Tạp chí Tin học và Điều khiển học, T.24, S.2 (2008), 141–151

MỘT CÁCH TIẾP CẬN THUẬT TOÁN GEN ĐỂ GIẢI BÀI TOÁN PHỦ TẬP HỢP

PHAN THỊ HOÀI PHƯƠNG[1], NGUYỄN MINH HẰNG[2], LƯƠNG CHI MAI[3]

[1]*Học viện Công nghệ Bưu chính-Viễn thông*
[2]*Trung tâm tính toán - Viện hàn lâm khoa học Liên bang Nga*
[3]*Viện Công nghệ thông tin, Viện Khoa học và Công nghệ Việt Nam*

Abstract. Bài toán phủ tập hợp (Set Covering Problem - SCP) là một mô hình toán học cho nhiều ứng dụng quan trọng như lập lịch biểu, quy hoạch dịch vụ, phân tích dữ liệu logic, đơn giản hóa biểu thức Boolean Trong bài này chúng ta đề xuất một cách tiếp cận dựa trên thuật toán gen (GA) để giải bài toán SCP và thử nghiệm đánh giá hiệu quả của nó trên các bài toán mẫu trong thư viện Beasley's OR Library.

Tóm tắt. The Set Covering Problem (SCP) is a main model for several important applications, including scheduling, service planning, logical data analysis, Boolean expression simplification. In this paper, a new genetic algorithm-based approach for solving SCP is proposed and its performance is evaluated on the test-bed instances of Beasley's OR Library.

1. BÀI TOÁN TÌM PHỦ TỐI THIỂU

Bài toán phủ tập hợp (Set Covering Problem - SCP) là một mô hình toán học cho nhiều ứng dụng quan trọng như lập lịch biểu [6], quy hoạch dịch vụ, phân tích dữ liệu logic, đơn giản hóa biểu thức Boolean [8]SCP là bài toán NP đầy đủ [4] nên việc xây dựng thuật toán tìm nghiệm tối ưu hay nghiệm xấp xỉ là rất khó khăn. Tuy nhiên cấu trúc các thể hiện của bài toán trong thế giới thực cung cấp thêm những thông tin ngữ cảnh cho phép giải quyết bài toán SCP kích thước khá lớn (vài trăm dòng và vài triệu cột) với nghiệm đạt được sai khác so với nghiệm tối ưu chỉ khoảng 1% trong khoảng thời gian tính toán chấp nhận được [8]. Các phương pháp giải bài toán SCP trong thực tế dựa trên nhiều cách tiếp cận khác nhau và có thể phân loại thành các lớp như lớp thuật toán dựa trên lý thuyết quy hoạch tuyến tính, lớp các thuật toán heuristics và lớp các thuật toán branch-and-bound [7–13]. Dưới đây chúng ta sẽ đưa ra phát biểu hình thức của bài toán và đề xuất thuật toán dựa trên nguyên lý của thuật toán gen (GA) để giải bài toán SCP.

1.1. Hình thức hóa bài toán tìm phủ tối thiểu

Bài toán phủ tập hợp (Set Covering Problem - SCP) là mô hình của nhiều ứng dụng quan trọng. SCP có thể được định nghĩa một cách hình thức như sau:

Giả sử $A = \{a_{ij}\}$ là ma trận nhị phân $m \times n$ và $C = \{c_j\}$ là vector thực n chiều với các thành phần không âm. Ta ký hiệu tập $M = \{1, 2, ..., m\}$; $Q = \{1, 2, ..., n\}$. Chúng ta coi ma trận A bao gồm các hàng và các cột và c_j là giá của cột $j \in Q$. Người ta nói rằng cột $j \in Q$ phủ hàng $i \in M$ nếu $a_{ij} = 1$. Một nghiệm chấp nhận được của SCP là tập con bất kỳ các cột $S,\ S \subseteq Q$, sao cho mỗi hàng $i \in M$ được phủ ít nhất bởi một cột $j \in S$. Bài toán SCP chính là việc tìm kiếm nghiệm chấp nhận được có giá tối thiểu với mô hình toán học như sau:

$$\sum_{j \in Q} c_j x_j \to \min. \tag{1}$$

Trên tập các ràng buộc:

$$\sum_{j \in Q} a_{ij} x_j \geq 1,\ i \in M, \tag{2}$$

$$x_j \in \{0, 1\},\ j \in Q \tag{3}$$

với $x_j = 1$ nếu $j \in S$ và $x_j = 0$ nếu ngược lại.

Mục sau sẽ trình bày cách tiếp cận dựa trên thuật toán gen để giải quyết bài toán (1)-(2)-(3).

1.2. Cách tiếp cận thuật toán gen giải bài toán tìm phủ tối thiểu

Một cách không hình thức, thuật toán gen có thể được mô tả như sau:
Bước 1: Lựa chọn một quần thể xuất phát.
Bước 2: Đánh giá mức độ thích nghi của từng cá thể trong quần thể.
Bước 3: Lựa chọn cha-mẹ từ quần thể theo mức độ thích nghi.
Áp dụng các toán tử gen lên cha-mẹ (crossover hoặc đột biến) để tạo ra các con.
Bước 4: Đánh giá mức độ thích nghi của các con được sinh ra.
Thay thể một vài cá thể trong quần thể (hoặc toàn bộ) bằng các con.
Bước 5: Nếu đạt được nghiệm mong muốn thì dừng. Nếu không lặp lại bước 3.

Để có thể áp dụng thuật toán gen giải bài toán tìm phủ tối thiểu, chúng ta sẽ lần lượt hình thức hóa các bước của thuật toán trên theo ngữ cảnh của bài toán.

Không mất tính tổng quát ta có thể giả thiết rằng các cột trong SCP được sắp xếp theo thứ tự giá tăng dần và với các cột có cùng giá ta sắp xếp chúng theo thứ tự giảm dần số lượng hàng được phủ bởi cột.

a. Biểu diễn bài toán và mức độ thích nghi

Bước đầu tiên trong việc thiết kế thuật toán gen là việc lựa chọn cách biểu diễn (mã hóa) thích hợp cho bài toán. Ở đây chúng ta chọn cách biểu diễn nhị phân. Mỗi cá thể i được biểu diễn bởi một nhiễm sắc thể là một vector n chiều S_j, trong đó thành phần thứ j của vector s_{ij} sẽ nhận giá trị 1 nếu cột j nằm trong nghiệm và nhận giá trị 0 nếu ngược lại. Với cách biểu diễn đó, mức độ thích nghi f_i của cá thể S_i được tính như sau:

$$f_i = \sum_{j=1}^{n} c_j s_{ij} \tag{4}$$

trong đó c_j là giá của cột j.

b. Kỹ thuật lựa chọn cha-mẹ

Việc lựa chọn cha-mẹ là chính là việc gán cho các cá thể cơ hội tái tạo lại. Có rất nhiều phương pháp đã được đề xuất. Ở đây chúng ta chọn phương pháp lựa chọn tương xứng. Với phương pháp này, mỗi cá thể được gán một xác suất lựa chọn tương xứng với mức độ thích nghi của nó và thủ tục lựa chọn được thực hiện dựa trên phân phối xác suất đã xây dựng. Xác suất lựa chọn của cá thể i được tính như sau:

$$p_i = \frac{1/f_i}{\sum\limits_{i=1}^{N} 1/f_i}. \tag{5}$$

Ở đây f_i là mức độ thích nghi của cá thể i trong quần thể và N là kích thước của quần thể. Chú ý rằng bài toán SCP là bài toán cực tiểu hóa nên xác suất lựa chọn của cá thể tỷ lệ nghịch với mức độ thích nghi.

c. Toán tử Crossover

Thông thường trong các thuật toán gen người ta sử dụng các toán tử crossover 1 điểm hoặc 2 điểm. Khi đó các điểm crossover được tạo ra một cách ngẫu nhiên sẽ chia nhiễm sắc thể thành nhiều đoạn gen. Các đoạn gen của cặp cha-mẹ sẽ được đổi cho nhau để tạo nên các con. Toán tử crossover 1 điểm có thể được định nghĩa một cách hình thức như sau:

Giả sử P_1 và P_2 là các nhiễm sắc thể cha-mẹ $(P_1[1], ..., P_1[n])$ và $(P_2[1], ..., P_2[n])$. Khi đó với điểm crossover là $k,\ 1 \leq k < n$, toán tử crossover sẽ tạo ra 2 cá thể con Ch_1 và Ch_2 như sau:

$$Ch_1 = (P_1[1], ..., P_1[k],\ P_2[k+1], ..., P_2[n]).$$

$$Ch_2 = (P_2[1], ..., P_2[k],\ P_1[k+1], ..., P_1[n]).$$

Nhận xét rằng các toán tử crossover kinh điển như trên không quan tâm đến mức độ thích nghi của chính cha-mẹ. Vì thế chúng không mô hình hóa được các tính chất như tính "trội", tính "lặn"... của các cặp gen tương ứng trong cha-mẹ. Để khắc phục được hạn chế này chúng ta đề xuất một toán tử crossover mới chỉ tạo ra 1 con từ một cặp cha-mẹ và cho phép cha-mẹ "di truyền" lại cho con gen "trội" của mình so với gen tương ứng của cá thể ghép cặp. Nếu giá trị của gen ở cùng một vị trí là giống nhau trong nhiễm sắc thể của cặp cá thể cha-mẹ thì chúng sẽ được sao lại cho con. Ngược lại nếu cặp gen tương ứng ở cùng vị trí trong 2 nhiễm sắc thể là khác nhau thì gen "trội" được xác định phụ thuộc vào mức độ thích nghi trung bình của các cá thể nằm trong hai phân hoạch của quần thể do cặp gen đó tạo ra. Giả sử G là quần thể đang xét. Khi đó ta ký hiệu $G_i(1)$ là tập các cá thể của G có giá trị gen thứ i là 1 và $G_i(0)$ là tập các cá thể của G có giá trị gen thứ i là 0. Ta có phân hoạch $G = G_i(1) \cup G_i(0)$ với mọi i. Toán tử crossover mới được xây dựng như sau.

Giả sử cặp cá thể cha-mẹ là $P_1,\ P_2$ và P là cá thể con thu được từ việc áp dụng toán tử crossover mới lên P_1 và P_2. Khi đó với mọi $i,\ 1 \leq i \leq n$, ta xác định $P[i]$ theo quy tắc:

1. Nếu $P_1[i] = P_2[i]$, đặt $P[i] = P_1[i] = P_2[i]$. (6)

2. Nếu $P_1[i] \neq P_2[i]$. Khi đó ta xác định hai tập hợp $G_i(P_1[i])$ và $G_i(P_2[i])$. Do $P_1[i] \neq P_2[i]$ nên hai tập hợp này đều khác rỗng và cũng tạo nên phân hoạch $G_i(P_1[i]) \cup G_i(P_2[i])$. Trên hai tập nhận được ta tính độ thích nghi trung bình của các cá thể trong trong chúng theo công thức sau:

$$\overline{f}_1 = \frac{1}{\|G_i(P_1[i])\|} \sum_{j \in G_i(P_1[i])} f_j \text{ và } \overline{f}_2 = \frac{1}{\|G_i(P_2[i])\|} \sum_{j \in G_i(P_2[i])} f_j \tag{7}$$

với $\|G_i(P_k[i])\|$ là lực lượng của các tập $G_i(P_k[i]),\ k = 1, 2$, và các f_j được xác định bởi (4). Khi đó:

$$P[i] = P_1[i] \text{ nếu } \overline{f}_1 \leq \overline{f}_2.$$

Ngược lại $P[i] = P_2[i]$. (8)

Toán tử crossover như trên không những cho phép phản ánh vai trò của gen trong từng cá thể theo (4) mà còn tính đến sự đóng góp của nó vào quá trình hình thành của quần thể. Cặp gen có giá trị khác nhau trong hai cá thể cha-mẹ sẽ chia quần thể thành hai phần với độ thích nghi trung bình khác nhau. Giá trị gen ứng với phần có độ thích nghi trung bình nhỏ hơn cho kỳ vọng rằng, nghiệm của bài toán có nhiều khả năng nằm ở nửa không gian với lược đồ xác định bởi giá trị đó tại vị trí đang xét. Bởi vậy gen của cá thể cha-mẹ có giá trị này có thể được coi là gen "trội" và được truyền cho con.

Nhận xét rằng khi $G_i(1) = G$ (hoặc $G_i(0) = G$) thì mọi cá thể trong quần thể đều có giá trị của gen i là giống nhau và giá trị đó bảo đảm được di truyền cho con theo nhánh (6). Theo thuật ngữ thường dùng, tập hợp những điểm như vậy được gọi là một "lược đồ" của quần thể. Lược đồ sẽ được truyền từ thế hệ trước cho thế hệ sau và phân chia nhiễm sắc thể thành các khối riêng biệt. Đặc điểm này cho phép chúng ta có thể áp dụng song song thuật toán gen trên các khối riêng biệt đó để nâng cao hiệu quả về mặt thời gian của thuật toán. Lược đồ cũng góp phần bảo đảm tính xác định của toán tử đột biến được xây dựng sau đây.

d. Tần xuất đột biến biến thiên

Nhiều tác giả đã đề xuất tần số đột biến cố định là $1/n$. Khi đó đột biến xảy ra trên một gen được lựa chọn ngẫu nhiên. Tần xuất đột biến này tỏ ra quá nhỏ khi thuật toán gen hội tụ, nhất là khi thuật toán tiến dần tới nghiệm tối ưu cục bộ. Những điểm tối ưu cục bộ có thể coi như các cá thể trong quần thể là "cận huyết thống". Bởi vậy để cải thiện quần thể đó, chúng ta cần giảm bớt tính "cận huyết thống" và tăng cường tính "đa dạng sinh học" cho quần thể. Việc sử dụng tần xuất đột biến biến thiên có thể cho phép chúng ta thực hiện ý tưởng này. Sau đây là một phương pháp lựa chọn tần xuất đột biến biến thiên.

Với mỗi gen thứ $i,\ 1 \leq i \leq n$, ta tính Entropy của gen thứ i

$$H_i = -p_0(i) \log p_0(i) - p_1(i) \log p_1(i) \tag{9}$$

ở đây $p_0(i)$ và $p_1(i)$ là tần xuất xuất hiện giá trị 0 và giá trị 1 của gen thứ i trong quần thể.

Ta coi rằng khi H_i càng nhỏ thì mức độ "cận huyết thống" của các cá thể trong quần thể càng lớn theo ngữ cảnh của gen thứ i. Ví dụ nếu đối với một gen i nào đó các giá trị của nó trong cặp cá thể cha-mẹ của quần thể là giống nhau. Khi đó bằng toán tử crossover như xây dựng ở trên, gen i trong cá thể con cũng mang cùng giá trị chung của cha-mẹ. Vì vậy việc tìm kiếm bị hạn chế tại một nửa không gian xác định bởi giá trị chung của gen i trong cặp cá thể cha-mẹ. Từ đó thuật toán gen có thể hội tụ về nghiệm tối ưu cục bộ không mong muốn. Về mặt trực quan, "thuần chủng" và "cận huyết thống" ở một góc độ nào đó là những khái niệm có hiệu ứng đối nghịch nhau nhưng không mâu thuẫn mà bổ xung cho nhau. Trong thuật toán gen, cặp toán tử crossover và đột biến phần nào phản ánh tính chất đó. Toán tử đột biến giúp mở rộng không gian tìm kiếm, hay nói cách khác, tạo nên tính "đa dạng sinh học" cho phép thuật toán thoát khỏi các điểm tối ưu cục bộ. Trong quá trình hình thành các quần thể, cặp toán tử crossover và đột biến được áp dụng một cách thích hợp để cân đối giữa "thuần chủng" và "đa dạng sinh học".

Nếu quần thể không tồn tại lược đồ thì với mọi $i,\ 1 \leq i \leq n,\ H_i \neq 0$. Trong trường hợp tồn tại lược đồ, ta có thể chia nhiễm sắc thể thành các khối riêng biệt mà trong mỗi khối riêng biệt đó không tồn tại lược đồ. Điều này bảo đảm Entropy của các gen nằm trong từng khối đều khác 0 và toán tử đột biến được áp dụng riêng rẽ trong từng khối. Do vậy không mất tính tổng quát và để thuận tiện cho việc trình bày ta luôn có thể coi với mọi $i,\ 1 \leq i \leq n$. Khi đó ta xác định xác suất đột biến của từng gen $i,\ 1 \leq i \leq n$, theo công thức sau

$$P_{mutation}(i) = \frac{1/H_i}{\sum_{j=1}^{n} 1/H_j}. \tag{10}$$

Toán tử đột biến được áp dụng cho các cá thể con P do toán tử crossover tạo ra. Các gen có xác suất đột biến lớn hơn $1/n$ sẽ được chọn để gây đột biến theo quy tắc đảo bit sau

1. Nếu $P_{mutation}(i) > 1/n,\ P[i] = 1,\ p_1(i) > p_0(i)$ thì đặt $P[i] = 0$. (11)
2. Nếu $P_{mutation}(i) > 1/n,\ P[i] = 0,\ p_0(i) > p_1(i)$ thì đặt $P[i] = 1$. (12)

e. Khôi phục tính chấp nhận được của nghiệm

Khi áp dụng các toán tử crossover hoặc đột biến lên các cá thể, tính chấp nhận được của nghiệm có thể bị vi phạm (tức là tập hợp nhận được không còn là một phủ). Để khắc phục hiện tượng này, chúng ta đưa ra một thuật toán heuristic khôi phục tính chấp nhận được của nghiệm và đồng thời cũng là một bước tối ưu cục bộ nâng cao hiệu quả chung của thuật toán gen.

Khi một cá thể được sinh ra không thỏa mãn tính chất chấp nhận được, ta gọi M' là tập con của tập các hàng M trên đó tính chấp nhận được của nghiệm bị vi phạm (tức là các dòng không được phủ bởi bất cứ cột nào nằm trong nghiệm). Đối với mỗi cột $j \in Q$ không nằm trong nghiệm ta xác định R_j là tập các hàng được phủ bởi cột j. Khi đó các cột sẽ được lựa chọn để bổ xung vào nghiệm theo thứ thự tăng dần của tỷ số:

$$\frac{c_j}{\|R_j \cap M'\|} \tag{13}$$

ở đây c_j là giá của cột j và $\|R_j \cap M'\|$ là lực lượng của tập hợp $R_j \cap M'$. Quá trình bổ xung kết thúc khi mọi dòng trong M' được phủ hết.

Quá trình bổ xung như trên khôi phục tính chấp nhận được của nghiệm nhưng đôi khi có thể gây nên tính dư thừa. Một cột được gọi là dư thừa nếu như việc loại bỏ cột đó ra khỏi nghiệm không làm vi phạm tính chấp nhận được. Thuật toán heuristic sau đây cho phép chúng ta đồng thời khôi phục tính chấp nhận được và loại bỏ tính dư thừa của nghiệm. Kết quả của thuật toán chính là nghiệm tối ưu cục bộ cho bài toán.

Đặt:

M : tập tất cả các hàng;

Q : tập tất cả các cột;

R_j :tập tất cả các hàng được phủ bởi cột $j,\ j \in Q$.

K_i : tập tất cả các cột phủ hàng $i,\ i \in M$;

S : tập tất cả các cột nằm trong nghiệm;

M' : tập tất cả các hàng không được phủ;

w_i : số lượng cột trong nghiệm phủ hàng $i,\ i \in M$.

Các bước của thuật toán:

Bước 1: Khởi tạo $w_i := \|S \cap K_i\|,\ \forall i \in M$.

Bước 2: Khởi tạo $M' := \{i | w_i = 0,\ \forall i \in M\}$.

Bước 3: Với mỗi hàng i theo thứ tự tăng nằm trong M'

a. Tìm cột j đầu tiên theo thứ tự tăng nằm trong K_i cực tiểu hóa (13).

b. Bổ xung j vào S và đặt

$$w_i := w_i + 1,\ \forall i \in R_j,$$

$$M' := M' \backslash R_j.$$

Bước 4: Với mỗi cột j theo thứ tự giảm trong S, nếu $w_i > 1$ với mọi $i \in R_j$ ta đặt

$$S := S - j,\ w_i := w_i - 1,\ \forall i \in R_j.$$

Bước 5: S là nghiệm chấp nhận được không chứa cột dư thừa và thuật toán kết thúc.

Trong thuật toán trên các bước 1 và 2 xác định các hàng không được phủ.

Bước 3 và 4 là heuristics "tham lam" theo nghĩa trong bước 3 những cột có tỷ số giá thấp sẽ được ưu tiên đưa vào nghiệm và bỏ đi những cột dư thừa với giá cao nhất trong bước 4 (do ban đầu chúng ta đã giả thiết sắp xếp các cột theo thứ tự giá tăng dần).

f. Kích thước và mô hình thay thế quần thể

Khi các cá thể con được tạo ra, chúng sẽ được dùng để thay thế quần thể cũ theo nguyên tắc thay thế trạng thái ổn định (Steady-State Replacement). Các cá thể có mức độ thích nghi lớn hơn mức thích nghi trung bình sẽ là các ứng viên bị thay thế bởi các cá thể có mức độ thích nghi nhỏ hơn theo công thức (4).

Trong quá trình thay thế quần thể theo nguyên tắc thay thế trạng thái ổn định, chúng ta cần tránh sự xuất hiện lặp của cá thể trong quần thể.

Kích thước quần thể N do người dùng lựa chọn. Trong tính toán thử nghiệm chúng ta chọn $N = 100$.

g. Thuật toán gen giải bài toán tìm phủ tối thiểu

Từ các phân tích trên chúng ta có thể đề xuất thuật toán gen để giải bài toán tìm phủ tối thiểu với các bước sau:
Bước 1: Tạo quần thể ban đầu gồm N cá thể ngẫu nhiên.
Bước 2: Chọn 2 cá thể P_1 và P_2 trong quần thể theo phương pháp đấu loại (tournament selection method) sử dụng xác suất lựa chọn được định nghĩa trong (5).
Bước 3: Áp dụng toán tử crossover (6)-(7)-(8) lên cặp cá thể P_1 và P_2 để tạo nên cá thể mới P.
Bước 4: Gây đột biến trên P bằng toán tử đột biến xác định bởi (10) và (11)-(12).
Bước 5: Áp dụng thuật toán khôi phục tính chấp nhận được lên P để thu được cá thể P' là chấp nhận được và không dư thừa. Nếu trùng với cá thể nào đó trong quần thể thì quay lại bước 2. Ngược lại thực hiện bước tiếp theo.
Bước 6: Thay thế P' cho một cá thể được chọn ngẫu nhiên và có mức độ thích nghi lớn hơn trung bình trong quần thể.
Bước 7: Lặp lại các bước 2-6 cho tới khi không có cá thể mới được sinh ra trong quần thể. Khi đó nghiệm của bài toán là cá thể có mức độ thích nghi nhỏ nhất trong quần thể.

2. KẾT QUẢ TÍNH TOÁN

Để đánh giá hiệu quả của thuật toán mới, có thể có 3 cách tiếp cận:

1. Chứng minh bằng lý thuyết những tính chất của thuật toán.

2. So sánh hiệu quả của thuật toán (thời gian, độ chính xác...) so với các thuật toán khác trên cùng một tập hợp các bài toán và các điều kiện thực nghiệm hoàn toàn giống nhau.

3. Kiểm chứng trực tiếp hiệu quả của thuật toán trên các bài toán mẫu mà nghiệm (hoặc nghiệm xấp xỉ) của bài toán đã biết trước.

Với các thuật toán heuristic cách tiếp cận thứ nhất hầu như không thể tiến hành được.

Cách tiếp cận thứ hai đòi hỏi sự thống nhất giữa các nhóm nghiên cứu về điều kiện thực nghiệm và bài toán chung để tiến hành so sánh. Trong hầu hết các công trình đã được công bố, những vấn đề này thường được mô tả rất tóm tắt không đủ thông tin để nhóm có thể thử nghiệm bằng thuật toán của mình và đưa ra các so sánh cần thiết.

Chính vì những lý do trên, cách tiếp cận thứ ba được lựa chọn để đánh giá hiệu quả của thuật toán đã đề xuất. Cụ thể là sử dụng các bài toán kiểm tra của thư viện OR Library [5]. Thư viện này gồm 65 bài toán với kích thước, độ phức tạp khác nhau cùng với nghiệm chính xác hoặc nghiệm xấp xỉ tốt nhất tìm được cho đến thời điểm hiện tại. Kết quả có được trong thư viện là do áp dụng nhiều thuật giải khác nhau. Các bài toán trong thư viện OR Library được chia thành 11 lớp có ký hiệu là 4, 5, 6, A, B, C, D, E, F, G và H. Những bài toán thuộc các lớp 4-6 và A-D là những bài toán có nghiệm tối ưu biết trước còn các bài toán thuộc các lớp E-H là các bài toán có kích thước lớn và chưa biết nghiệm tối ưu. Để phục vụ cho việc đánh giá hiệu quả của thuật toán được đề xuất trong bài này, chúng ta sử dụng nghiệm tối ưu của các bài toán 4-6 và A-D trong [2] và nghiệm chấp nhận được của các bài toán E-H là nghiệm do Jacobs và Brusco đưa ra trong [3] trên cơ sở áp dụng áp dụng thuật toán Simulated annealing-based heuristic. Chi tiết của các bài toán được cho trong Bảng 1.

Bảng 1. Cấu trúc các bài toán trong thư viện OR Library

Lớp bài toán	Số lượng hàng	Số lượng cột	Số bài toán
4	200	1000	10
5	200	2000	10
6	200	1000	5
A	300	3000	5
B	300	3000	5
C	400	4000	5
D	400	4000	5
E	500	5000	5
F	500	5000	5
G	1000	10000	5
H	1000	10000	5

Do kết quả của việc áp dụng thuật toán gen trong thực tế phụ thuộc vào việc tạo lập quần thể ban đầu [12] nên với mỗi bài toán chúng ta lần lượt thử nghiệm với 10 quần thể ban đầu khác nhau được chọn ngẫu nhiên có cùng kích thước là 100. Kết quả tính toán được đưa ra trong Bảng 2 với cấu trúc như sau:

Cột 1(BT) : Số hiệu bài toán đưa vào đánh giá.

Cột 2(Opt): Nghiệm tối ưu của các bài toán thuộc lớp 4-6 và A-D hoặc nghiệm chấp nhận được cho các bài toán thuộc lớp E-H.

Cột 3 đến cột 12(T_i): chứa kết quả của các lần thử nghiệm tương ứng với các quần thể ban đầu khác nhau. Nếu phép thử cho kết quả trùng với nghiệm tối ưu (hoặc nghiệm chấp nhận được) thì trong cột tương ứng sẽ chứa ký hiệu (x).

Cột 13 (%): Chứa độ lệch trung bình theo % của các kết quả nhận được trong 10 lần thử so với nghiệm tối ưu (hoặc nghiệm chập nhận được) của cột 2 theo công thức sau:

$$\sum_{i-1}^{10}(S_{T_i} - S_0)/10 \times 100\%$$

trong đó S_{T_i} là nghiệm của lần thử thứ T_i và S_0 là nghiệm tối ưu (hoặc nghiệm chấp nhận được).

Bảng 2. Kết quả tính toán thử nghiệm

BT	Opt	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	%
4.1	429	x	x	x	432	x	430	430	430	x	430	0.2
4.2	512	x	x	x	x	x	x	x	x	x	x	0.0
4.3	516	x	x	x	x	x	x	x	x	x	x	0.0
4.4	494	x	x	x	x	x	502	x	x	x	x	0.2
4.5	512	x	x	x	x	x	x	x	x	x	x	0.0
4.6	560	x	x	x	x	x	x	x	x	x	x	0.0
4.7	430	x	x	432	x	x	x	x	x	x	x	0.05

4.8	492	493	x	x	x	x	x	x	x	x	x	0.02
4.9	641	x	645	x	x	x	x	x	650	645	645	0.3
4.10	514	x	x	x	x	x	x	x	x	x	x	0.0
5.1	253	x	x	x	x	x	x	x	x	x	x	0.0
5.2	302	305	x	305	x	305	x	x	305	305	x	0.5
5.3	226	228	x	228	228	228	228	228	228	228	228	0.9
5.4	242	x	x	x	x	243	x	x	x	x	x	0.04
5.5	211	x	x	x	x	x	x	x	x	x	x	0.0
5.6	213	x	x	x	x	x	x	x	x	x	x	0.0
5.7	293	x	x	x	x	x	x	x	x	x	x	0.0
5.8	288	289	289	289	x	289	289	289	289	289	x	0.3
5.9	279	x	x	x	x	x	x	x	x	x	x	0.0
5.10	265	x	x	x	x	x	x	x	x	x	x	0.0
6.1	138	x	x	x	x	x	x	x	x	x	x	0.0
6.2	146	x	147	x	x	x	x	x	x	x	147	0.1
6.3	145	x	x	x	x	x	x	x	x	x	x	0.0
6.4	131	x	x	x	x	x	x	x	x	x	x	0.0
6.5	161	x	x	x	x	x	x	164	x	x	x	0.3
A.1	253	x	254	x	x	x	x	x	x	x	254	0.2
A.2	252	x	x	x	x	x	x	x	x	x	x	0.0
A.3	232	x	x	x	x	233	233	x	233	233	233	0.5
A.4	234	x	x	x	x	x	x	x	x	x	x	0.0
A.5	236	x	x	x	x	x	x	x	x	x	x	0.0
B.1	69	x	x	x	x	x	x	x	x	x	x	0.0
B.2	76	x	x	x	x	x	x	x	x	x	x	0.0
B.3	80	x	x	x	x	x	x	x	x	x	x	0.0
B.4	79	x	x	x	x	x	x	x	x	x	x	0.0
B.5	72	x	x	x	x	x	x	x	x	x	x	0.0
C.1	227	x	x	229	x	x	x	x	x	x	x	0.2
C.2	219	x	x	221	221	221	221	x	x	x	221	1.0
C.3	243	x	248	245	247	251	248	248	244	247	x	1.4
C.4	219	x	x	x	220	x	x	x	x	x	x	0.1
C.5	215	x	x	x	x	x	x	216	x	x	x	0.1
D.1	60	x	x	x	x	x	x	x	x	x	x	0.0
D.2	66	x	x	x	x	x	67	x	x	x	x	0.1
D.3	72	x	73	x	x	x	73	x	x	x	x	0.2
D.4	62	x	x	x	x	x	x	x	x	x	x	0.0
D.5	61	x	x	x	x	x	x	63	x	x	x	0.2
E.1	29	x	x	x	x	x	x	x	x	x	x	0.0
E.2	30	31	x	31	31	x	x	31	31	31	x	0.1
E.3	27	28	28	x	28	28	28	x	28	28	28	0.1
E.4	28	x	x	x	x	x	x	x	x	x	x	0.0
E.5	28	x	x	x	x	x	x	x	x	x	x	0.0
F.1	14	x	x	x	x	x	x	x	x	x	x	0.0
F.2	15	x	x	x	x	x	x	x	x	x	x	0.0

F.3	14	x	x	x	x	x	x	x	x	x	x	0.0
F.4	14	x	x	x	x	x	x	x	x	x	x	0.0
F.5	14	x	13	x	13	x	x	x	x	x	13	-2.0
G.1	179	178	178	x	176	178	178	178	176	178	167	-0.7
G.2	158	155	155	157	x	155	x	157	x	155	155	-1.0
G.3	169	x	166	168	168	168	x	167	168	168	168	-0.6
G.4	172	171	170	170	174	x	168	171	171	x	171	-0.5
G.5	168	x	x	169	170	174	169	169	169	169	169	0.8
H.1	64	x	x	x	x	x	x	x	x	x	x	0.0
H.2	64	x	x	x	x	x	x	x	x	x	x	0.0
H.3	60	59	x	59	59	59	59	59	59	59	59	-1.5
H.4	59	x	60	58	58	58	x	60	x	60	58	-0.2
H.5	55	x	x	x	56	x	x	x	x	x	x	0.2

Kết quả chỉ ra trong bảng cho thấy với các bài toán thuộc các lớp 4-6 và A-D, thuật toán cho nghiệm tối ưu trong ít nhất một lần thử và độ lệch trung bình dao động không quá 1,5%. Đối với các bài toán thuộc các lớp E-H, thuật toán cho nghiệm tốt hơn trong ít nhất một lần thử cho từng bài toán. Điều này cho thấy những thông tin ngữ cảnh được mô phỏng trong các toán tử gen mới đã nâng cao đáng kể tính ổn định cũng như độ chính xác của thuật toán đã đề xuất.

3. KẾT LUẬN VÀ HƯỚNG NGHIÊN CỨU

Bài bào trình bày một thuật toán heurisctic để giải bài toán tìm phủ tối thiểu có giá không thuần nhất dựa trên thuật toán gen với các toán tử crossover và đột biến mô phỏng các tính chất khác nhau của sự tiến hóa trong thế giới tự nhiên như tính “trội”, tính “thuần chủng” và tính “đa dạng sinh học”. Kết quả tính toán cho thấy thuật toán có thể cho nghiệm tối ưu đối với các bài toán có kích thước nhỏ và nghiệm chất lượng tốt đối với các bài toán kích thước lớn.

Các kết quả trên sẽ được áp dụng để giải một biến thể của bài toán phủ tập hợp phát biểu trong [1] nhằm nâng cao hiệu quả của hệ thống đã được xây dựng trong lĩnh vực tối ưu hoá sản xuất.

TÀI LIỆU THAM KHẢO

[1] Phan Thi Hoai Phuong et al., A model combining genetic algorithm and simplex method for solving a production expense minimizing problem, *Journal of Computer Science and Cybernetics* **22** (4) (Hanoi, Vietnam, 2006) 319–324 (in Vietnamese).

[2] J.E. Beasley and K. Jornsten, Enhancing an algorithm for set covering problems, *European Journal of Operational Research* **58** (1992) 293–300.

[3] L.W. Jacobs and M.J. Brusco, “A simulated annealing-based heuristic for the set covering problem”. Working paper, Operations Management and Information Systems Department, Northern Illinois University, Dekalb, IL 60115, USA, 1993.

[4] M.R. Garey and D.S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[5] J.E. Beasley, OR-Library: distributing test problems by electronic mail, *Journal of the Operational Research Society* **41** (1990) 1069–1072.

[6] A. Capara et all., Algorithms for railway crew management, *Mathematical Programming* **79** (1997) 125–141.

[7] S. Ceria et all., Set Covering Problem. In Dell'Amico, F. Maffioli and S. Martello (eds.), *Annotated Bibliographies in Combinatorial Optimization*, Wiley and Sons, 1997 (415–428).

[8] Alberto Caprara, Matteo Fischetti, Paolo Toth, "Algorithms for the set covering problem". Working Paper, DEIS, University of Bologna, 1998.

[9] Alberto Caprara, Matteo Fischetti, Paolo Toth, A heuristic method for the set covering problem, *Operation Research* **47** (1999) 730–743.

[10] Kakuzo Iwamura, Norio Okaday, Yozo Deguchiz, Recent advancements of a genetic algorithm to solve the set covering problem, *RIMS Kokyuroku* **1325** (2003) 51–56.

[11] Philippe Galinier and Alain Heztz, Solution techniques for the large set covering problem,*Discrete Applied Mathematics* **155** (3) (2007).

[12] Guanghui Lan and Gail W. DePuy, On the effectiveness of incorporating randomness and memory into a multi-start metaheuristic with application to the Set Covering Problem, *Computers and Industrial Engineering* **51** (3) (2006).

[13] G. Lan, G.W. DePuy, and G.E. Whitehouse, An effective and simple Heuristic for the set covering problem, *European Journal of Operational Research* **176** (3) (2007) 1387–1403.

Nhận bài ngày 3 - 10 - 2007
Nhận lại sau sửa ngày 5 - 8 -2008