Kỷ yếu Hội nghị Quốc gia lần thứ VIII về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR); Hà Nội, ngày 9-10/7/2015
DOI: 10.15625/vap.2015.000209

THUẬT TOÁN LẬP LỊCH LUỒNG CÔNG VIỆC TRONG MÔI TRƯỜNG ĐIỆN TOÁN ĐÁM MÂY

Phan Thanh Toàn[1], Nguyễn Thế Lộc[2], Nguyễn Doãn Cường[3]

[1] Khoa Sư phạm kỹ thuật, Trường Đại học Sư phạm Hà Nội
[2] Khoa Công nghệ thông tin, Trường Đại học Sư phạm Hà Nội
[3] Viện Công nghệ thông tin, Viện Khoa học Công nghệ Quân sự

pttoan@hnue.edu.vn, locnt@hnue.edu.vn, cuongvncntt@yahoo.com

TÓM TẮT - *Điện toán đám mây (Cloud Computing) là mô hình dịch vụ phân tán dựa trên sự kết hợp của các máy chủ nằm tại các vị trí địa lý khác nhau. Một trong những yếu tố quyết định hiệu năng của đám mây là vấn đề lập lịch luồng công việc. Khi khách hàng gửi yêu cầu tới, trung tâm điều khiển phải tìm cách phân chia công việc cho các máy chủ có cấu hình khác nhau sao cho thời gian thực hiện là ngắn nhất. Bài toán lập lịch từ lâu đã được chứng minh là thuộc lớp NP-khó trong khi mô hình dịch vụ yêu cầu phải tìm ra lời giải trong thời gian ngắn để khách hàng không phải chờ đợi. Bài báo này đề xuất thuật toán metaheuristic PSOi để tìm kiếm phương án lập lịch dựa trên phương pháp Tối ưu bầy đàn. Thực nghiệm được tiến hành trên công cụ mô phỏng CloudSim đã chứng tỏ thuật toán đề xuất cho kết quả tốt hơn ba thuật toán đối chứng là PSO, Random và RoundRobin và lời giải tìm được có độ sai lệch rất bé so với lời giải tối ưu.*

Từ khóa: *workflow scheduling, particle swarm optimization, cloud computing*

I. ĐẶT VẤN ĐỀ

Luồng công việc (workflow) là một chuỗi có thứ tự các tác vụ (task) có thể được thực hiện đồng thời hay tuần tự nếu dữ liệu đầu ra của tác vụ này là đầu vào của tác vụ kế tiếp. Quy trình xử lý đơn hàng, thủ tục yêu cầu bồi thường bảo hiểm, quá trình xử lý công văn hành chính là những ví dụ về luồng công việc trong thực tế. Vấn đề lập lịch luồng công việc trong môi trường điện toán đám mây về bản chất là tìm phương án ánh xạ những tác vụ của luồng công việc tới các máy chủ của đám mây sao cho thời gian xử lý toàn bộ luồng công việc là nhỏ nhất, biết rằng khối lượng tính toán và yêu cầu dữ liệu của các tác vụ, tốc độ tính toán và truyền thông của các máy chủ là khác nhau.

Phần tiếp theo của bài báo có cấu trúc như sau. Phần II giới thiệu một số công trình nghiên cứu có liên quan về bài toán lập lịch luồng công việc. Trong phần III chúng tôi trình bày mô hình lý thuyết để biểu diễn năng lực tính toán và truyền thông của đám mây, dựa trên mô hình lý thuyết này, phần IV đề xuất:

(i) phương thức mới để cập nhật vị trí của cá thể (mục 4.2)

(ii) giải pháp để chương trình thoát ra khỏi vùng cực trị địa phương và di chuyển tới một vùng mới trong không gian tìm kiếm (mục 4.3)

(iii) thuật toán lập lịch mới tên là PSOi (mục 4.4).

Phần V mô tả các thực nghiệm được tiến hành dựa trên công cụ mô phỏng Cloudsim [1] và phân tích những số liệu thực nghiệm thu được. Phần VI tóm tắt những kết quả chính của bài báo và hướng nghiên cứu sẽ tiến hành trong tương lai.

II. CÁC CÔNG TRÌNH LIÊN QUAN

2.1. Các hướng tiếp cận bài toán

Bài toán lập lịch luồng công việc đã được chứng minh là thuộc lớp NP-đầy đủ [2] nghĩa là thời gian để tìm ra lời giải tối ưu là rất lớn, vì vậy đã có nhiều giải thuật metaheuristic được nghiên cứu nhằm tìm ra lời giải gần đúng trong thời gian ngắn. S. Parsa [3] đã đề xuất một thuật toán lập lịch nhằm tối thiểu thời gian thực thi trong môi trường lưới tính toán Grid. J.M. Cope và đồng nghiệp đã phân tích hiệu năng của giải thuật FRMTL và FRMAS [4] trong môi trường lưới tính toán TeraGrid, một dạng đặc biệt của đám mây điện toán. A. Agarwal đã đề xuất thuật toán tham lam [5] trong đó mỗi tác vụ được gán một thứ tự ưu tiên dựa vào khối lượng công việc của tác vụ, mỗi máy chủ cũng được gán một thứ tự ưu tiên theo tốc độ xử lý của máy chủ sau đó gán các tác vụ vào các máy chủ theo các thứ tự ưu tiên đã tính toán. Cách làm này có nhược điểm là khiến những tác vụ có mức ưu tiên thấp phải chờ đợi lâu và bỏ qua yếu tố tốc độ truyền dữ liệu giữa các máy chủ trong đám mây.

Một số tác giả khác như M. Wieczorek [6] đã nghiên cứu và đề xuất thuật toán lập lịch thực thi luồng công việc theo phương pháp GA (Genetic Algorithm - Gen di truyền), tuy nhiên các nghiên cứu [7] [8] đã nhận định rằng phương

pháp PSO (Particle Swarm Optimization - Tối ưu bầy đàn) có ưu thế hơn so với phương pháp GA khi giải bài toán lập lịch luồng công việc trong những môi trường tính toán phân tán như Lưới (Grid Computing) hay Đám mây (Cloud Computing). Theo hướng đó, S. Pandey [9] đã đề xuất thuật toán theo phương pháp PSO nhằm cực tiểu hóa chi phí thực thi. Thay vì tìm phương án có tổng chi phí thực thi tại các máy chủ là bé nhất, S. Pandey lại định nghĩa hàm mục tiêu để tìm phương án có chi phí thực thi của máy chủ tốn kém nhất (máy có tổng chi phí lớn hơn mọi máy khác) là nhỏ nhất so với các phương án khác. Cách làm này có xu hướng "cào bằng" nghĩa là thiên về các lời giải có chi phí thực thi của các máy chủ là xấp xỉ nhau. Chúng tôi nhận thấy, qua lý thuyết và các thực nghiệm kiểm chứng, cách làm này thường khiến chương trình sớm hội tụ về những giá trị cực tiểu địa phương thay vì tìm ra cực trị toàn cục.

2.2. Phương pháp Tối ưu bầy đàn

Phương pháp tối ưu bầy đàn (PSO - Particle Swarm Optimization) được đề xuất bởi Kennedy và Eberhart [10] là phương pháp tìm kiếm tiến hóa dựa theo hành vi tìm thức ăn theo đàn của các loài động vật như chim hay cá, mỗi cá thể trong đàn sẽ di chuyển dựa theo kinh nghiệm của bản thân và của các cá thể khác trong quần thể. Tại bước lặp thứ k, hướng di chuyển của cá thể thứ i trong đàn được cập nhật theo các công thức sau:

$$v_i^{k+1} = \omega \times v_i^k + c_1\, rand_1 \times (pbest_i - x_i^k) + c_2 rand_2 \times (gbest - x_i^k) \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^k \quad (2)$$

Trong đó

- v_i^k , v_i^{k+1} : vector dịch chuyển của cá thể i ở bước lặp k và k+1
- x_i^k, x_i^{k+1} : vị trí của cá thể i ở bước lặp thứ k và k+1
- ω : hệ số quán tính
- c_1, c_2 : hệ số gia tốc
- $rand_1$, $rand_2$: các hệ số ngẫu nhiên trong đoạn [0,1]
- $pbest_i$: vị trí tốt nhất của cá thể i tính tới thời điểm hiện tại
- $gbest$: vị trí tốt nhất mà cả quần thể đã tìm được cho tới hiện tại

III. MÔ HÌNH LÝ THUYẾT

Giả sử cần sắp xếp lịch biểu cho một luồng công việc trong môi trường đám mây với các giả thiết như sau:

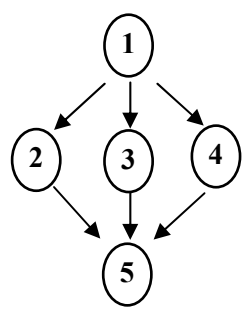
- Luồng công việc được biểu diễn bởi đồ thị G=(V, E), với V là tập đỉnh của đồ thị, mỗi đỉnh biểu thị cho một tác vụ.
- T ={T_1, T_2,…,T_M} là tập các tác vụ, M là số lượng tác vụ của luồng công việc đang xét.
- E là tập cạnh thể hiện mối quan hệ cha-con giữa các tác vụ. Cạnh (T_i, T_j) ∈ E cho biết tác vụ T_i là cha của tác vụ T_j, dữ liệu đầu ra của T_i sẽ là dữ liệu đầu vào cho tác vụ T_j (xem Hình 1).
- Tập máy chủ của đám mây ký hiệu là S = {S_1, S_2,….,S_N}, N là số lượng máy chủ của đám mây.
- Mỗi tác vụ có thể được thực thi trên một máy chủ bất kì, máy chủ đó phải thực hiện toàn bộ tác vụ từ đầu đến cuối.
- Khối lượng tính toán (Workload) của tác vụ T_i kí hiệu là W_i với đơn vị đo là flop (floating point operations: phép tính trên số thực dấu phảy động). W_i được cho trước ($\forall i$ = 1,2, …M)
- Tốc độ tính toán của máy chủ S_i , đơn vị là MI/s (million instructions/second), được ký hiệu bởi P(S_i), là giá trị được cho trước ($\forall i$ = 1,2, …M)
- Giữa hai máy chủ S_i, S_j bất kỳ ($1 \le i,j \le N$) có một đường truyền với băng thông, đơn vị là Megabit/s, được biểu thị bởi hàm hai biến B() được định nghĩa như sau:

$$B: S \times S \rightarrow R^+$$
$$(S_i, S_j) \rightarrow B(S_i, S_j)$$

- Giả thiết hàm băng thông B() thỏa mãn các điều kiện sau:
  - $B(S_i,S_i) = \infty$: thời gian truyền tại chỗ bằng không
  - $B(S_i,S_j) = B(S_j,S_i)$: tốc độ truyền hai chiều bằng nhau
  - Giá trị $B(S_i,S_j)$ được cho trước ($\forall i,j$).
- Khối lượng dữ liệu do tác vụ T_i chuyển tới tác vụ T_j, kí hiệu là D_{ij} với đơn vị là Megabit, là giá trị cho trước ($\forall i,j$).
- Mỗi phương án xếp lịch thực thi luồng công việc tương đương với một hàm f()

$$f: T \rightarrow S$$
$$T_i \rightarrow f(T_i)$$

Trong đó $f(T_i)$ là máy chủ chịu trách nhiệm thực thi tác vụ T_i



Hình 1. Đồ thị biểu diễn một luồng công việc với 5 tác vụ

Từ các giả thiết trên ta suy ra:

- Thời gian tính toán của tác vụ T_i là: $\frac{W_i}{P(f(T_i))}$ (i=1,2, ... M) (3)

- Thời gian truyền dữ liệu giữa tác vụ T_i và tác vụ con T_j là $\frac{D_{ij}}{B(f(T_i), f(T_j))}$ (4)

- Bài báo này định nghĩa hàm mục tiêu là: *Makespan* → *min* trong đó Makespan là thời gian hoàn thành luồng công việc, được tính từ khi tác vụ gốc được khởi động cho tới thời điểm tác vụ cuối cùng được thực hiện xong.

IV. THUẬT TOÁN ĐỀ XUẤT

4.1. Mã hóa cá thể

Theo phương pháp PSO, tại bước lặp thứ *k*, cá thể thứ *i* trong đàn được xác định bởi vector vị trí x_i^k (cho biết vị trí hiện tại) và vector dịch chuyển v_i^k (cho biết hướng dịch chuyển hiện tại). Trong bài toán xếp lịch đang xét, hai vector đó đều có số chiều bằng số tác vụ trong luồng công việc, ký hiệu là *M*. Các thành phần của vector vị trí x_i^k là số nguyên nhưng các thành phần của vector dịch chuyển v_i^k lại là số thực do công thức (1) tính vector dịch chuyển có những tham số là số thực như $rand_1$, $rand_2$, c_1,c_2. Cả vector vị trí và vector dịch chuyển đều được biểu diễn bằng cấu trúc dữ liệu bảng băm.

Ví dụ 1: giả sử luồng công việc gồm tập tác vụ T={T_1, T_2, T_3, T_4, T_5}, đám mây có tập máy chủ S = {S_1, S_2, S_3}. Khi đó cá thể x_i được biểu diễn bằng vector vị trí [1 ; 2 ; 1 ; 3 ; 2] chính là phương án xếp lịch mà theo đó tác vụ T_1, T_3 được bố trí thực hiện bởi máy chủ S_1, tác vụ T_2, T_5 được thực hiện trên S_2 còn tác vụ T_4 được thực hiện bởi S_3 như dưới đây

T_1	T_2	T_3	T_4	T_5
S_1	S_2	S_1	S_3	S_2

4.2. Phương thức cập nhật vị trí của cá thể

Khi áp dụng công thức cập nhật vị trí của cá thể (2) vào bài toán lập lịch đang xét, chúng ta gặp một vấn đề. Như đã giải thích ở trên, các thành phần của vector dịch chuyển v_i^k phải là số thực do công thức (1) tính vector dịch chuyển có những tham số là số thực như $rand_1$, $rand_2$, c_1,c_2. Nhưng vì tập máy chủ S là hữu hạn và đếm được nên các thành phần của vector vị trí x_i phải là số nguyên để có thể ánh xạ tới một máy chủ nào đó nơi mà tác vụ tương ứng sẽ được thực hiện, chẳng hạn vector vị trí x_i trong ví dụ 1 có các thành phần là $x_i[1]$ =1, $x_i[2]$ =2, $x_i[1]$ =1, $x_i[4]$ =3, $x_i[5]$ =2. Hậu quả là hai vế của phép gán (2) khác kiểu nhau, vế trái $x_i^{k+1}[t]$ thuộc kiểu số nguyên còn vế phải $x_i^k[t] + v_i^k[t]$ thuộc kiểu số thực.

Để giải quyết mâu thuẫn này, một số nghiên cứu trước đây như [9] [11] đã làm tròn giá trị số thực ở vế phải rồi gán cho biến vị trí $x_i^{k+1}[t]$ ở vế trái. Kết quả là nếu giá trị của vế phải là 3.2 thì phân phối tác vụ tới thực thi tại máy chủ có số thứ tự là 3, còn nếu vế phải là 3.8 thì tác vụ sẽ được phân cho máy chủ có số thứ tự là 4. Cách làm có vẻ tự nhiên này thực chất là gán một vị trí được tính toán cẩn thận theo chiến lược PSO cho máy chủ mà số thứ tự của nó tình cờ đúng bằng giá trị nguyên sau khi làm tròn. Cách làm như vậy đã phá hỏng quá trình tiến hóa từng bước của phương pháp PSO.

Để giải quyết vấn đề trên, bài báo này đề xuất cách giải quyết như sau: giá trị thực của vế phải ($x_i^k[t] + v_i^k[t]$) sẽ được để nguyên không làm tròn, còn vế trái $x_i^{k+1}[t]$ sẽ được gán bởi định danh của máy chủ có tốc độ tính toán gần với giá trị của vế phải nhất so với các máy chủ còn lại. Làm như vậy tác vụ sẽ được gán cho máy chủ có năng lực phù hợp với giá trị được tính toán theo PSO.

$$x_i^{k+1}[t] \leftarrow j \text{ nếu } |P(S_j) \quad (x_i^k[t] + v_i^k[t])| \le |P(S_r) \quad (x_i^k[t] + v_i^k[t])| \; \forall S_r \in S, t = 1,2 \dots M \qquad (5)$$

Ví dụ 2: giả thiết tập máy chủ S trong ví dụ 1 có tốc độ tính toán được liệt kê trong bảng 1 sau đây

Bảng 1. Tốc độ tính toán của các máy chủ

Máy chủ S_i	Tốc độ xử lý $P(S_i)$ (MI/s)
S_1	3.1
S_2	5.2
S_3	4.1

Giả sử ở bước thứ *k*+1 tổng $x_i^k + v_i^k$ = [4.4 ; 2.1 ; 6.7 ; 5.6 ; 10.2] thì vector vị trí x_i^{k+1} sẽ được gán bằng [3; 1; 2; 2; 2] nghĩa là cá thể đó tương ứng với phương án xếp lịch sau đây:

T_1	T_2	T_3	T_4	T_5
S_3	S_1	S_2	S_2	S_2

Thật vậy, thành phần thứ nhất của vector vị trí, $x_i^{k+1}[1]$, sẽ nhận giá trị 3, nghĩa là tác vụ T_1 sẽ được gán cho máy chủ S_3 bởi vì :

$$x_i^{k+1}[1] \leftarrow 3 \text{ vì } |P(S_3) \quad 4.4| \le |P(S_r) \quad (4.4)| \; \forall S_r \in S$$

Nghĩa là trong 3 máy chủ thì máy S_3 có tốc độ tính toán gần với giá trị 4.4 nhất so với 2 máy chủ còn lại, theo bảng 1, do đó tác vụ T_1 được gán cho máy chủ S_3 để thực hiện, tức là $f(T_1) = S_3$. Phép gán tương tự cũng được thực hiện với bốn tác vụ còn lại : T_2, T_3,T_4,T_5.

Vấn đề tương tự cũng xảy ra với phép trừ hai vector vị trí trong công thức (1): ($pbest_i$ - x_i^k) và ($gbest$ - x_i^k). Một số công trình hiện có như [9] [11] chỉ đơn giản thực hiện phép trừ các thành phần số nguyên rồi gán cho máy chủ có số thứ tự tương ứng. Ví dụ nếu $pbest_i$ = [2,4,3,3,5] và x_i^k = [1,3,2,1,2] thì $pbest_i$ - x_i^k =[2-1,4-3,3-2,3-1,5-2] = [1,1,1,2,3]. Như đã giải thích ở trên, cách làm này thực chất là gán các tác vụ cho những máy chủ mà số thứ tự của nó tình cờ đúng bằng kết quả phép trừ. Cách làm mang tính ngẫu nhiên như vậy đã phá hỏng quá trình từng bước tiếp cận tới vị trí cực trị của phương pháp PSO. Bài báo này đề xuất một "phép trừ vector" áp dụng riêng cho công thức (1) như sau. Giả sử:

$$pbest_i = [x_{i1}, x_{i2},\ldots x_{iM}] \text{ với } x_{ik} \in S\ (\forall k) \text{ và } x_j = [x_{j1}, x_{j2},\ldots x_{jM}] \text{ với } x_{jk} \in S\ (\forall k)$$

Khi đó kết quả phép trừ $pbest_i$ - x_j được tính như sau: $pbest_i$ - x_j =[y_1, y_2,…y_M] với các thành phần y_k là các số thực được tính như sau

$$y_k = \left\{F(x_{ik}) + \frac{\sum_{q \neq s} B(x_{ik}, x_q)}{N-1}\right\} - \left\{F(x_{jk}) + \frac{\sum_{q \neq s} B(x_{jk}, x_q)}{N-1}\right\} \text{ với } k = 1,2,\ldots M \qquad (6)$$

Theo cách tính này, các máy chủ được xếp thứ tự theo tốc độ tính toán và băng thông của những đường truyền kết nối tới nó. Ví dụ 3 sau đây sẽ minh họa cụ thể hơn.

Ví dụ 3:

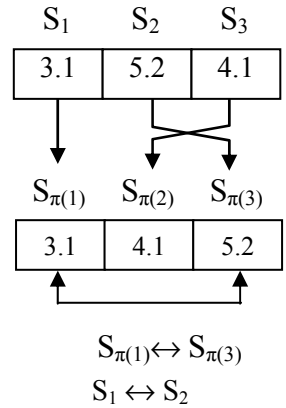
Ta tiếp tục sử dụng tập máy chủ trong ví dụ 2.

Giả sử $gbest$ = [2,1,2,1,1] ; x_j = [3,2,1,2,1] ;

Vậy $gbest - x_j$ = [y_1, y_2, y_3,y_4,y_5] với y_1 được tính như sau

$$y_1 = \left\{F(S_2) + \frac{B(S_2,S_1) + B(S_2,S_3)}{3-1}\right\} - \left\{F(S_3) + \frac{B(S_3,S_1) + B(S_3,S_2)}{3-1}\right\}$$

Cách tính tương tự được áp dụng cho các thành phần y_2, y_3 … y_5 còn lại.



Hình 2. Thủ tục Inverse

4.3. Biện pháp thoát khỏi cực trị địa phương

Phương pháp PSO nói riêng và các phương pháp tìm kiếm tiến hóa nói chung đôi khi bị mắc kẹt tại các lời giải cực trị địa phương mà không thể thoát ra để đi tới lời giải tốt hơn. Bài báo này đề xuất thủ tục Inverse như một biện pháp được áp dụng mỗi khi vòng lặp chương trình sa vào một cực trị và các cá thể bị hút vào gần lời giải cực trị đó mà không thể thoát ra. Khi đó thủ tục Inverse sẽ chuyển các cá thể tới một miền không gian mới nhiều khả năng chưa được lục soát (xem Hình 3)

Function Inverse (vector vị trí x_i)

Input: vector vị trí x_i

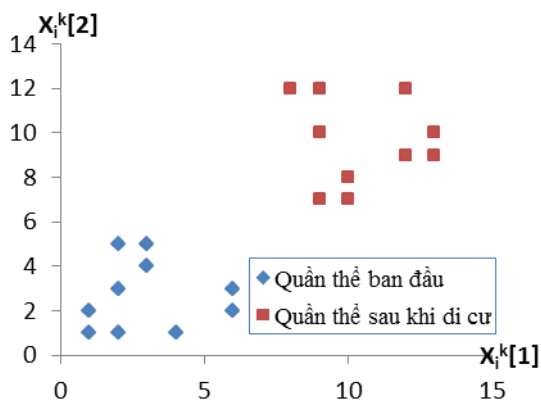
Output: vector vị trí x_i sau khi đã biến đổi

1. Sắp xếp các máy chủ theo thứ tự tăng dần của tốc độ tính toán ta thu được dãy ($S_{\pi(1)}$, $S_{\pi(2)}$,…, $S_{\pi(N)}$), trong đó
   $P(S_{\pi(1)}) \le P(S_{\pi(2)}) \le \ldots \le P(S_{\pi(N)})$ với $S_{\pi(i)} \in S$ (i=1,2 … N)
2. Thay thế các thành phần trong vector vị trí của cá thể theo nguyên tắc đảo ngược:
   $S_{\pi(1)}$ được thay bởi $S_{\pi(N)}$, $S_{\pi(2)}$ được thay bởi $S_{\pi(N-1)}$, …
3. return x_i

End Function

Thủ tục Inverse hoạt động theo nguyên tắc đảo chiều nhằm di chuyển mỗi cá thể tới một vị trí mới nằm xa vị trí hiện tại. Giả sử hiện tại một tác vụ đang được gán cho máy chủ có tốc độ xử lý nhanh nhất thực hiện, vậy thì sau khi thực hiện thủ tục Inverse nó sẽ được gán cho máy chủ có tốc độ chậm nhất, nếu tác vụ đó đang được gán cho máy chủ có tốc độ nhanh thứ 2 thì thủ tục Inverse sẽ chuyển nó tới thực hiện tại máy chủ có tốc độ chậm thứ 2.

Ví dụ 4: ta vẫn tiếp tục xét tập máy chủ S ở các ví dụ trước, sắp xếp chúng theo thứ tự tăng dần của tốc độ xử lý ta thu được dãy (S_1, S_3 , S_2), ký hiệu dãy đó là ($S_{\pi(1)}$, $S_{\pi(2)}$, $S_{\pi(3)}$) như hình 2. Giả sử cá thể x_i đang có vector vị trí là [3, 1, 2, 2, 1, 1], áp dụng thủ tục Inverse với x_i ta thu được cá thể mới có giá trị là [3, 2, 1, 1, 2, 2]. Thật vậy, thành phần x_i[1] = 3 tức là tác vụ T_1 được gán cho máy chủ S_3, mà S_3 chính là $S_{\pi(2)}$ (xem hình 2) nên nó đứng ở giữa danh sách, do đó không bị thay đổi khi danh sách đảo ngược. Thành phần x_i[2] = 1 tức là tác vụ T_2 được gán cho máy chủ S_1, mà S_1 chính là $S_{\pi(1)}$ (xem hình 2) nên thủ tục Inverse sẽ biến nó thành $S_{\pi(3)}$ tức là S_2, vì thế giá trị mới của thành phần x_i[2] sau khi gọi thủ tục Inverse là 2.



Hình 3. Quần thể được di cư tới vùng tìm kiếm mới thông qua thủ tục Inverse với M=20, N=8

4.4. Thuật toán đề xuất PSOi

Tổng hợp những cải tiến nói trên, thuật toán đề xuất với tên gọi PSOi (PSO Inverse) được mô tả như sau.

Algorithm PSOi

Input: tập T, tập S, mảng W[1×*M*], mảng P[1×*N*], mảng B[*N*×*N*], mảng D[*M*×*M*], hằng số *K*, độ lệch ε, số cá thể *SCT*
Output: lời giải tốt nhất *gbest*

```
1. Khởi tạo vector vị trí và vector dịch chuyển của cá thể i  một cách ngẫu nhiên
2. Khởi tạo bước lặp t← 0 ;
3. while (điều kiện lặp) do
4.     for i=1 to SCT do
5.           Tính vector vị trí x_i theo công thức (5)
6.     end for
7.     for i=1 to SCT do
8.           Cập nhật pbest_i
9.     end for
10.    Cập nhật gbest
11.    for i=1 to SCT do
12.          Cập nhật vector dịch chuyển v_i theo (1)
13.          Tính x_i theo (2)
14.    end for
15.    t++ ;
16.    if (sau K thế hệ mà độ lệch giữa các gbest không vượt quá ε) then
17.          for i=1 to SCT do
18.                Inverse(x_i) //Thực hiện thao tác Inverse trên cá thể thứ i
19           end for
20     end if
21. end while
22. return gbest
```

Thuật toán hoạt động theo phương pháp PSO theo đó tại mỗi bước các cá thể cập nhật vị trí của mình hướng tới vị trí tốt nhất của cả quần thể (*gbest*) đồng thời có dựa trên kinh nghiệm cá nhân ($pbest_i$). Nếu sau *K* thế hệ liên tiếp mà cả quần thể không cải thiện được một cách đáng kể giá trị *gbest* (mức chênh không vượt quá ε) thì chứng tỏ quần thể đang hội tụ tại một cực trị địa phương. Khi đó thủ tục Inverse được gọi để di cư cả quần thể tới một vùng không gian mới, tại đó quá trình tìm kiếm được tái khởi động.

Điều kiện lặp ở đây là mức chênh của giá trị *gbest* so với *K* vòng lặp trước đó lớn hơn độ lệch ε (ε ấn định từ trước), nghĩa là thuật toán PSOi sẽ dừng nếu như sau *K* lần di cư (thông qua thủ tục Inverse) mà giá trị *gbest* tìm được vẫn không cải thiện được một cách đáng kể (mức chênh không vượt quá ε).

Trong trường hợp thuật toán hội tụ nhanh nhất, nghĩa là sau *K* lần thực hiện Inverse thì chương trình hội tụ tới cực trị, điều kiện dừng lặp được thỏa mãn nên chương trình kết thúc sau K^2 thế hệ. Ngược lại, trong trường hợp tồi nhất, chương trình luôn tìm được lời giải tốt hơn sau mỗi lần di cư (thông qua thủ tục Inverse) thì các vùng tìm kiếm sẽ lần lượt được khảo sát cho tới khi toàn bộ không gian lời giải được duyệt hết, thuật toán trở thành duyệt vét cạn. Để tránh tình huống này, chúng tôi cũng sử dụng giải pháp chung thường được áp dụng trong các giải thuật tiến hóa, đó là đặt một giá trị ngưỡng tối đa, khi quá trình tiến hóa của quần thể đạt tới số thế hệ vượt quá giá trị ngưỡng đã định thì quá trình tìm kiếm kết thúc. Trong phần thực nghiệm tiếp theo giá trị ngưỡng cho số thế hệ là 3000, giá trị *K* được đặt là 30 và độ lệch ε được ấn định là 0.21.

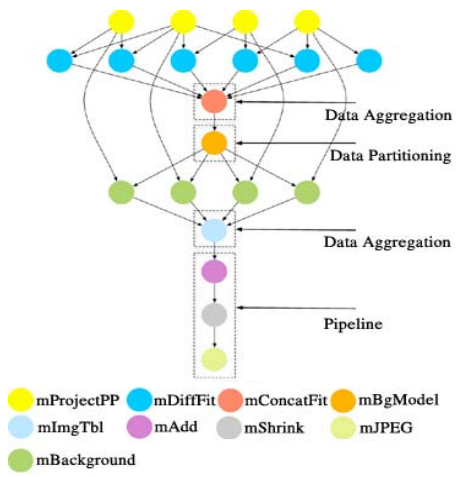
V. THỰC NGHIỆM

5.1. Phân nhóm dữ liệu thực nghiệm

Dữ liệu sử dụng trong các thực nghiệm bao gồm:

- Dữ liệu thực tế thu được từ các công ty cung cấp dịch vụ Cloud trong nước [13] [14] và quốc tế [15] [16]
- Dữ liệu mô phỏng được lấy từ nguồn cung cấp Cloud Sim [8]
- Dữ liệu luồng công việc được lấy từ dự án Montage [17], đây là dự án nghiên cứu thiên văn được tổ chức và tài trợ bởi NASA/IPAC, dữ liệu của ứng dụng được thu thập từ các kính thiên văn, kích thước luồng công việc phụ thuộc vào vùng ảnh chụp được từ bầu trời. Hình 4 minh họa một luồng công việc gồm 20 tác vụ trong ứng dụng Montage

Những dữ liệu đó được tổng hợp lại và chia thành bốn nhóm (được trình bày trong Bảng 3) dựa theo số lượng máy chủ *N* và số lượng tác vụ *M* bao gồm:



Hình 4. Luồng công việc Montage với 20 tác vụ

- Nhóm 1: *M*=5, *N*=3
- Nhóm 2: *M*=10, *N*=3
- Nhóm 3: *M*=20, *N*=8
- Nhóm 4: M=25, N=8 (luồng công việc từ ứng dụng Montage)

Mỗi nhóm lại bao gồm ba thực nghiệm khác nhau về tỷ lệ số cạnh trên số đỉnh của đồ thị luồng công việc, ký hiệu là α

$$\alpha = \frac{|E|}{N \times (N-1)/2}$$

Tham số α cho biết đồ thị G phân thành bao nhiêu cấp, mỗi cấp có nhiều hay ít tác vụ, nói cách khác α phản ánh độ trù mật của đồ thị G. Khi làm thực nghiệm với mỗi nhóm, số máy chủ và số tác vụ được giữ cố định còn tỷ lệ α lần lượt thay đổi như trong các hình 4, 5, 6.

5.2. Tham số cấu hình hệ thống

Các tham số cấu hình của đám mây được thiết lập trong miền giá trị như sau:

- Tốc độ tính toán P của các máy chủ: từ 1 đến 250 (million instructions/s)
- Khối lượng dữ liệu D giữa các tác vụ: từ 1 đến 10000 (Mega bit)
- Băng thông giữa các máy chủ B: từ 10 đến 100 (Mega bit/s)
- Hệ số quán tính: $\omega = 0.729$
- Hệ số gia tốc: $c_1 = c_2 = 1.49445$
- Hằng số: $K = 30$
- Số cá thể *SCT*: *SCT*=25
- Độ lệch ε: 0.21
- α: từ 0.2 tới 0.7

5.3. Quá trình tiến hành thực nghiệm

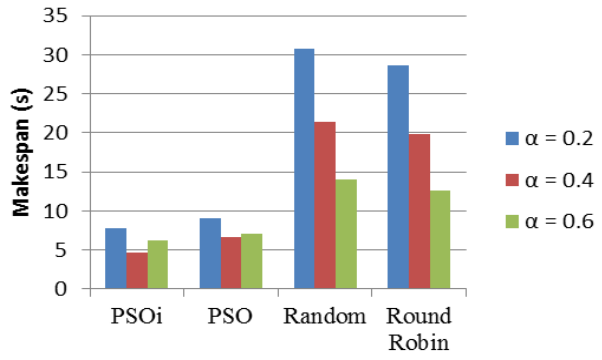
Để kiểm chứng thuật toán đề xuất PSOi chúng tôi đã sử dụng công cụ mô phỏng Cloudsim [1] để tạo lập môi trường đám mây kết hợp với dữ liệu luồng công việc của ứng dụng Montage [17]. Các hàm của gói thư viện Jswarm [1] được sử dụng để thực hiện các phương thức Tối ưu bầy đàn. Đối tượng so sánh là thuật toán PSO [9], thuật toán Random [12] và thuật toán Round Robin [13].

Các chương trình mô phỏng được viết bằng ngôn ngữ Java và chạy trên máy tính cá nhân với bộ vi xử lý Intel Core i5 2.2 GHz, RAM 4GB, hệ điều hành Windows 7 Ultimate. Thực nghiệm được lặp lại 300 lần trên mỗi nhóm thực nghiệm.

Để kiểm chứng hiệu quả của thủ tục Inverse trong quá trình tìm kiếm lời giải chúng tôi đã thực hiện thuật toán PSOi trên các bộ dữ liệu khác nhau và thống kê tỉ lệ thủ tục Inverse giúp thoát khỏi cực trị địa phương, kết quả được trình bày trong bảng 3.

5.4. Kết quả thực nghiệm

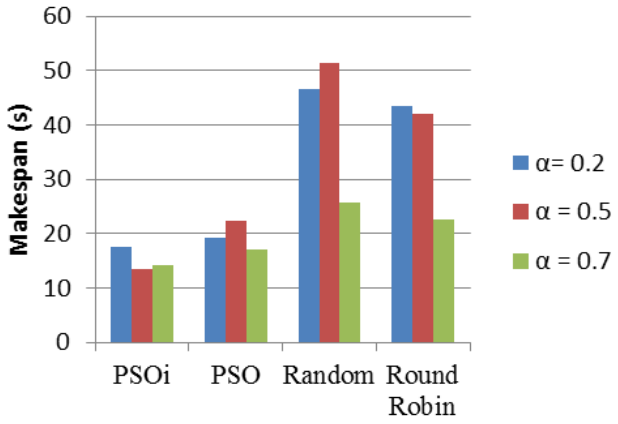
Hình 5, 6, 7, 8 cho thấy sự chênh lệch về thời gian xử lý (makespan) của lời giải tốt nhất mà thuật toán đề xuất PSOi và các thuật toán đối chứng (PSO, Random và Round Robin) tìm được khi chạy trên 4 nhóm dữ liệu thực nghiệm khác nhau.



Hình 5. Lời giải tìm được bởi các thuật toán trong trường hợp M=5, N=3

Bảng 2. Kết quả thực nghiệm

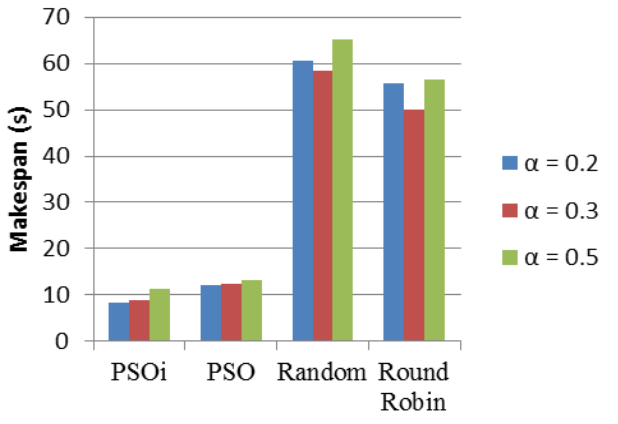
M	N	α	PSOi	PSO	Random	Round Robin
5	3	0.2	7.8	9.0	30.75	28.6
5	3	0.4	4.7	6.6	21.45	19.9
5	3	0.6	6.2	7.1	14.1	12.6
10	3	0.2	15.7	19.3	46.6	43.6
10	3	0.4	13.6	22.4	51.5	42.0
10	3	0.7	14.3	17.0	25.8	22.6
20	8	0.2	8.4	12.2	60.5	55.7
20	8	0.3	8.8	12.5	58.5	50.1
20	8	0.5	11.2	13.1	65.1	56.5
25	8	0.2	2.9	3.9	15.1	13.5



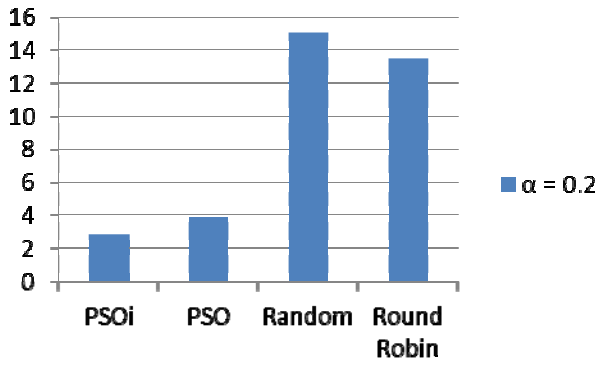
Hình 6. Lời giải tìm được bởi các thuật toán trong trường hợp M=10, N=3

Bảng 3. Kết quả thử nghiệm thủ tục Inverse

M	N	α	Tỉ lệ cá thể mới sinh ra sau mỗi lần Inverse
5	3	0.2	12.9%
5	3	0.4	13.1%
10	3	0.2	13.4%
10	3	0.4	15.2%
10	3	0.7	13.8%
20	8	0.2	22.9%
20	8	0.3	20.6%
20	8	0.5	21.5%



Hình 7. Lời giải tìm được bởi các thuật toán trong trường hợp M=20, N=8



Hình 8. Lời giải tìm được bởi các thuật toán cho luồng công việc Montage với M=25, N=8

Các thông số ở bảng 2 cho thấy thuật toán được kiểm chứng trên nhiều bộ dữ liệu khác nhau về quy mô của luồng công việc (số tác vụ M và số máy chủ N) và độ trù mật α của đồ thị liên kết G. Kết quả thực nghiệm cho thấy trong hầu hết các trường hợp thuật toán đề xuất PSOi đều cho lời giải tốt hơn các thuật toán PSO, Random và Round Robin. Riêng với nhóm thực nghiệm thứ nhất (số tác vụ bằng 5 và số máy chủ bằng 3) thì thuật toán PSOi cho lời giải gần xấp xỉ với lời giải tốt tuyệt đối tìm được bằng phương pháp duyệt vét cạn: 7.8 giây so với 7.7 giây.

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bài báo này đã trình bày một giải thuật tìm kiếm theo phương pháp Tối ưu bầy đàn để tìm lời giải gần đúng cho bài toán lập lịch thực thi luồng công việc trong môi trường điện toán đám mây. Những kết quả chính gồm có:

- Đề xuất một phương thức mới để cập nhật vị trí của cá thể bằng cách ánh xạ một giá trị thực tới máy chủ có tốc độ tính toán và băng thông gần với giá trị đó nhất.
- Đề xuất thủ tục Inverse để chương trình thoát ra khỏi cực trị địa phương bằng cách chuyển các cá thể tới một miền không gian tìm kiếm mới.
- Đề xuất thuật toán PSOi sử dụng phương thức cập nhật vị trí cá thể và thủ tục Inverse để tìm kiếm lời giải cho bài toán lập lịch thực thi luồng công việc trong môi trường đám mây.

Những kết quả thực nghiệm được tiến hành với nhiều bộ dữ liệu thực nghiệm khác nhau đã chứng tỏ chất lượng lời giải tìm được bởi thuật toán đề xuất tốt hơn so với các thuật toán đối chứng là thuật toán PSO gốc, thuật toán Random và thuật toán Round Robin. Về hướng công việc tiếp theo, ngoài hai yếu tố hiện nay là kinh nghiệm cá nhân (pbest) và kinh nghiệm từ cả quần thể (gbest) chúng tôi dự định đưa thêm vào yếu tố kinh nghiệm của các cá thể trong một lân cận xác định theo lược đồ Ring hoặc Star để cập nhật vị trí mới cho mỗi cá thể nhằm đạt được lời giải có chất lượng tốt hơn.

TÀI LIỆU THAM KHẢO

[1]. Công cụ mô phỏng CloudSim http://www.cloudbus.org/cloudsim/

[2]. J. D. Ullman, "NP-complete scheduling problems", Journal of Computer and System Sciences, Volume 10, Issue 3, 1975

[3]. S. Parsa, R. E. Maleki "RASA: A New Task Scheduling Algorithm in Grid Environment", International Journal of Digital Content Technology and its Applications, Vol. 3, No. 4, 2009

[4]. J. M. Cope, N. Trebon, H. M. Tufo, P. Beckman, "Robust data placement in urgent computing environments", IEEE International Symposium on Parallel & Distributed Processing, IPDPS 2009

[5]. A. Agarwal, S. Jain, "Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment", International Journal of Computer Trends and Technology (IJCTT), vol. 9, 2014

[6]. M. Wieczorek, "Marek Scheduling of Scientific Workflows in the ASKALON Grid Environment", ACM SIGMOD Record Journal, Vol. 34, Issue 3, 2005.

[7]. A. Salman, "Particle swarm optimization for task assignment Problem", Microprocessors and Microsystems, 2002.

[8]. S. Pandey, A. Barker, K. K. Gupta, R. Buyya, "Minimizing Execution costs when using globally distributed cloud services", 24th IEEE International Conference on Advanced Information Networking and Applications, 2010.

[9]. J. Kennedy, RC. Eberhart, "Particle swarm optimization", in Proc. IEEE Int'l. Conference on Neural Networks, vol. IV, 1995.

[10]. T. Davidovic, M. Selmic, D. Teodorovic, D. Ramljak "Bee colony optimization for scheduling independent tasks to identical processors", Journal of Heuristics, 2012.

[11]. M. Mitzenmacher, E. Upfal, "Probability and Computing: Randomized Algorithms and Probabilistic Analysis", Cambridge University Press, 2005.

[12]. Don Fallis, "The Reliability of Randomized Algorithms", British Journal for the Philosophy of Science, 2000.

[13]. https://www.ngoisaoso.net/Cloud-Server.html

[14]. http://vdo.vn/may-chu-vps-vdc

[15]. http://www.rackspace.com/cloud/servers

[16]. http://aws.amazon.com/ec2/pricing/

[17]. http://montage.ipac.caltech.edu

THUẬT TOÁN LẬP LỊCH LUỒNG CÔNG VIỆC TRONG MÔI TRƯỜNG ĐIỆN TOÁN ĐÁM MÂY

Phan Thanh Toan, Nguyen The Loc, Nguyen Doan Cuong

ABSTRACT *- The key factor which rules the cloud's performance is the workflow scheduling, one of the well-known problems have proven to be NP-complete. Many algorithm in the literature have been targeting the workflow scheduling problem, however, handful efficient solutions have been proposed. This paper proposes a metaheuristic algorithm called PSOi which based on the Particle Swarm Optimization method. Our experiments which arranged by using the simulation tool CloudSim show that PSOi is superior to the general algorithms called Random and RoundRobin, moreover the deviation between the solution found by PSOi and the optimal solution is negligible.*

Keywords: *workflow scheduling, particle swarm optimization, cloud computing.*