Toán tử lân cận mới cho thuật toán Tabu Search và PSO giải bài toán lập lịch luồng công việc trong môi trường điện toán đám mây

Article *in* Journal on Information Technologies & Communications · December 2019

DOI: 10.32913/mic-ict-research-vn.v2019.n2.865

CITATIONS
0

READS
811

3 authors:

Phan Toan
Northwestern University
19 PUBLICATIONS **53** CITATIONS
SEE PROFILE

Huu Dang Quoc
Thuongmai University
18 PUBLICATIONS **44** CITATIONS
SEE PROFILE

Loc Nguyen The
Đại học Sư phạm Hà Nội
22 PUBLICATIONS **162** CITATIONS
SEE PROFILE

Toán tử lân cận mới cho thuật toán Tabu search và PSO giải bài toán lập lịch luồng công việc trong môi trường điện toán đám mây

New Effective Neighborhoods for the Tabu Search and Particle Swarm Optimization Algorithm Solves the Workflow Scheduling in Cloud Computing

Phan Thanh Toàn, Đặng Quốc Hữu, Nguyễn Thế Lộc

***Abstract**: Cloud computing is a new trend of information and communication technology that enables resource distribution and sharing at a large scale. The Cloud consists of a collection of virtual machine that promise to provision on-demand computational and storage resources when needed. End-users can access these resources via the Internet and have to pay only for their usage. Workflow scheduling is a big issue in the era of Cloud computing. Basically it is the issue related to the discovering resources and allocating tasks on suitable resources. Workflow scheduling plays a vital role in the system management. In this work, we propose a new algorithm for workflow scheduling that is derived from the Particle Swarm Optimization and Tabu Search Algorithm.*

***Keyword**: Workflow scheduling, Tabu Search, Particle Swarm Optimization, cloud computing.*

I. GIỚI THIỆU

Với sự phát triển của công nghệ thông tin và truyền thông, điện toán đám mây đã được ứng dụng rộng rãi trong nghiên cứu khoa học và thực tiễn. Mọi tài nguyên trong môi trường điện toán đám mây đều được cung cấp cho người dùng dưới dạng dịch vụ, như các dịch vụ về phần mềm Software as a Service (SaaS), dịch vụ cơ sở hạ tầng Infrastructure as a Service (IaaS), dịch vụ nền tảng hạ tầng Platform as a Service (PaaS). Rất nhiều ứng dụng được mô hình hóa dưới dạng luồng công việc bao gồm tập các tác vụ và các phụ thuộc giữa những tác vụ theo kiểu cha-con: tác vụ con chỉ được bắt đầu sau khi tác vụ cha đã hoàn thành. Ứng dụng dạng luồng công việc được sử dụng rộng rãi trong nhiều lĩnh vực nghiên cứu khoa học và thực tiễn như thiên văn học, tin sinh, dự báo động đất,…Hơn nữa ngày nay các ứng dụng ngày càng phức tạp và đòi hỏi phải xử lí một khối lượng lớn dữ liệu, chính vì vậy các ứng dụng này cần phải được thực hiện trên các hệ thống siêu máy tính, hệ thống tính toán lưới, hay điện toán đám mây. Lập lịch luồng công việc thực chất là tìm phương án để gán các tác vụ của luồng công việc vào thực hiện trên các máy ảo (VM – Virtual Machine) của môi trường điện toán đám mây nhằm cực tiểu thời gian hoàn thành và (hoặc) chi phí thực hiện luồng công việc.

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. 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.

Bài toán lập lịch luồng công việc đã được nghiên cứu từ những năm 1950 và đã được chứng minh là thuộc lớp NP-Khó. Trong những năm gần đây đã có rất nhiều ứng dụng khoa học được mô hình hóa bởi dạng đồ thị luồng công việc như ứng dụng Montage [1], CyberShake [2], Epigenomics [3], LIGO [4], v.v.

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.

(ii) Toán tử lân cận mới cho thuật toán Tabu search nhằm thoát khỏi cực trị địa phương trong phương pháp PSO.

(iii) Thuật toán lập lịch mới tên là TSPSO.

Phần V mô tả các thực nghiệm được tiến hành dựa trên công cụ mô phỏng Cloudsim [5] 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. NHỮNG CÔNG TRÌNH LIÊN QUAN

Bài toán lập lịch luồng công việc được ứng dụng trong nhiều lĩnh vực của khoa học và thực tiễn, bài toán này đã được chứng minh thuộc lớp NP-Khó [6]. Lập lịch luồng công việc bản chất là tìm phương án gán các tác vụ của luồng công việc vào thực hiện trên các tài nguyên sao cho đảm bảo thứ tự của các tác vụ trong luồng công việc và cực tiểu hóa thời gian hoàn thành và (hoặc) chi phí thực thi luồng công việc. Do bài toán thuộc lớp NP-Khó nên việc tìm lời giải đúng cho các luồng công việc có số lượng tác vụ lớn là không khả thi, vì vậy đã có nhiều công trình nghiên cứu nhằm tìm ra lời giải gần đúng cho bài toán này.

K. Dubey và các cộng sự [7] đã đề xuất thuật toán lập lịch điều phối các tác vụ của luồng công việc trong môi trường điện toán đám mây dựa trên việc cải tiến thuật toán HEFT nhằm cực tiểu hóa thời gian hoàn thành luồng công việc, trong công trình nhóm tác giả đã trình bày tóm tắt các công trình nghiên cứu liên quan đến bài toán lập lịch luồng công việc và đề xuất một thuật toán lập lịch mới dựa trên thuật toán HEFT, kết quả thực nghiệm đã chỉ ra thuật toán mới có thời gian hoàn thành luồng công việc tốt hơn thuật toán CPOP và HEFT.

Ahmad M. Manasrah [8] đã đề xuất thuật toán lập lịch luồng công việc trong môi trường điện toán đám mây dựa trên kết hợp giữa thuật toán di truyền và thuật toán tối ưu bày đàn, trong công trình nhóm tác giả đã thực hiện khởi tạo quần thể ban đầu một cách ngẫu nhiên, sau đó thực hiện các toán tử cơ bản của thuật toán di truyền cho quần thể, tiếp theo sẽ áp dụng thuật toán tối ưu bày đàn dựa trên quần thể đã được tiến hóa bởi thuật toán di truyền. Kết quả thực nghiệm đã chỉ ra thuật toán đề xuất làm việc tốt hơn thuật toán GA và PSO.

N.S.Grigoreva [9] đã đề xuất thuật toán lập lịch điều phối các tác vụ của luồng công việc vào thực hiện trên một hệ thống đa bộ vi xử lý nhằm cực tiểu hóa thời gian hoàn thành luồng công việc. Tác giả đã sử dụng kết hợp phương pháp nhánh cận và kỹ thuật tìm kiếm nhị phân để tìm ra phương án xếp lịch có thời gian hoàn thành luồng công việc là nhỏ nhất.

R. Rajkumar [10] đã đề xuất thuật toán lập lịch luồng công việc dựa trên nhu cầu của khách hàng như thời gian hoàn thành, chi phí thực thi,… qua đó sẽ điều phối các tác vụ vào thực hiện trên các máy chủ nhằm thỏa mãn tốt nhất nhu cầu của khách hàng.

Các tác giả trong bài báo [11] đã đề xuất thuật toán EGA (Enhanced Genetic Algorithm) lập lịch bằng phương pháp di truyền. Trong công trình các tác giả sử dụng thuật toán Enhanced Max Min trong bước khởi tạo quần thể nhằm tìm ra các cá thể tốt cho quá trình tiến hóa.

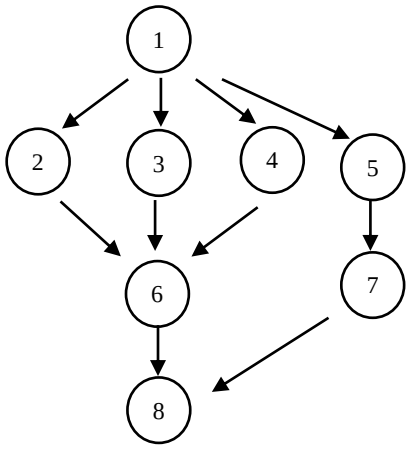
Pandey [12] đã đề xuất thuật toán lập lịch luồng công việc PSO Heuristic (Particle Swarm Optimization Heuristic – PSO_H) trong môi trường điện toán đám mây dựa trên chiến lược tối ưu bày đàn. Rajkumar đã đề xuất một thuật toán lập lịch phân cấp [12] và đưa vào các tham số dịch vụ khác nhau, chẳng hạn như thời gian đáp ứng. Thuật toán sử dụng tham số này như một quyền ưu tiên để lựa chọn các tác vụ lập lịch. Q.Cao và các đồng nghiệp đã trình bày thuật toán lập lịch dựa trên giải thuật ABC (Activity Based Costing) [13]. Thuật toán này gán mức ưu tiên cho mỗi tác vụ trong luồng công việc theo các tham số về thời gian, không gian, các tài nguyên và chi phí, quá trình lập lịch sẽ sử dụng mức ưu tiên này để quyết định các tác vụ được chọn trong quá trình lập lịch.

III. MÔ HÌNH LÝ THUYẾT

Đồ thị luồng công việc được biểu diễn bởi đồ thị có hướng, không có chu trình $G=(V,E)$, ví dụ như ở Hình 1.

Trong đó:

- V là tập đỉnh, mỗi đỉnh tương ứng với một tác vụ trong đồ thị luồng công việc.
- $T=\{T_1, T_2,\ldots,T_M\}$ là tập các tác vụ.



Hình 1. Đồ thị biểu diễn một luồng công việc với 8 tác vụ.

- E là tập cạnh, biểu diễn mối quan hệ giữa các tác vụ. Nếu $e = (T_i, T_k)$ là một cạnh của đồ thị G, có nghĩa tác vụ T_i là tác vụ cha của tác vụ T_k, và tác vụ T_i sẽ gửi tới tác vụ T_k một khối lượng dữ liệu là $tdata^k$.
- $S = \{S_1, S_2, ..., S_N\}$ là tập N máy chủ trong môi trường điện toán đám mây. Mỗi máy chủ được xác định bởi một năng lực tính toán xác định là $P(S_i)$, đơn vị đo là MI/s (million instructions/second)
- Khối lượng tính toán của tác vụ T_i kí hiệu là W_i (flop-floating point operations)
- $P(S_i)$ là năng lực tính toán của máy chủ S_i (MI/s: million instructions/second)
- Mỗi cặp máy chủ đều được kết nối với nhau bởi một đường truyền riêng, và có băng thông là $B(S_i, S_j)$. Băng thông được xác định bởi hàm:

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

  Do tính chất của băng thông, hiển nhiên ta có:

  $B(S_i,S_i) = \infty$ và $B(S_i,S_j) = B(S_j,S_i)$ $\forall i,j$

Khái niệm lịch biểu

Mỗi lịch biểu được biểu diễn bởi hàm:

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

Trong đó $f(T_i)$ là máy chủ được giao để thực hiện tác vụ T_i.

Các tham số và điều kiện ràng buộc

- x_j^k là biến logic và được xác định như sau: $x_j^k = 1$ nếu tác vụ T_k được gán vào thực hiện trên máy chủ S_j.
- $d_{i,j}^k$ là khối lượng dữ liệu được truyền từ máy chủ S_i tới S_j cho tác vụ T_k nếu $x_j^k = 1$
- $tftime_{i,j}$ là thời gian truyền dữ liệu từ máy chủ S_i tới S_j cho tác vụ T_k nếu $d_{i,j}^k > 0$ and $x_j^k = 1$

  $$tftime_{i,j} = \frac{d_{i,j}^k}{B(S_i,S_j)} \quad (1)$$

- $extime_j^k$ thời gian thực thi tác vụ T_k trên máy chủ S_j nếu $x_j^k = 1$

  $$etime_j^k = \frac{W_k}{P(S_j)} \quad (2)$$

- Thời gian thực hiện tác vụ T_k kí hiệu là ET^k

  $$ET^k = \sum_{i=1}^N \sum_{j=1}^N d_{i,j}^k \times tftime_{i,j} \times x_j^k + \sum_{j=1}^N extime_j^k \times x_j^k \quad (3)$$

- Thời gian tính toán của tác vụ T_i là: $\frac{W_i}{P(f(T_i))}$

  (i=1,2, ... M)
- Thời gian truyền dữ liệu giữa tác vụ T_i và tác vụ con T_j là $\frac{D_{ij}}{B(f(T_i), f(T_j))}$
- Kí hiệu thời gian hoàn thành luồng công việc (Makespan) là CT :

  $$CT = \max\{ET^k\} ; k=1,2,\ldots,M$$

Các điều kiện ràng buộc như sau:

a) $x_j^k \geq 0;\ \forall k = 1,2,..,M \text{ and } j = 1,2,\ldots,N$

b) $d_{i,j}^k \geq 0, \forall i,j = 1,2,..,N \text{ and } k = 1,2,\ldots,M$

c) $tdata_j^k \geq 0, \forall k = 1,2,..,M$

d) $tftime_{i,j} \geq 0;\ \forall i,j = 1,2,\ldots,N$

e) $extime_j^k \geq 0;\ \forall k = 1,2,..,M \text{ and } j = 1,2,\ldots,N$

f) $\sum_{j=1}^N x_j^k = 1$

g) $\sum_{i=1}^N \sum_{j=1}^N x_j^k \times d_{i,j}^k = tdata^k$

h) $\sum_{k=1}^M \sum_{i=1}^N \sum_{j=1}^N x_j^k \times d_{i,j}^k = \sum_{k=1}^M tdata^k$

Hàm mục tiêu: cần tối thiểu hóa Makespan

$$max_{k=1,2,\ldots M} \left\{ \sum_{i=1}^N \sum_{j=1}^N d_{i,j}^k \times tftime_{i,j} \times x_j^k + \sum_{j=1}^N extime_j^k \times x_j^k \right\} \rightarrow min$$

trong đó CT 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. GIẢI PHÁP 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. Cả vector vị trí và vector dịch chuyển đều được biểu diễn bằng cấu trúc dữ liệu bảng băm trong ngôn ngữ lập trình java.

Kí hiệu vector vị trí $x_i = (\pi_{i1}, \pi_{i2}, \ldots, \pi_{iM})$; với $\pi_{ij} \in S$; j=1, 2,…, M

Ví dụ 1: giả sử luồng công việc gồm tập tác vụ T={T_1, T_2, T_3, T_4, T_5, T_6 , T_7}, 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 ; 3 ;1] chính là phương án xếp lịch mà theo đó tác vụ T_1, T_3, T_7 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, T_6 được thực hiện bởi S_3.

T_1	T_2	T_3	T_4	T_5	T_6	T_7
1	2	1	3	2	3	1

4.2. Phương pháp cập nhật vị trí cá thể

Xét công thức cập nhật vị trí cá thể theo công thức gốc của PSO:

$$v_i^{k+1} = \omega v_i^k + c_1\, rand_1 \times (pbest_i - x_i^k) + c_2\, rand_2 \times (gbest - x_i^k) \quad (4)$$

$$x_i^{k+1} = x_i^k + v_i^k \quad (5)$$

(4) và (5) cho thấy, giống như đa số các metaheuristic khác, PSO vốn được thiết kế cho dữ liệu liên tục, các thành phần của vector dịch chuyển v_i^k là số thực do công thức (4) 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 (5) 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 [12] 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.

$$x_i^{k+1}[t] \leftarrow j \text{ nếu } \left|P(S_j) - \left(x_i^k[t] + v_i^k[t]\right)\right| \leq \left|P(S_r) - \left(x_i^k[t] + v_i^k[t]\right)\right| \forall S_r \in S; t = 1,2 \ldots M \quad (6)$$

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 :

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

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 (4): ($pbest_i$ - x_i^k) và ($gbest$ - x_i^k). Một số công trình hiện có như [10] 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 (4) 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

$$y_k = \left\{P(x_{ik}) + \frac{\sum_{q\in S} B(x_{ik}, x_q)}{N-1}\right\} - \left\{P(x_{jk}) + \frac{\sum_{q\in S} B(x_{jk}, x_q)}{N-1}\right\} \text{ với } k = 1,2,\ldots M \quad (7)$$

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 $lbest_j$ = [2,1,2,1,1] ; x_j = [3,2,1,2,1] ;

Vậy $lbest_j - x_j = [y_1, y_2, y_3, y_4, y_5]$ 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.

4.4. 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 các toán tử lân cận mới và sử dụng phương pháp tìm kiếm lân cận Tabu Search cho cá thể gbest để tìm kiếm phần tử gbest mới tốt hơn và qua đó định hướng quần thể dịch chuyển sang vùng tìm kiếm mới.

4.4.1.Thủ tục tìm kiếm lân cận

Tabu search là phương pháp tìm kiếm lân cận được để xuất bởi Glove vào năm 1986 [14], [15], phương pháp này đã được ứng dụng vào tìm lời giải gần đúng cho nhiều bài toán tối ưu tổ hợp. Phương pháp Tabu search bắt đầu từ một lời giải ngẫu nhiên của bài toán, sau đó sẽ áp dụng một số các toán tử lân cận để sinh ra lời giải mới, quá trình này được lặp đi lặp lại cho đến khi tìm ra lời giải tối ưu hoặc thỏa mãn điều kiện của bài toán. Bài báo này để xuất các toán tử lân cận mới sử dụng cho thuật toán Tabu Search

Gọi *remove(π, p)* là toán tử loại bỏ phần tử π_p trong vector vị trí của cá thể (π_1, π_2,…., π_n). Sau khi thực hiện toán tử này ta có vector mới là : (π_1, π_2,…, π_{p-1}, π_{p+1}, ., π_n). *Insert(π, p, S_i)* là toán tử chèn vào vị trí p trong vector vị trí của cá thể máy chủ mới là S_i. Sau khi thực hiện toán tử này ta có vector mới là: (π_1, π_2,…, π_{p-1}, S_i, π_p, ., π_n). Toán tử lân cận Best Exchange: hoán đổi vị trí của hai máy chủ tại vị trí p_1 và p_2 tùy theo năng lực của máy chủ.

Function BE (p_1, p_2)

Input: vector vị trí (π_1, π_2,.., π_{p1},.. ., π_{p2},.. π_n)
Output: vector vị trí (π_1, π_2,.., π_{p2},.. ., π_{p1},.. π_n)
1. $S_1 \leftarrow$ remove(π, p_1)
2. if(P(π_{p2}) > P(π_{p1}))
3. insert(π, p_2-1, S_1)
4. else insert(π, p_2, S_i)
5. if(P(π_{p2}) > P(π_{p1}))
6. $S_2 \leftarrow$ remove(π, p_2)
7. else $S_2 \leftarrow$ remove(π, p_{2+1})
8. insert(π, p_1-1, S_2)
End Function

Toán tử Best Swap (BS): toán tử này tìm ra vị trí tốt nhất p trong một cá thể xi = (π1, π2,…, ..., πn) và thực hiện hoán chuyển hai vị trí p và p+1. Vị trí tốt nhất được tính toán dựa trên giá trị makespan của phương án hiện thời.

Function BS ()

Input: vector vị trí (π_1, π_2,…, π_n)
Output: vector vị trí (π'_1, π'_2,..,π'_n)
1. $S_i \leftarrow$ Max{P(S_1), P(S_2),…, P(S_n)}
2. for(i=1 ; i< M ; i++)

```
3.  Mi = makespan((π1, π2,.., πi-1,Si,
    πi+1 ..., πn))
4. k ← min(Mi)
5. j ← remove(π, k)
6. insert(π, k+1, j)
End Function
```

4.4.2. Giải thuật tìm kiếm lân cận

Function Tabu_Search (vector vị trí x_i)

```
Input: vector vị trí xi
Output: vector vị trí xk có f(xk) < f(xi)
1. Khởi tạo bước lặp t ← 0
2. while (điều kiện lặp)
3.    Khởi tạo giá trị r1, r2 ngẫu
      nhiên trong đoạn [1, M]
4.    xi ← BE(xi, r1, r2)
6.    xk ← BS (xi)
7.    if f(xk) < f(xi)  then return xk
5.    else return xi
6.    t ← t+1
7. End while
End Function
```

4.5. Thuật toán đề xuất TSPSO

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

Algorithm TSPSO

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 xi theo
         công thức (6)
6.       Mi = Makespan của cá thể xi
7.    end for
8.    for i=1 to SCT do
9.       if pbesti < Mi then pbesti= Mi
10.   end for
11.   gbest = Min{Mi}
12.   for i=1 to SCT do
13.   Cập nhật vector dịch chuyển vi^k
      theo công thức (4) và (6)
14.   Tính xi theo (5)
15.   end for
16.   t++ ;
17.   if (sau K thế hệ mà độ lệch giữa
      các gbest không vượt quá ε) then
18.   gbest=Tabu_Search (gbest);
19.   end if
20. end while
  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 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 Tabu_Search được gọi để 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.

Độ phức tạp của thuật toán TSPSO. Trước khi thực hiện thuật toán chính TSPSO 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 TSPSO 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). Thuật toán Tabu_Search có độ phức tạp là O(M^2)

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 TSPSO là (Số thế hệ)× O(M^2).

V. KẾT QUẢ 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 [16] 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 [17]
- 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.

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

- 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.

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 (Megabit/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 TSPSO chúng tôi đã sử dụng công cụ mô phỏng Cloudsim [5] 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 [5] được sử dụng để thực hiện các phương thức Tối ưu bày đàn. Đối tượng so sánh là thuật toán PSO_H [12], thuật toán EGA [18] và thuật toán Round Robin [19].

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 tiến hành một cách độc lập 30 lần trên mỗi bộ dữ liệu thực nghiệm.

5.4. Kết quả thực nghiệm

Hình 4,5,6 cho thấy sự chênh lệch về thời gian xử lý (makespan) của lời giải tốt nhất mà thuật toán đề xuất TSPSO và các thuật toán đối chứng (PSO_H, EGA 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.

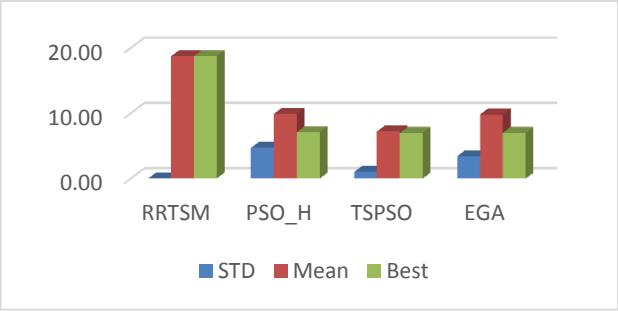
Bảng 2: Kết quả thực hiện thuật toán với các bộ dữ liệu ngẫu nhiên

Kí hiệu	RRTSM			PSO_H			TSPSO			EGA		
	STD	Mean	Best	STD	Mean	Best	STD	Mean	Best	STD	Mean	Best
T532	-	18.7	18.7	4.7	9.9	7.1	1.0	7.2	7.0	3.4	9.8	7.0
T1031	-	33.1	33.1	2.4	20.4	17.4	1.1	18.3	16.5	1.2	20.4	16.8
T1032	-	16	16	2.0	8.2	3.5	1.2	5.1	3.5	1.8	8.1	3.5
T1035	-	18.9	18.9	2.5	9.7	5.2	1.8	6.8	3.9	2.3	9.7	4.9
T1051	-	23.7	23.7	1.5	21.5	16.4	1.1	17.6	14.8	1.4	19.1	16.1
T1054	-	25.2	25.2	2.0	20.7	18.9	0.9	19.0	17.3	1.3	20.5	17.6
T2081	-	72.7	72.7	5.2	44.2	34.1	2.7	37.8	32.4	3.7	41.9	35.1
T2083	-	20.3	20.3	2.2	21.2	19.8	0.5	19.4	18.2	1.4	20.3	19.6
T2084	-	72.0	72.0	6.1	44.7	37.4	2.7	39.3	34.1	3.2	42.5	36.5
T2086	-	32.5	32.5	1.5	26.2	22.8	1.2	21.4	17.4	1.4	25.4	22.6

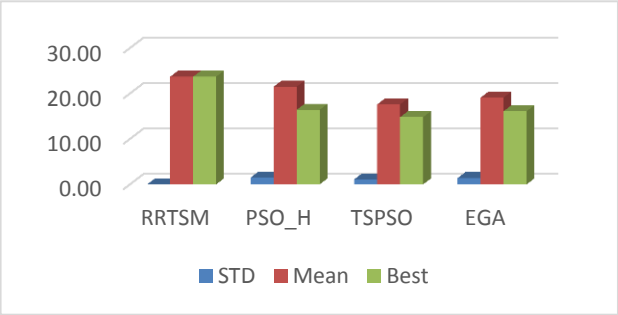
Bảng 3: Kết quả thực hiện thuật toán với các bộ dữ liệu từ ứng dụng Montage

Kí hiệu	RRTSM			PSO_H			TSPSO			EGA		
	STD	Mean	Best	STD	Mean	Best	STD	Mean	Best	STD	Mean	Best
M2032	-	162.7	162.7	4.9	142.7	131.6	3.6	123.4	129.0	5.3	135.5	130.1
M2051	-	146.6	146.6	5.4	132	121.8	3.3	123.4	115.7	4.5	131.7	123.3
M2531	-	465.0	465.0	18.7	373.4	345.5	13.0	346.4	336.1	7.0	352.7	339.4

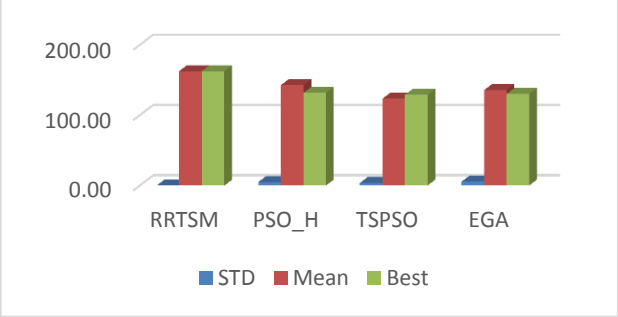
M2532	-	183.8	183.8	3.9	110.7	101.5	1.5	104.7	101.2	1.4	112.1	104.5
M2533	-	322.9	322.9	4.0	311.4	311.7	0.5	312.5	311.6	0.5	315.2	311.8
M2581	-	300.6	300.6	15	261.3	232.1	6.1	236.1	223.4	6.5	246.2	231.9
M2582	-	133.9	133.9	5.1	84.8	77.9	4.7	81.4	72.3	2.9	85.1	77.6
M2583	-	236.5	236.5	8.3	239	224	5.3	221.3	215.7	4.7	233.5	221.4
M5081	-	155.8	155.8	6.3	108.0	95.0	5.5	101.7	91.5	3.2	107.8	96.6
M5082	-	82.1	82.1	0.9	14.0	18.1	0.8	12.6	13.1	0.5	14.0	14.8
M5083	-	101.7	101.7	4.3	98.3	89.8	4.3	90.0	80.2	1.8	95.3	88.5



Hình 4: So sánh các thuật toán với bộ dữ liệu T532



Hình 5: So sánh các thuật toán với bộ dữ liệu T1051



Hình 6: So sánh các thuật toán với bộ dữ liệu M2032

Kết quả thực nghiệm được trình bày chi tiết trong bảng 2, 3 và các hình 4,5,6, kết quả so sánh giữa giá trị trung bình tính được bởi thuật toán TSPSO với các thuật toán đối sánh, trong hầu hết các trường hợp thuật toán TSPSO đều cho kết quả tốt hơn các thuật toán đối sánh, giá trị trung bình tìm được bởi TSPSO nhỏ hơn giá trị trung bình tìm được bởi PSO_H từ 4% - 11% và nhỏ hơn giá trị trung bình tìm được bởi thuật toán EGA từ 2% - 7%.

Hình 1,2,3,4 cũng so sánh giữa giá trị tốt nhất tìm được bởi thuật toán TSPSO với các thuật toán đối sánh, qua đó ta thấy giá trị tốt nhất tìm được bởi TSPSO nhỏ hơn giá trị tốt nhất tìm được bởi PSO_H từ 2% - 9%, và nhỏ hơn giá trị tốt nhất tìm được bởi Random từ 20% - 40%, giá trị tốt nhất tìm được bởi thuật toán TSPSO nhỏ hơn giá trị tốt nhất tìm được bởi thuật toán EGA từ 1% - 8% . Cuối cùng các hình 1,2,3,4 chỉ ra kết quả so sánh giữa độ lệch chuẩn tìm được bởi thuật toán TSPSO với các thuật toán đối sánh, giá trị độ lệch chuẩn của TSPSO đều nhỏ hơn độ lệch chuẩn của các thuật toán RRTSM, PSO_H và EGA, điều đó chứng tỏ thuật toán TSPSO có chất lượng lời giải tốt hơn các thuật toán đối sánh và độ ổn định trong các lần chạy cũng tốt hơn.

VI. KẾT LUẬN

Bài báo này đã trình bày một kiến trúc lân cận mới cho thuật toán Tabu Search và thuật toán 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 $pbest_i$
- Đề xuất thủ tục hai toán tử lân cận mới cho thuật toán tìm kiếm lân cận Tabu Search để chương trình thoát ra khỏi cực trị địa phương bằng cách dịch chuyển các cá thể tới một miền không gian tìm kiếm mới.
- Đề xuất thuật toán TSPSO sử dụng phương thức cập nhật vị trí cá thể và thủ tục Tabu Search 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 EGA và thuật toán Round Robin. Về hướng công việc tiếp theo chúng tôi dự định đề xuất một kiến trúc lân cận mới phù hợp với bài toán nhằm nâng cao khả năng tìm kiếm tổng thể qua đó nhằm đạt được lời giải có chất lượng tốt hơn.

TÀI LIỆU THAM KHẢO

[1] G. B. Berriman, et al, *Montage: A Grid Enabled Engine for Delivering Custom Science-Grade Mosaics On Demand*", in *SPIE Conference*, 2004.

[2] P. Maechling, et al, *SCEC CyberShake Workflows, Automating Probabilistic Seismic Hazard Analysis Calculations*", Springer, 2006.

[3] "USC Epigenome Center". http://epigenome.usc.edu. [Online]. http://epigenome.usc.edu

[4] LIGO Project. LIGO - Laser Interferometer Gravitational Wave Observatory. [Online]. http://www.ligo.caltech.edu.

[5] 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) 1–11. (http://www.cloudbus.org/cloudsim/).

[6] J.D. Ullman, *NP-complete scheduling problems*, Journal of Computer and System Sciences, pages 384-393, volume 10, issue 3, 1975

[7] K. Dubey, M. Kumar, S.C. Sharma, *Modified HEFT Algorithm for Task Scheduling in Cloud Enveronment*, International conference on Smart Computing and Communications, ICSCC 2017

[8] Ahmad M. Manasrah, Hanan Ba Ali, *Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing*, Wireless Communications and Mobile Computing Volume 2018, Article ID 1934784, 16 pages

[9] N.S.Grigoreva, *Branch and Bound Method for Scheduling Precedence Constrained Tasks on Parallel Identical Processors*, Proceedings of the World Congress on Engineering, Vol II, 2014

[10] R. Rajkumar, T. Mala, *Achieving Service Level Agreement in Cloud Environment using Job Prioritization in Hierarchical Scheduling*, Proceeding of International Conference on Information System Design and Intelligent Application, 2012, vol 132, pp 547-554.

[11] S. Singh, M.Kalra, *Task scheduling optimization of independent tasks in cloud computing using enhanced genetic algorithm*, International Journal of Application or Innovation in Engineering & Management, V.3, Issue 7, 2014.

[12] S. Pandey, L. Wu1, S. M. Guru, R. Buyya, *A Particle Swarm Optimization (PSO)-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments*, Proc. of 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), pages 400-407, 2010

[13] Q. Cao, W. Gong and Z. Wei, *An Optimized Algorithm for Task Scheduling Based On Activity Based Costing in Cloud Computing*, In Proceedings of Third International Conference on Bioinformatics and Biomedical Engineering, 2009, pp.1-3

[14] Glover, F., Tabu search - part I, ORSA J. Comput. 1, 190-206, 1989.

[15] Glover, F., De Werra D., Tabu Search, Annuals of Operations Research, 41, 1993

[16] Vliet J. V., Paganelli F. - Programming Amazon EC2, O'Reilly Media, ISBN 1449393683, 2011.

[17] 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)

[18] S. Singh, M.Kalra, *Task scheduling optimization of independent tasks in cloud computing using enhanced genetic algorithm*, International Journal of Application or Innovation in Engineering & Management, V.3, Issue 7, 2014.

[19] M. Mitzenmacher, E. Upfal, “Probability and Computing: Randomized Algorithms and Probabilistic Analysis”, Cambridge University Press (2005).

SƠ LƯỢC VỀ TÁC GIẢ

PHAN THANH TOÀN

Sinh năm 1974 tại Thái Nguyên, tốt nghiệp đại học và thạc sĩ tại trường Đại học Bách khoa Hà Nội, nhận bằng tiến sĩ năm 2018 tại Học viện Khoa học Công nghệ Quân sự. Hiện đang công tác tại trường Đại học Sư phạm Hà Nội.

Lĩnh vực nghiên cứu: phương pháp tiến hóa, tối ưu, xử lý song song và phân tán.

Điện thoại : 0912.069.762

E-mail: pttoan@hnue.edu.vn

NGUYỄN THẾ LỘC

Tốt nghiệp đại học khoa Toán Tin, đại học Sư phạm Hà Nội năm 1993, tốt nghiệp thạc sĩ CNTT tại trường đại học Bách khoa Hà Nội, nhận bằng tiến sỹ tại Viện Nghiên cứu Khoa học Công nghệ Nhật Bản JAIST năm 2007. Hiện đang công tác tại trường Đại học Sư phạm Hà Nội.

Lĩnh vực nghiên cứu: mạng máy tính và truyền thông, xử lý song song và phân tán.

Điện thoại : 0988.765.837.

E-mail: locnt@hnue.edu.vn.

ĐẶNG QUỐC HỮU

Tốt nghiệp đại học và thạc sĩ khoa CNTT, Đại học Quốc Gia Hà Nội năm 2006 và 2015, đang làm nghiên cứu sinh tại Viện Khoa học và Công nghệ Quân sự từ năm 2017. Hiện đang công tác tại đại học Thương Mại.

Lĩnh vực nghiên cứu: mạng máy tính và truyền thông, xử lý song song và phân tán.

Điện thoại : 0988710727.

E-mail: huudq@tmu.edu.vn.