Tạp chí Khoa học Kỹ thuật Mỏ - Địa chất
Trang điện tử: http://tapchi.humg.edu.vn

Nghiên cứu thuật toán Matheuristic để tối thiểu hóa tổng thời gian trễ của bài toán lập lịch flow shop *m* máy

Tạ Quang Chiểu*

Khoa Công nghệ Thông tin, Trường đại học Mỏ - Địa chất, Việt Nam

THÔNG TIN BÀI BÁO

Quá trình:
Nhận bài 20/6/2017
Chấp nhận 26/7/2017
Đăng online 30/10/2017

Từ khóa:
Matheuristic
Tổng thời gian trễ
Flow shop
Di truyền

TÓM TẮT

Trong bài báo này, chúng tôi xem xét bài toán lập lịch flow shop m máy ($F_m//\sum T_j$) để tối thiểu hóa tổng thời gian trễ. Các phép toán lận cận (neighborhood operators) và thuật toán Maheuristic được đề xuất cho bài toán này. Các thuật toán Matheuristic là các thuật toán gần đúng, là kỹ thuật được thực hiện bằng cách "nhúng" các quy hoạch toán học vào trong các thuật toán metaheuristic. Các kết quả tính toán chỉ ra rằng thuật toán Matheuristic thực hiện tốt hơn thuật toán di truyền (Genetic alogorithm-GA). Trong tương lai, chúng tôi sẽ đánh giá thuật toán Matheuristic với các thuật toán khác trên cùng bài toán; nghiên cứu và ứng dụng thuật toán Matheuristic này cho các bài toán lập lịch khác.

1. Mở đầu

Trong bài báo này chúng tôi xem xét bài toán lập lịch flow shop cho *m* máy (*machine*) để tối thiểu hóa tổng thời gian trễ. Bài toán được xem xét gồm một tập $J = \{1,2,...,n\}$ của *n* công việc (*job*) được xử lý trên một tập $M = (M_1, M_2,..., M_m)$ của *m* máy (*machine*) và có đặc trưng như sau:

- Mỗi công việc *j* được xử lý trên máy thứ nhất, máy 2, máy 3,... Mỗi công việc chỉ được bắt đầu trên máy thứ M_{i+1} nếu nó được hoàn thành trên máy thứ M_i và máy thứ M_{i+1} rỗi.

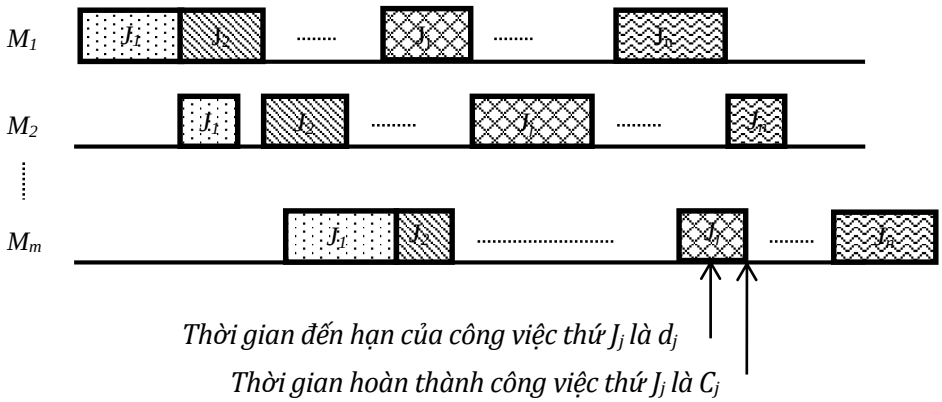
- Trình tự xử lý các công việc trên các máy M_1, $M_2,..., M_m$ là như nhau. Nghĩa là, nếu công việc thứ *i* được xử lý trên máy M_1 thì nó cũng có thứ tự xử lý thứ *i* trên máy $M_2,..., M_m$.

Chúng tôi ký hiệu $p_{i,j}$ là thời gian xử lý công việc thứ J_j trên máy M_i và d_j là thời gian đến hạn của công việc thứ J_j; C_j được là thời gian hoàn thành công việc thứ J_j; T_j là thời gian trễ của công việc J_j và được xác định $T_j = \max(0, C_j - d_j)$ với $\forall j$, $1 \le j \le n$.

Bài toán này được ký hiệu là $F_m//\sum T_j$, trong đó $\sum T_j = \sum_{j=1}^{n} T_j$, và thuộc lớp bài toán *NP-hard* và là một trong những bài toán tối ưu tổ hợp rất khó (Du and Leung, 1990; Lenstra et al., 1977). Mục tiêu của của bài toán cần được tối thiểu hóa tổng thời gian trễ $\sum T_j$ hay *Minimize* ($\sum T_j$).

Bài toán này đã có một số nhà khoa học đề xuất các phương pháp giải quyết với các tiêu chuẩn khác nhau như: Cmax (makespan), Σ Cj, Σ Tj,... Một số trong các phương pháp đã giải quyết cho trường hợp đặc biệt với số lượng máy (machine) bằng 2. Trong trường hợp này, một vài

*Tác giả liên hệ
E-mail: taquangchieu@humg.edu.vn



Hình 1. Sơ đồ Grand của bài toán Flow shop.

phương pháp chính xác (exact method) được đưa ra như: thuật toán nhánh cận (Kim, 1993a; Pan and Fan, 1997; Sen et al., 1989).

Trong (Pan et al., 2002), số lượng công việc đạt đến n = 24 và cho kết quả tối ưu. Một vài thuật toán gần đúng (*heuristic algorithm*) được đề xuất như: thuật toán tham lam (*greedy algorithm*) sử dụng các luật ưu tiên, thuật toán NEH (Nawaz et al., 1983a) hay phương pháp dịch chuyển điểm nghẽn (Koulamas, 1998) (*shifting bottleneck procedure*). Một vài phương pháp metaheuristic cũng đã được đề xuất như: thuật toán mô phỏng luyện kim (*simulated annealing*) (Osman and Laporte, 1996), thuật toán tìm kiếm Tabu (*Tabu search algorithms*) (Grabowski and Wodecki, 2004; Kim, 1993b; Nowicki and Smutnicki, 1996), thuật toán di truyền (*genetic algorithms*) (Onwubolu and Mutingi, 1999), thuật toán tối ưu bầy đàn (*particle swarm optimization*) (Liao et al., 2007; Tasgetiren et al., 2007), v.v..

Đối với bài toán lập lịch flow shop *m* máy, Onwubolu and Mutingi đã đề xuất trong (Onwubolu and Mutingi, 1999) một thuật toán di truyền để tính tổng thời gian trễ cho số lượng các công việc trễ. Trong một nghiên cứu khác của Vallada et al (Vallada et al., 2008), một vài thuật toán đã được cài đặt và so sánh để thực hiện cho bài toán flow shop với các tiêu chuẩn khác nhau.

Vài năm trở lại đây, một phương pháp gần đúng mới được đưa ra để giải quyết cho các bài toán tổ hợp tối ưu. Phương pháp này gồm các phương pháp chính xác được cài đặt "*nhúng*" bên trong các phương pháp gần đúng, nó đã cài đặt và nhận được nhiều các kết quả thú vị cho các bài toán khó (Maniezzo et al., 2010; Talbi, 2013). Các phương pháp này được gọi là *Matheuristic.* Trong (Della Croce et al., 2011), tác giả đã đề xuất một phương pháp matheuristic cho bài toán flow shop để tối thiểu hóa tổng thời gian hoàn thành của các công việc. (Ta, 2017; Ta et al., 2015, 2013), các tác giả đưa ra một phương pháp matheuristic cho bài toán lập lịch flowshop. (Pessan et al., 2008) đề xuất thuật toán nhánh cận và thuật toán di truyền để giải quyết cho bài toán lập lịch với các máy song song.

Trong bài báo này, chúng tôi đề xuất thuật toán *Matheuristic* được khởi tạo bởi thuật toán tham lam. Thuật toán này được chạy thử nghiệm với bộ dữ liệu gồm 108 mẫu được đưa ra bởi (Vallada et al., 2008). Các kết quả này được so sánh với thuật toán di truyền (*Genetic algorithm*).

Các phần tiếp theo của bài báo này được tổ chức như sau: *Phần 2*, miêu tả các phép toán lân cận (*neighborhood operator*). *Ở phần 3*, đưa ra mô hình quy hoạch tuyến tính hỗn hợp và thuật toán Matheuristic. Trong *phần 4*, trình bày các kết quả và thảo luận. Các kết luận và một vài hướng nghiên cứu trong tương lai được đề xuất *ở phần 5*.

2. Các phép toán lận cận (Neighborhood operators)

Trong phần này các phép toán lân cận được được miêu tả và sử dụng: phép toán hoán vị 2 công việc - *SWAP*, chèn trước - EBSR (*Extraction and Backward Shifted Re-insertion*) và chèn sau - EFSR (*Extraction and Forward Shifted Re-insertion*) (Della Croce et al., 2004).

S được ký hiệu là chuỗi hiện tại. Các phép toán lân cận được được ứng dụng cho chuỗi $S = S_1/ S_{[i]}/ S_2/ S_{[j]}/ S_3$, trong đó S_1, S_2 và S_3 là 3 chuỗi con của *S*, $S_{[i]}$ và $S_{[j]}$ là các công việc tại vị trí *i* và vị trí *j* trong *S* ($i \neq j$) và được thực hiện (Hình 2) như sau:

SWAP: Một lân cận của S được tạo ra bằng cách hoán vị 2 vị trí của công việc ở vị trí *i* và vị trí *j*, kết quả được chuỗi S' = $S_1/ S_{[j]}/ S_2/ S_{[i]}/ S_3$.

EBSR: Một lân cận của S được tạo ra bằng cách lấy công việc ở vị trí $S_{[j]}$ và chèn vào ngay trước công việc ở vị trí $S_{[i]}$, kết quả được chuỗi S' = $S_1/ S_{[j]} / S_{[i]}/ S_2/ S_3$.

EFSR: Một lân cận của S được tạo ra bằng cách lấy công việc ở vị trí $S_{[i]}$ và chèn vào ngay sau công việc ở vị trí $S_{[j]}$, kết quả được chuỗi S' = $S_1/ S_2/ S_{[j]} / S_{[i]}/ S_3$.

3. Mô hình quy hoạch tuyến tính hỗn hợp và thuật toán Matheuristic

Trong phần này chúng tôi đề xuất mô hình toán học quy hoạch tuyến tính hỗn hợp và thuật toán Matheuristic cho bài toán.

3.1. Mô hình quy hoạch tuyến tính hỗn hợp - MILP (Mixed Integer Linear Programming)

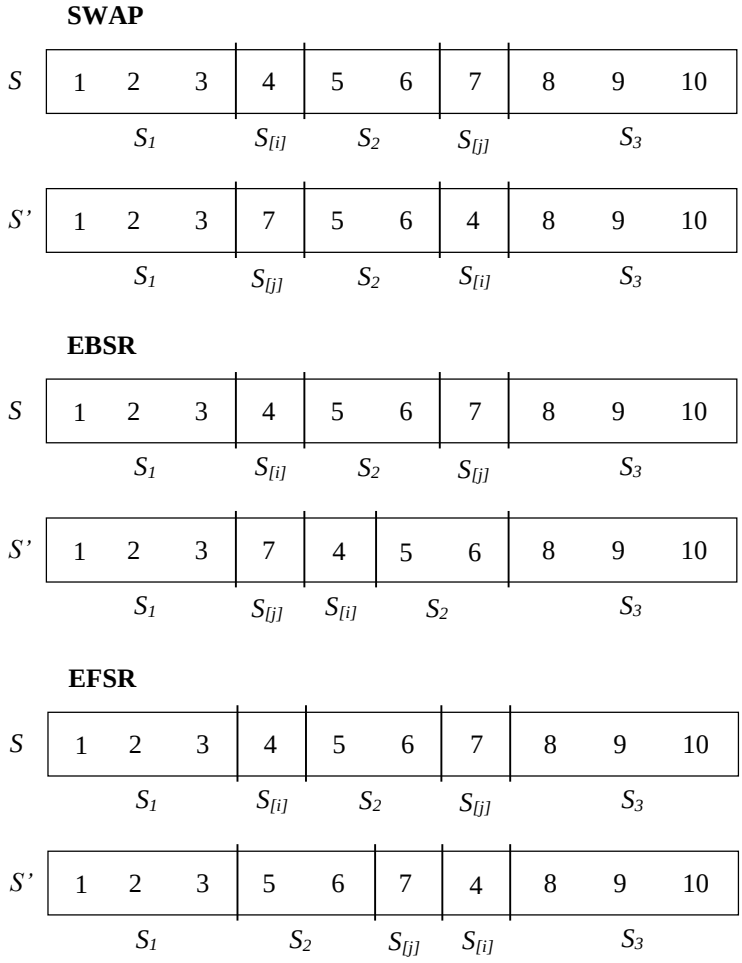
Mô MILP cho bài toán dựa vào vị trí của các biến. Biến nhị phân $x_{j,k}$ bằng 1 nếu công việc J_j ở vị trí k bên trong chuỗi và ngược lại thì bằng 0, $\forall j \in \{1,..,n\}, \forall k \in \{1,..,n\}$. Biến liên tục (continuous variable) $C_{i,k} \geq 0$ là thời gian hoàn thành của công việc ở vị trí k trên máy M_i, $\forall i \in \{1,..,m\}, \forall k \in \{1,..,n\}$ và $T_k \geq 0$ là thời gian trễ của công việc ở vị trí k, $\forall k \in \{1,..,n\}$.

Mô hình MILP như sau:

$$Minmize \sum_{k=1}^{n} T_k \quad (1)$$

$$subject\ to \sum_{k=1}^{n} x_{j,k} = 1, \quad \forall j \in \{1,\dots,n\} \quad (2)$$

$$\sum_{j=1}^{n} x_{j,k} = 1, \quad \forall k \in \{1,\dots,n\} \quad (3)$$



Hình 2. Mô tả các phép toán SWAP, EBSR, EFSR

$$C_{1,1} = \sum_{j=1}^{n} p_{1,j} x_{j,1} \quad (4)$$

$$C_{1,k} = C_{1,k-1} + \sum_{j=1}^{n} p_{1,j} x_{j,k}, \quad \forall k \in \{2, \dots, n\} \quad (5)$$

$$C_{i,1} = C_{i-1,1} + \sum_{j=1}^{n} p_{i,j} x_{j,1}, \quad \forall i \in \{2, \dots, m\} \quad (6)$$

$$C_{i,k} \geq C_{i-1,k} + \sum_{j=1}^{n} p_{i,j} x_{j,k}, \quad \forall i \in \{2, \dots, m\} \quad (7)$$

$$\forall k \in \{1, \dots, n\}$$

$$C_{i,k} \geq C_{i,k-1} + \sum_{j=1}^{n} p_{i,j} x_{j,k}, \quad \forall i \in \{2, \dots, m\} \quad (8)$$

$$\forall k \in \{1, \dots, n\}$$

$$T_k \geq C_{m,k} - \sum_{j=1}^{n} d_j x_{j,k}, \quad \forall k \in \{1, \dots, m\} \quad (9)$$

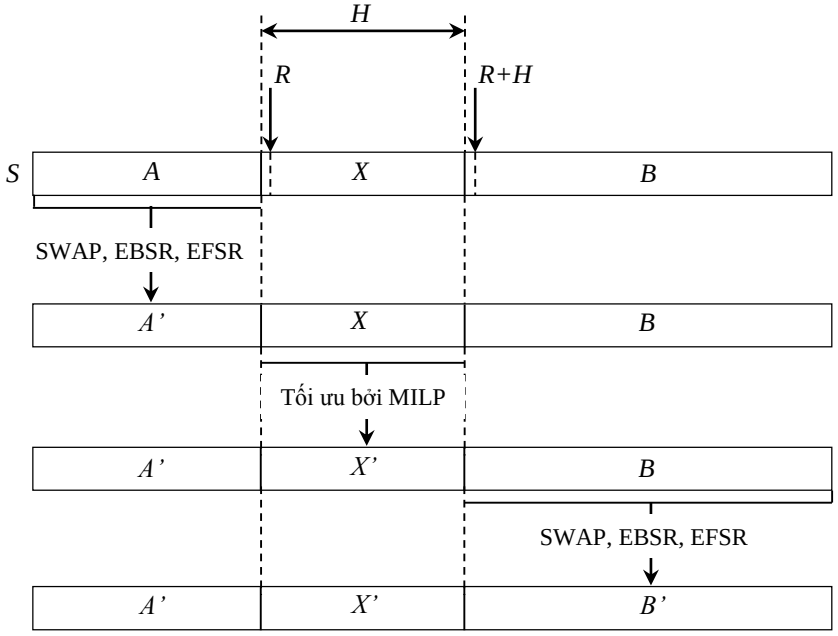
Ràng buộc (2) và (3) đảm bảo rằng chắc chắn chỉ có một công việc ở một vị trí và mỗi vị trí chỉ có một công việc. Các ràng buộc (4) và (6) tính toán thời gian hoàn thành trên máy thứ nhất M_1. Các ràng buộc (5), (7) và (8) xác định thời gian hoàn thành trên máy M_i. Ràng buộc (9) xác định tổng thời gian trễ.

3.2. Thuật toán Matheuristic - MH_{V1}

Trong phần này, chúng tôi đề xuất thuật toán *Matheuristic* thực hiện bằng cách gọi lặp lại mô hình MILP được miêu tả trong *phần 3.1* để tính toán và thực hiện với các bài toán con (*chuỗi con của S*) và cho các kết quả chính xác.

Các kí hiệu và hoạt động của thuật toán Matheuristic - MH_{V1}:

Cho chuỗi *S = AXB*, một chỉ số (*index*) *R* và cỡ một "cửa sổ" được kí hiệu bởi *H*. Trong phương pháp này, *A* là tập các công việc từ vị trí *1* đến vị trí thứ *R-1*; *B* là các công việc từ vị trí *R+H* đến vị trí cuối cùng của *S*; các công việc từ vị trí *R* đến vị trí *R+H-1* được ký hiệu bởi *X*. *Đầu tiên*, các phép toán *SWAP*, *EBSR*, và *EFSR* (*Miêu tả trong phần 2*) được áp dụng đối với chuỗi *A* và kết quả đạt được là chuỗi *A'*. *Tiếp theo*, chuỗi X được tối ưu bởi MILP và cho kết quả là chuỗi *X'*. *Cuối cùng*, các phương pháp *SWAP*, *EBSR*, và *EFSR* được áp dụng đối với chuỗi *B* và được kết quả là chuỗi *B'*. Chuỗi *S'=A'X'B'* là một chuỗi mới, với hy vọng chuỗi *S'* sẽ tốt hơn *S* hay $\sum T_j(S') < \sum T_j(S)$ và chuỗi *S* được thay thế chuỗi *S*. Công việc này được lặp lại với các giá trị khác nhau của *R* cho tới *R = n-H*. Quá trình được thực hiện cho đến khi "*tiêu chuẩn dừng*" đạt được. Chúng tôi dùng tiêu chuẩn dừng là thời gian giới hạn và được kí hiệu là *TimeLimitMH*. Chi tiết thực hiện của thuật toán được đưa ra trong Hình 3 và Thuật toán 1.



Hình 3. Minh họa thuật toán Matheuristic cho chuỗi "AXB"

Hàm mục tiêu để tìm chuỗi con tốt nhất A' là tổ hợp tuyến tính của tổng thời gian trễ của các công việc của chuỗi A' và *makespan* hay C_{max} của A' (*khoảng thời gian hoàn thành kể từ khi bắt đầu xử lý các công việc cho tới khi hoàn thành việc xử lý tất cả các công việc được gọi là makespan của bài toán và được ký hiệu là* $\boldsymbol{C_{max}}$). Hàm mục tiêu này là:

$$Z_A = \alpha\left(\sum_{i=1}^{R-1} T_i\right) + (1-\alpha)C_{m,R-1}$$

Trong đó, $C_{m,R-1}$ là thời gian hoàn thành trên máy M_m của các công việc ở vị trí *R-1* (*công việc ở vị trí cuối cùng của X'*).

Thuật toán 1: Thuật toán Matheuristic

```
1: Input: S = initial solution
2: improved = true
3: while (CPU ≤ TimeLimMH) and (improved = true) do
4:    improved = false; R = 0
5:    while R ≤ n - H do
6:       A = (S[1], S[2], …, S[R-1],)
7:       A' = Neighborhood operator (A)
8:       X = (S[R], S[R+1],…, S[R+H-1])
9:       X' = re-optimization of X
10:         B = (S[R+H], S[R+H+1],…, S[n])
11:         B' = Neighborhood operator (B)
12:         S' = A'X'B'
13:         if (∑Tj(S') < ∑Tj(S)) then
14:            improved = true; S = S'
15:            if (R + H ≤ n - H) then
16:                R = R + H - 1
17:            end if
18:         end if
19:         R = R + 1
20:      end while
21:   end while
```

Thuật toán 2: Hoán vị các công việc trong S (Swap(S))

```
1: Input: S = initial solution
2: for i = 1 to n-1 do
3:    j = i + 1
4:    while (j ≤ n) and (j - i ≤ n/2) do
5:       S' = S[1], S[2], …, S[i-1], S[j], S[i+1],…, S[j-1] S[i], S[j+1],…,
         S[n]
6:       if (∑Tj(S') < ∑Tj(S)) then
7:          S = S'
8:       end if
9:       j = j + 1
10:      end while
11:   end for
12: return (S)
```

Tương tự, hàm mục tiêu để tìm chuỗi con tốt nhất có thể cũng là tổ hợp tuyến tính của các công việc của *X'* và *makespan* hay C_{max} của *X'*. Hàm mục tiêu có được như sau:

$$Z_X = \alpha\left(\sum_{k=R}^{R+H-1} T_k\right) + (1-\alpha)C_{m,R+H-1}$$

Trong đó, $C_{m,R+H-1}$ là thời gian hoàn thành trên máy M_m của các công việc ở vị trí *R+H-1* (*công việc ở vị trí cuối cùng của X'*).

Đầu tiên, tổng thời gian trễ của *A'* và thời gian hoàn thành công việc cuối cùng của chuỗi *A'* trên mỗi máy được tính toán và ký hiệu là CA_i, các ràng buộc với CA_i được miêu tả như sau:

$$C_{1,R} = CA_1 + \sum_{j=1}^{n} p_{1,j}x_{j,R}$$

$$C_{i,R} \geq CA_i + \sum_{j=1}^{n} p_{i,j}x_{j,R}, \quad \forall i \in \{2,\dots,m\}$$

Với $C_{i,R}$ là thời gian hoàn thành trên máy M_i của công việc *ở* vị trí *R* của chuỗi *S*.

Tiếp theo, tối ưu chuỗi *X* bởi *MILP* và đạt được chuỗi *X'*. Thời gian hoàn thành của công việc cuối cùng của chuỗi *X'* trên mỗi máy được kí hiệu là CX_i. Sau đó chuỗi *B* được sắp xếp sau chuỗi *X*. Cuối cùng, giá trị của *S'* được tính như sau:

$$\sum T_j(S') = \sum T_j(A) + \sum T_j(X^*) + \sum T_j(B)$$

Trong Thuật toán 1 và Thuật toán 2, $S_{[kj}$ là công việc ở vị trí *k* trong chuỗi *S*. Đối với Thuật toán 1, nếu ở lần thực hiện của thuật toán cho các công việc ở vị trí trong đoạn [*R, R+H-1*] mà tổng thời gian trễ tại lần thực hiện này nhỏ hơn lần thực hiện trước đó ($\sum T_j(S') < \sum T_j(S)$) thì vị trí của các công việc được xem xét trong lần lặp tiếp theo là [*R+H, R+2H-1*] và *R+2H-1*≤*n*. Ngược lại, các vị trí cho lần lặp tiếp theo trong đoạn [*R+1, R+H*].

Trong Thuật toán 2, một cặp công việc được hoán vị nếu sự khác nhau giữa hai vị trí của các công việc không vượt quá *n/2*. Nếu một hoán vị được thực hiện mà cho kết quả tốt hơn thì chuỗi hiện tại được cập nhật luôn và sử dụng cho lần hoán vị tiếp hoán vị tiếp theo.

Ở Thuật toán 1, chúng ta có thể sử dụng một thuật toán bất kì cho chuỗi khởi tạo (*S = initial solution*). Các Thuật toán 3 - EDD

Thuật toán 3: Thuật toán EDD

```
1: Input: S = a set of jobs,
2: Sorted: the jobs by non decreasing order of d_j
3: Output: A set of jobs sorted in non decreasing order of d_j
```

Thuật toán 4: Thuật toán NEH

```
1: Input: S = jobs sorted in EDD order,
2: Consider the partial sequence with minimum ∑T_j and minimum
makespan in case of ties among {(S[1], S[2]), (S[2]; S[1])}
3: for k = 3 to n do
4:    Test the insertion of S[k] at any possible position in S'
      from 1 to k + 1
5:    Keep the best insertion, i.e. the insertion with minimum
      total tardiness, and the insertion with minimum makespan in
      case of ties.
6: end for
```

(*Earliest Due Date*), Thuật toán 4 - NEH (*Nawaz Enscore Ham*) (Nawaz et al., 1983b) đã được chúng tôi sử dụng để thử nghiệm cho bài toán trường hợp này.

4. Kết quả và thảo luận

Chúng tôi đã thử nghiệm thuật toán này trên máy tính PC Intel core™ i5 CPU 2.4GHz. Tập dữ liệu để kiểm tra thực nghiệm được tạo ra bởi (Vallada et al., 2008) đã được sử dụng để đánh giá. 9 bộ dữ liệu sử dụng giá trị của m và n với $m \in \{10, 30, 50\}$ và $n \in \{50, 150, 250, 350\}$.

Các thuật toán Matheuristic, GA được thử nghiệm với giải pháp khởi tạo (*initial solution*) là EDD và NEH. Từ các thử nghiệm chỉ ra rằng kết quả tốt nhất đạt được với phương pháp khởi tạo *EDD*.

Thời gian giới hạn của thuật toán được đặt cố định là *TimeLimMH = (200 + n + m)* giây. Cỡ của "cửa sổ" H = 7, hệ số $\propto = 0.5$.

Trong Bảng 1, trên mỗi dòng là kết quả của 9 tệp dữ liệu và các số in đậm tương ứng với kết quả tốt nhất của dòng đó. Ở bảng này, cột 'Best' của MH_{V1} chỉ ra số lần thuật toán MH_{V1} thực hiện tốt hơn thuật toán *GA*. Cột 'Δ_{MH}' chỉ ra trung bình chênh lệch giữa MH_{V1} và *GA*:

$$\Delta_{MH} = \frac{MH_{V1} - GA}{MH_{V1}}$$

Tương tự, cột 'Best' của *GA* chỉ ra số lần thuật toán *GA* thực hiện tốt hơn thuật toán MH_{V1}. Cột 'Δ_{GA}' chỉ ra trung bình chênh lệch giữa *GA* và MH_{V1}:

$$\Delta_{GA} = \frac{GA - MH_{V1}}{GA}$$

Từ kết quả của Bảng 1 chỉ ra rằng thuật toán Matheuristic thực hiện tốt hơn thuật toán di truyền GA trong hầu hết các trường hợp. Tuy nhiên, thuật toán GA thực hiện tốt hơn Matheuristic trong trường hợp *(n x m) = (350 x 50)*, và với *n = 150* công việc, *m = 50* máy thì cả 2 thuật toán cho kết quả như nhau.

5. Kết luận

Trong bài báo này, chúng tôi xem xét bài toán flow shop *m* máy với mục đích là tối thiểu hóa tổng thời gian trễ. Các phép toán lận cận

Bảng 1. So sánh giữ thuật toán Matheuristic và GA

n x m	MH_{V1}			GA		
	Best	CPU(s)	ΔMH_{V1}	Best	CPU(s)	Δ_{GA}
50 × 10	6	260,11	8,91%	4	260,01	-1,30%
50 × 30	5	280,17	0,06%	4	280,01	-0,20%
50 × 50	6	300,40	-0,62%	5	300,02	0,59%
150 × 10	7	360,21	6,30%	3	360,02	4,06%
150 × 30	7	380,61	-1,21%	4	380,02	1,14%
150 × 50	5	402,81	1,35%	5	400,05	-3,67%
250 × 10	9	460,73	-3,28%	3	460,03	3,08%
250 × 30	8	483,64	8,07%	2	480,05	2,90%
250 × 50	9	503,42	-3,59%	2	500,07	3,43%
350 × 10	7	562,58	7,39%	4	560,08	3,46%
350 × 30	6	591,63	7,42%	6	580,1	3,31%
350 × 50	2	606,78	23,36%	7	600,1	-19,68%
Sum/avg	**77**	**432,76**	**4,51%**	**49**	**430,05**	**-0,24%**

(*neighborhood operators***),** thuật toán *Maheuristic* được đề xuất cho bài toán này và so sánh với thuật toán di truyền GA. Các kết quả tính toán chỉ ra rằng thuật toán Matheuristic thực hiện tốt hơn thuật toán GA trong hầu hết các trường hợp, ngoại trừ trường hợp *n = 350* công việc và *m = 50* máy.

Một vài hướng nghiên cứu có thể được xem xét trong tương lai. *Đầu tiên*, so sánh thuật toán Matheuristic với các thuật metaheuristic khác như: thuật toán tìm kiếm Tabu - TS, thuật toán tối ưu bầy đàn - PSO, thuật toán mô phỏng luyện kim - SA; thuật toán MH_{V1} cũng sẽ được so sánh với các thuật toán Matheuristic khác. *Tiếp theo*, thuật toán Matheuristic được đề xuất ở đây có thể áp dụng cho bài toán điều phối xe - VRP (*Vehicle Routing Problem*) (Lenstra and Rinnooy Kan, 1981; Ta et al., 2014).

Lời cảm ơn

Tác giả gửi lời cám ơn chân thành đến Giáo sư Jean-Charles Billaut, Tiến sĩ Jean-Louis Bouquard trường đại học Fran ç ois Rabelais, thành phố Tours, cộng hòa Pháp đã hỗ trợ để tôi hoàn thành bài báo này.

Tài liệu tham khảo

Della Croce, F., Ghirardi, M., Tadei, R., 2004. Recovering Beam Search: enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics* 10, 89-104.

Della Croce, F., Grosso, A., Salassa, F., 2011. A matheuristic approach for the total completion time two-machines permutation flow shop problem. *Lect. Notes Comput. Sci*. 6622 LNCS 38-47.

Du, J., Leung, J.Y.T., 1990. Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*. 19, 483-495.

Grabowski, J., Wodecki, M., 2004. A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Computers & Operations Research*. 31, 1891-1909.

Kim, Y. D., 1993a. A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops. *Computers & Operations Research*. 20(4), 391-401.

Kim, Y. D., 1993b. Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal of the Operational Research Society*. 44, 19-28.

Koulamas, C., 1998. A guaranteed accuracy shifting bottleneck algorithm for the two-machine flowshop total tardiness problem. *Computers & Operations Research*. 25, 83-89.

Lenstra, J.K., Rinnooy Kan, A.H.G., 1981. Complexity of vehicle routing and scheduling problems. *Networks* 11, 221-227.

Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P., 1977. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*. 1, 343-362.

Liao, C.-J., Chao-Tang Tseng, Luarn, P., 2007. A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Research*. 34, 3099-3111.

Maniezzo, V., Stutzle, T., Voss, S., 2010. Matheuristics: Hybridizing Metaheuristics and Mathematical Programming. *Annals of Information Systems* 10, Springer.

Nawaz, M., Enscore, E., Ham, I., 1983a. A heuristic algorithm for the m-machine n-job flow shop sequencing problem. *Omega* 11, 91-95.

Nawaz, M., Enscore, E., Ham, I., 1983b. A heuristic algorithm for the m-machine n-job flow shop sequencing problem. *Omega* 11, 91-95.

Nowicki, E., Smutnicki, C., 1996. A fast tabu search algorithm for the permutation flowshop problem. *European Journal of Operational Research.* 91, 160-175.

Onwubolu, G.C., Mutingi, M., 1999. Genetic algorithm for minimizing tardiness in flow-shop scheduling. *Production Planning and Control* 10, 462-471.

Osman, I. H., Laporte, G., 1996. Metaheuristics: A bibliography. *Ann. Oper. Res.* 63, 513-623.

Pan, J.C.-H., Fan, E.-T., 1997. Two-machine flowshop scheduling to minimize total

tardiness. *International Journal of Systems Science.* 28, 405-414.

Pan, J. C. H., Chen, J. S., Chao, C. M., 2002. Minimizing tardiness in a two-machine flow-shop. *Computers & Operations Research.* 29, 869-885.

Pessan, C., Bouquard, J. L., Neron, E., 2008. Genetic branch-and-bound or exact genetic algorithm? *Lecture Notes in Computer Science.* 4926 LNCS, 136-147.

Sen, T., Dileepan, P., Gupta, J. N. D., 1989. The two-machine flowshop scheduling problem with total tardiness. *Computers & Operations Research.* 16(4), 333-340.

Ta, Q. C., 2017. *Matheuristic algorithms to minimize total tardiness in flow shop scheduling*, Éditions Universitaires Européennes.

Ta, Q.C., Billaut, J.-C., Bouquard, J.-L., 2015. Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem. *Journal of Intelligent Manufacturing*. 1-12.

Ta, Q. C., Billaut, J. C., Bouquard, J. L., 2014. Minimisation de la somme des retards pour un problème d'ordonnancement de type flowshop à deux machines et un problème de livraison intégrés. *Le 15ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision (ROADEF)*.

Ta, Q. C., Billaut, J. C., Bouquard, J. L., 2013. Recovering beam search and Matheuristic algorithms for the F2||sum T_j scheduling problem. *5th International Conference on Industrial Engineering and Systems Management*, 218-220.

Talbi, E.G ., 2013. *Hybrid Metaheuristics*. Studies in computational intelligence, Springer.

Tasgetiren, M. F., Liang, Y. C., Sevkli, M., Gencyilmaz, G., 2007. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*. 177, 1930-1947.

Vallada, E., Ruiz, R., Minella, G., 2008. Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research*. 35, 1350-1373.

ABSTRACT

Study of Matheuristic algorithm for minimizing total tardiness in the *m*-machine flow-shop scheduling problem

Chieu Quang Ta

Faculty of Information Technology, Hanoi University of Mining and Geology, Vietnam

In this paper, we consider m-machine permutation flow shop problem with total tardiness minimization. We propose neighbor operators and matheuristic algorithm. Mathueristic are an hybridization of a local search and an exact resolution method. The matheuristic is compared to a genetic algorithm. Computational experiments are performed on benchmark instances and the results show the good performances of the matheuristic algorithm. Finally, some future research directions are proposed.

Keywords: Matheuristic, total tardiness, flow shop, Genetic.