Tạp chí Khoa học và Công nghệ ***54*** *(3) (2016) 287-299*

DOI: 10.15625/0866-708X/54/3/6483

THUẬT TOÁN LPSO LẬP LỊCH CHO CÁC ỨNG DỤNG KHOA HỌC 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, 136 Xuân Thủy, Cầu Giấy, Hà Nội*

[2]*Khoa Công nghệ thông tin, Trường Đại học Sư phạm Hà Nội, 136 Xuân Thủy,Cầu Giấy, Hà Nội*

[3]*Viện Công nghệ thông tin, Viện Khoa học công nghệ quân sự, 17 Hoàng Sâm, Cầu Giấy, Hà Nội*

[*]Email: *pttoan@hnue.edu.vn*

Đến Tòa soạn: 29/6/2015; Chấp nhận đăng: 4/3/2016

TÓM TẮT

Ứng dụng dạng luồng công việc đã được sử dụng rộng rãi trong nhiều công trình nghiên cứu khoa học, đây là loại ứng dụng có quy mô phức tạp và thường phải xử lí một lượng dữ liệu rất lớn do vậy các môi trường tính toán phân tán như điện toán lưới (grid computing), hay điện toán đám mây (cloud computing) thường được sử dụng. Bài toán lập lịch từ lâu đã được chứng minh là thuộc lớp NP-complete trong khi mô hình dịch vụ trên môi trường điện toán đám mây 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 LPSO để tìm kiếm phương án lập lịch dựa trên phương pháp Tối ưu bày đàn. Thực nghiệm được tiến hành trên công cụ mô phỏng CloudSim đã chứng tỏ thuật toán đề xuất cho kết quả tốt hơn hai 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***:** lập lịch luồng công việc, tối ưu bày đàn, điện toán đám mây.

1. 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. Rất nhiều ứng dụng trong các lĩnh vực khoa học khác nhau đều yêu cầu phải xử lí một lượng lớn dữ liệu được tổ chức theo dạng luồng công việc. 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 lí 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 2 giới thiệu một số công trình nghiên cứu có liên quan về bài toán lập lịch luồng công việc.Trong phần 3 chúng tôi trình bày mô hình

lí thuyết để biểu diễn năng lực tính toán và truyền thông của đám mây, dựa trên mô hình lí 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 LPSO (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 6 tóm tắt những kết quả chính của bài báo và hướng nghiên cứu sẽ tiến hành trong tương lai.

2. 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 lí 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 – Thuật giải di truyền), tuy nhiên các nghiên cứu [7, 8] đã nhận định rằng phương pháp PSO (Particle Swarm Optimization - Tối ưu bày 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). T. Davidovic và các công sự đã đề xuất giải thuật lập lịch cho các ứng dụng luồng công việc trong môi trường đa bộ vi xử lí đồng nhất theo phương pháp tối ưu bày ong [10]. Theo hướng đó, S. Pandey [8] 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 lí 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. Zahraa Tarek [9] và các cộng sự đã đề mô hình cho bài toán lập lịch luồng công việc trong môi trường điện toán đám mây và thuật giải dựa theo phương pháp tối ưu bày đàn (PSO), mô hình và thuật giải Zahraa Tarek đề xuất cũng tương tự như S. Pandey [8] đã đề xuất, tuy nhiên Zahraa Tarek đã thực hiện cực tiểu hóa đồng thời chi phí thực thi luồng công việc và tổng thời gian thực hiện các tác vụ.

2.2. Phương pháp Tối ưu bày đàn

Phương pháp tối ưu bày đàn (PSO - Particle Swarm Optimization) được đề xuất bởi Kennedy và Eberhart [12] là phương pháp tìm kiếm tiến hóa dựa theo hành v_i tìm thức ăn theo đàn của các loài động vật như chim hay cá. Mỗi cá thể được xác định bởi 2 tham số là vector vị trí $x_i = [x_{i1}, x_{i2},\ldots,x_{iM}]$, và vector dịch chuyển $v_i = [v_{i1}, v_{i2},\ldots,v_{iM}]$, ban đầu thuật toán sẽ khởi tạo quần thể với vector vị trí và vector dịch chuyển một cách ngẫu nhiên, sau đó trong mỗi bước lặp của thuật toán vector dịch chuyển v_i và vector vị trí x_i của mỗi cá thể sẽ được cập nhật theo công thức (1) và (2).

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

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

trong đó

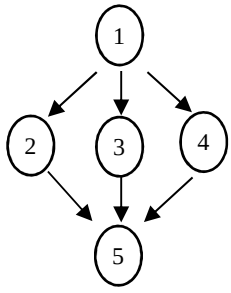
- v_i^k , v_i^{k+1} : vector dịch chuyển của cá thể *i* ở bước lặp *k* và *k*+1
- x_i^k, x_i^{k+1} : vị trí của cá thể *i* ở bước lặp thứ *k* và *k*+1
- ω : hệ số quán tính
- c_1, c_2 : hệ số gia tốc
- $rand_1$, $rand_2$: các hệ số ngẫu nhiên trong đoạn [0,1]
- $pbest_i$: vị trí tốt nhất của cá thể *i*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

3. MÔ HÌNH LÍ 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 kí 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 kí 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 kì ($1 \le i, j \le N$) 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:

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



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

- 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 $f()$

$$f : \mathrm{T} \rightarrow \mathrm{S}$$
$$T_i \rightarrow f(T_i)$$

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

Từ các giả thiết trên ta suy ra:

- Thời gian tính toán của tác vụ T_i là:

$$\frac{W_i}{P(f(T_i))} \quad (i=1,2, \ldots M) \tag{3}$$

- Thời gian truyền dữ liệu giữa tác vụ T_i và tác vụ con T_j là

$$\frac{D_{ij}}{B(f(T_i), f(T_j))} \tag{4}$$

- Bài báo này định nghĩa hàm mục tiêu là: *Makespan* $\rightarrow$ *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.

4. 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, kí hiệu là *M*. 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. Các thành phần của vector dịch chuyển v_i^k 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 [7, 8] 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.

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

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 lí $P(S_i)$ (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 :

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

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 [7] [8] 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:

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

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

$$y_k = \left\{P(x_{ik}) + \frac{\sum_{q\in S} B(x_{ik}, x_q)}{N-1}\right\} - \left\{P(x_{jk}) + \frac{\sum_{q\in S} B(x_{jk}, x_q)}{N-1}\right\} \text{ với } k = 1,2,\ldots M \quad (6)$$

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

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

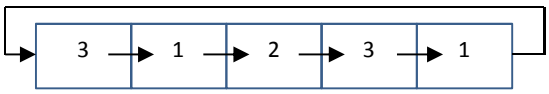
Cách tính tương tự được áp dụng cho các thành phần $y_2, y_3 \ldots y_5$ còn lại.

4.3. Biện pháp thoát khỏi cực trị địa phương

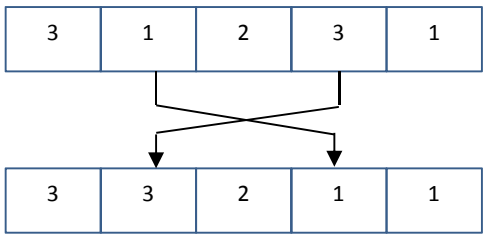
Phương pháp PSO nói riêng và các phương pháp tìm kiếm tiến hóa nói chung đôi khi bị mắc kẹt tại các lời giải cực trị địa phương mà không thể thoát ra để đi tới lời giải tốt hơn. Bài báo này đề xuất sử dụng phương pháp PSO kết hợp với thủ tục tìm kiếm lân cận để định hướng cá thể tốt nhất chuyển sang vùng tìm kiếm mới mỗi khi chương trình bị sa vào vùng cực trị địa phương.

4.3.1. Thủ tục tìm kiếm lân cận

Tìm kiếm lân cận là phương pháp tìm kiếm bắt đầu từ một giải pháp ban đầu s_0 của bài toán và sử dụng các toán tử để di chuyển sang một giải pháp khác của bài toán theo một cấu trúc lân cận xác định nhằm tìm ra một lời giải tốt hơn. Bài báo này đề xuất 2 toán tử Exchange và RotateRight sử dụng cho quá trình tìm kiếm lân cận (Hình 2.a và 2.b).



Hình 2.a. Toán tử RotateRight.



Hình 2.b. Toán tử Exchange.

4.3.2. Giải thuật tìm kiếm lân cận

Function LocalSearch (vector vị trí x_i)

Input: vector vị trí x_i
Output: vector vị trí x_k có $f(x_k) < f(x_i)$
1. Khởi tạo bước lặp t ← 0
2. while (điều kiện lặp)
3. Khởi tạo giá trị r ngẫu nhiên trong đoạn [1, M]
4. x_i ← RotateRight(x_i, r)
5. Khởi tạo 2 giá trị ngẫu nhiên $rand_1$, $rand_2$ trong đoạn [1,M]
6. x_k ← Exchange (x_i, $rand_1$, $rand_2$)
7. if $f(x_k) < f(x_i)$ then return x_k
5. else return x_i
6. t ← t+1
7. End while
End Function

4.4. Thuật toán đề xuất LPSO

Tổng hợp những cải tiến nói trên, thuật toán đề xuất với tên gọi LPSO được mô tả như sau.

Algorithm LPSO

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 gbest*
13. *end for*
14. for *i*=1 to *SCT* do
15. Cập nhất vector dịch chuyển v_i^k theo công thức (1) và (6)
16. Tính x_i theo (2)
17. end for
18. *t*++ ;
19. if (sau *K* thế hệ mà độ lệch giữa các *gbest* không vượt quá ε) then

```
20.    gbest = LocalSearch(gbest);
21  end if
22. end while
23. return gbest
```

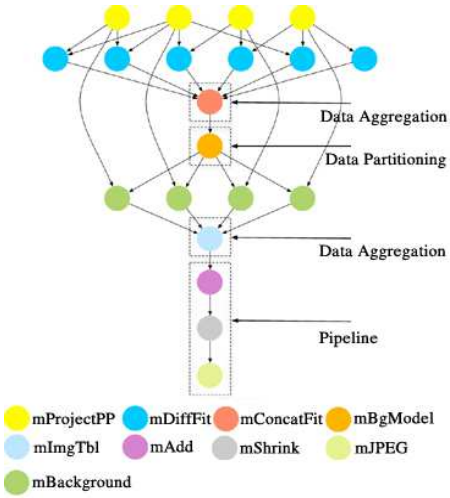
Thuật toán hoạt động theo phương pháp PSO theo đó tại mỗi bước lặp các cá thể cập nhật vị trí của mình hướng tới vị trí tốt nhất của cả quần thể (gbest) đồng thời có dựa trên kinh nghiệm cá nhân ($pbest_i$). Nếu sau K thế hệ liên tiếp mà cả quần thể không cải thiện được một cách đáng kể giá trị gbest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

Điều kiện lặp của thuật toán chúng tôi sử dụng giải pháp mà các giải thuật tiến hóa thường áp dụng đó là đặt một 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 300, giá trị K được đặt là 10 và độ lệch ε được ấn định là 0,21.

Độ phức tạp của thuật toán LPSO. Trước khi thực hiện thuật toán chính LPSO ta cần phải sắp xếp các máy chủ thực thi theo thứ tự tăng dần của tốc độ thực hiện, giải thuật sắp xếp có độ phức tập về thời gian là O(n.log(n)), thủ tục tính ma trận thời gian thực thi của mỗi tác vụ trên các máy chủ có độ phức tạp thời gian tính toán là O(M×N); với M là số tác vụ, N là số máy chủ. Thủ tục tính ma trận thời gian truyền dữ liệu giữa các máy chủ có độ phức tạp tính toán là $O(N^2)$. Trong thuật toán LPSO thì thủ tục khởi tạo sẽ khởi tạo các cá thể của quần thể một cách ngẫu nhiên, mỗi cá thể được mã hóa bởi một véc tơ độ dài M do vậy độ phức tạp của thủ tục khởi tạo là O(M×SCT) ; với SCT là số cá thể trong quần thể, trong thực nghiệm chúng tôi sử dụng SCT là 100. Hàm tính thời gian thực hiện (makespan) của mỗi phương án xếp lịch là $O(M^2)$.

Thủ tục RotateRight có độ phức tạp là O(M), thủ tục Exchange có độ phức tạp là hằng số, do vậy thủ tục tìm kiếm lân cận LocalSearch có độ phức tạp là O(M).

Trong bài toán lập lịch luồng công việc thường số tác vụ lớn hơn số máy chủ (M > N) do vậy độ phức tạp của thuật toán LPSO là (Số thế hệ)× $O(M^2)$.



Hình 3. Luồng công việc Montage với 20 tác vụ. (http://montage.ipac.caltech.edu)

5. THỰC NGHIỆM

5.1. Phân nhóm dữ liệu thực nghiệm

Dữ liệu sử dụng trong các thực nghiệm bao gồm:

- Dữ liệu về tốc độ tính toán của các máy chủ và băng thông giữa các máy chủ được lấy từ các công ty cung cấp dịch vụ cloud [14] và địa chỉ website (http://aws.amazon.com/ec2/pricing).
- Dữ liệu luồng công việc được lấy từ các bộ dữ liệu thử nghiệm được xây dựng theo độ trù mật khác nhau và các luồng công việc từ các ứng

dụng thực tế như ứng dụng Montage [15]

- Những dữ liệu đó được tổng hợp lại và chia thành hai nhóm, nhóm 1 là các luồng công việc ngẫu nhiên với sự khác nhau về hệ số α, nhóm 2 là các luồng công việc từ ứng dụng Montage (Hình 3).

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

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 4,5,6,7.

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 LPSO 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 [18]. Các hàm của gói thư viện Jswarm [1] được sử dụng để thực hiện các phương thức Tối ưu bày đàn. Đối tượng so sánh là thuật toán PSO_H [8], thuật toán Random [11] 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 lí Intel Core i5 2.2 GHz, RAM 4GB, hệ điều hành Windows 7 Ultimate. Thực nghiệm được tiến hành một cách độc lập 30 lần trên mỗi bộ dữ liệu thực nghiệm.

5.4. Kết quả thực nghiệm

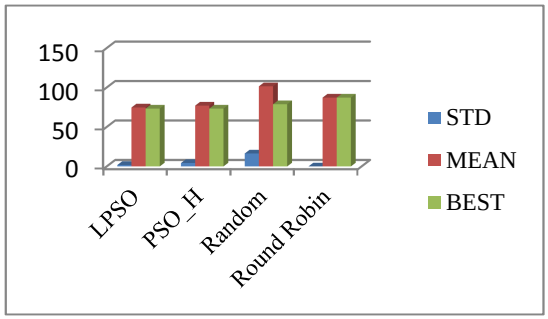
Hình 5, 6, 7, 8, 9 cho thấy sự chênh lệch về thời gian xử lí (makespan) của lời giải tốt nhất mà thuật toán đề xuất LPSO và các thuật toán đối chứng (PSO_H, Random và Round Robin) tìm được khi chạy trên các bộ dữ liệu khác nhau thuộc cả 2 nhóm luồng công việc ngẫu nhiên và luồng công việc từ ứng dụng Montage.

Bảng 2. Kết quả thực nghiệm cho các bộ dữ liệu ngẫu nhiên.

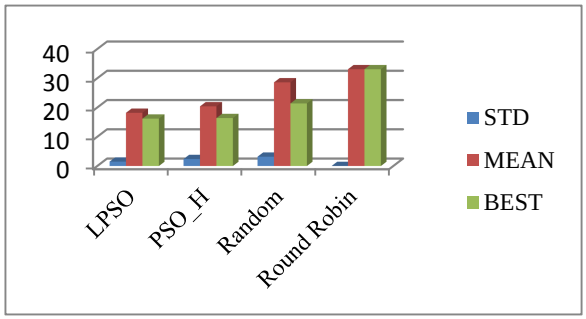
Bộ dữ liệu	Random			Round Robin			PSO_H			LPSO		
	STD	Mean	Best	STD	Mean	Best	STD	Mean	Best	STD	Mean	Best
T532	5.6	20.8	12.6	-	18.7	18.7	4.7	9.9	6.8	0.0	6.8	6.8
T534	16.4	101.5	78.8	-	87.2	87.2	4.3	77.1	73.3	1.8	74.7	73.3
T1031	3.2	28.6	21.4	-	33.1	33.1	2.4	20.4	16.4	1.5	18.2	16.2
T1034	3.7	29.7	22.1	-	28.6	28.6	1.4	22.9	19.7	1.0	21.5	19.6
T1051	3.7	27.0	19	-	23.7	23.7	1.8	19.5	16.4	1.0	17.6	14.8
T1054	3.4	27.4	20	-	25.2	25.2	2.0	20.7	17.9	0.9	19.0	17.7
T2031	6.9	66.2	50.7	-	72.2	72.2	5.4	41.2	33.2	1.5	34.1	31.2
T2034	7.9	65.6	47.4	-	77.3	77.3	4.7	41.9	33.2	1.5	36.2	34.0
T2051	6.8	59.0	45.8	-	71.9	71.9	5.0	41.0	29.6	3.1	34.2	28.5
T2054	7.4	59.7	45.3	-	71.9	71.9	3.6	43.2	35.9	3.2	36.0	30.9
T2081	3.9	63.3	56.3	-	72.7	72.7	5.2	44.2	35.0	2.7	37.8	32.3
T2084	6.8	67.6	51.6	-	72.0	72.0	6.1	44.7	37.1	2.7	39.3	34.8

Bảng 3. Kết quả thực nghiệm cho các luồng công việc từ ứng dụng Montage.

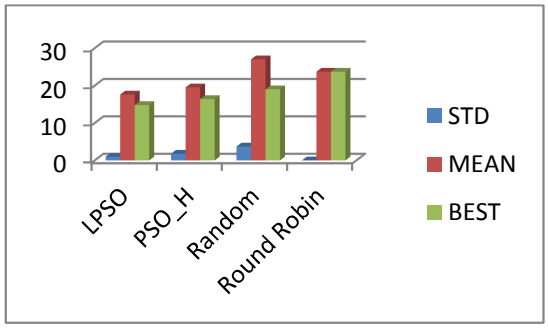
Bộ dữ liệu	Random			Round Robin			PSO_H			LPSO		
	STD	Mean	Best	STD	Mean	Best	STD	Mean	Best	STD	Mean	Best
M2031	48.6	482.5	354.6	-	491.8	491.8	36.6	337.3	255.2	11.7	242.8	229.5
M2032	8.5	161.9	140	-	162.7	162.7	4.9	142.1	130.6	3.6	128.4	122.7
M2033	33.2	485.6	435.2	-	491.8	491.8	16.7	484.9	427.5	7.2	459.5	427.5
M2051	9.0	156.9	143.8	-	146.6	146.6	5.4	132	122.8	3.3	123.4	118.7
M2052	38.0	337.3	264.2	-	356.1	356.1	23.5	257.5	180.4	18.6	195.6	180.4
M2053	44.6	540.2	450.2	-	448.8	448.8	10.9	461.1	440.1	9.3	437.7	421.4



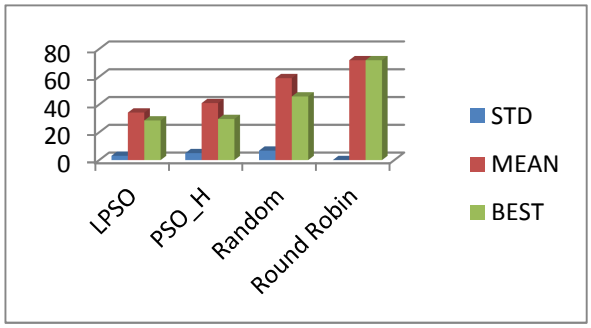
Hình 4. Kết quả thực nghiệm với bộ dữ liệu T534.



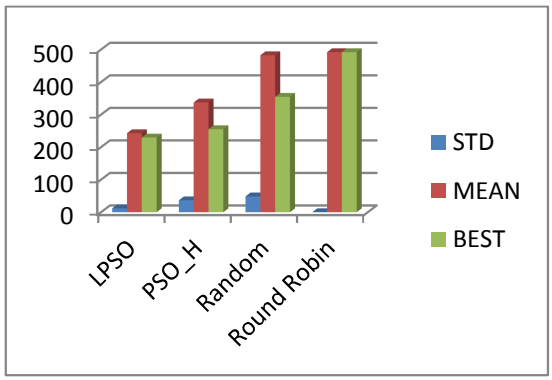
Hình 5. Kết quả thực nghiệm với bộ dữ liệu T1031.



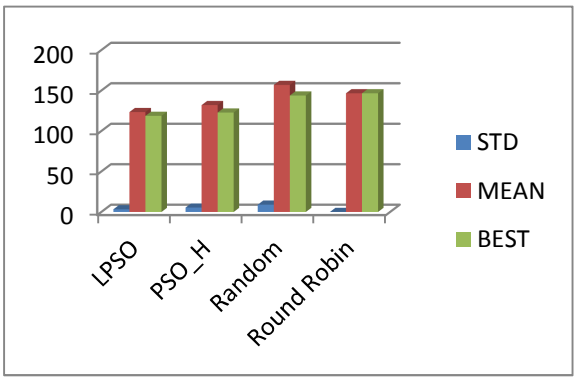
Hình 6. Kết quả thực nghiệm với bộ dữ liệu T1051



Hình 7. Kết quả thực nghiệm với bộ dữ liệu T2051.



Hình 8. Kết quả thực nghiệm với bộ dữ liệu M2031.



Hình 9. Kết quả thực nghiệm với bộ dữ liệu M2051.

Các thông số ở Bảng 2 và 3 cho thấy thuật toán được kiểm chứng trên nhiều bộ dữ liệu khác nhau về qui 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 LPSO đều cho lời giải tốt hơn các thuật toán PSO_H, Random và Round Robin ở cả 3 tham số là độ lệch chuẩn, giá trị trung bình và giá trị tốt nhất. Chúng tôi đã tiến hành thuật giải vét cạn trên một số bộ dữ liệu thực nghiệm để tìm lời giải tối ưu tuyệt đối và so sánh với giá trị tốt nhất tìm được bởi thuật toán LPSO kết quả được trình bày trong Bảng 4, lời giải tốt nhất tìm được bởi thuật toán LPSO có độ chênh so với giá trị tối ưu tuyệt đối từ 100% đến 105%.

Bảng 4. So sánh giá trị tối ưu tìm được bởi thuật toán LPSO và thuật toán vét cạn.

Bộ dữ liệu	Giá trị tối ưu tìm được bằng thuật toán vét cạn	Giá trị tối ưu tìm được bằng thuật toán LPSO	Độ chênh
T532	6.8	6.8	100 %
T534	73.3	73.3	100 %
T1031	16.2	16.2	100 %
T1034	19.3	19.6	102 %
T1051	14.3	14.8	103 %
T1054	16.8	17.7	105 %
M2032	119.8	122.7	102 %

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bài báo này đã trình bày một giải thuật tìm kiếm theo phương pháp Tối ưu bày 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 công thức tính vector dịch chuyển của cá thể thứ i theo giá trị gbest và $lbest_i$

- Đề xuất thủ tục LocalSearch 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 LPSO sử dụng phương thức cập nhật vị trí cá thể và thủ tục LocalSearch 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_H, 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 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, Von Neuman 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. Buyya R., Ranjan R., Calheiros R.N. - Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities, Proceedings of the Conference on High Performance Computing and Simulation, 2009, pp. 1–11. (http://www.cloudbus.org/cloudsim/).
2. Ullman J. D. - NP-complete scheduling problems, Journal of Computer and System Sciences **10** (3) (1975) 384-393.
3. Pars S., Maleki R. E. - RASA: A New Task Scheduling Algorithm in Grid Environment, International Journal of Digital Content Technology and its Applications **3** (4) (2009) 152-160.
4. Cope J.M, Trebon N., Tufo H.M., Beckman P. - Robust data placement in urgent computing environments, IEEE International Symposium on Parallel & Distributed Processing, IPDPS (2009) 1-13.
5. Agarwal A., Jain S. - Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment, International Journal of Computer Trends and Technology (IJCTT) **9** (2014) 344-349.
6. Wieczorek. M., - Marek Scheduling of Scientific Workflows in the ASKALON Grid Environment, ACM SIGMOD Record Journal **34**(3) (2005) 56-62.
7. Salman. A., Ahmad. I. - Particle swarm optimization for task assignment Problem, Microprocessors and Microsystems **26**(8) (2002) 363-371.
8. Pandey.S., Barker. A., Gupta. K.K., Buyya. R. - Minimizing Execution costs when using globally distributed cloud services, 24th IEEE International Conference on Advanced Information Networking and Applications (2010) 222-229.
9. Tarek. Z., Zakria. M., Omara. A. - Pso Optimization algorithm for Task Scheduling on The Cloud Computing Environment, internation journal of computer and technology **13**(9) 4885-4897.
10. Davidovic T., Selmic M., Teodorovic D., Ramljak D. - Bee colony optimization for scheduling independent tasks to identical processors, Journal of Heuristics **18** (2012) 549-569.

11. Don Fallis - The Reliability of Randomized Algorithms, British Journal for the Philosophy of Science **51** (2000) 255-271.
12. Kennedy J., Eberhart R.C. - Particle swarm optimization, in Proc. IEEE Int'l. Conference on Neural Networks **4** (1995) 1942-1948.
13. Mitzenmacher M., Upfal E., - Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge University Press, January 31, 2005.
14. Vliet J. V., Paganelli F. - Programming Amazon EC2, O'Reilly Media, ISBN 1449393683, 2011.
15. Bruce Berriman G., Deelman E. - Montage: A grid enabled engine for delivering custom science-grade mosaics on demand, in SPIE, 2004 (http://montage.ipac.caltech.edu)

ABSTRACT

LPSO: AN EFFICIENT ALGORITHM SCHEDULING OF SCIENTIFIC WORKFLOWS IN CLOUD COMPUTING

Toan Phan Thanh[1], Loc Nguyen The[2], Cuong Nguyen Doan[3]

[1]*Faculty of Technology Education, Hanoi National University of Education, 136 Xuan Thuy, Hanoi*

[2]*Faculty of Information Technology, Hanoi National University of Education, 136 Xuan Thuy, Hanoi*

[3]*Institute of Information Technology, Military Institute of Science and Technology, 17 Hoang Sam, Cau Giay, Hanoi*

[*]Email: *pttoan@hnue.edu.vn*

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 algorithms in the literature have been targeting the workflow scheduling problem, however, handful efficient solutions have been proposed. This paper proposes a metaheuristic algorithm called LPSO, which based on the Particle Swarm Optimization method. Our experiments, which arranged by using the simulation tool CloudSim show that LPSO is superior to the general algorithms called Random and RoundRobin, moreover the deviation between the solution found by LPSO and the optimal solution is negligible.

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