GIẢI BÀI TOÁN LẬP LỊCH JOBSHOP BẰNG THUẬT DI TRUYỀN

1. Mô tả bài toán Job Shop tổng quát

JSP là bài toán cho một tập n công việc $\{J_i\}_{1 \le i \le n}$ được xử lý ở trên một tập m máy $\{M_j\}_{1 \le j \le m}$ và thoả mãn các ràng buộc sau đây:

1. Mỗi công việc phải được xử lý ở trên một tập các máy theo một thứ tự cho trước (thứ tự này còn được gọi là tuần tự công nghệ).

2. Tại một thời điểm mỗi máy chỉ có thể xử lý một công việc.

3. Thao tác của công việc J_i được xử lý ở trên máy M_j được ký hiệu là O_{ij}.

4. Thời gian xử lý thao tác O_{ij} của công việc J_i ở trên máy M_j được ký hiệu là p_{ij}.

5. Thời gian bắt đầu xử lý và thời gian hoàn thành việc xử lý thao tác O_{ij} được ký hiệu lần lượt là s_{ij} và c_{ij}.

6. Thời gian hoàn thành việc xử lý tất cả các công việc được gọi là makespan và được ký hiệu là C_{max}: $C_{max} = \max c_{ij}$.

Bài toán lập lịch Job Shop là trường hợp tổng quát của hai bài toán mà chúng tôi đã trình bày trước đây: FSP và PFSP là bởi vì:

1. Trình tự các thao tác của mỗi công việc được xử lý ở trên các máy (tuần tự công nghệ) có thể là khác nhau (với bài toán Flow Shop, tuần tự công nghệ của cả các công việc là như nhau).

2. Trình tự công việc được xử lý ở trên mỗi máy có thể cũng khác nhau (với bài toán Flow Shop hoán vị, trình tự này là như nhau).

Như vậy, bài toán lập lịch Flow Shop hoán vị và bài toán Flow Shop chỉ là hai trường hợp riêng của bài toán Job Shop.

Mục tiêu của bài toán lập lịch Job Shop là tìm một lịch biểu sao cho C_{max} là nhỏ nhất.

Một ví dụ về JSP 3 × 3 được cho trong bảng 1. Dữ liệu được cho bao gồm tuần tự công nghệ và thời gian xử lý mỗi công việc ở trên mỗi máy (trong dấu ngoặc đơn).

Theo bảng 1, các thao tác của công việc 1 được xử lý theo thứ tự $O_{11} \longrightarrow O_{12} \longrightarrow O_{13}$, các thao tác của công việc 2 được xử lý theo thứ tự $O_{21} \longrightarrow O_{23} \longrightarrow O_{22}$, các thao tác của công việc 3 được xử lý theo thứ tự $O_{32} \longrightarrow O_{31} \longrightarrow O_{33}$. Điều này có nghĩa là công việc 1 trước tiên được xử lý ở trên máy 1 với thời gian xử lý là 3 đơn vị thời gian, sau đó được xử lý ở trên máy 2 với 3 đơn vị thời gian và cuối cùng được xử lý ở trên máy 3 cũng với 3 đơn vị thời gian. Các công việc 2 và 3 cũng được lý giải tương tự.

Công việc	Máy (thời gian xử lý)		
1	1 (3)	2 (3)	3 (3)
2	1 (2)	3 (3)	2 (4)
3	2 (3)	1 (2)	3 (1)

Bảng 1: Bài toán lập lịch Job Shop 3 công việc, 3 máy

Bài toán trên có thể được cho một cách tương đương bởi một ma trận tuần tự công nghệ $\{T_{ik}\}$ và một ma trận thời gian xử lý $\{p_{ik}\}$ như trong hình 1.

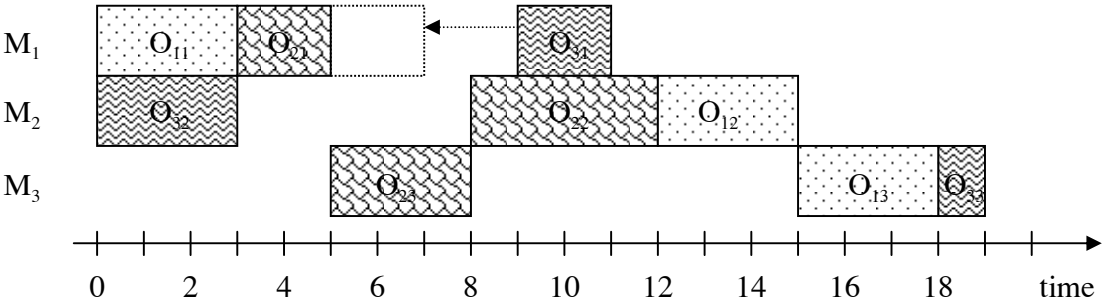
$$\{T_{ik}\} = \begin{vmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{vmatrix} \qquad \{p_{ik}\} = \begin{vmatrix} 3 & 3 & 3 \\ 2 & 3 & 4 \\ 3 & 2 & 1 \end{vmatrix}$$

Hình 1: Ma trận tuần tự công nghệ và ma trận thời gia xử lý cho bài toán lập lịch Job Shop 3 công việc, 3 máy

2. Các lịch biểu tích cực và bán tích cực

2.1. Các lịch biểu bán tích cực

Một lịch biểu được gọi là bán tích cực khi không thao tác nào có thể được bắt đầu sớm hơn mà không thay đổi thứ tự xử lý các thao tác ở trên một máy nào đó.



Hình 2: Một biểu đồ Gantt biểu diễn một lời giải cho bài toán được cho trong bảng 1

Ví dụ, thao tác O_{31} ở trên hình 2 có thể được bắt đầu xử lý tại đơn vị thời gian 5, như là được chỉ ra bởi các đường nét đứt, mà không cần thay đổi thứ tự các thao tác ở trên một máy bất kỳ, và lịch biểu mới thu được là lịch biểu bán tích cực.

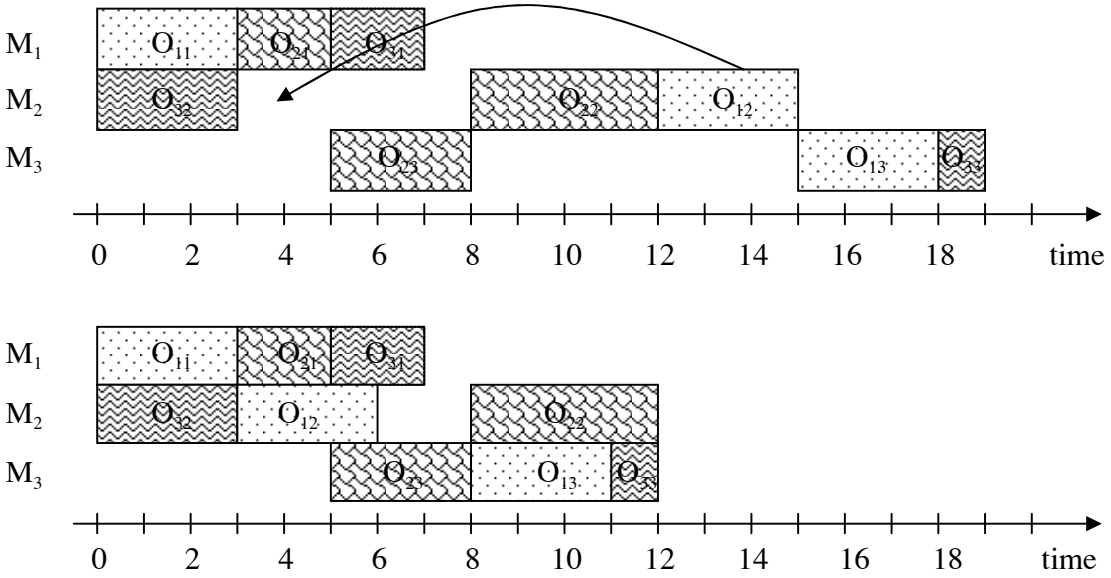
Một lịch biểu bán tích cực có thể được biểu diễn bởi một ma trận lời giải cấp $m \times n$: $S = \{S_{jk}\}$. Tức là, nếu $S_{jk} = i$ tương ứng với thao tác thứ k ở trên máy M_j là của công việc J_i. Ở đây: $1 \le j \le m$, $1 \le i \le n$, $1 \le k \le n$.

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

Hình 3: Ma trận lời giải của bài toán được biểu diễn trong hình 2

Hình 3 là ma trận biểu diễn một lời giải của bài toán 3 × 3 trong bảng 1. Trong hình này, ví dụ các thao tác ở trên máy 2 được xử lý theo thứ tự O_{32}, O_{22}, O_{12}. Cho một ma trận $S = \{S_{jk}\}$, có thể dễ ràng có được một lịch biểu bán tích cực tương ứng. Mỗi thao tác O có 2 thao tác đi trước: thao tác đi trước cùng công việc và thao tác đi trước cùng một máy. thao tác đi trước cùng công việc của O được ký hiệu là PJ(O), là thao tác ngay trước O trong tuần tự công nghệ. Thao tác đi trước cùng máy của O được ký hiệu là PM(O), là thao tác ngay trước O trong ma trận lời giải. Ví dụ, giả sử bài toán đã cho trong bảng 1 chúng ta có $PJ(O_{12}) = O_{11}$ và nếu lời giải được cho như là trong hình 3 thì $PM(O_{12}) = O_{22}$. Một thao tác O được lập lịch tại đơn vị thời gian 0 nếu nó không có các thao tác cùng công việc đi trước và cùng máy đi trước. Nếu chỉ có thao tác cùng công việc được xử lý trước hoặc cùng máy được xử lý trước thì O được lập lịch ngay sau khi thao tác xử lý trước hoàn thành, còn không nó được lập lịch khi cả hai thao tác đi trước hoàn thành tức là: $s(O) = \max\{c(PJ(O)); c(PM(O))\}$ lời giải cho trong hình 3 tương ứng với biểu đồ Grannt trong hình 2 chỉ ra rằng O_{31} có thể bắt đầu tại đơn vị thời gian 5.

2.2 Các lịch biểu tích cực



Hình 4: Một ví dụ về phép dịch trái chấp nhận được

Makespan của một lịch biểu bán tích cực có thể được giảm bớt bằng cách dịch chuyển một thao tác nào đó về bên trái mà không làm ảnh hưởng tới sự thi hành các công việc khác. Xét một lịch biểu bán tích cực S và 2 thao tác O_{ij} và O_{kj} được xử lý trên cùng

một máy M_j . Nếu O_{kj} được xử lý trước O_{ij} và máy M_j có một thời gian nghỉ dài hơn p_{ij} trước khi xử lý O_{kj}, thì một sự hoán đổi thứ tự là có thể được sao cho O_{ij} được xử lý trước O_{kj} mà không làm ảnh hưởng tới bất kỳ thao tác nào khác. Một phép hoán đổi thứ tự như thế được gọi là "một phép dịch trái chấp nhận được". Một lịch biểu mà không có một phép dịch trái nào chấp nhận được gọi là một "lịch biểu tích cực". Hình 4 trình bày một ví dụ về phép dịch trái chấp nhận được. Lịch biểu phía trên của hình 4 chính là lịch biểu trong hình 2 nhưng thao tác O_{31} được bắt đầu xử lý tại đơn vị thời gian 5 và makespan của nó là 19.

Trên máy M_2 trong lịch biểu phía trên của hình 4, O_{12} là thao tác của J_1 có thể được dịch chuyển tới ngay phía sau O_{32} mà không làm trễ bất kỳ thao tác nào khác. Sau đó thao tác O_{13} được dịch chuyển tới ngay sau 2 thao tác đi trước O_{12} và O_{23}. O_{33} ở trên máy 3 cũng được dịch chuyển. Lịch biểu thu được với makespan = 12 được biểu diễn trong hình phía dưới của hình 4.

Người ta đã chứng minh rằng luôn tồn tại một lịch biểu tối ưu là tích cực, nên chúng ta có thể giới hạn không gian tìm kiếm là tập tất cả các lịch biểu tích cực để thu hẹp không gian tìm kiếm mà vẫn đảm bảo tìm ra được lịch biểu tối ưu.

3. Thuật toán GT

Một bài toán lập lịch Job Shop được cho bởi ma trận tuần tự công nghệ $\{T_{ik}\}$ và ma trận thời gian xử lý $\{p_{ik}\}$. Thuật toán GT sinh lịch biểu tích cực cho bài toán được trình bày như sau:

1. Khởi tạo G là tập các thao tác đầu tiên trong tuần tự công nghệ (cột đầu của ma trận $\{T_{ik}\}$), tức là $G = \{O_1T_{11}; O_2T_{21}, ..., O_nT_{n1}\}$. Đối với mỗi thao tác $O \in G$, tập ES (O) := 0 và EC(O) := p(O).

2. Tìm thao tác hoàn thành sớm nhất $O_{*j} \in G$. Một tập con của G chứa các thao tác được xử lý ở trên máy M_j ký hiệu là G_j.

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

4. Chọn một thao tác trong $C[M_j, k]$ một cách ngẫu nhiên. Gọi thao tác được chọn là O_{i*j}.

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

6. Đối với tất cả các thao tác $O_{ij} \in G_j \setminus \{O_{i*j}\}$:

- Cập nhật ES(O_{ij}) như sau: ES (O_{ij}) := max{ES(O_{ij}), EC(O_{i*j})}.

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

7. Xoá O_{i*j} khỏi G (và do đó khỏi G_j), và bổ sung thêm thao tác O_{is} kế tiếp O_{i*j} 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\{O_{i*j}} $\cup$ {O_{is}}.

Tính ES(O_{is}) và EC(O_{is}) như sau:

- ES (O_{is}) := max{EC(O_{i*j}), EC(PM(O_{is}))}.

- EC(O_{is}) := ES (O_{is}) + p(O_{is}).

8. Lặp lại từ bước 2 đến bước 7 cho tới khi tất cả các thao tác được lập lịch.

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}.

Một lịch biểu tích cực có thể được sinh ra bằng cách sử dụng thuật toán GT được đề suất bởi Giffer và Thompson [12] được trình bày trên. Trong giải thuật GT có một số vấn đề cần làm rõ sau:

1. Như đã trình bày trong mục trước, một thao tác O có 2 thao tác đi trước. Thao tác đi trước cùng công việc được ký hiệu bởi PJ(O) và thao tác đi trước cùng máy PM(O). Một thao tác O chưa được lập lịch được gọi là có thể được lập lịch khi cả thao tác đi trước cùng công việc và thao tác đi trước cùng máy (nếu chúng tồn tại) đều đã được lập lịch. Tập tất cả các thao tác có thể được lập lịch được ký hiệu là G.

2. Thời gian có thể bắt đầu sớm nhất ES(O) của O được xác định bằng max thời gian hoàn thành của PJ(O) và PM(O): ES(O) = max{c(PJ(O)), c(PM(O))}. Thời gian có thể hoàn thành sớm nhất EC(O) = ES(O) + p(O).

3. Thao tác có thể hoàn thành sớm nhất O_{*j} trong G_j với máy M_j là thao tác mà EC(O_{*j}) là nhỏ nhất ở trong G_j:

$$O_{*j} = \arg\min \{EC(O) / O \in G_j\}. \quad (2.1)$$

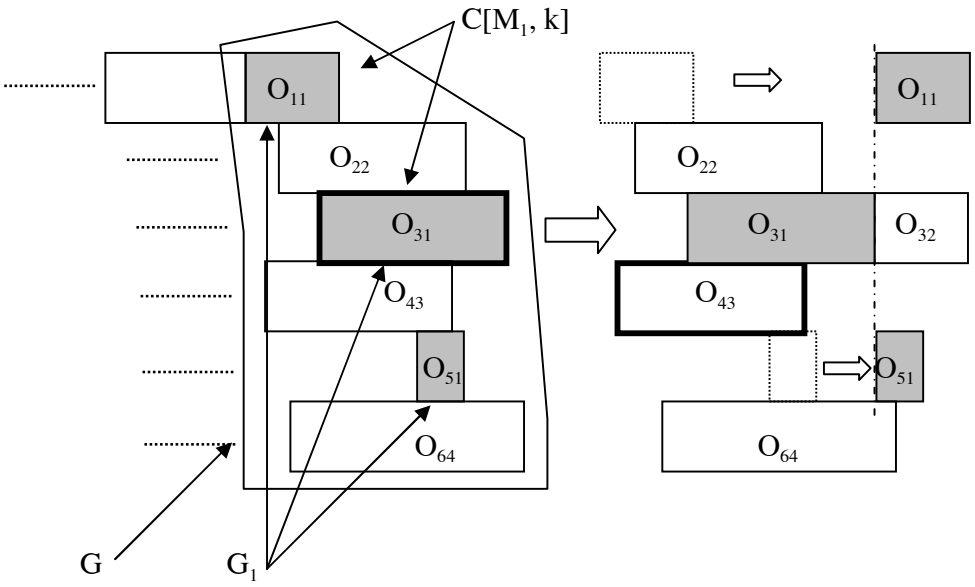
4. Cho một thao tác có thể hoàn thành sớm nhất O_{*j} và nếu có k - 1 thao tác đã được lập lịch ở trên máy M_j , một tập cạnh tranh C[M_j, k] là một tập các thao tác ứng cử viên cho việc xử lý tiếp theo ở trên máy M_j được định nghĩa như sau:

$$C[M_j, k] = \{O_{ij} \in G_j / ES(O_{ij}) < EC(O_{*j})\}. \quad (2.2)$$

lưu ý rằng $O_{*j} \in C[M_j, k]$.

Mục đích của thuật toán GT là lập lịch cho các thao tác sao cho tránh được thời gian nghỉ của máy đủ dài cho phép sự dịch trái có thể. Vì mục đích này, sự lựa chọn thao

tác lập lịch tiếp theo trong tập các thao tác có thể lập lịch G_j phải cẩn thận để không tạo ra thời gian nghỉ đủ dài. Do đó mà tập cạnh tranh $C[M_j, k] \subset G_j$ được duy trì. Chỉ cần thao tác tiếp theo được chọn từ tập cạnh tranh, một thời gian nghỉ sẽ được giữ đủ ngắn và lịch biểu kết quả được đảm bảo là tích cực. Một lịch biểu tích cực có được bởi việc lặp lại thuật toán GT cho tới khi tất cả các thao tác được lập lịch.



Hình 5: Lập lịch sử dụng thuật toán GT

Hình 5 minh hoạ cách làm việc của thuật toán GT. Hình này trình bày một đoạn giữa của quá trình lập lịch cho tất cả các thao tác. Trong hình này, O_{22}; O_{31}; O_{43}; O_{51} và O_{64} là có thể được lập lịch và cấu thành tập G. Thao tác hoàn thành sớm nhất được nhận diện là O_{11}, điều đó dẫn đến $G_1 = \{O_{11}; O_{31}; O_{51}\}$. Trong G_1 chỉ O_{11} và O_{31} thoả mãn bất đẳng thức 2.2, do đó $C[M_1, k] = \{O_{11}; O_{3j}\}$. Nếu O_{31} được chọn ngẫu nhiên từ $C[M_1, k]$, thì O_{11} và O_{51} được chuyển dịch về phía trước theo bước 6 trong thuật toán GT.

4. Áp dụng thuật toán di truyền giải bài toán lập lịch jobshop

4.1. Mã hoá lời giải

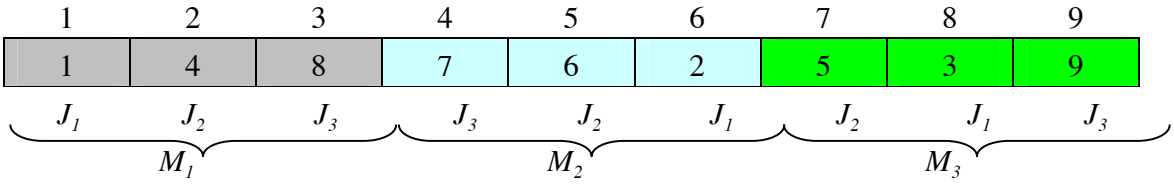
Giả sử có *n* công việc được xử lý trên *m* máy. Số công đoạn của công việc thứ *i* được ký hiệu là *job*[*i*] (không quá *m* với mọi *i*). Tổng số các thao tác cần được xử lý của tất cả các công việc là $L = \sum_{i=1}^{n} job[i]$. Chúng ta mã hoá các thao tác của J_1 từ 1 đến *job*[*1*], của J_2 từ *job*[1] + 1 đến *job*[1] + *job*[2],..., của J_n từ *job*[1] + *job*[2] + ... + *job*[*n*-1] +1 to *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 1. Các thao tác được mã hoá bằng các số tự nhiên như trong bảng 2:

Công việc	Mã hoá thao tác		
J_1	1	2	3
J_2	4	5	6
J_3	7	8	9

Bảng 2: Mã hoá các công đoạn bằng số tự nhiên

Giải thích: theo $\{T_{ik}\}$, các thao tác đầu tiên của J_1 và J_2 được xử lý trên M_1, thao tác thứ 2 của J_3 được xử lý trên máy 1. Vì vậy, mã của các thao tác trên M_1 là một hoán vị nào đó của {1, 4, 8}. Tương tự 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}.



Hình 6: Một lời giải hợp lệ cho JSP 3 × 3

Một lời giải hợp lệ cho bài toán có thể có dạng như hình 6.

Lời giải trong hình 6 có thể được biểu diễn một cách tương đương 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.

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

4.2. 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 lý $\{p_{ik}\}$, chúng ta sử dụng thuật toán GT đã được trình bày trong mục 3.

4.3. 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).

4.4. Các toán tử di truyền

a. Toán tử chọn lọc

Toán tử này nhằm chọn các lời giải tốt cho thế hệ sau dựa trên độ tốt xấu của hàm đánh giá hay sác xuất được chọn của mỗi lời giải. Trong bài báo này, chúng tôi chọn lọc các lời giải cho thế hệ sau theo nguyên lý bánh xe sổ xố.

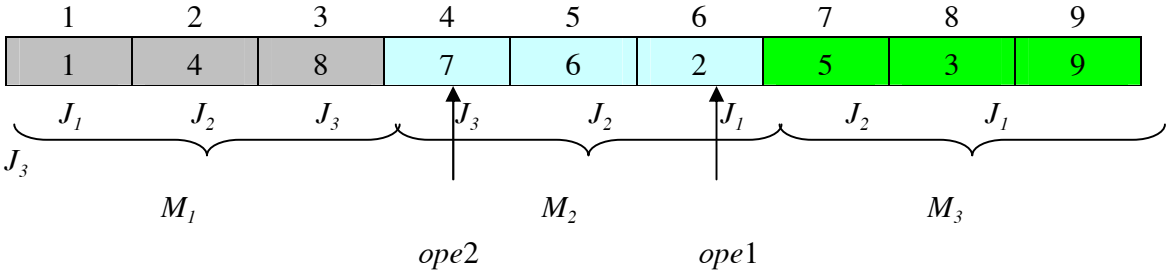
b. Toán tử đột biến

+ Chọn ngẫu nhiên một thao tác *ope*1 trong cá thể cha. Xác định máy thực hiện thao tác đó M_{ope1} nhờ hàm Findmach(*ope*1) và vị trí của thao tác đó *pos*1 nhờ hàm Findpos(*ope*1).

+ Chọn ngẫu nhiên một thao tác *ope*2 trong cá thể cha. Xác định máy thực hiện thao tác đó M_{ope2} nhờ hàm Findmach(*ope*2) và vị trí của thao tác đó *pos*2 nhờ hàm Findpos(*ope*2).

+ Nếu $M_{ope1} = M_{ope2}$ thì tiến hành đột biến (hoán đổi vị trí của hai thao tác). 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 thao tác trong cá thể con. Ví dụ, cá thể cha được chọn để đột biến hoán đổi như hình 3:

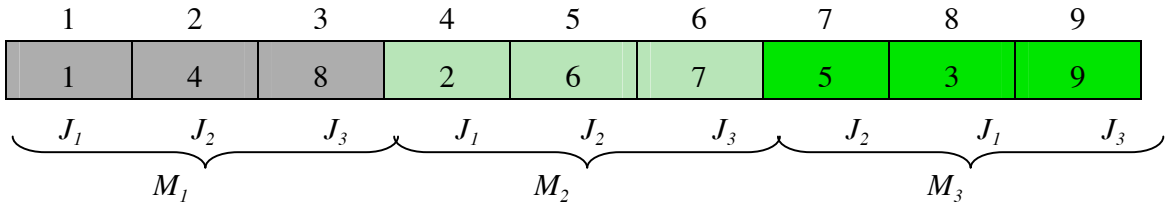


Hình 3: Cá thể cha chọn đột biến hoán đổi

+ Chẳng hạn: *ope*1 = 2 ---> M_{ope1} = 2 và *pos*1 = 6.

+ *ope*2 = 7 ---> M_{ope2} = 2 và *pos*2 = 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 hoán đổi được biểu diễn trong hình 7:

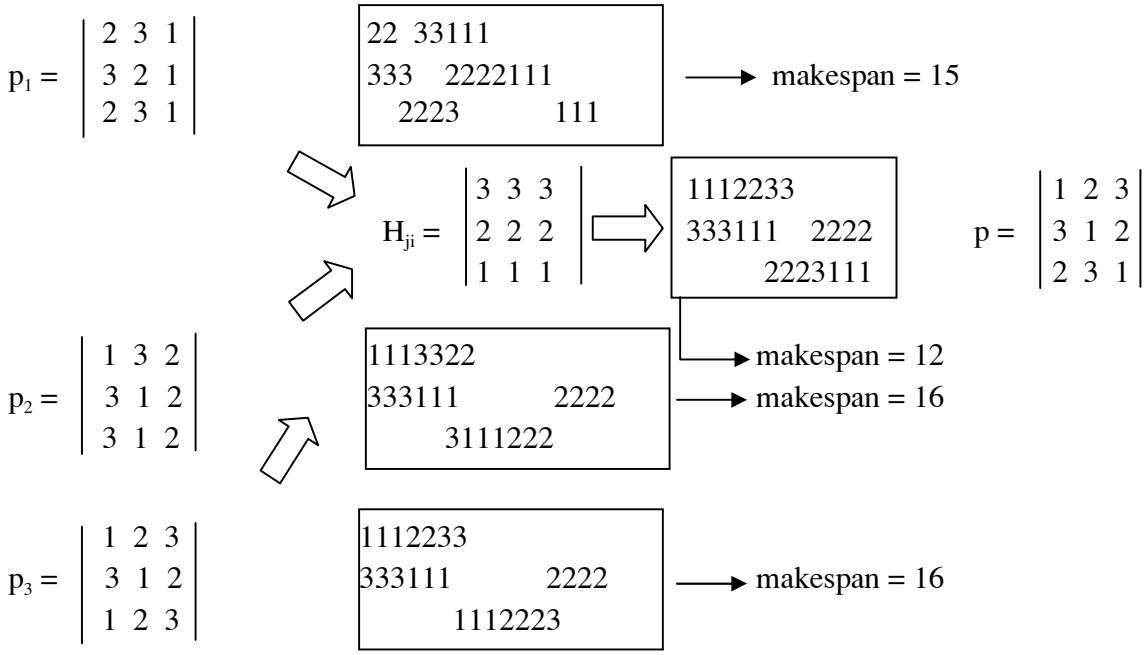


Hình 7: Cá thể con sau khi đột biến hoán đổi

Toán tử đột biến được tiến hành đồng thời trên tất cả các máy, khi đó việc chọn ngẫu nhiên được tiến hành m lần (cho m máy).

c. Toán tử lai ghép

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 là sự 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. Do sử dụng GT nên sau khi lai ghép cá thể con vẫn là lời giải tích cực.



Hình 8: Trao đổi chéo đồng nhất áp dụng thuật toán GT

Phép lai ghép được mô tả vắn tắt như 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 lý ở trên máy M_j ký hiệu là G_j. (theo 2.1)

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. (theo 2.2)

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:

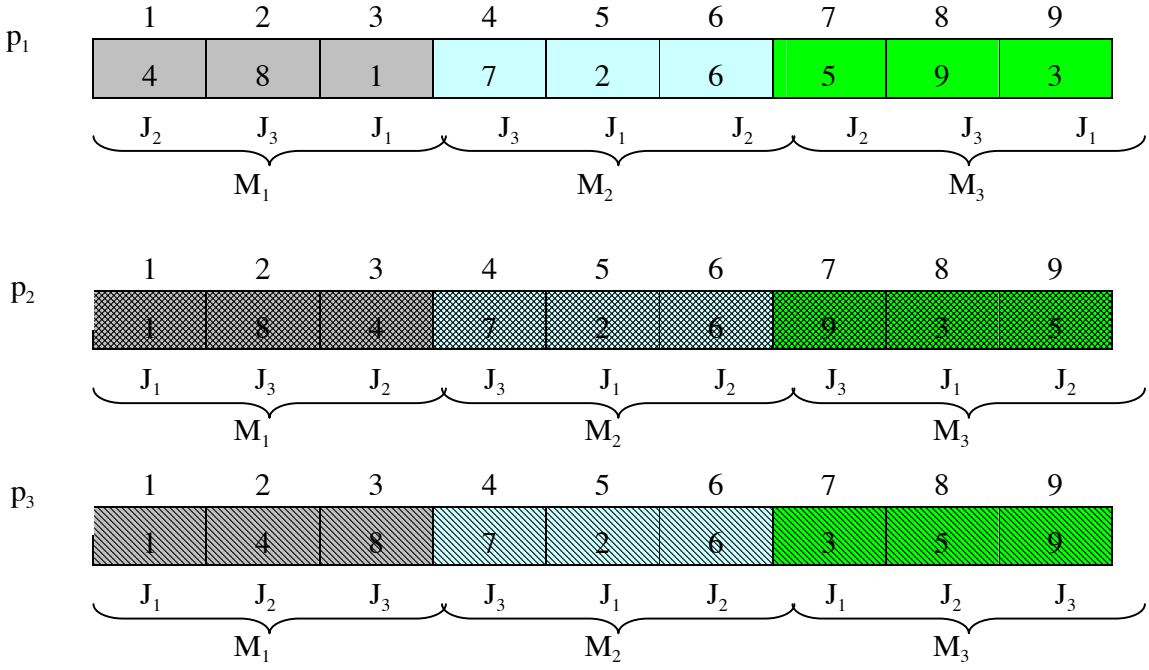
- $ES(O_{rs}) := \max\{EC(O_{rj}), EC(PM(O_{rs}))\}$.

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

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.

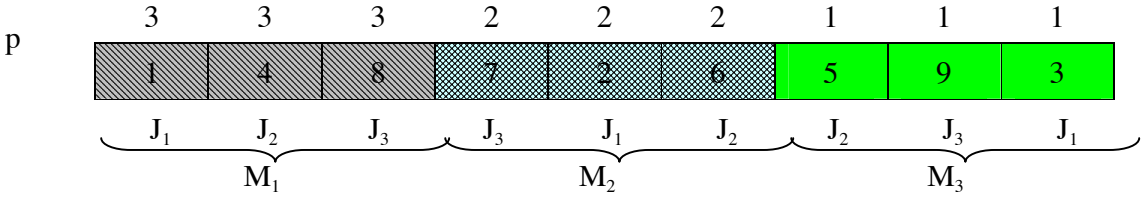
9. Ma trận lời giải ra $\{S_{jk}\}$là lịch biểu tích cực thu được với tập thời gian bắt đầu và thời gian hoàn thành là $\{s(O_{ij})\}$ và $\{c(O_{ij})\}$. ở đây $i = S_{jk}$.

Hình 9 trình bày một ví dụ về trao đổi chéo đồng nhất sử dụng thuật toán GT, 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 kết quả ra có thể được biểu diễn bằng hình vẽ như hình 9 a, b:



Hình 9a: Các cha tham gia trao đổi chéo GT

Ma trận ngẫu nhiên và cá thể con tương ứng sau lai ghép:



Hình 9b: Cá thể con sau lai ghép GT

4.5. Thuật toán tiến hoá

Thuật toán tiến hoá cho JSP được đặc tả như sau:

```
Procedure GA_JSP
Begin
        t ← 0
        Khởi tạo P(t)    {dùng thuật toán GT}
        Đánh giá P(t)
        while ( not điều kiện dừng ) do
           Begin
              Xây dựng tập lời giải trung gian P'(t):
                 + áp dụng toán tử đột biến đối với P(t) được P1(t)
                 + áp dụng toán tử trao đổi chéo đối với P(t) được P2(t)        {áp dụng GT}
                 + P'(t) = P(t) ∪ P1(t) ∪ P2(t)
              Đánh giá P'(t)
              t ← t + 1
              Chọn lọc P(t) từ P'(t-1)
           End
End
```