GIẢI THUẬT DI TRUYỀN GIẢI BÀI TOÁN CỰC TIỂU HOÁ ĐỘ TRỄ
GENETIC ALGORITHM FOR SOLVING THE MINIMUM LATENCY PROBLEM

Ban Hà Bằng, Nguyễn Đức Nghĩa
Trường Đại học Bách Khoa Hà Nội

TÓM TẮT

Bài toán cực tiểu hoá độ trễ (Minimum Latency Problem – MLP) – hay còn gọi là bài toán thợ sửa chữa lưu động – là một trong những lớp bài toán tối ưu tổ hợp có nhiều ứng dụng trong thực tế. Trong trường hợp tổng quát, MLP được chứng minh là bài toán NP–khó. Hiện nay, đã có nhiều thuật toán giải gần đúng bài toán MLP được đề xuất, song chất lượng lời giải thu được chưa cao. Bài báo trình bày thuật toán phát triển dựa trên sơ đồ của thuật toán di truyền (Genetic Algorithm – GA – thuật toán áp dụng hiệu quả cho lớp bài toán tối ưu tổ hợp) để giải bài toán MLP. Kết quả thực nghiệm cho thấy, thuật toán đề xuất đưa ra được lời giải với chất lượng tốt hơn so với các lời giải của các thuật toán gần đúng tốt nhất hiện biết.

ABSTRACT

Minimum Latency Problem (MLP) - also known as traveling repairman problem - is a class of combinatiorial optimization problems that have many practical applications. In general case, MLP is proved to be NP-hard. Recently, there are several efficient approximation algorithms for solving MLP, however the quality of the provided solution is not actually high. This paper presents a new algorithm based on the scheme of the genetic algorithm for solving MLP. The experimental result on the proposed algorithm shows that it gives a better solution than the best one of the approximation algorithms.

I. ĐẶT VẤN ĐỀ

Bài toán MLP được phát biểu dưới dạng đồ thị như sau: Cho đồ thị đầy đủ có trọng số K_n với tập đỉnh $V = \{1, 2, \ldots, n\}$ và ma trận trọng số không âm đối xứng $C = \{c_{ij} \mid i, j = 1, 2, \ldots, n\}$. Giả sử $P = u_1, u_2, \ldots, u_n$ là một đường đi trên K_n. Ta gọi độ trễ của đỉnh u_k $(1 < k \le n)$ trên đường đi P là tổng $\rho(u_k) = \sum_{i=1}^{k-1} c(u_i, u_{i+1})$, trong đó $c(u_i, u_{i+1})$ là trọng số của cạnh (u_i, u_{i+1}). Độ trễ của đường đi P được định nghĩa như là tổng độ trễ của tất cả các đỉnh trên đường đi:

$$\rho(u_k) = \sum_{k=2}^{n} \rho(u_k).$$

Bài toán MLP yêu cầu tìm một đường đi đơn bắt đầu từ một đỉnh xuất phát cố định u_1 đi qua tất cả các đỉnh còn lại của đồ thị với độ trễ là cực tiểu.

Một số ứng dụng của bài toán (có thể xem chi tiết trong [1, 2]):

- Một máy chủ phải phục vụ một tập các yêu cầu. Cần tìm lịch thực hiện các yêu cầu trên máy chủ đó sao cho thời gian chờ đợi trung bình của mỗi yêu cầu là cực tiểu.

- Một ứng dụng khác của bài toán MLP là bài toán cực tiểu hoá thời gian tìm kiếm thông tin trên mạng.

Trong một số trường hợp đặc biệt, bài toán MLP có thể giải được trong thời gian đa thức [3]. Thế nhưng trong tình huống tổng quát MLP đã được chứng minh là thuộc lớp NP-khó [1], nghĩa là ngoại trừ P=NP, không có thuật toán thời gian đa thức để giải nó. Vì vậy, việc xây dựng các thuật toán gần đúng hiệu quả là cách tiếp cận tự nhiên để phát triển thuật toán giải bài toán MLP trong trường hợp tổng quát. Blum đưa ra thuật toán gần đúng với cận tỷ lệ 144 [2], Gemans và Klein Berg giảm cận này xuống còn 21.55 [4], Grag tiếp tục giảm xuống còn 10.78 [5]. Aaron Archer, Asaf Levin, David Williamson đưa ra thuật toán gần đúng với cận tỷ lệ 3.01 [6] – là cận nhỏ nhất hiện nay.

Một lớp thuật toán khác cũng được áp dụng cho bài toán là lớp thuật toán heuristic [7].

Lớp thuật toán này tập trung tìm kiếm lời giải cực trị địa phương.

Bài báo này trình bày thuật toán phát triển dựa trên sơ đồ của thuật toán di truyền với kết quả thực nghiệm đạt được tốt hơn so với kết quả đạt được trong [6] và [7].

II. GIẢI THUẬT DI TRUYỀN GIẢI BÀI TOÁN MLP

GA dựa trên thuyết chọn lọc tự nhiên của Darwin được Holland đề xuất vào những năm 1970. Hiện nay, GA được áp dụng vào việc giải quyết các bài toán trong nhiều lĩnh vực. Sơ đồ chung của thuật toán có thể mô tả như sau [8]:

Khởi tạo quần thể ban đầu;

LOOP

Lựa chọn các cá thể cha mẹ trong quần thể bằng toán tử lựa chọn;

Lai ghép các cá thể cha mẹ được chọn để tạo ra các cá thể con cháu bằng toán tử lai ghép;

Đột biến các cá thể con cháu bằng toán tử đột biến;

Loại bỏ các cá thể cha mẹ ra khỏi quần thể;

Bổ sung các cá thể con cháu vào quần thể;

IF thoả mãn điều kiện dừng THEN exit LOOP;

END LOOP

- Kỹ thuật mã hoá thực hiện việc mã hoá các cá thể lời giải của bài toán. Với bài toán MLP, cá thể được biểu diễn bằng một danh sách các đỉnh. Chẳng hạn, cá thể đường đi: 1, 3, 4, 2, 5 được biểu diễn bằng danh sách P = (1, 3, 4, 2, 5). Ưu điểm của kỹ thuật mã hoá này là đơn giản, dễ cài đặt cho lớp bài toán MLP.

- Khởi tạo quần thể ban đầu với kích thước quần thể là k: Chọn một đỉnh bất kỳ là đỉnh xuất phát, cố định đỉnh này. Sinh ngẫu nhiên k hoán vị của n-1 đỉnh còn lại. Mỗi hoán vị kết hợp với đỉnh xuất phát cho ta một cá thể đường đi.

- Toán tử lựa chọn: Chọn ngẫu nhiên một nhóm cá thể lời giải với kích thước nhóm cho trước (kí hiệu: N). Sau đó, chọn ra hai cá thể có độ trễ nhỏ nhất làm cá thể cha mẹ. Ưu điểm của toán tử là lực lựa chọn thay đổi một cách dễ dàng bằng cách thay đổi kích thước nhóm. Chẳng hạn: Khi giá trị N nhỏ, các cá thể có độ thích nghi thấp sẽ có nhiều cơ hội được lựa chọn hơn khi giá trị N lớn.

- Toán tử lai ghép lai ghép hai cá thể cha và mẹ với xác suất lai ghép (kí hiệu: P_c) cho trước để tạo ra cá thể con. Do đỉnh xuất phát cố định nên toán tử sẽ lai ghép cá thể cha và mẹ từ n-1 đỉnh còn lại như sau: Lựa chọn ngẫu nhiên một số vị trí trong cá thể cha. Sao chép các đỉnh ở các vị trí được lựa chọn trong cá thể cha vào các vị trí tương ứng trong cá thể con. Các đỉnh ứng với những vị trí không được lựa chọn trong cá thể cha, được điền vào những vị trí còn khuyết từ trái qua phải trong cá thể con, theo thứ tự mà các đỉnh đó xuất hiện trong cá thể mẹ. Toán tử lai ghép này giống với toán tử lai ghép được trình bày trong [9]. Toán tử lai ghép giúp cho GA nâng cao được chất lượng trung bình của các cá thể lời giải trong quần thể.

- Toán tử đột biến đột biến cá thể với xác suất đột biến cho trước (kí hiệu: P_m). Toán tử đột biến giúp cho GA hạn chế được sự hội tụ sớm. Ta xét hai toán tử đột biến. Toán tử đột biến thứ nhất thực hiện việc đột biến đối với cá thể $u = (u_1, u_2, ..., u_n)$ như sau: Chọn ngẫu nhiên hai vị trí i, j $(1 \le i < j < n)$, sau đó đột biến u thành $u^* = (u_1, u_2, ..., u_i, u_{j+1}, u_{j+2}, ..., u_n, u_{i+1}, u_{i+2}, ..., u_j)$, nghĩa là u^* thu được từ u bằng cách di chuyển toàn bộ đoạn từ vị trí $j+1$ đến n vào sau vị trí i. Toán tử đột biến thứ hai thực hiện đột biến cá thể theo toán tử đột biến thứ nhất, sau đó, áp dụng thuật toán tìm kiếm địa phương 2-opt (trong [7]) cho cá thể đó.

- Sau mỗi thế hệ, các cá thể cha mẹ bị loại bỏ ra khỏi quần thể. Trong khi đó, các cá thể con cháu được bổ sung sẽ đóng vai trò là cá thể cha mẹ ở thế hệ tiếp theo.

- Điều kiện dừng thuật toán: Thuật toán sẽ dừng nếu sau 10 thế hệ lời giải tốt nhất không được cải thiện.

III. TÍNH TOÁN THỰC NGHIỆM

3.1 Dữ liệu thực nghiệm

Dữ liệu thực nghiệm được lấy từ bộ dữ liệu TSPLIB95 [10]. Bộ dữ liệu này gồm một số file, mỗi file chứa toạ độ của n điểm. 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 của các điểm trong một file, đặt $\Delta_x = (X_{max} - X_{min})/n$ và $\Delta_y = (Y_{max} - Y_{min})/n$. Ta phân các bộ dữ liệu trên thành ba nhóm dựa trên các giá trị Δ_x, Δ_y. Nhóm 1 với Δ_x, Δ_y nhỏ (Δ_x, $\Delta_y \le 3$, các điểm được phân bố gần nhau), nhóm 2 với Δ_x, Δ_y lớn

(Δ_x, $\Delta_y \geq 9$, các điểm được phân bố thưa), nhóm 3 các điểm được phân bố đặc biệt (chẳng hạn, nhiều điểm cách đều nhau, hoặc nằm trên một đường thẳng). Đối với bộ dữ liệu TSPLIB95, chúng ta chọn ra trong mỗi nhóm một bộ dữ liệu đại diện để tiến hành làm thực nghiệm xác định giá trị của các tham số cho thuật toán. Sau đó, với giá trị tham số tìm được, tiến hành thực nghiệm với bộ dữ liệu TSPLIB95. Do có nhiều tham số đầu vào, khi tiến hành thực nghiệm xác định tham số, chúng ta sẽ thay đổi giá trị của một tham số được chọn và cố định các giá trị tham số còn lại, từ đó có thể xem xét ảnh hưởng của tham số được chọn đến kết quả.

Dữ liệu đầu vào của giải thuật là *n* điểm được cho bởi toạ độ trên mặt phẳng.

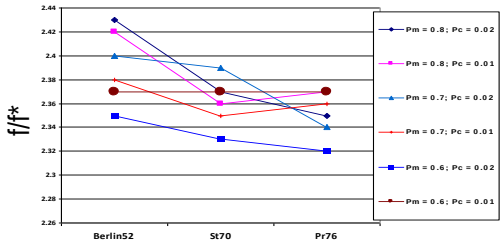
Tham số của GA, GAH gồm: k, P_c, P_m N (các ký hiệu đã giải thích ở trên).

Chọn các file dữ liệu: St70 (nhóm 1), Berlin52 (nhóm 2), Pr76 (nhóm 3). Chúng ta sẽ chọn các giá trị tham số mà thuật toán cho ra giá trị $\overline{f/f^*}$ và Δf nhỏ nhất, trong đó f là độ trễ của lời giải và f* là độ trễ tối ưu, $\overline{f/f^*}$ và Δf lần lượt là trung bình cộng và độ lệch chuẩn của các f/f* ứng với các file dữ liệu thực nghiệm. Ký hiệu: GA là thuật toán sử dụng toán tử đột biến thứ nhất, GAH là thuật toán sử dụng toán tử đột biến thứ hai và k/n là tỷ số giữa k và n.

3.2 Kết quả thực nghiệm

1. Thực nghiệm lựa chọn giá trị các tham số cho thuật toán

Thực nghiệm chọn P_c, P_m. Tham số cố định: N = 5, k/n = 20; tham số thay đổi: $P_c \in (0.6, 0.8)$, $P_m \in (0.01, 0.02)$. Kết quả thực nghiệm được diễn tả trong hình 1.



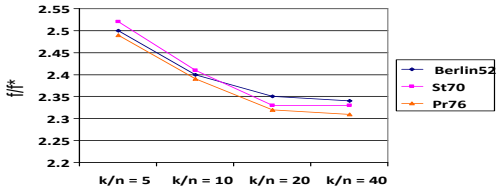
Hình 1. So sánh kết quả lời giải với Pc, Pm khác nhau

Trên nhìn vẽ, mỗi đường gấp khúc ứng với kết quả lời giải với P_c, P_m khác nhau.

Nhận xét: P_c = 0.6, P_m = 0.02, GA cho ra kết quả lời giải với $\overline{f/f^*}$ và Δf thấp nhất ($\overline{f/f^*}$ = 2.33, Δf = 0.00025). Chúng ta sẽ sử dụng các giá trị tham số này.

Thực nghiệm chọn *k*. Tham số cố định: N = 5, P_c = 0.6, P_m = 0.02; tham số thay đổi: k/n = (5, 10, 20, 40). Kết quả thực nghiệm được cho trong hình 2.

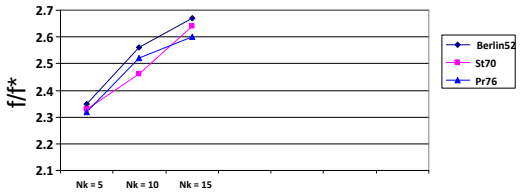
Nhận xét: Khi tăng *k*, chất lượng lời giải cũng tăng theo. Tuy nhiên, khi *k* tăng đến một giá trị đủ lớn, chất lượng lời giải gần như không được cải thiện (giá trị f/f* gần như không được cải thiện khi tăng *k* lên từ k/n = 20 đến k/n = 40). Như vậy, việc tăng *k* không phải lúc nào cũng tăng chất lượng lời giải, thậm chí còn làm tăng thời gian chạy chương trình. Vậy, chọn k/n = 20 là phù hợp.



Hình 2. So sánh kết quả lời giải với k khác nhau

Trên nhìn vẽ, mỗi đường gấp khúc ứng với kết quả lời giải với tỷ số k/n khác nhau.

Thực nghiệm chọn N. Tham số cố định k/n = 20, P_c = 0.6, P_m = 0.02; tham số thay đổi: N = (5, 10, 15). Hình 3 trình bày kết quả thực nghiệm chọn *N*.

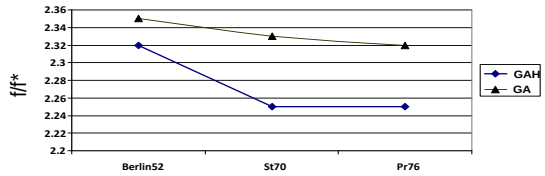


Hình 3. So sánh kết quả lời giải với N_k khác nhau

Trên nhìn vẽ, mỗi đường gấp khúc ứng với kết quả lời giải với *N* khác nhau.

Nhận xét: *N* càng lớn, lực lựa chọn càng lớn dẫn đến GA càng hội tụ sớm, kết quả lời giải đạt được không cao. Thực nghiệm cho thấy với N = 5, kết quả lời giải đạt được tốt nhất.

Thực nghiệm so sánh kết quả lời giải của GAH và GA (hình 4). Tham số cố định: N = 5, P_c = 0.6, P_m = 0.02, k/n = 20.



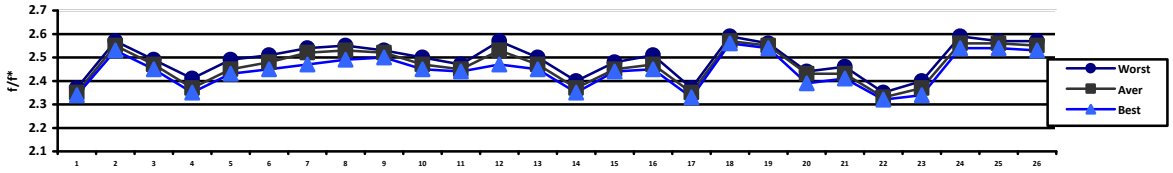
Hình 4. So sánh kết quả lời giải của GA và GAH

Nhận xét: Khi áp dụng phép toán đột biến thứ hai, chất lượng lời giải của GAH được cải thiện so với GA. Tuy nhiên, sự cải thiện này vẫn còn thấp ($\overline{f/f^*}$= 2.27 so với $\overline{f/f^*}$= 2.33). Nguyên nhân chủ yếu là do GAH cũng như GA đều hội tụ sớm.

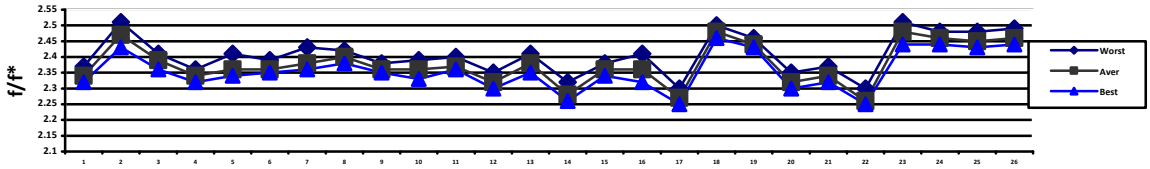
2. Thực nghiệm với bộ dữ liệu TSPLIP95

Tiến hành thực nghiệm thuật toán trên bộ dữ liệu TSPLIB95 với các giá trị tham số tìm được ở trong các thực nghiệm trước. Thực nghiệm tiến hành như sau**:** Mỗi file chạy 5 lần với GA và GAH. Kết quả tốt nhất ứng với cột Best, kết quả tồi nhất ứng với cột Worst, kết quả trung bình 5 lần chạy ứng với cột Aver. Bảng 1 trình bày các kết quả thực nghiệm. Các kết quả trong bảng 1 được diễn tả trực quan hơn trong các hình 5 – 9.

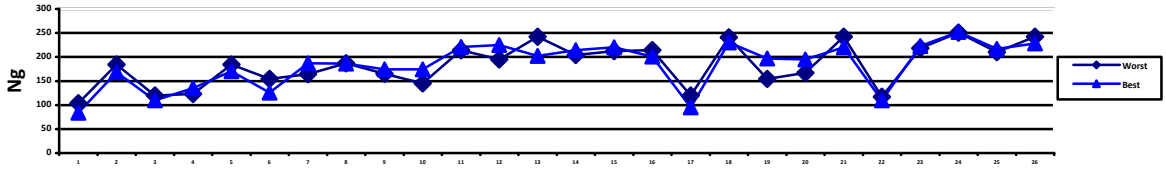
Kí hiệu: Các ký hiệu số ở trục hoành trong các hình 5–9 tương ứng với các file dữ liệu được sắp thứ tự trong bảng 1. AA là thuật toán gần đúng của Archer, Levin, Williamson [6]. LS là thuật toán heuristic 2-opt kết hợp với thuật toán k-láng giềng [7]. N_g là số thế hệ được khảo sát trong thuật toán.



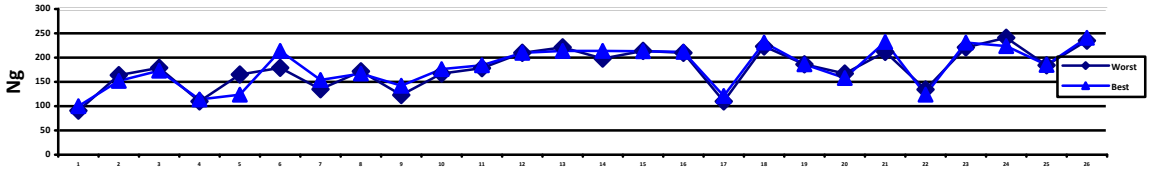
Hình 5. So sánh kết quả lời giải của GA ứng với kết quả tốt nhất, tồi nhất, trung bình



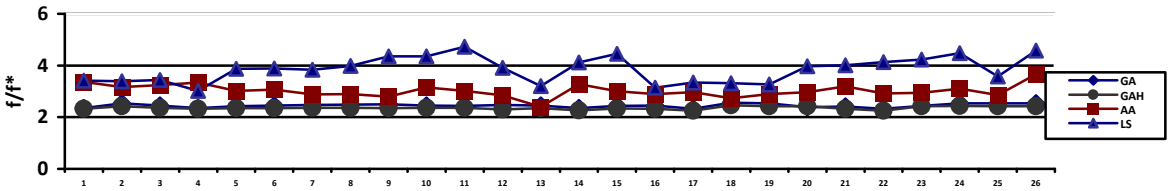
Hình 6. So sánh kết quả lời giải của GAH ứng với kết quả tốt nhất, tồi nhất, trung bình



Hình 7. So sánh tổng số lượng thế hệ của GA ứng với lời giải tồi nhất và tốt nhất



Hình 8. So sánh tổng số lượng thế hệ của GAH ứng với lời giải tồi nhất và tốt nhất



Hình 9. So sánh kết quả lời giải với các thuật toán khác nhau

Nhận xét: Hình 5, hình 6 cho thấy sự chênh lệch kết quả thực nghiệm giữa lời giải tốt nhất, tồi nhất và trung bình của GA, GAH là thấp. Điều đó chứng tỏ kết quả của GA, GAH với các bộ dữ liệu khác nhau có độ ổn định tương đối cao. Hình 7, hình 8 cho thấy GA, GAH hội tụ khá sớm đến lời giải cực trị địa phương (tổng số thế hệ trong cả hai trường hợp đối với bộ dữ liệu TSPLIB95 dao động ổn định trong khoảng từ 95 đến 252 thế hệ). Kết quả thực nghiệm từ bảng 1 cũng cho thấy, chất lượng lời giải không hoàn toàn phụ thuộc vào số lượng thế hệ (ví dụ: Pr76 khi áp dụng GA: f/f* = 2.37 với N_g = 120, trong khi, f/f* = 2.33 với N_g = 95). Điều đó cho thấy việc tránh sự hội tụ sớm, nâng cao chất lượng lời giải không chỉ đơn thuần là nâng cao số lượng thế hệ. Hình 9 chứng tỏ kết quả thu được bởi GA và GAH là tốt hơn so với AA và LS. Tuy nhiên, chất lượng lời giải đạt được vẫn thấp: Giá trị tốt nhất đạt được bởi GA và GAH trung bình còn lớn hơn cỡ 2.37 lần giá trị tối ưu. Nguyên nhân bởi GA, cũng như GAH hội tụ sớm đến cực trị địa phương.

IV. KẾT LUẬN

Kết quả thực nghiệm đạt được khi áp dụng thuật toán di truyền giải bài toán MLP được trình bày trong bài báo tốt hơn so với kết quả đạt được từ các thuật toán gần đúng tốt nhất hiện biết. Có thể thấy thuật toán di truyền là hướng có triển vọng để giải quyết bài toán MLP. Tuy nhiên, kết quả thực nghiệm đạt được vẫn còn thấp (giá trị tốt nhất trung bình tìm được vẫn còn lớn hơn cỡ 2.37 giá trị tối ưu). Điều đó cho thấy GA, GAH đề xuất trong bài này còn hội tụ sớm. Việc khắc phục sự hội tụ sớm và tổng quát hoá kết quả thực nghiệm đạt được để nâng cao chất lượng của lời giải sẽ được bàn đến trong những công trình tiếp theo.

Bảng 1. So sánh chất lượng lời giải của các thuật toán

Dữ liệu	GA					GAH					LS			AA
	Worst		Aver	Best		Worst		Aver	Best		Worst	Aver	Best	
	f/f*	N_g	f/f*	f/f*	N_g	f/f*	N_g	f/f*	f/f*	N_g				
(1) Berlin52	2.37	104	2.35	2.34	84	2.37	91	2.34	2.32	100	3.83	3.51	3.42	3.36
(2) Eil101	2.57	184	2.55	2.53	168	2.51	164	2.47	2.43	152	3.45	3.42	3.39	3.17
(3) Eil76	2.49	120	2.47	2.45	110	2.42	179	2.39	2.36	173	3.56	3.52	3.44	3.24
(4) Eil51	2.41	123	2.37	2.35	135	2.36	110	2.34	2.32	114	2.72	2.65	3.02	3.34
(5) KroA100	2.49	184	2.45	2.43	170	2.41	165	2.36	2.34	124	4.21	4.01	3.87	3.02
(6) KroA150	2.51	154	2.48	2.45	126	2.39	179	2.37	2.35	214	4.52	4.21	3.89	3.07
(7) KroB100	2.54	164	2.52	2.47	187	2.43	135	2.38	2.36	154	4.29	3.92	3.84	2.88
(8) KroB150	2.55	187	2.53	2.49	186	2.42	172	2.40	2.38	167	4.25	4.14	3.98	2.89
(9) KroC100	2.53	165	2.52	2.50	174	2.38	123	2.36	2.35	142	4.51	4.41	4.35	2.79
(10) KroD100	2.50	145	2.47	2.45	174	2.39	167	2.36	2.33	176	4.63	4.42	4.35	3.14
(11) KroE100	2.47	214	2.45	2.44	221	2.40	178	2.37	2.36	184	4.85	4.75	4.73	3.01
(12) Lin105	2.57	195	2.53	2.50	225	2.35	210	2.32	2.30	210	4.18	4.05	3.91	2.84
(13) Pr107	2.50	242	2.47	2.45	202	2.40	221	2.38	2.35	214	3.68	3.56	3.21	2.40
(14) Pr124	2.40	214	2.37	2.35	214	2.32	198	2.28	2.26	214	4.37	4.21	4.12	3.28
(15) Pr136	2.48	212	2.45	2.44	220	2.38	214	2.36	2.34	213	4.68	4.53	4.45	3.01
(16) Pr144	2.50	214	2.47	2.45	201	2.41	210	2.36	2.32	212	3.21	3.17	3.14	2.89
(17) Pr76	2.37	120	2.35	2.33	95	2.30	110	2.27	2.25	121	3.58	3.46	3.34	2.97
(18) Rat195	2.59	241	2.57	2.56	230	2.50	223	2.48	2.46	231	3.48	3.42	3.31	2.73
(19) Rat99	2.56	154	2.55	2.54	197	2.46	186	2.44	2.43	186	3.65	3.48	3.27	2.89
(20) Rd100	2.44	167	2.43	2.39	195	2.35	167	2.32	2.30	158	4.17	4.05	3.97	2.97
(21) Rd400	2.46	242	2.43	2.41	220	2.37	212	2.34	2.32	232	4.20	4.15	4.13	3.19
(22) St70	2.35	117	2.33	2.32	110	2.30	134	2.26	2.25	124	4.12	4.05	4.01	2.94
(23) U195	2.52	215	2.47	2.44	234	2.50	221	2.44	2.42	231	4.63	4.36	4.23	2.94
(24) U574	2.59	251	2.56	2.54	252	2.48	241	2.46	2.44	224	4.54	4.51	4.48	3.10
(25) Ts225	2.57	210	2.56	2.54	217	2.48	184	2.45	2.43	185	3.96	3.79	3.54	2.86
(26) Vm1084	3.20	242	3.16	3.12	228	3.18	235	3.13	3.10	241	4.64	4.61	4.58	3.66

TÀI LIỆU THAM KHẢO

1. *S. Sahni, T. Gonzalez*; P-complete approximation problems, Journal of the ACM 23 (1976) 555–565.
2. *A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan and M. Sudan*; The minimum latency problem. Proceedings of 26th ACM Sympon Theory Of Computing(STOC), pp 163{171, 1994}.
3. *B.Y. Wu, Z. N. Huang and F.-J. Zhan* (2004/12); Exact algorithms for the minimum latency problem; Information Processing Letters, vol. 92(6), pp. 303-309.
4. *M. Goemans and J. Kleinberg*; An improved approximation ratio for the minimum latency problem; Proc. 7th ACMSIAM Symposium on Discrete Algorithms (SODA), pp 152-158, 1996.
5. *N. Garg*; A 3-approximation for the minimum tree spanning k vertices. Proc. 37th IEEE Symp; On Foundations of Computer Science (FOCS), pp.302 {309, 1996}.
6. *Aaron Archer, Asaf Levin, David Williamson*; A Faster, Better Approximation Algorithm for the Minimum Latency Problem; Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003.
7. http://www.postech.ac.kr/~bkim/tsp_report.doc
8. *Melanie Mitchel*; An introduction to genetic algorithms**;** MIT Press Cambridge, MA, USA, 1996.
9. *P. Larranaga, C.M.H, Kuijpers, R.H.Murga I. Inza and S. Dizdarevic*; A review of representations and operators; Department of Computer science and Atificial Intelligence, P.O. Box 649, University of Basque, spain.
10. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

Địa chỉ liên hệ: Nguyễn Đức Nghĩa - Tel: 0903.210.111, email: nghiand@it-hut.edu.vn
Ban Hà Bằng – Tel: 0985.819.467, email: bangbh_bkit@yahoo.com
Khoa Công nghệ thông tin, Trường Đại học Bách khoa Hà Nội