến loại một, hai và ba sẽ được xác định một cách ngẫu nhiên với tỷ lệ tương ứng là 50%, 30% và 20% (theo nhận xét đã nêu ở trên, ta giữ 20% số lượng kiến không có khả năng sử dụng vết mùi để tìm đường đi). Để thực hiện điều này, khi cần xác định cá thể kiến hiện tại thuộc loại nào trong ba loại trên, ta gieo một số ngẫu nhiên rd trong khoảng từ 1 đến 100. Nếu $rd \leq 50$ thì cá thể kiến đó gán với loại đầu tiên. Nếu $50 < rd \leq 80$, thì cá thể kiến đó gán với loại thứ hai và cuối cùng nếu $rd > 80$, thì cá thể kiến đó gán với loại thứ ba. Sau khi xác định loại cho cá thể kiến, việc lựa chọn đỉnh kế tiếp được thực hiện theo phương pháp bánh xe roulette, trong đó đỉnh kế tiếp của cá thể kiến được lựa chọn ngẫu nhiên theo xác suất: Đỉnh nào có xác suất cao hơn sẽ có khả năng được chọn cao hơn, nhưng không có nghĩa là các đỉnh có xác suất thấp hơn không bao giờ được chọn mà nó được chọn với cơ hội thấp hơn. Chi tiết việc lựa chọn đỉnh kế tiếp được trình bày trong Thuật toán 2.

- Cập nhập thông tin vết mùi: Các cá thể kiến lần lượt (con kiến này đi xong mới đến lượt con kiến khác) thực hiện việc tìm đường đi và khi di chuyển theo đường đi được chọn sẽ để lại vết mùi trên các cạnh đã đi qua. Giả sử cá thể kiến thứ k thực hiện việc di chuyển theo đường đi T_k, khi đó lượng vết mùi mà nó để lại trên các cạnh đi qua được tính bởi công thức

$$\Delta\tau_{kij} = \begin{cases} \sigma/L(T_k), \text{ nếu } (v_i, v_j) \in T_k, \\ 0, \text{ nếu trái lại} \end{cases} \tag{2.4}$$

trong đó σ là tham số. Khi đó, sau khi cá thể kiến thứ k đã di chuyển, tổng lượng vết mùi τ_{kij} có trên mỗi cạnh (v_i, v_j) được cập nhật theo $\Delta\tau_{kij}$:

$$\tau_{kij} = (1-p)\tau_{k-1ij} + \Delta\tau_{kij}, \tag{2.5}$$

trong đó tham số $p \in (0, 1)$ là xác suất bay hơi của vết mùi.

- Khởi tạo quần thể ban đầu cho thuật toán di truyền: Mỗi đường đi được tạo bởi một cá thể kiến được xem như một cá thể trong quần thể ban đầu cho thuật toán di truyền. Như vậy, với S_p cá thể kiến thì quần thể ban đầu P sẽ có S_p cá thể đường đi.

- Hàm ước lượng: Hàm ước lượng tính tổng độ trễ cho mỗi cá thể đường đi. Tổng độ trễ càng nhỏ thì độ thích nghi của cá thể đó càng cao. Nói một cách khác, tổng độ trễ tỷ lệ nghịch với độ thích nghi của cá thể.

- Toán tử lựa chọn: Một nhóm gồm các cá thể với kích thước cho trước (ký hiệu thông số này là NG) được lựa chọn ngẫu nhiên từ quần thể. Sau đó, hai cá thể có độ trễ nhỏ nhất trong nhóm được lựa chọn làm cá thể cha mẹ. Toán tử lựa chọn được mô tả chi tiết trong Thuật toán 3.

- Toán tử lai ghép: Toán tử lai ghép thực hiện việc lai ghép các cá thể cha TP và cá thể mẹ TM với một xác suất lai ghép (Pc) cho trước. Trong [3], chúng tôi đã đề xuất toán tử lai ghép cho bài toán MLP. Trong bài báo này sẽ sử dụng lại toán tử lai ghép này. Tuy nhiên, thay vì chỉ lựa chọn đỉnh bổ sung vào cá thể con là đỉnh đi ngay trước hoặc đỉnh đi sát sau vị trí đỉnh hiện tại, ta lựa chọn đỉnh bổ sung trong dãy gồm l đỉnh đi trước và l đỉnh đi sau vị trí đỉnh hiện tại, trong đó l là thông số chọn trước. Đỉnh được lựa chọn bổ sung là đỉnh trong dãy nói trên mà việc bổ sung nó cho ta cá thể con với độ trễ nhỏ nhất. Việc thực hiện toán tử lai ghép được trình bày trong Thuật toán 4.

- Toán tử đột biến: Sau khi thu được cá thể con từ bước lai ghép, ta tiến hành đột biến cá thể này với xác suất đột biến (P_m) cho trước. Mục đích của toán tử đột biến là tạo sự đa dạng cho quần thể. Ta sử dụng toán tử đột biến đơn giản sau đây: Chọn ngẫu nhiên hai đỉnh trong cá thể và thực hiện việc hoán đổi chúng. Việc thực hiện toán tử đột biến được mô tả trong Thuật toán 5.

- Tìm kiếm địa phương: Trong bước này, ta kết hợp thuật toán di truyền với thuật toán tìm kiếm địa phương. Sau bước đột biến, ta thực hiện tìm kiếm địa phương xuất phát từ cá thể con cháu theo các thuật toán tìm kiếm địa phương swap-adjacent, swap, 2-opt-edge, và 2-opt-node (xem [12]). Để giảm độ phức tạp tính toán của các thuật toán, ta sẽ thu hẹp lân cận tìm kiếm bằng cách cố định ngẫu nhiên một đỉnh và chỉ thực hiện tìm kiếm trong lân cận gồm các lời giải thu được nhờ phép biến đổi liên quan đến đỉnh này. Nhờ đó, độ phức tạp tính toán của các thuật toán tìm kiếm địa phương chỉ là $O(n)$. Mô tả chi tiết của thuật toán được trình bày trong Thuật toán 6.

Algorithm 4. Function crossoverOperator(*TP*, *TM*)

Input: Cá thể cha *TP* và cá thể mẹ *TM*.
Output: Cá thể con *TC*.

```
TC[1] = v1; TC[2] = TM[2]; jend = 2;
for (i = 3; i < n; i++)
{   Tìm q sao cho TP[q] = TC[jend];
    //chọn đỉnh kế tiếp để cá thể TC có độ trễ nhỏ nhất
    minL = +∞; TC_min = ∅;
    for (j = -l; j < l; j++) // duyệt dãy đỉnh TP[i-l .. i+l]
        if (i+j> 0) &&(i+j ≤ n) && (TP[i+l] ∉ TC)
        {   Bổ sung TP[i+l] vào TC;
            if (L(TC) < minL)
                TC_min = TC; minL = L(TC);
            TC = TC \ TP[i+l];
        }//end of if (i+j > 0) ...
    //nếu mọi đỉnh trong TP[i–l ... i+l] đều có trong TC
    if (minL = +∞)
    {   chọn ngẫu nhiên một đỉnh v ∈ {u | u ∉ TC};
        TC = TC ∪ {v};
    }//end of if (minL = +∞)
     Hoán đổi vai trò của TM và TP;
}//end of for (i = 3; i < n; i++)
return T;
```

Algorithm 5. Function mutationOperator(*T*)

Input: Cá thể *T*.
Output: Cá thể sau khi đột biến *T′*.

```
//chọn ngẫu nhiên hai vị trí i và j trong cá thể T
i = random(n);
j = random(n);
Hoán đổi hai vị trí i, j trong cá thể T để thu được T′
return T′;
```

Algorithm 6. Function localSearch(*T*)

Input: Cá thể ban đầu *T*.
Output: Cá thể sau tìm kiếm địa phương *T′*.

```
//chọn ngẫu nhiên vị trí của một đỉnh trong T
i = random(n);
//Thực hiện lần lượt các thuật toán tìm kiếm địa phương
//bắt đầu từ cá thể T với đỉnh cố định ở vị trí i;
T′ = swap-adjacent (swap, 2-opt-edge, 2-opt-node) (T, i);
return T′;
```

- Điều kiện dừng của thuật toán di truyền: Ta sẽ dừng thuật toán nếu như sau một số lượng thế hệ định trước (ký hiệu số này là N_p) thuật toán không tìm được lời giải tốt hơn.

- Cập nhập thông tin di truyền: Gọi G^* là cá thể đường đi tốt nhất tìm được bởi thuật toán di truyền. Khi đó, thông tin di truyền Δg^*_{ij} của cá thể này được tính bởi công thức sau

$$\Delta g^*_{ij} = \begin{cases} \sigma/L(G^*), \text{ nếu } (v_i, v_j) \in G^*, \\ 0, \text{ nếu trái lại.} \end{cases} \tag{2.6}$$

Thông tin di truyền g_{ij} trên mỗi cạnh (v_i, v_j) được khởi tạo cho quần thể kiến ở bước lặp kế tiếp của ACO-GA được cập nhập theo công thức

$$g_{ij} := g_{ij} + \Delta g^*_{ij}. \tag{2.7}$$

- Điều kiện dừng của thuật toán ACO-GA: Thuật toán ACO-GA dừng nếu như sau m vòng lặp (m là thông số xác định trước) lời giải tốt nhất không được cải thiện. Khi đó, hành trình tốt nhất tìm được trong quá trình thực hiện thuật toán sẽ được đưa ra như lời giải cần tìm.

Bây giờ, ta đánh giá độ phức tạp tính toán của thuật toán ACO-GA. Một vòng lặp của ACO-GA đòi hỏi thực hiện các công việc sau: 1) Việc khởi tạo thông tin vết mùi và di truyền đòi hỏi thời gian $O(n^2)$; 2) Việc xây dựng quần thể kiến P đòi hỏi thời gian $O(|N_k|+n)$ (trong đó xây dựng từng cá thể kiến theo phương pháp roulette_Wheel mất thời gian $O(|N_k|)$ và cập nhập thông tin vết mùi từng cá thể kiến mất thời gian $O(n)$, lưu ý là $|N_k| < n$); 3) Việc thực hiện thuật toán di truyền đòi hỏi thời gian $O(n^2)$ (bởi vì, thời gian đòi hỏi bởi việc thực hiện toán tử lựa chọn là $O(NG^2)$ ($NG < n$), toán tử lai ghép: $O(n^2)$, toán tử đột biến: $O(1)$, thuật toán tìm kiếm địa phương: $O(n)$); 4) cập nhập thông di truyền: $O(n)$. Giả sử, thuật toán di truyền kết thúc sau k_1 vòng lặp, khi đó một vòng lặp của ACO-GA đòi hỏi thời gian $O(k_1 \times n^2 \times SP)$. Nếu ACO-GA kết thúc sau k_2 vòng lặp, thì độ phức tạp tính toán của ACO-GA là $O(k_1 \times k_2 \times SP \times n^2)$.

3. THẢO LUẬN

Đối với một bài toán NP-khó như bài toán MLP, hiện tại có ba hướng tiếp cận chính để phát triển thuật toán giải: 1) hướng tiếp cận đúng, 2) hướng tiếp cận gần đúng, 3) hướng tiếp cận meta-heuristic. Đầu tiên, các thuật toán đúng tìm lời giải tối ưu có đánh giá độ phức tạp tính toán bùng nổ tổ hợp: $O(n!)$ trong tình huống tồi nhất, nhưng trong thực tế chúng thường chạy nhanh hơn đánh giá lý thuyết này. Mặc dù vậy, các thuật toán dạng này (chẳng hạn, xem [4, 15]) cũng chỉ giải được bài toán MLP với kích thước nhỏ (đồ thị với ít hơn 40 đỉnh). Trong cách tiếp cận thứ hai, các thuật toán gần đúng với cận tỷ lệ α có ưu điểm lớn là về mặt lý thuyết chúng đảm bảo đưa ra lời giải có chi phí không lớn hơn α lần chi phí của lời giải tối ưu [1, 2, 5, 6, 8]. Tuy nhiên, cận tỷ lệ tốt nhất đạt được hiện nay là 3.59 còn là quá lớn và chưa đáp ứng được yêu cầu của thực tế ứng dụng. Cuối cùng, các thuật toán meta-heuristic là những thuật toán có độ phức tạp tính toán thường là không quá lớn, nhưng chất lượng lời giải tìm được bởi các thuật toán này chỉ có thể đánh giá thông qua thực nghiệm. Trong hướng tiếp cận meta-heuristic giải bài toán MLP, các kết quả thực nghiệm cho thấy các thuật toán được đề xuất trong [3, 10, 12] đều đưa ra lời giải tốt hơn rất nhiều so với lời giải tìm được bởi các thuật toán gần đúng với cận tỷ lệ.

Trong hướng tiếp cận meta-heuristic, các thuật toán trong [10, 12] xuất phát từ một lời giải ban đầu (hay một điểm trong không gian lời giải) và dựa trên thuật toán đa láng giềng (Variable Neighborhood Search-VNS) để tìm lời giải tốt hơn cho bài toán MLP. Tuy nhiên, khi bài toán tối ưu có không gian lời giải lớn, thì hướng tiếp cận đơn lời giải thường chỉ khai thác một phần không gian lời giải của bài toán. Do đó, các thuật toán này dễ rơi vào cực trị địa phương. Để khắc phục nhược điểm của cách tiếp cận đơn lời giải, thuật toán theo hướng tiếp cận đa lời giải có thể được sử dụng. Các thuật toán này có thể đồng thời tìm kiếm từ nhiều điểm trong không gian lời giải của bài toán. Bởi vậy, không gian tìm kiếm lời giải của các thuật toán này được mở rộng hơn. Kết quả là, chúng có nhiều cơ hội thoát khỏi cực trị địa phương hơn so với các thuật toán đơn lời giải. Thuật toán di truyền trong [3] và thuật toán đề xuất trong bài báo là thuật toán theo hướng tiếp cận đa lời giải, trong đó, mỗi cá thể đóng vai trò như một lời giải. Tuy nhiên, thuật toán di truyền trong [3] gặp vấn đề hội tụ sớm do mất đi sự đa dạng của quần thể sau khi thực hiện một số vòng lặp. Đó là lý do chính dẫn đến chất lượng lời giải thu được bởi thuật toán là chưa cao. Để khắc phục tình trạng này, chúng tôi đề xuất thuật toán kế thừa ưu điểm của thuật toán di truyền và thuật toán đàn kiến để cân bằng giữa hai đặc tính khai phá (exploration) không gian tìm kiếm lời giải mới và khai thác (exploitation) không gian lời giải đã có. Cụ thể, thuật toán đàn kiến đóng vai trò khởi tạo quần thể đa dạng cho thuật toán di truyền bằng việc duy trì ba loại kiến như đã trình bày. Ngược lại, thông tin di truyền từ thuật toán di truyền đóng vai trò định hướng cho đàn kiến đến không gian lời giải tiềm năng trong lần khởi tạo quần thể kế tiếp. Tuy nhiên, các thuật toán theo hướng tiếp cận đa lời giải có khối lượng tính toán lớn khi chúng phải tính toán đồng thời trên nhiều lời giải. Do đó, thời gian chạy của các thuật toán này thường lớn hơn so với các thuật toán đơn lời giải.

4. KẾT QUẢ THỰC NGHIỆM

Thuật toán đề xuất được cài đặt trên C++ và thực nghiệm được tiến hành trên máy tính cá nhân với bộ xử lý Intel Pentium core 2 duo 2.13Ghz và 4GB bộ nhớ. Dữ liệu thực nghiệm bao gồm bộ dữ liệu TSPLIB [16] và ba bộ dữ liệu ngẫu nhiên. Tất cả các bộ dữ liệu đều thỏa mãn điều kiện metric. Bộ dữ liệu TSPLIB là một thư mục các file dữ liệu. Mỗi file lưu trữ tọa độ của các đỉnh. Bộ dữ liệu này được sử dụng trong nhiều công trình nghiên cứu liên quan [1, 3, 10, 12]. Trong ba bộ dữ liệu ngẫu nhiên, bộ dữ liệu đầu tiên là do A. Salehipour et al. [10] cung cấp. Bộ dữ liệu này được A. Salehipour et al. và được M. Silva et al. sử dụng trong các thực nghiệm được trình bày trong các công trình [10, 12]. Để đánh giá chính xác hiệu quả của thuật toán, ta xét thêm hai bộ dữ liệu ngẫu nhiên với kích thước nhỏ mà lời giải tối ưu của chúng tìm được nhờ sử dụng thuật toán giải đúng [4].

Để đánh giá hiệu quả của thuật toán, hai thực nghiệm được tiến hành. Dữ liệu trong thực nghiệm đầu tiên bao gồm bộ dữ liệu TSPLIB và các bộ dữ liệu ngẫu nhiên của A. Salehipour et al. Trong thực nghiệm này, chúng ta so sánh kết quả thu được từ thuật toán ACO-GA với các kết quả từ các công trình nghiên cứu liên quan [3, 10, 12]. Do lời giải tối ưu của các bộ dữ liệu này chưa được biết (ngoại trừ lời giải tối ưu cho bộ dữ liệu ngẫu nhiên của A. Salehipour et al. khi $n = 50$ được lấy từ [10]), cho nên hiệu quả của các thuật toán chỉ được đánh giá một cách tương đối. Thực nghiệm thứ hai được thực thi trên các bộ dữ liệu ngẫu nhiên mà lời giải tối ưu của chúng tìm được nhờ sử dụng thuật toán giải đúng trong [4]. Khi biết lời giải tối ưu, ta có thể đánh giá một cách chính xác hiệu quả của thuật toán đề xuất. Trong các thực nghiệm, ta lựa chọn giá trị cho các tham số trong thuật toán ACO-GA như sau: $m = 10,\ S_p = 300,\ NG = 5,\ Pc = 0.7,\ Pm = 0.2,\ l = 5,\ \beta_\tau = 1,\ \beta_\mu = 5,\ \beta_g = 5,\ \tau =$

$10,\ \tau_0 = 10,\ g_0 = 1,\ p = 0.3$ và $Np = 20$.

Bảng 1. Kết quả thực nghiệm các thuật toán cho bộ dữ liệu của A. Salehipour et al. ($n = 50$)

File dữ liệu	*OPT*	M. Silva et al.	GA			ACO–GA			
		Best sol	Best sol	Aver Sol	$\overline{T}$	Best sol	Aver Sol	Gap[%]	$\overline{T}$
TRP-50-R1	12198	12198	12709	12709	3.0	**12198**	12198	0.00	0.50
TRP-50-R2	11621	11621	12845	12921	3.2	**11621**	11674	0.00	0.52
TRP-50-R3	12139	12139	12904	12904	3.5	**12139**	12139	0.00	0.51
TRP-50-R4	13071	13071	13925	13925	3.6	**13071**	13071	0.00	0.48
TRP-50-R5	12126	12126	12726	12726	2.9	**12126**	12284	0.00	0.47
TRP-50-R6	12684	12684	13245	13245	3.1	**12684**	12684	0.00	0.50
TRP-50-R7	11176	11176	11991	11991	3.2	**11176**	11176	0.00	0.51
TRP-50-R8	12910	12910	13558	13754	3.1	**12910**	12945	0.00	0.52
TRP-50-R9	13149	13149	13845	13845	3.3	**13149**	13149	0.00	0.48
TRP-50-R10	12892	12892	13740	13740	2.9	**12892**	12892	0.00	0.47
TRP-50-R11	12103	12103	12565	12565	2.8	**12103**	12181	0.00	0.45
TRP-50-R12	10633	10633	11657	11657	2.9	**10633**	10633	0.00	0.47
TRP-50-R13	12115	12115	12870	12870	3.0	**12115**	12115	0.00	0.40
TRP-50-R14	13117	13117	13659	13721	3.1	**13117**	13117	0.00	0.50
TRP-50-R15	11986	11986	12793	12793	3.2	**11986**	11986	0.00	0.51
TRP-50-R16	12138	12138	12803	12803	3.3	**12138**	12138	0.00	0.43
TRP-50-R17	12176	12176	13387	13387	3.4	**12176**	12176	0.00	0.42
TRP-50-R18	13357	13357	13988	13988	3.5	**13357**	13357	0.00	0.44
TRP-50-R19	11430	11430	11911	12012	3.0	**11430**	11430	0.00	0.47
TRP-50-R20	11935	11935	12409	12409	3.0	**11935**	11935	0.00	0.42
Trung bình					3.1			0.00	0.47

Bảng 2. Kết quả thực nghiệm các thuật toán cho bộ dữ liệu của A. Salehipour et al. ($n = 100$)

File dữ liệu	A.Salehipour et al.	M. Silva et al.	GA			ACO–GA			
	Best Sol	Best Sol	Best Sol	Aver Sol	$\overline{T}$	Best Sol	Aver Sol	Gap[%]	$\overline{T}$
TRP-100-R1	35334	32779	35334	35334	6.2	**32779**	32779	0.00	1.12
TRP-100-R2	38442	33435	36773	36773	6.4	**33435**	33435	0.00	1.21
TRP-100-R3	37642	32390	35150	35320	7.1	**32390**	32390	0.00	0.84
TRP-100-R4	37508	34733	37508	37751	7.3	**34733**	34733	0.00	0.92
TRP-100-R5	37215	32598	35254	35254	6.7	**32598**	32598	0.00	1.04
TRP-100-R6	40422	34159	36988	36988	6.5	34212	34212	0.16	1.25
TRP-100-R7	37367	33375	36897	36941	6.3	**33375**	33375	0.00	0.81
TRP-100-R8	38086	31780	35408	35510	6.4	**31780**	31780	0.00	0.91
TRP-100-R9	36000	34167	36415	36574	6.8	**34167**	34167	0.00	0.72
TRP-100-R10	37761	31605	34018	34018	7.0	31812	31812	0.65	0.84
TRP-100-R11	37220	34188	37006	37006	7.1	**34188**	35534	0.00	0.85
TRP-100-R12	34785	32146	35437	35437	6.5	32899	32899	2.34	0.77
TRP-100-R13	37863	32604	35239	35512	6.7	**31989**	**31989**	-1.89	0.81
TRP-100-R14	36362	32433	34869	34869	6.8	33601	33601	3.60	0.92
TRP-100-R15	39381	32574	35640	35640	7.0	**32574**	32574	0.00	0.81
TRP-100-R16	39823	33566	37083	37083	6.9	34818	34818	3.73	0.85
TRP-100-R17	41824	34198	37058	37421	7.1	36297	36297	6.14	0.91
TRP-100-R18	39091	31929	35109	35109	7.2	**31929**	31929	0.00	0.84
TRP-100-R19	39941	33463	36457	36457	6.4	**33463**	33463	0.00	0.87
TRP-100-R20	39888	33632	35998	35998	6.3	**33632**	33632	0.00	0.82
Trung bình					6.5			0.74	0.91

4.1. Thực nghiệm 1

Trong thực nghiệm này, thuật toán ACO-GA thực thi trên bộ dữ liệu TSPLIB và bộ dữ liệu ngẫu nhiên của A. Salehipour et al. [10]. Mỗi bộ dữ liệu bao gồm các file dữ liệu thực nghiệm. Mỗi file lưu trữ tọa độ của các đỉnh. Ta thực thi thuật toán 10 lần trên mỗi file dữ liệu. Kết quả thực nghiệm được trình bày trong các Bảng từ 1 đến 3. Trong các bảng này, cột Best Sol, Aver Sol lần lượt là kết quả tốt nhất và kết quả trung bình thu được sau 10 lần chạy các thuật toán. Cột gap[%] ghi nhận giá trị $gap[\%] = \dfrac{(UB_2 - UB_1)}{UB_1}100\%$, trong đó UB_2 và UB_1 lần lượt là lời giải tốt nhất thu được từ thuật toán ACO-GA và thuật toán của M. Silva et al. Cột $\bar{T}$ là thời gian chạy trung bình của thuật toán tính bằng phút. Trong Bảng 1, cột OPT tương ứng với giá trị tối ưu. Trong Bảng 3, kết quả thực nghiệm thuật toán của Archer et al. được lấy từ [1]. Dấu "-" nghĩa là không có kết quả thực nghiệm với thuật toán tương ứng.

Kết quả thực nghiệm trình bày trong các Bảng từ 1 đến 3 chứng tỏ thuật toán ACO-GA hiệu quả hơn thuật toán di truyền GA [3] cả về chất lượng lời giải và thời gian chạy. So với lời giải tối ưu có được trong Bảng 1, thuật toán ACO-GA đưa ra được lời giải tối ưu cho tất cả các file dữ liệu. Kết quả thực nghiệm cũng cho thấy, thuật toán ACO-GA cho lời giải tốt hơn thuật toán của Archer et al. và A. Salehipour et al. ở tất cả các file dữ liệu. Trong các bộ dữ liệu ngẫu nhiên ACO-GA cho lời giải với chất lượng rất sát với chất lượng lời giải được đưa ra bởi thuật toán của M. Silva et al. Cụ thể, $gap[\%]$ trung bình trong Bảng 1 và 2 lần lượt là 0.00% và 0.74%. Đặc biệt, đối với file dữ liệu TPR-S100-13, thuật toán ACO-GA đưa ra lời giải tốt hơn so với thuật toán của M.Silva et al. Đối với các bộ dữ liệu trong TSPLIB, kết quả thực nghiệm trong Bảng 3 cho thấy thuật toán ACO-GA đưa ra lời giải với chất lượng

Bảng 3. Kết quả thực nghiệm các thuật toán cho bộ dữ liệu TSPLIB

File dữ liệu	Archer et al.	A.Salehipour et al.	M. Silva et al.	GA		ACO-GA	
	Best Sol	Best Sol	Best Sol	Best Sol	$\bar{T}$	Best Sol	$\bar{T}$
st70	26384	19553	19215	19893	4.20	19215	0.61
rat195	280900	213371	210191	245412	9.10	210191	4.30
kroD100	1297932	976830	949594	976830	6.00	**948325**	1.50
pr226	10421449	7226554	7100308	7212541	10.1	7100308	4.12
lin105	780662	585823	585823	598375	6.15	585823	1.56
pr107	2205490	1983475	1980767	21750029	6.17	**1973726**	1.62
lin318	7475822	5876537	5560679	5876537	13.2	5560679	9.25
pr439	24126010	18567170	17688561	18567170	20.0	17688561	13.1
att532	-	18448435	5581240	18448435	23.0	18448435	18.5
rat99	75048	56994	54984	58327	5.5	57896	1.41
eil101	38582	-	-	30235	6.0	29123	1.48
eil76	26128	-	-	18243	4.3	18023	0.65
eil51	14683	-	-	10281	3.0	9952	1.51
kroE100	1345314	-	-	984759	5.8	955272	1.41
kroA200	3387616	-	-	2962453	8.7	2845318	4.12
kroB200	3731218	-	-	3000409	8.5	2939494	4.23
rd100	458419	-	-	370645	5.7	355985	1.43
tsp225	537080	-	-	463666	9.4	454714	4.62
d198	1380470	-	-	1309819	9.1	1204997	4.12
u159	3837650	-	-	3169661	6.7	2836575	3.25
bier127	5929120	-	-	5011031	7.8	4768611	1.81
gil262	393641	-	-	327534	11.5	325679	4.23

tốt hơn so với các thuật toán còn lại ở hầu hết các file dữ liệu (ngoại trừ đối với hai file dữ liệu att532 và rat99, thuật toán đưa ra lời giải với chất lượng kém hơn so với thuật toán của M. Silva et al.). Tuy nhiên, thời gian chạy của thuật toán ACO-GA lớn hơn thời gian chạy của thuật toán M.Silva et al.

Bảng 4. Kết quả thực nghiệm các thuật toán cho các bộ dữ liệu ngẫu nhiên (n = 40)

File dữ liệu	*OPT*	GA				ACO–GA			
		Best Sol	**Aver Sol**	**Gap[%]**	$\bar{T}$	**Best Sol**	**Aver Sol**	**Gap[%]**	$\bar{T}$
eil51	6140	**6140**	6140	0.00	2.5	**6140**	6140	0.00	0.30
eil76	5894	**5894**	5894	0.00	2.6	**5894**	5894	0.00	0.25
st70	7801	**7801**	7801	0.00	2.5	**7801**	7801	0.00	0.24
rat195	18058	18265	18285	1.15	2.7	**18058**	18058	0.00	0.21
kroA100	239680	241012	241741	0.56	2.6	**239680**	239854	0.00	0.20
kroB100	245999	247541	248041	0.63	2.5	**245999**	24612	0.00	0.21
kroC100	1122890	1145124	1147815	1.98	2.8	**1122890**	1122890	0.00	0.22
berlin52	1372572	**1372572**	1372952	0.00	2.9	**1372572**	1372854	0.00	0.24
pr76	1122891	1123541	1123654	0.06	2.7	**1122891**	1122301	0.00	0.21
tsp225	27523	27523	27618	0.00	2.8	**27523**	27523	0.00	0.22
tss225	1138068	1138068	1138230	0.00	2.7	**1138068**	1138068	0.00	0.24
lin105	140450	**140450**	1405412	0.00	2.8	**140450**	140450	0.00	0.21
test 1	8105	**8105**	8105	0.00	2.7	**8105**	8105	0.00	0.24
test 2	9248	**9248**	9248	0.00	2.8	**9248**	9248	0.00	0.22
test 3	8584	8630	8658	0.54	2.7	**8584**	8584	0.00	0.25
test 4	9240	9263	9267	0.25	2.6	**9240**	9278	0.00	0.24
test 5	8097	**8097**	8097	0.00	2.7	**8097**	8138	0.00	0.23
test 6	9490	9506	9519	0.17	2.8	**9490**	9490	0.00	0.24
test 7	8450	8517	8564	0.79	2.7	**8450**	8450	0.00	0.25
test 8	8426	**8426**	8426	0.00	3.0	**8426**	8426	0.00	0.25
test 9	8987	9018	9023	0.34	2.7	**8987**	8987	0.00	0.24
test 10	9705	9713	9719	0.08	2.8	**9705**	9705	0.00	0.23
test 11	9526	**9526**	9556	0.00	2.7	**9526**	9545	0.00	0.22
test 12	8827	8838	8872	0.12	2.5	**8827**	8827	0.00	0.21
test 13	9440	9450	9475	0.11	2.6	**9440**	9440	0.00	0.20
test 14	8539	**8539**	8539	0.00	2.7	**8539**	8562	0.00	0.27
test 15	8321	**8321**	8321	0.00	2.7	**8321**	8321	0.00	0.24
test 16	8327	8434	8457	1.28	2.6	**8327**	8327	0.00	0.26
test 17	8300	8302	8302	0.02	2.5	**8300**	8300	0.00	0.24
test 18	9520	**9520**	9520	0.00	2.7	**9520**	9520	0.00	0.22
test 19	9759	9769	9792	0.10	2.8	**9759**	9759	0.00	0.21
test 20	8782	**8785**	8813	0.03	2.7	**8782**	8782	0.00	0.22
Trung bình				0.26	2.69			0.00	0.23

4.2. Thực nghiệm 2

Hai bộ dữ liệu ngẫu nhiên được sử dụng là một bộ dữ liệu Euclid và một bộ dữ liệu phi Euclid. Mỗi bộ dữ liệu bao gồm các file dữ liệu có kích thước nhỏ (số đỉnh của đồ thị trong mỗi file dữ liệu $n \leq 40$). Trong bộ dữ liệu ngẫu nhiên phi Euclid, các file dữ liệu được tạo ngẫu nhiên với chi phí cạnh là các số nguyên từ 1 đến 100. Trong bộ dữ liệu ngẫu nhiên Euclid, tọa độ các đỉnh trong mỗi file dữ liệu được tạo ngẫu nhiên trong hình vuông 200×200 đơn vị. Chi phí cạnh được tính theo khoảng cách Euclid giữa hai đỉnh. Đối với mỗi bộ dữ liệu, ta

tạo ra mười file dữ liệu khác nhau có kích thước bằng 40 đỉnh. Ngoài ra, để có thêm nhiều file dữ liệu với kích thước nhỏ, có thể trích chọn ngẫu nhiên 40 đỉnh từ các file dữ liệu có kích thước lớn hơn trong TSPLIB. Như đã biết, bộ dữ liệu TSPLIB gồm các file dữ liệu. Mỗi file dữ liệu lưu trữ toạ độ của các đỉnh. Gọi $X_{\max}$, $Y_{\max}$ là hoành độ và tung độ lớn nhất, $X_{\min}$, $Y_{\min}$ là hoành độ và tung độ nhỏ nhất trong một file dữ liệu. Gọi $\Delta x = \frac{X_{\max}-X_{\min}}{n}$ và $\Delta y = \frac{Y_{\max}-Y_{\min}}{n}$. Ta chia bộ dữ liệu trên thành ba nhóm khác nhau dựa trên giá trị $\Delta x, \Delta y$. Nhóm 1 với $\Delta x, \Delta y$ nhỏ ($\Delta x, \Delta y \leq 3$, các đỉnh được phân bố gần nhau). Nhóm 2 với $\Delta x, \Delta y$ lớn ($\Delta x, \Delta y \geq 9$, các đỉnh được phân bố thưa). Nhóm 3 các đỉnh được phân bố đặc biệt (chẳng hạn, nhiều đỉnh cách đều nhau hoặc nằm trên một đường thẳng). Chia các file dữ liệu trong TSPLIB trên thành ba nhóm. Trong mỗi nhóm, chọn ra bốn file dữ liệu làm đại diện và thực hiện trích chọn ngẫu nhiên 40 đỉnh từ các file dữ liệu trong nhóm đó. Cụ thể, nhóm 1 gồm các file dữ liệu được trích chọn từ eil51, st70, eil76 và rat195. Nhóm 2 gồm các file dữ liệu được trích chọn từ kroA100, kroB100, KroC100 và Berlin52. Cuối cùng, các file dữ liệu được trích chọn từ tsp225, tss225, pr76 và lin105 thuộc về nhóm 3. Kết quả thực nghiệm được trình bày trong Bảng 4.

Trong Bảng 4, cột OPT, Best Sol (UB) và Aver Sol lần lượt là giá trị tối ưu, kết quả tốt nhất và kết quả trung bình thu được từ các thuật toán sau 10 lần chạy. Cột $gap[\%]$ vẫn ghi giá trị $gap[\%] = \dfrac{(UB - OPT)}{OPT}100\%$ như trong ba bảng trước và cột $\bar{T}$ là thời gian chạy trung bình của các thuật toán tính bằng phút.

Kết quả thực nghiệm trong Bảng 4 cho thấy thuật toán ACO-GA hiệu quả hơn thuật toán di truyền GA [3] cả về chất lượng lời giải và thời gian chạy thuật toán. Hơn nữa, đối với tất cả các bộ dữ liệu kích thước nhỏ này, thuật toán ACO-GA luôn tìm được lời giải tối ưu.

5. KẾT LUẬN

Bài báo đã đề xuất thuật toán di truyền lai ghép với thuật toán đàn kiến giải bài toán MLP, trong đó thuật toán đàn kiến có vai trò khởi tạo quần thể cho thuật toán di truyền, còn thuật toán di truyền có vai trò định hướng cho đàn kiến lựa chọn đường đi. Thuật toán được chạy thử nghiệm trên các bộ dữ liệu chuẩn để so sánh hiệu quả với các thuật toán hiện biết. Đối với bộ dữ liệu ngẫu nhiên, thuật toán đề xuất cho lời giải tốt hơn các thuật toán tốt nhất hiện biết trên nhiều bộ dữ liệu. Đối với bộ dữ liệu TSPLIB, thuật toán tỏ ra hiệu quả hơn khi luôn đưa ra lời giải tốt hơn hoặc không thua kém các thuật toán tốt nhất. Tuy nhiên, thời gian chạy của thuật toán vẫn cần được cải thiện để đáp ứng được các yêu cầu thực tiễn. Một trong những hướng để giải quyết vấn đề này là thực hiện song song hóa thuật toán đề xuất. Đó là vấn đề sẽ tiếp tục nghiên cứu trong thời gian tới.

TÀI LIỆU THAM KHẢO

[1] A. Archer, A. Levin, and D. Williamson, A faster, better approximation algorithm for the minimum latency problem, *SIAM Journal on Computing* **37** (1) (2007) 1472–1498.

[2] S. Arora, and G. Karakostas, Approximation schemes for minimum latency problems, *SIAM Journal on Computing* **32** (2) (1999) 688–693.

[3] H.B. Ban, and D.N. Nguyen, Improved genetic algorithm for minimum latency problem, *Proceedings of ACM SOICT*, Hanoi, Vietnam, 2010 (9–15).

[4] H.B. Ban, K. Nguyen, M.C. Ngo, and D.N. Nguyen, An efficient exact algorithm for minimum latency problem, *Journal on Progress of Informatics* (10) (2013) 1–8.

[5] A. Blum, P. Chalasani, D. Coppersmith, W. Pulleyblank, P. Raghavan, and M. Sudan, The minimum latency problem, *Proceedings of ACM Symposium on the Theory of Computing*, Quebec, Canada, 1994 (163–171).

[6] K. Chaudhuri, B. Goldfrey, S. Rao, and K. Talwar, Path, Tree and minimum latency tour, *Proceedings of IEEE Foundations of Computer Science*, MA, USA, 2003 (36–45).

[7] M. Dorigo, and T. Stutzle, *Ant Colony Optimization*, Bradford Books, London, 2004.

[8] M. Goemans, and J. Kleinberg, An improved approximation ratio for the minimum latency problem, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, MA, USA, 1996 (152–158).

[9] H. Hasegawa, Optimization of GROUP behavior, *Japan Ethological Society Newsletter* (43) (2004) 22-–23.

[10] A. Salehipour, K. Sorensen, P. Goos, and O.Braysy, Efficient GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem, *Journal on Operations Research* **9** (2) (2011) 189-–209.

[11] S. Sahni, and T. Gonzalez, P-complete approximation problem, *Journal on ACM* **23** (3) (1976) 555–565.

[12] M. Silva, A. Subramanian, T. Vidal, and L. Ochi, A simple and effective metaheuristic for the Minimum Latency Problem, *Journal on Operations Research* **221** (3) (2012) 513-–520.

[13] S. Shimomura, M. Sugimotoy, T. Haraguchiy, H. Matsushitaz, and Y. Nishioy, Ant colony optimization with intelligent and dull ants, *Proceedings of Symposium on Nonlinear Theory and its Applications*, Krakow, Poland, 2010 (504–507).

[14] D. S. Johnson, and L. A. McGeoch, The traveling salesman problem: A case study in local optimization, *Local Search in Combinatorial Optimization*, E. Aarts and J. K. Lenstra, eds, Wiley, 215–310, 1997.

[15] B.Y. Wu, Z.N. Huang, and F.J. Zhan, Exact algorithms for the minimum latency problem, *Journal on Inf. Process. Lett.* **92** (6) (2004) 303–309.

[16] http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95.

Ngày nhận bài 01 - 3 - 2013
Nhận lại sau sửa ngày 10 - 8 - 2013