# MỘT THUẬT TOÁN DI TRUYỀN HIỆU QUẢ CHO BÀI TOÁN LẬP LỊCH JOB SHOP

**Nguyễn Hữu Mùi*, Vũ Đình Hòa**

*Khoa Công nghệ Thông tin, Trường Đại học Sư phạm Hà Nội*

*Email: *muithu@yahoo.com*

Đến Tòa soạn: 20/5/2011; Chấp nhận đăng: 12/12/2012

## TÓM TẮT

Bài báo này trình bày một thuật toán di truyền mới cho bài toán lập lịch job shop (Jobshop Scheduling Problem - JSP). Thuật toán mới này có một số đổi mới sau đây: Một lịch biểu được mã hoá bởi các số tự nhiên; các luật ưu tiên của Giffler và Thompson được dùng để tạo ra các lịch biểu tích cực; phép đột biến được thực hiện trên các cá thể tiềm năng và kết hợp với kĩ thuật tìm kiếm lân cận; phép trao đổi chéo mới kết hợp trao đổi chéo đồng nhất với thuật toán GT và được thực hiện trên 3 cá thể cha.

Dựa trên phương pháp đã đề nghị, chúng tôi đã thực hiện một chương trình tìm lịch biểu tối ưu gần đúng cho JSP. Chương trình đã chạy trên các bài toán test chuẩn do Muth và Thompson đề nghị. Phương pháp được đề nghị của chúng tôi ưu điểm hơn hẳn so với các phương pháp khác ở chỗ thời gian tính toán và tỉ lệ các lời giải tối ưu có thể tìm được. Để minh chứng cho điều đó, trong mục 4 chúng tôi trình bày các kết quả thử nghiệm và so sánh các kết quả của chúng tôi với các kết quả của Yamada.

*Từ khóa*: lập lịch job shop, lập lịch, thuật toán di truyền.

## 1. GIỚI THIỆU

Lập lịch là một chủ đề quan trọng thuộc lĩnh vực vận trù học xuất hiện từ đầu những năm 1950. Mục tiêu của lập lịch là phân phối tài nguyên dùng chung một cách hiệu quả nhất cho các tác vụ đồng thời trong toàn bộ thời gian xử lí. Các bài toán lập lịch xuất hiện trong nhiều lĩnh vực khác nhau như trong: Sản xuất, chăm sóc sức khỏe, giáo dục - đào tạo, xử lí tính toán, vận tải,... Một mô hình điển hình về lập lịch đó là bài toán lập lịch job shop (JSP), bài toán này có tầm quan trọng đặc biệt trong cả lí thuyết lẫn thực tiễn. Bài toán lập lịch job shop có nhiều khả năng được đề xuất và nghiên cứu lần đầu tiên bởi Akers và Fiedman [1]. Trong các tài liệu của Liên Xô cũ, JSP thường được gọi là bài toán Akers-Friedman hay bài toán AF. Bài toán này thuộc lớp *NP*-hard và nổi tiếng là một trong những bài toán tối ưu tổ hợp khó tính toán nhất được biết cho tới nay. Người ta đã chứng minh được rằng chỉ có một số ít trường hợp của JSP có thể giải được trong thời gian đa thức sau đây:

1. JSP 2 máy với số công đoạn của mỗi công việc không quá 2 ($J2|op \leq 2$), được giải trong thời gian $O(n.logn)$. Thuật giải cho trường hợp này do Jackson [2] đề xuất bằng cách áp dụng có mở rộng thuật toán của Johnson cho bài toán Flow Shop -FSP 2 máy.

2. JSP 2 máy với thời gian xử lí mỗi công đoạn của tất cả các công việc đều bằng 1 ($J2|p_{ij} = 1$), bài toán này có thể giải được trong thời gian $O(n)$ bởi một thuật toán do Hefetz và Adiri [3] đề xuất.

3. Akers [4] đã chứng minh rằng JSP chỉ có 2 công việc ($J| n = 2$) có thể được xem như là một bài toán đường đi ngắn nhất và vì thế có thuật giải thời gian đa thức.

4. Dựa trên ý tưởng của Kravchenko và Sotskov [5], Brucker [6] đưa ra thuật toán thời gian đa thức cho JSP với 2 máy và $k$ công việc ($J2|n = k$) với $k$ là hằng số.

Ngoài các trường hợp kể trên, các trường hợp còn lại của JSP đều là *NP*-hard. Đây là kết quả nghiên cứu của Sotskov và Shaklevich [7], bảng 1 tổng kết về độ phức tạp của JSP.

*Bảng 1*. Độ phức tạp tính toán của JSP.

| | | Số *m* máy | | | |
|---|---|---|---|---|---|
| | | 2 | 3 | Hằng số | Tuỳ ý |
| Số *n* công việc | 2 | *P* | *P* | *P* | *P* |
| | 3 | *P* | *NP*-hard | *NP*-hard | *NP*-hard |
| | Hằng số | *P* | *NP*-hard | *NP*-hard | *NP*-hard |
| | Tuỳ ý | *NP*-hard | *NP*-hard | *NP*-hard | *NP*-hard |

Thuật toán di truyền (Genetic Algorithm - GA) phỏng theo các quá trình tiến hoá tự thích nghi của các quần thể sinh học để tối ưu hoá các hàm mục tiêu. Thuật toán này lần đầu tiên được phát triển bởi John Holland [8] và đã được áp dụng rất thành công trong nhiều lĩnh vực khác nhau của khoa học máy tính bao gồm: Học máy, lí thuyết điều khiển và tối ưu tổ hợp,... GA được đặc trưng bởi chiến lược tìm kiếm dựa trên quần thể và 3 toán tử di truyền: Chọn lọc, đột biến và trao đổi chéo. Nakano và Yamada [9] là những người đầu tiên đã áp dụng GA cổ điển dùng phép biểu diễn các lời giải theo số nhị phân cho JSP. Sau đó họ còn đề xuất phương pháp kết hợp GA với các kĩ thuật tìm kiếm khác và đã thu được thành tựu đáng kể trong việc chinh phục JSP. Ulder và những người khác [10] là những người đầu tiên đề xuất phương pháp tìm kiếm cục bộ di truyền cho JSP, phương pháp này lai ghép giữa GA và các kĩ thuật tìm kiếm cục bộ. Gần đây hơn, các nhà nghiên cứu có xu hướng kết hợp GA với nhiều kĩ thuật tìm kiếm khác để tạo ra các giải pháp mạnh cho bài toán phức tạp này. Chẳng hạn như: Lee Hui Peng và những người khác [11], F. Guerriero [12], Rui Zhang và những người khác [13], Yamada [14], ...

Sau nhiều năm nghiên cứu về GA và JSP, chúng tôi nhận thấy rằng phương pháp kết hợp GA với các kĩ thuật tìm kiếm khác để tạo ra các giải pháp mạnh cho JSP là một hướng nghiên cứu còn nhiều tiềm năng. Đã có nhiều công trình nghiên cứu theo hướng này trong những năm gần đây, tuy nhiên cho tới nay vẫn chưa có giải pháp nào giải quyết triệt để JSP. Trong bài báo này chúng tôi trình bày một thuật toán di truyền lai mới cho JSP. Trong phương pháp mới này, chúng tôi đề xuất một số điểm mới trong việc dùng GA kết hợp với các kĩ thuật khác để giải JSP sau đây:

1. Một lịch biểu được mã hoá bởi các số tự nhiên. Phương pháp mã hóa này có ưu điểm hơn hẳn các phương pháp mã hóa đã sử dụng trước đây vì nó tạo điều kiện thuận lợi cho việc thực thi các toán tử di truyền và đơn giản hóa trong cài đặt chương trình.

2. Sử dụng chiến lược tìm kiếm lân cận trong phép đột biến, ở đây một lân cận là một lời giải thu được sau một thay đổi nhỏ trong phép đột biến gien. Cải tiến này khắc phục yếu điểm không hội tụ của GA đơn thuần.

3. Đề xuất một toán tử trao đổi chéo mới, toán tử này kết hợp trao đổi chéo đồng nhất dùng các luật ưu tiên của Giffler và Thompson và được thực hiện trên 3 cá thể cha. Cải tiến này nhằm tạo ra một cá thể con mang các thuộc tính tốt của nhiều cá thể cha và vẫn là một lịch biểu tích cực.

4. Toán tử chọn lọc mới được tiến hành theo 2 bước:

Bước 1: Chọn 1 cá thể có độ thích nghi tốt nhất kể từ thế hệ đầu cho tới thế hệ hiện tại cho thế hệ tiếp theo (cá thể này gọi là phần tử tinh hoa) để đảm bảo cho tính hội tụ của thuật toán.

Bước 2: Các cá thể còn lại được chọn ngẫu nhiên theo nguyên tắc bánh xe xổ số để đảm bảo được thuộc tính ngẫu nhiên của GA.

## 2. MÔ TẢ BÀI TOÁN LẬP LỊCH JOB SHOP

Bài toán lập lịch job shop có thể được mô tả vắn tắt như sau:

Cho $n$ công việc $\{J_i\}_{1 \le i \le n}$ được xử lí ở trên một tập $m$ máy $\{M_j\}_{1 \le j \le m}$ và có các đặc trưng sau đây:

1. Tại một thời điểm mỗi máy chỉ có thể xử lí một công việc.

2. Mỗi máy $M_j$ tùy ý có đều có khả năng thực hiện một công việc $J_i$ nào đó, phần công việc $J_i$ trên máy $M_j$ này được gọi là công đoạn $O_{ij}$. Với mỗi $J_i$ trình tự thực hiện các công đoạn lần lượt trên các máy được ấn định trước và được gọi là tuần tự công nghệ.

3. Thời gian bắt đầu xử lí và thời gian hoàn thành việc xử lí công đoạn $O_{ij}$ được kí hiệu lần lượt là $s_{ij}$ và $c_{ij}$. Thời gian xử lí công đoạn $O_{ij}$ được kí hiệu là $p_{ij}$.

4. Thời gian hoàn thành tất cả các công việc được gọi là makespan và được kí hiệu là $C_{max}$: $C_{max} = maxc_{ij}$.

Để minh họa cho bài toán, một ví dụ về JSP $3 \times 3$ được cho trong bảng 2. Dữ liệu vào bao gồm tuần tự công nghệ của các máy cho mỗi công việc và thời gian xử lí mỗi công việc ở trên mỗi máy (trong dấu ngoặc đơn).

*Bảng 2.* JSP 3 công việc, 3 máy.

| Công việc | Máy (thời gian xử lí) | | |
|---|---|---|---|
| 1 | 1 (3) | 2 (3) | 3 (3) |
| 2 | 1 (2) | 3 (3) | 2 (4) |
| 3 | 2 (3) | 1 (2) | 3 (1) |

Theo bảng 2, các công đoạn của $J_1$ được xử lí theo thứ tự $O_{11}$, $O_{12}$, $O_{13}$; các công đoạn của $J_2$ được xử lí theo thứ tự $O_{21}$, $O_{23}$, $O_{22}$; các công đoạn của $J_3$ được xử lí theo thứ tự $O_{32}$, $O_{31}$, $O_{33}$. Bảng 2 có thể được thay bởi một ma trận tuần tự công nghệ $\{T_{ik}\}$ và một ma trận thời gian xử lí $\{p_{ik}\}$:

$$\{T_{ik}\} = \begin{vmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{vmatrix} \qquad \{p_{ik}\} = \begin{vmatrix} 3 & 3 & 3 \\ 2 & 3 & 4 \\ 3 & 2 & 1 \end{vmatrix}$$

## 3. THUẬT TOÁN DI TRUYỀN MỚI CHO BÀI TOÁN LẬP LỊCH JOB SHOP

Qua nghiên cứu các công trình đã công bố về áp dụng GA cho JSP chúng tôi nhận thấy rằng các kĩ thuật lai chiếm ưu thế hơn các kĩ thuật sử dụng GA đơn thuần. GA đang được kết hợp khá phổ biến với các phương pháp tìm kiếm khác và đã tạo ra các phương pháp lai khá hiệu quả cho JSP. Điểm lại các công trình được công bố trong những năm gần đây về các giải pháp cho JSP, có lẽ số các công trình có liên quan tới áp dụng GA là nhiều nhất. Điều này cho thấy xu hương và vai trò của kĩ thuật tìm kiếm này hiện nay và trong tương lai. Trong bài báo này chúng tôi đề xuất một thuật toán di truyền với nhiều đổi mới cho JSP. Tính vượt trội của thuật toán được đánh giá thông qua thử nghiệm sẽ được trình bày trong mục 4 của bài báo.

### 3.1. Mã hoá lời giải

Giả sử có n công việc được xử lí trên m máy. Số công đoạn của công việc thứ $i$ được kí hiệu là job[$i$] (không quá m với mọi $i$). Tổng số các công đoạn cần được xử lí của tất cả các công việc là: $L = \sum_{i=1}^{n} job[i]$.

Chúng ta mã hoá các công đoạn của $J_1$ từ 1 đến job[1], của $J_2$ từ job[1] + 1 đến job[1] + job[2],..., của $J_n$ từ job[1] + job[2] + ... + job[$n$-1] +1 đến $L$. Như vậy một lời giải là một hoán vị nào đó của dãy số tự nhiên {1, 2, 3, ..., $L$} thoả mãn các ràng buộc của bài toán.

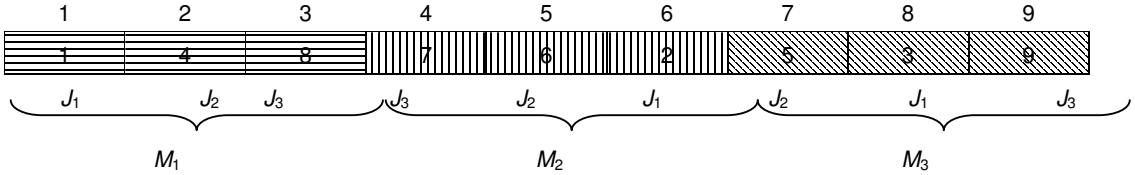
Ví dụ, với bài toán 3 công việc, 3 máy đã cho trong bảng 2. Các công đoạn được mã hoá bằng các số tự nhiên như trong bảng 3:

*Bảng 3*. Mã hoá các công đoạn.

| Công việc | Mã hoá công đoạn | | |
|---|---|---|---|
| $J_1$ | 1 | 2 | 3 |
| $J_2$ | 4 | 5 | 6 |
| $J_3$ | 7 | 8 | 9 |

*Giải thích:* theo $\{T_{jk}\}$, các công đoạn đầu tiên của $J_1$ và $J_2$ được xử lí trên $M_1$, công đoạn thứ 2 của $J_3$ được xử lí trên máy 1. Vì vậy, mã của các công đoạn trên $M_1$ là một hoán vị nào đó của {1, 4, 8}. Tương tự các công đoạn trên $M_2$ là một hoán vị nào đó của {2, 6, 7}, trên $M_3$ là một hoán vị nào đó của {3, 5, 9}.

Một lời giải có thể của bài toán được biểu diễn bằng hình vẽ có dạng như hình 1.




*Hình 1*. Một lời giải hợp lệ cho JSP 3 × 3.

Lời giải trong hình 1 cũng có thể được biểu diễn bởi một ma trận lời giải $S_{jk}$. Trong đó, $S_{jk} = i$, tức là hoạt động thứ $k$ trên máy $M_j$ là của $J_i$.

$$\{S_{jk}\} = \begin{vmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \end{vmatrix}$$

### 3.2. Khởi tạo tập lời giải ban đầu

Để sinh ra một tập lời giải ban đầu bao gồm các lịch biểu tích cực cho JSP được cho bởi ma trận tuần tự công nghệ $\{T_{ik}\}$, và ma trận thời gian xử lí $\{p_{ik}\}$, chúng ta sử dụng thuật toán GT được đề xuất bởi B. Giffler and G. L. Thompson [15].

### 3.3. Xây dựng hàm thích nghi

Hàm thích nghi được xây dựng như sau:

fitness = $M$ - $C_{max}$, trong đó $C_{max}$ là makespan của lời giải, $M$ là tham số được đưa vào để chuyển bài toán tìm min của $C_{max}$ thành bài toán tìm max (vì GA chỉ áp dụng trực tiếp cho bài toán tìm max).

### 3.4. Toán tử đột biến

+ Chọn ngẫu nhiên một công đoạn (kí hiệu là *ope*1) trong cá thể cha. Xác định máy thực hiện công đoạn đó (kí hiệu là $M_{ope1}$) và vị trí của công đoạn đó (kí hiệu là *pos*1).

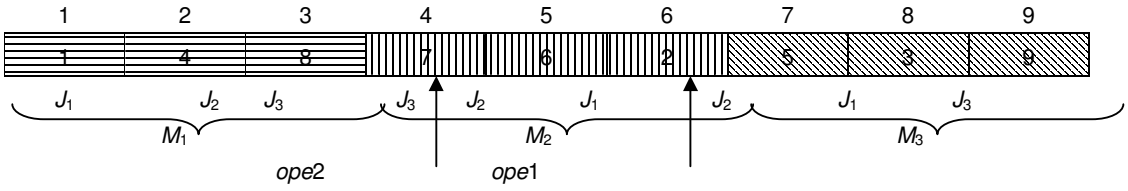
+ Chọn ngẫu nhiên một công đoạn (kí hiệu là *ope*2) trong cá thể cha. Xác định máy thực hiện công đoạn đó (kí hiệu là $M_{ope2}$) và vị trí của công đoạn đó (kí hiệu là *pos*2).

+ Nếu $M_{ope1} = M_{ope2}$ thì tiến hành đột biến (hoán đổi vị trí của hai công đoạn). Kết quả cho chúng ta cá thể con. Trong trường hợp $M_{ope1} \neq M_{ope2}$ thì cá thể cha được giữ nguyên.

+ Cập nhật lại thời gian bắt đầu và thời gian hoàn thành của tất cả các công đoạn trong cá thể con.

+ Cá thể con chỉ được chấp nhận khi có độ thích nghi tốt hơn cá thể cha hoặc số lần đột biến lại vượt quá ngưỡng cho phép (do người lập trình qui định). Mỗi cá thể con thu được sau phép đột biến có thể xem như là một lân cận của cá thể cha. Toán tử đột biến được tiến hành đồng thời trên tất cả các máy.

Ví dụ, cá thể cha được chọn để đột biến như trong hình 2:

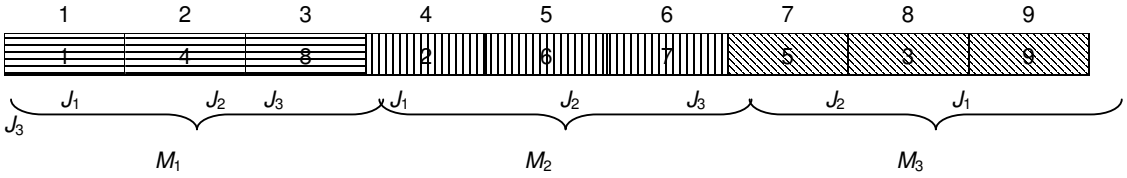



*Hình 2*. Cá thể cha cho phép đột biến.

+ Chẳng hạn: $ope1 = 2$ ---> $M_{ope1} = 2$ và $pos1 = 6$.

+ $ope2 = 7$ ---> $M_{ope2} = 2$ và $pos2 = 4$.

+ $M_{ope1} = M_{ope2}$ ---> hoán đổi các công đoạn ở vị trí 4 và vị trí 6 cho nhau, cá thể con sau khi đột biến được biểu diễn trong hình 3.




*Hình 3*. Cá thể con thu được sau phép đột biến.

### 3.5. Toán tử lai ghép hay còn gọi là toán tử trao đổi chéo

Toán tử lai ghép được thực hiện trên 3 cá thể cha $p_1$, $p_2$ và $p_3$ được biểu diễn bởi các ma trận lời giải tương ứng $S^1 = \{S^1_{jk}\}$, $S^2 = \{S^2_{jk}\}$ và $S^3 = \{S^3_{jk}\}$. Các gien trong cá thể con $p = \{S_{jk}\}$ sẽ được tái kết hợp từ các gien trong 3 cá thể cha. Trong bài báo này, chúng tôi đề xuất một toán tử lai ghép mới kết hợp đồng thời của phép lai ghép đồng nhất, thuật toán GT và được thực hiện trên 3 cá thể cha để tăng tính đa dạng của cá thể con. Hơn nữa, do sử dụng thuật toán GT nên sau khi lai ghép cá thể con vẫn là một lịch biểu tích cực.

Phép lai ghép này sử dụng thuật toán GT nên đa số các bước giống như trong thuật toán GT [15]. Sự khác biệt giữa thuật toán GT và phép lai ghép của chúng tôi chủ yếu là ở bước 4.

Toán tử lai ghép mới được đề xuất bởi chúng tôi bao gồm các bước sau:

1. Khởi tạo $G$ là tập các công đoạn đầu tiên trong tuần tự công nghệ của tất cả các công việc (cột đầu tiên của ma trận $\{T_{ik}\}$), tức là $G = \{ O_{1T_{11}}, O_{2T_{21}}, ..., O_{nT_{n1}} \}$. Đối với mỗi công đoạn $O \in G$, $ES(O) := 0$ và $EC(O) := p(O)$.

2. Tìm công đoạn hoàn thành sớm nhất $O_{*j} \in G$. Một tập con của $G$ chứa các công đoạn được xử lí ở trên máy $M_j$ kí hiệu là $G_j$.

3. Tính tập cạnh tranh $C[M_j, k] \subset G_j$, ở đây $k$ - 1 là số các công đoạn đã được lập lịch trên máy $M_j$.

4. Chọn một trong các cha $\{p_1, p_2, p_3\}$ theo giá trị của $H_{ji}$, $p := p_{H_{ij}}$ và $S^p = S^{H_{ij}}$. Đối với mỗi $O_{ij} \in C[M_j, k]$, tồn tại một chỉ số $l$ sao cho $S_{jl} = i$. Gọi $l_m$ là chỉ số nhỏ nhất, tức là $l_m = \min \{l / S_{jl} = i$ và $O_{ij} \in C[M_j, k]\}$. Gọi $r := S_{jlm}$, $O_{rj}$ sẽ được chọn để lập lịch ở trong $p$ sớm nhất trong các thành viên của $C[M_j, k]$.

5. Lập lịch cho $O_{rj}$ là công đoạn thứ $k$ trên máy $M_j$; tức là $S_{jk} := r$, với thời gian bắt đầu và thời gian hoàn thành của nó là $ES(O_{rj})$ và $EC(O_{rj})$: $s(O_{rj}) = ES(O_{rj})$; $c(O_{rj}) = EC(O_{rj})$.

6. Đối với tất cả các công đoạn $O_{ij} \in G_j \setminus \{O_{rj}\}$:

- Cập nhật $ES(O_{ij})$ như sau: $ES(O_{ij}) := \max\{ES(O_{ij}), EC(O_{rj})\}$.

- Cập nhật $EC(O_{ij})$ như sau: $EC(O_{ij}) := ES(O_{ij}) + p(O_{ij})$.

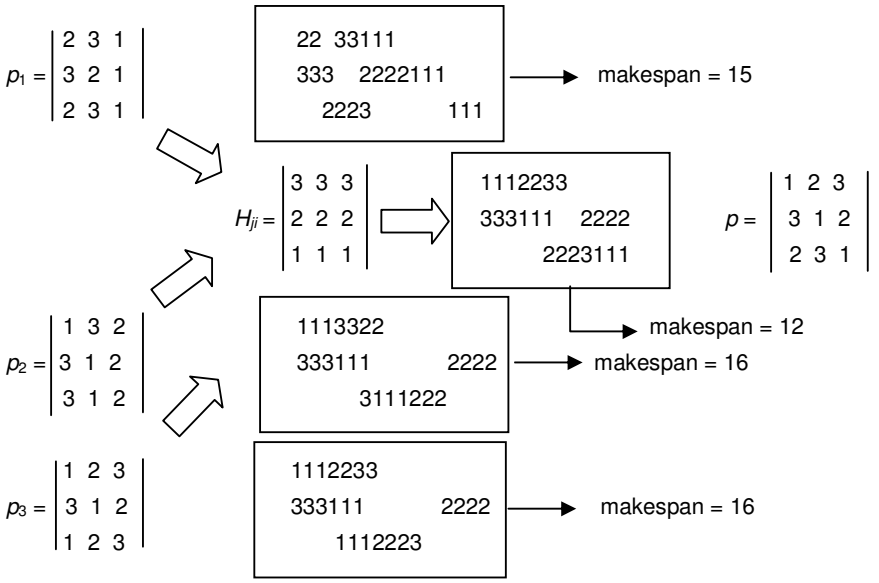
7. Xoá $O_{rj}$ khỏi $G$ (và do đó khỏi $G_j$), và bổ sung thêm công đoạn $O_{rs}$ kế tiếp $O_{rj}$ trong tuần tự công nghệ vào $G$ nếu nó tồn tại. Tức là, nếu $j = T_{ik}$ và $k < m$, thì $s := T_{i,k+1}$ và $G := (G\backslash\{O_{rj}\} \cup \{O_{rs}\}$. Tính $ES(O_{rs})$ và $EC(O_{rs})$ như sau:

- $ES\ (O_{rs}) := \max\{EC(O_{rj}), EC(PM(O_{rs}))\}$.

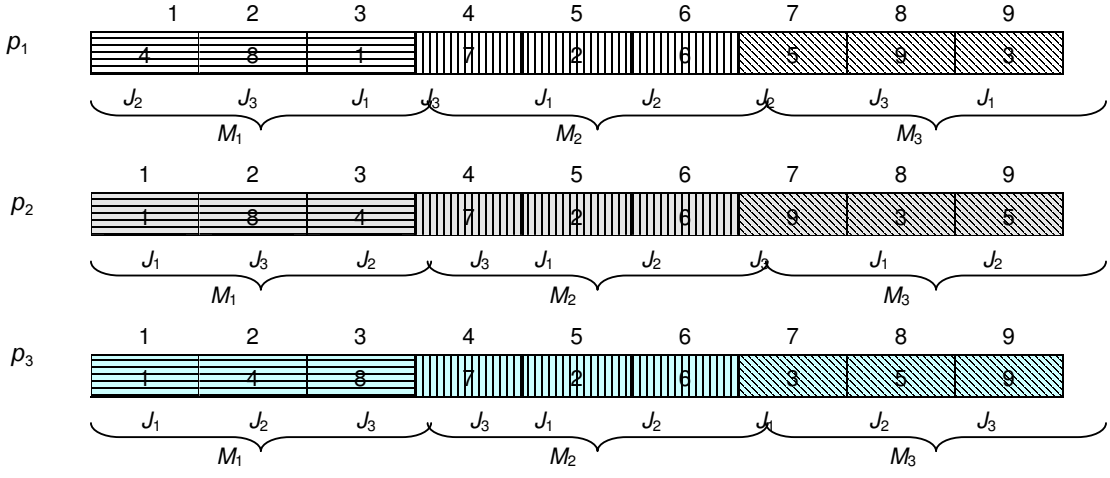
- $EC(O_{rs}) := ES\ (O_{rs}) + p(O_{rs})$.

8. Lặp lại từ bước 2 đến bước 7 cho tới khi tất cả các công đoạn được lập lịch trong cá thể con $p$.

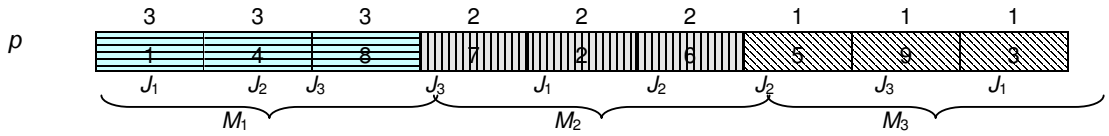
9. Ma trận lời giải ra $\{S_{jk}\}$là lịch biểu tích cực thu được với tập thời gian bắt đầu và thời gian hoàn thành là $\{s(O_{ij})\}$ và $\{c(O_{ij})\}$. Ở đây $i = S_{jk}$.




*Hình 4*. Trao đổi chéo đồng nhất áp dụng thuật toán GT trên 3 cá thể cha.




*Hình 5*. Các cha tham trao đổi chéo.




*Hình 6*. Ma trận ngẫu nhiên và cá thể con tương ứng sau lai ghép.

Hình 4 trình bày một ví dụ về trao đổi chéo đồng nhất sử dụng thuật toán GT và được áp dụng cho ba cha $p_1$, $p_2$ và $p_3$ với một ma trận ngẫu nhiên $H_{ji}$. Con $p$ là kết quả của phép trao đổi chéo. Các cha tham gia trao đổi chéo và cá thể con thu được có thể biểu diễn bằng hình vẽ như hình 5 và hình 6.

### 3.6. Toán tử chọn lọc

Toán tử chọn lọc chọn pop_size cá thể cho thế hệ $t + 1$ được tiến hành như sau:

1. Chọn cá thể có độ thích nghi tốt nhất kể từ thế hệ đầu đến thế hệ $t$.
2. Xây dựng tập lời giải trung gian $P'(t)$ áp dụng phép đột biến và phép lai ghép:

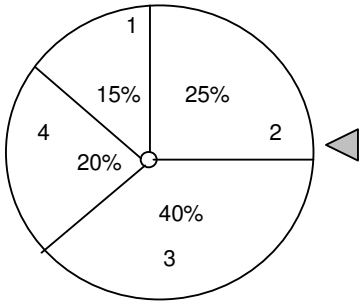
+ Áp dụng toán tử đột biến đối với $P(t)$ được $P_1(t)$

+ Áp dụng toán tử trao đổi chéo đối với $P(t)$ được $P_2(t)$

+ $P'(t) = P(t) \cup P_1(t) \cup P_2(t)$

3. Chọn pop_size -1 cá thể còn lại trong tập $P'(t)$ theo nguyên lí bánh xe sổ xố.

Bánh xe xổ số được minh họa trong hình 7, lời giải 1 có xác suất chọn lọc là 0.15. Như vậy, mỗi lần quay bánh xe xổ số nó có khả năng được chọn là 15%. Tương tự, xác suất chọn lọc của các lời giải 2, 3, 4 tương ứng là 0.25, 0.40, và 0.20.




*Hình 7*. Bánh xe xổ số.

### 3.7. Thuật toán di truyền cho JSP

```
Procedure GA_JSP
Begin
    t ← 0                    {t là số thế hệ tiến hóa}
    Khởi tạo P(t)            {dùng thuật toán GT}
    Đánh giá P(t)            {tính hàm thích nghi của mỗi cá thể trong P(t)}
```

while ( not điều kiện dừng ) do

Begin

Xây dựng tập lời giải $P'(t)$ bằng cách áp dụng phép đột biến và phép lai ghép:

+ Áp dụng toán tử đột biến đối với $P(t)$ được $P_1(t)$

+ Áp dụng toán tử trao đổi chéo đối với $P(t)$ được $P_2(t)$

+ $P'(t) = P(t) \cup P_1(t) \cup P_2(t)$

Đánh giá $P'(t)$

$t \leftarrow t + 1$

Chọn lọc $P(t)$ từ $P'(t\text{-}1)$

End

End

- ***Tính đúng đắn của thuật toán***

Tính đúng đắn của thuật toán được khẳng định thông qua các đặc trưng sau đây:

1. Do sử dụng thuật toán GT để sinh ra các lịch biểu, cho nên mỗi cá thể con được sinh ra đều là một lịch biểu hợp lệ và hơn nữa nó còn là một lịch biểu tích cực.

2. Trong phép đột biến, cá thể tham gia đột biến được sửa đổi bằng cách thay đổi thứ tự sắp xếp trong một máy nào đó, sau đó thời gian bắt đầu và thời gian kết thúc của toàn lịch biểu được cập nhất lại nên cá thể con sau đột biến vẫn đảm bảo là một lịch biểu hợp lệ.

3. Phép trao đổi chéo sử dụng thuật toán GT để sinh ra các lịch biểu con, cho nên mỗi cá thể con được sinh ra đều là một lịch biểu hợp lệ và hơn nữa chúng vẫn còn là lịch biểu tích cực.

4. Thuật toán sẽ dừng khi gặp điều kiện kết thúc hay khi đạt kết quả tối ưu.

5. Vì thuật toán luôn duy trì lời giải tốt nhất trong quần thể, trước hoặc sau khi chọn lọc, vì vậy thuật toán được chứng tỏ hội tụ tới tối ưu toàn cục nhờ thuộc tính không giảm của các thế hệ tiếp theo. Kết quả này được thảo luận với định lí lược đồ và được chứng minh bởi Günter Rudolph [16].

- ***Đánh giá độ phức tạp của thuật toán***

Trong vòng lặp while, độ phức tạp của toán tử trao đổi chéo là cao nhất. Vì vậy, theo quy tắc xác định độ phức tạp thì độ phức tạp của các hoạt động trong vòng lặp while tùy thuộc vào độ phức tạp của phép trao đổi chéo. Độ phức tạp của phép trao đổi chéo có thể được xác định ngắn gọn như sau:

- Số lần chọn thao tác cho mỗi lời giải con là $n \times m$, với $n$ là số công việc và $m$ là số máy.

- Số các cá thể con trong phép trao đổi chéo < pop_size, với pop_size là số cá thể trong quần thể.

Vậy, độ phức tạp của phép trao đổi chéo là: $O(n \times m \times \text{pop\_size})$.

Số lần lặp của vòng lặp while là $t$. Vậy độ phức tạp của GA_JSP là: $O(t \times n \times m \times \text{pop\_size})$.

## 4. KẾT QUẢ THỬ NGHIỆM

Dựa vào phương pháp đề xuất trong bài báo này, chúng tôi đã cài đặt một chương trình bằng ngôn ngữ C# và chạy thử nghiệm trên máy PC với bộ vi xử lí Core 2 Dual có tốc độ 3.0 GHz. Chúng tôi sử dụng dữ liệu vào cho chương trình là các bài toán benchmark do Muth và G. L. Thompson [17] đề nghị. Đây là các bài toán test nổi tiếng về khó giải quyết và đã biết kết quả tối ưu.

*Bảng 4*. Kết quả chạy thử nghiệm.

| Bài toán | Cỡ lời giải | Số thế hệ | Sác xuất trao đổi chéo ($p_c$) | Sác xuất đột biến ($p_m$) | Kết quả chạy | Tối ưu thực sự |
|---|---|---|---|---|---|---|
| *mt*06 ($6 \times 6$) | 50 | 200 | 0.8 | 0.05 | 55 | 55 |
| *mt*10 ($10 \times 10$) | 500 | 200 | 0.8 | 0.05 | 930 | 930 |
| *mt*20 ($20 \times 5$) | 1000 | 200 | 0.8 | 0.05 | 1185 | 1165 |

Bộ tham số và kết quả chạy thử nghiệm chương trình do chúng tôi đề nghị được trình bày trong bảng 4. Kết quả tối ưu của bài toán *mt*06 tìm được rất nhanh chỉ sau vài lần chạy thử. Với bài toán *mt*10 kết quả tối ưu tìm được trung bình là 1 lần sau 100 lần chạy thử.

Trong các công trình nghiên cứu áp dụng GA cho JSP, các phương pháp do Yamada đề suất cho tới nay vẫn được xem là một trong những phương pháp cho kết quả tốt nhất. Vì vậy, chúng tôi sử dụng kết quả của hai phương pháp do Yamada đề nghị là: phương pháp sử dụng thuật toán di truyền đơn (SGA - bảng 5) và phương pháp sử dụng thuật toán di truyền kết hợp với thuật toán GT (GT/GA - bảng 6) để so sánh với kết quả của phương pháp do chúng tôi đề xuất.

*Bảng 5*. Kết quả chạy thử nghiệm phương pháp SGA của Yamada

| Bài toán | Cỡ lời giải | Số thế hệ | $p_c$ | $p_m$ | Kết quả chạy | Tối ưu thực sự |
|---|---|---|---|---|---|---|
| *mt*06 | 100 | 200 | 0.9 | 0.01 | 55 | 55 |
| *mt*10 | 1000 | 200 | 0.9 | 0.01 | 965 | 930 |
| *mt*20 | 2000 | 200 | 0.9 | 0.01 | 1215 | 1165 |

*Bảng 6*. Kết quả chạy thử nghiệm phương pháp GA/GT của Yamada.

| Bài toán | Cỡ lời giải | Số thế hệ | $p_c$ | $p_m$ | Kết quả chạy | Tối ưu thực sự |
|---|---|---|---|---|---|---|
| *mt*06 | 100 | 200 | 0.9 | 0.01 | 55 | 55 |
| *mt*10 | 1000 | 200 | 0.9 | 0.01 | 930 | 930 |
| *mt*20 | 2000 | 200 | 0.9 | 0.01 | 1185 | 1165 |

Kết quả tối ưu của bài toán *mt*06 tìm được rất nhanh cho cả hai phương pháp do Yamada đề nghị. Với bài toán *mt*10 phương pháp SGA không tìm được lời giải tối ưu thực sự, phương pháp GA/GT tìm được 4 lần sau 600 lần chạy thử. Bộ tham số và kết quả chạy các phương pháp do Yamada đề nghị được trình bày trong bảng 5 và bảng 6.

Thông qua kết quả thử nghiệm có thể cho thấy phương pháp mà chúng tôi đề nghị có các điểm mạnh sau đây:

1. Do dùng mã hóa là số tự nhiên nên việc cài đặt chương trình khá đơn giản. Các sinh viên không hiểu sâu về thuật toán di truyền cũng có thể cài đặt được khi được hướng dẫn trong thời gian ngắn.

2. Thời gian tính toán giảm so với các phương pháp khác do tham số cỡ quần thể được chọn nhỏ hơn khi chạy thử nghiệm chương trình mà vẫn tìm kiếm được lời giải tối ưu. Chương trình có thể chạy được trên các máy tính có cấu hình thấp. Đây cũng là ưu điểm nổi bật của phương pháp do chúng tôi đề nghị.

3. Tỉ lệ tìm được lời giải tối ưu thực sự cao hơn phương pháp của Yamada đề nghị.

## 5. KẾT LUẬN

Một thuật toán di truyền lai cho JSP đã được đề nghị, phương pháp này có sử dụng hợp lí các kết quả của những người đi trước kết hợp với những đề xuất mới của chúng tôi. Chúng tôi đã cài đặt một chương trình xác định lịch biểu tối ưu gần đúng cho JSP để thẩm định thuật toán. Chương trình đã được chạy thử nghiệm với dữ liệu vào là các bài toán benchmark do Muth và Thompson đề nghị cho kết quả tốt. Với những bài toán test có kích cỡ nhỏ, chương trình cho kết quả tối ưu. Với những bài toán kích cỡ lớn, chương trình cho kết quả gần tối ưu. Việc cài đặt thuật toán này không mấy khó khăn do tính đơn giản của thuật toán. Qua kết quả thử nghiệm đã cho thấy những điểm mạnh của thuật toán do chúng tôi đề nghị với các phương pháp khác đó là thời gian tính toán và tỉ lệ các lời giải tối ưu tìm được. Tuy nhiên, đối với các bài toán cỡ lớn như bài toán test $20 \times 5$, chương trình vẫn chưa tìm được lịch biểu tối ưu. Đây là vấn đề được đặt ra cho những nghiên cứu tiếp theo về việc áp dụng GA cho các bài toán lập lịch. Trong thời gian tới, chúng tôi sẽ tiếp tục nghiên cứu để tìm ra những phương pháp hiệu quả hơn để có thể giải quyết các bài toán job shop có kích cỡ lớn hơn.

## TÀI LIỆU THAM KHẢO

1. Akers S. B. - A graphical approach to production scheduling problems, Operations Research **4** (1956) 244.
2. Jackson J. R. - An extension of Johnson's result on job lot scheduling, Naval Research Logistics Quarterly **3** (1956) 202-203.
3. Hefetz N. and Adiri I. - An efficient optimal algorithm for the two-machines, unit-time, Job Shop, schedule-length problem, Mathematics of Operations Reseach **7** (1982) 354-360.
4. Akers S. B. - A graphical approach to production scheduling problems, Operations Research **4** (1956) 244.
5. Kravchenko S. A. and Sotskov Y. N. - Optimal makespan schedule for three jobs on two machines, ZOR - Mathematical Methods of Operations Research **43** (1996) 233-238.

6. Brucker P. - A polynomial time algorithm for the two machines Job Shop scheduling problem with a fixed number of jobs, OR Spektrum **16** (1994) 5-7.
7. Sotskov Y. N. and Shaklevich N. V. - NP-hardness of shop-scheduling problems with three jobs, Descrete Applied Mathemayics **59** (1995) 237-266.
8. Holland J. H. - Adaptation in Natural and Artificial Systems. Unuv. of Michigan Press, 1975.
9. Nakano R. and Yamada T. - Conventional genetic algorithm for job shop problems, In Proceedings of International Conference on Genetic Algorithms (ICGA '91), 1991, pp. 474-479.
10. Ulder N. L. J., Pesch E., Van Laarhoven P. J. M., Bandelt J. H., and Aarts E. H. L. - Genetic local search algorithm for the traveling salesman problem, In Parallel Problem Solving from Nature, Vol. 1, 1994, pp. 109-116.
11. Lee Hui Peng, Sutinah Salim - A Modified Giffer and Thompson Genetic Algorithm on the job shop scheduling problem, MATEMATIKA (University Teknology Malaysia) **22** (2) (2006) 91-107.
12. Guerriero F. - Hybrid Rollout Approaches for the Job Shop Scheduling Problem, Jounal Optimization Theory and Applications **139** (2008) 419-438.
13. Rui Zhang and Cheng Wu - A hybrid approach to large-scale job shop scheduling, Application Intelligence **32** (2010) 47-59.
14. Yamada T. - studies on Metaheuristics for Jobshop and Flowshop scheduling problems, Kyoto University, Kyoto - Japan, 2003.
15. Giffler B. and Thompson G. L. - Algorithms for Solving Production Scheduling Problems, Operations Research **8** (4) (1960) 487-503.
16. Günter Rudolph - Convergence Analysis of Canonical Genetic Algorithms, IEEE Transactions on Neural Networks, special issue on evolutonary computation **5** (1) (1994).
17. Muth J. F. and Thompson G. L. - Industrial Scheduling, Prentice-Hall, Englewood Cli_s, N. J., 1963.

## ABSTRACT

AN EFFECTIVE GENETIC ALGORITHM FOR JOB SHOP SCHEDULING PROBLEM

Nguyen Huu Mui*, Vu Dinh Hoa

*Faculty of Information Technology, Hanoi University of Education*

*Email: *muithu@yahoo.com*

This paper presents an new genetic algorithm for the job shop scheduling problem - JSP. This new algorithm has its innovations follow: A schedule is encoded by natural numbers; the priority rules of Giffler and Thompson is used to create active schedules; the mutation is implemented on potential individuals and combined with neighborhood search technique; new crossover combines the uniform crossover with GT algorithm and implemented on 3 parents.

Based on the proposed method, we implement a program to find an approximation optimal schedule for JSP. The program ran on Muth and Thompson's benchmark problems. Our proposed method is preeminent in comparison with other methods, they are calculation time and ratio of the findable optimal solutions. To prove those, we present an empirical experiment and to compare our results with results of Yamada in the section 4.

*Keywords:* Jobshop Scheduling, Schedule, Genetic Algorithm.