# 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

Phan Thanh Toàn[1], Đặng Quốc Hữu[2], Nguyễn Thế Lộc[3]

[1] Khoa Sư phạm Kỹ thuật, Trường Đại học Sư phạm Hà Nội

[2] Trung tâm Công nghệ Thông tin, Trường Đại học Thương mại, Hà Nội

[3] Khoa Công nghệ Thông tin, Trường Đại học Sư phạm Hà Nội

Tác giả liên hệ: Phan Thanh Toàn, pttoan@hnue.edu.vn

Ngày nhận bài: 11/06/2019, ngày sửa chữa: 27/10/2019, ngày duyệt đăng: 27/10/2019

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

Biên tập lĩnh vực điều phối phản biện và quyết định nhận đăng: PGS.TS. Huỳnh Thị Thanh Bình

**Tóm tắt:** Điện toán đám mây là xu thế mới của công nghệ thông tin và truyền thông. Trong mô hình điện toán đám mây mọi khả năng liên quan đến công nghệ thông tin đều được cung cấp dưới dạng dịch vụ, cho phép người sử dụng truy cập đến các dịch vụ công nghệ (phần cứng và phần mềm) từ các nhà cung cấp dịch vụ. Điện toán đám mây là sự tập hợp của nhiều máy chủ vật lý và máy chủ ảo, được cấu hình để làm việc với nhau trên môi trường mạng Internet. Một trong số các vấn đề lớn nhất trong môi trường điện toán đám mây là bài toán lập lịch luồng công việc. Hiệu năng của các hệ thống điện toán đám mây phụ thuộc rất nhiều vào việc sắp xếp các tác vụ trong luồng thực thi trên các máy tính trong môi trường đám mây để hoàn thành luồng công việc một cách tối ưu. Trong bài báo này chúng tôi đề xuất một thuật toán lập lịch luồng công việc mới dựa trên chiến lược tối ưu bày đàn và tìm kiếm Tabu.

**Từ khóa:** *Lập lịch luồng công việc, tìm kiếm Tabu, tối ưu bày đàn, điện toán đám mây.*

**Title:** **New Effective Neighborhoods for Tabu Search and Particle Swarm Optimization to Schedule Workflow in Cloud Computing**

**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 machines that promises 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 cloud computing. Basically the issue relates to 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 particle swarm optimization and Tabu search.

**Keywords:** *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 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: dịch vụ về phần mềm (SaaS: Software as a Service), dịch vụ cơ sở hạ tầng (IaaS: Infrastructure as a Service), dịch vụ nền tảng hạ tầng (PaaS: Platform as a Service). Nhiều ứng dụng được mô hình hóa dưới dạng luồng công việc (workflow) bao gồm tập các tác vụ (task) và các phụ thuộc giữa chúng 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: thiên văn học, tin sinh, dự báo động đất, v.v. 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 (workflow scheduling) 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 giảm thiểu thời gian và chi phí thực hiện.

Luồng công việc là một chuỗi có thứ tự các tác vụ 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 (NP-Hard) [1]. 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 [2], CyberShake [3], Epigenomics [4], LIGO [5].

Phần tiếp theo của bài báo có cấu trúc như sau. Phần II giới thiệu một số công trình nghiên cứu liên quan đến bài toán lập lịch luồng công việc. Trong phần III chúng tôi trình bày mô hình lý thuyết 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 tìm kiếm Tabu nhằm thoát khỏi cực trị địa phương trong phương pháp PSO; và (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 [6] 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 trong tương lai.

## II. NHỮNG CÔNG TRÌNH LIÊN QUAN

Bài toán lập lịch luồng công việc 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. Có nhiều công trình nghiên cứu nhằm tìm ra lời giải gần đúng cho bài toán này. Trong [7], Dubey và các cộng sự 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.

Manasrah và Ba Ali [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.

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.

Rajavel và Mala [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, v.v. 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.

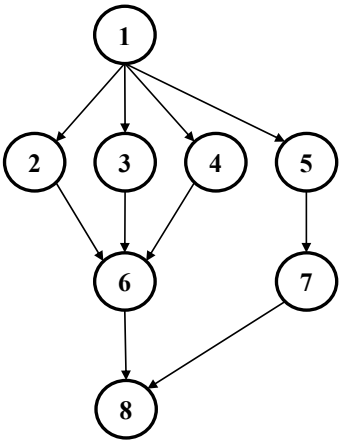
Pandey và cộng sự [12] đã đề xuất thuật toán lập lịch luồng công việc PSO Heuristic (PSO_H: Particle Swarm Optimization Heuristic) trong môi trường điện toán đám mây dựa trên chiến lược tối ưu bày đàn. Cụ thể, họ đã đề xuất một thuật toán lập lịch phân cấp và đưa vào các tham số dịch vụ khác nhau, chẳng 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. Cao và cộng sự 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

Trong bài báo này, chúng tôi sử dụng một số ký hiệu sau:

- $\mathbf{T} = \{T_1, T_2, \ldots, T_M\}$ là tập các tác vụ.
- $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$ là tập $N$ máy chủ trong môi trường điện toán đám mây; mỗi máy chủ $S_i$ có một năng lực tính toán xác định bởi $P(S_i)$, đơn vị tính là MI/s (million instructions/second).
- $W_i$ là khối lượng tính toán của tác vụ $T_i$, đơn vị tính là flop (floating point operations).
- Mỗi cặp máy chủ đều được kết nối với nhau bởi một đường truyền riêng có băng thông là $B(S_i, S_j)$, với $B(\cdot,\cdot)$ là hàm xác định băng thông $B : S \times S \rightarrow R^+$. Do tính chất của băng thông, hiển nhiên ta có:

$$\forall i, j : B(S_i, S_i) = \infty \text{ và } B(S_i, S_j) = B(S_j, S_i).$$




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

- Xuyên suốt bài báo, chúng tôi sử dụng hai tập chỉ số $\mathcal{M} = \{1, 2, \ldots, M\}$ và $\mathcal{N} = \{1, 2, \ldots, N\}$.

Đồ thị luồng công việc được biểu diễn bởi đồ thị có hướng không có chu trình $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, ví dụ như ở hình 1, trong đó $\mathbf{V}$ là tập đỉnh, mỗi đỉnh tương ứng với một tác vụ trong đồ thị luồng công việc và $\mathbf{E}$ là tập cạnh, biểu diễn mối quan hệ giữa các tác vụ. Nếu $e = (T_i, T_k)$ là một cạnh của đồ thị $\mathbf{G}$ thì $T_i$ 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 $tdata^k$.

**Định nghĩa 1 (Khái niệm lịch biểu):** Mỗi lịch biểu được biểu diễn bởi hàm

$$f : \mathbf{T} \to \mathbf{S},$$

trong đó $f(T_i) \in \mathbf{S}$ là máy chủ thực hiện tác vụ $T_i$.

Dưới đây là một số tham số của mô hình:

- $x_j^k$ là biến logic, với $x_j^k = 1$ nếu tác vụ $T_k$ được gán vào thực hiện trên máy chủ $S_j$, nếu không thì $x_j^k = 0$.
- $d_{i,j}^k$ là khối lượng dữ liệu được truyền từ máy chủ $S_i$ tới máy chủ $S_j$ 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 máy chủ $S_j$ cho tác vụ $T_k$ nếu $d_{i,j}^k > 0$ và $x_j^k = 1$:

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

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

$$extime_j^k = \frac{W_k}{P(S_j)}.$$

- $ET^k$ là thời gian thực hiện tác vụ $T_k$:

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

- $W_i / P(f(T_i))$ là thời gian tính toán tác vụ $T_i$ với $i \in \mathcal{M}$.
- $D_{ij} / B(f(T_i), f(T_j))$ là thời gian truyền dữ liệu giữa tác vụ $T_i$ và tác vụ con $T_j$.
- $CT$ là thời gian hoàn thành luồng công việc (Makespan):

$$CT = \max_{k \in \mathcal{M}} \{ET^k\}.$$

Tiếp theo là các điều kiện ràng buộc của mô hình:

a) $x_j^k \geq 0$ với mọi $k \in \mathcal{M}$ và $j \in \mathcal{N}$;
b) $d_{i,j}^k \geq 0$ với mọi $i, j \in \mathcal{N}$ và $k \in \mathcal{M}$;
c) $tdata_j^k \geq 0$ với mọi $k \in \mathcal{M}$;
d) $tftime_{i,j} \geq 0$ với mọi $i, j \in \mathcal{N}$;
e) $extime_j^k \geq 0$ với mọi $k \in \mathcal{M}$ và $j \in \mathcal{N}$;
f) $\sum_{j=1}^{N} x_j^k = 1$;
g) $\sum_{i=1}^{N} \sum_{j=1}^{N} x_j^k \times d_{i,j}^k = tdata^k$; và
h) $\sum_{k=1}^{M} \sum_{i=1}^{N} \sum_{j=1}^{N} x_j^k \times d_{i,j}^k = \sum_{k=1}^{M} tdata^k$.

**Định nghĩa 2 (Hàm mục tiêu):** Hàm mục tiêu được xác định bằng cách tối thiểu hóa Makespan như sau:

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

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. GIẢI PHÁP ĐỀ XUẤT

### 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 $\mathbf{x}_i^k$ (cho biết vị trí hiện tại) và vector dịch chuyển $\mathbf{v}_i^k$ (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 $\mathbf{x}_i = (\pi_{i1}, \pi_{i1}, \ldots, \pi_{iM})$ với $\pi_{ij} \in \mathbf{S}$ và $j \in \mathcal{M}$.

**Ví dụ 1:** Giả sử luồng công việc gồm tập tác vụ $\mathbf{T} = \{T_1, T_2, T_3, T_4, T_5, T_6, T_7\}$ và đám mây có tập máy chủ $\mathbf{S} = \{S_1, S_2, S_3\}$. Khi đó, cá thể $\mathbf{x}_i$ được biểu diễn bằng vector vị trí $(1; 2; 1; 3; 2; 3; 1)$ chính là phương án xếp lịch mà theo đó các tác vụ $T_1$, $T_3$, và $T_7$ được bố trí thực hiện bởi máy chủ $S_1$, tác vụ $T_2$ và $T_5$ được thực hiện trên $S_2$, còn tác vụ $T_4$ và $T_6$ được thực hiện bởi $S_3$ (Hình 2).

## 2. Phương pháp cập nhật vị trí cá thể

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

$$\mathbf{v}_i^{k+1} = \omega \mathbf{v}_i^k + c_1 rand_1 \times (pbest_i - \mathbf{x}_i^k) + c_2 rand_2 \times (gbest - \mathbf{x}_i^k), \quad (1)$$

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^k. \quad (2)$$

Hai công thức (1) và (2) 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 $\mathbf{v}_i^k$ là số thực do công thức (1) tính vector dịch chuyển có những tham số là số thực như $rand_1$, $rand_2$, $c_1$, và $c_2$. Nhưng vì tập máy chủ $\mathbf{S}$ là hữu hạn và đếm được nên các thành phần của vector vị trí $\mathbf{x}_i$ 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 $\mathbf{x}_i$ trong ví dụ 1 có các thành phần là $\mathbf{x}_i[1] = 1$, $\mathbf{x}_i[2] = 2$, $\mathbf{x}_i[3] = 1$, $\mathbf{x}_i[4] = 3$ và $\mathbf{x}_i[5] = 2$. Hậu quả là hai vế của phép gán (2) khác kiểu nhau, vế trái, $\mathbf{x}_i^{k+1}[t]$, thuộc kiểu số nguyên còn vế phải, $\mathbf{x}_i^k[t] + \mathbf{v}_i^k[t]$, 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 $\mathbf{x}_i^{k+1}t$ 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, $\mathbf{x}_i^k[t] + \mathbf{v}_i^k[t]$, sẽ được để nguyên không làm tròn, còn vế trái, $\mathbf{x}_i^{k+1}[t]$, 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. Như vậy,

$$\mathbf{x}_i^{k+1}[t] \leftarrow j$$

nếu, với mọi $S_r \in \mathbf{S}$ và $t \in \mathcal{M}$, ta có

$$|P(S_j) - (\mathbf{x}_i^k[t] + \mathbf{v}_i^k[t])| \leq |P(S_r) - (\mathbf{x}_i^k[t] + \mathbf{v}_i^k[t])|. \quad (3)$$

**Ví dụ 2:** Giả thiết tập máy chủ $\mathbf{S}$ trong ví dụ 1 có tốc độ tính toán được liệt kê trong bảng I. Giả sử ở bước thứ k+1 tổng $\mathbf{x}_i^k + \mathbf{v}_i^k = (4{,}4; 2{,}1; 6{,}7; 5{,}6; 10{,}2)$ thì vector vị trí $\mathbf{x}_i^{k+1}$ sẽ được gán bằng $(3; 1; 2; 2; 2)$. Nghĩa là cá thể đó tương ứng với phương án xếp lịch như mô tả trong hình 3. Thật vậy, thành phần thứ nhất của vector vị trí $\mathbf{x}_i^{k+1}[1]$ sẽ nhận giá trị 3 ($\mathbf{x}_i^{k+1}[1] \leftarrow 3$), nghĩa là tác vụ $T_1$ sẽ được gán cho máy chủ $S_3$ bởi vì

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ |
|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 2 | 3 | 1 |

Hình 2. Mô tả luồng công việc trong ví dụ 1.

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ |
|---|---|---|---|---|
| $S_3$ | $S_1$ | $S_2$ | $S_2$ | $S_2$ |

Hình 3. Mô tả phương án xếp lịch trong ví dụ 2.

Bảng I
TỐC ĐỘ TÍNH TOÁN CỦA CÁC MÁY CHỦ

| **Máy chủ $S_i$** | **Tốc độ xử lý $P(S_i)$** |
|---|---|
| $S_1$ | 3,1 |
| $S_2$ | 5,2 |
| $S_3$ | 4,1 |

$$|P(S_3) - 4{,}4| \leq |P(S_r) - 4{,}4| \ \forall S_r \in \mathbf{S}.$$

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 I, 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 là $T_2$, $T_3$, $T_4$ và $T_5$.

Vấn đề tương tự cũng xảy ra với phép trừ hai vector vị trí trong công thức (1): $(pbest_i - \mathbf{x}_i^k)$ và $(gbest - \mathbf{x}_i^k)$. 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 $\mathbf{x}_i^k = (1; 3; 2; 1; 2)$ thì

$$pbest_i - \mathbf{x}_i^k = (2-1; 4-3; 3-2; 3-1; 5-2) = (1; 1; 1; 2; 3).$$

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

$$pbest_i = (x_{i1}, x_{i2}, \ldots, x_{iM}) \text{ với } x_{ik} \in \mathbf{S} \ \forall k,$$
$$\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jM}) \text{ với } x_{jk} \in \mathbf{S} \ \forall k.$$

Khi đó kết quả phép trừ $pbest_i - \mathbf{x}_j$ được tính như sau:

$$pbest_i - \mathbf{x}_j = (y_1, y_2, \ldots, y_M),$$

**Thuật toán 1:** Toán tử lân cận BE($\mathbf{x}_i, p_1, p_2$)

**Dữ liệu vào:** Vector $(\pi_1, \ldots, \pi_{p1}, \ldots, \pi_{p2}, \ldots, \pi_n)$.
**Dữ liệu ra:** Vector $(\pi_1, \ldots, \pi_{p2}, \ldots, \pi_{p1}, \ldots, \pi_n)$.

$S_1 \leftarrow remove(\pi, p1)$;
**if** $P(\pi_{p2}) > P(\pi_{p1})$ **then**
  $insert(\mathbf{x}_i, \pi, p_2 - 1, S_1)$;
**else**
  $insert(\mathbf{x}_i, \pi, p_2, S_i)$;
**end**
**if** $P(\pi_{p2}) > P(\pi_{p1})$ **then**
  $S_2 \leftarrow remove(\mathbf{x}_i, \pi, p_2)$;
**else**
  $S_2 \leftarrow remove(\mathbf{x}_i, \pi, p_{2+1})$;
**end**
$insert(\mathbf{x}_i, \pi, p_1 - 1, S_2)$;

với các thành phần $y_k$ ($k \in \mathcal{M}$) là các số thực, được cho bởi

$$y_k = \left\{P(x_{ik}) + \frac{\sum_{q \in \mathbf{S}} B(x_{ik}, x_q)}{N-1}\right\} - \left\{P(x_{jk}) + \frac{\sum_{q \in \mathbf{S}} B(x_{jk}, x_q)}{N-1}\right\}.$$

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 $lbest_j = (2;1;2;1;1)$ và $\mathbf{x}_j = (3;2;1;2;1)$. Vậy ta có $lbest_j - \mathbf{x}_j = (y_1, y_2, y_3, y_4, y_5)$ 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 $y_2, \ldots, y_5$.

## 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 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.

***Thủ tục tìm kiếm lân cận:*** Tabu Search là phương pháp tìm kiếm lân cận được để xuất bởi Glove 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.

**Thuật toán 2:** Toán tử BS($\mathbf{x}_i$)

**Dữ liệu vào:** Vector $(\pi_1, \pi_2, \ldots, \pi_n)$.
**Dữ liệu ra:** Vector $(\pi''_1, \pi''_2, \ldots, \pi''_n)$.

$S_i \leftarrow \max\{P(S_1), P(S_2), \ldots, P(S_n)\}$;
**for** $(i = 1;\ i < M;\ i++)$ **do**
  $M_i = \text{makespan}((\pi_1, \ldots, \pi_{i-1}, S_i, \pi_{i+1}, \ldots, \pi_n))$;
**end**
$k \leftarrow \min\{M_i\}$;
$j \leftarrow remove(\mathbf{x}_i, \pi, k)$;
$insert(\mathbf{x}_i, \pi, k+1, j)$;

**Thuật toán 3:** Tabu_Search($\mathbf{x}_i$)

**Dữ liệu vào:** Vector vị trí $\mathbf{x}_i$.
**Dữ liệu ra:** Vector vị trí $\mathbf{x}_k$ có $f(\mathbf{x}_k) < f(\mathbf{x}_i)$.

Khởi tạo bước lặp $t \leftarrow 0$;
**while** điều kiện lặp **do**
  Khởi tạo ngẫu nhiên $r_1$ và $r_2$ trong đoạn $[1, M]$;
  $\mathbf{x}_i \leftarrow \text{BE}(\mathbf{x}_i, r_1, r_2)$;
  $x_k \leftarrow \text{BS}(\mathbf{x}_i)$;
  **if** $f(\mathbf{x}_k) < f(\mathbf{x}_i)$ **then**
    **return** $\mathbf{x}_k$;
  **else**
    **return** $\mathbf{x}_i$;
  **end**
  $t \leftarrow t + 1$;
**end**

Gọi $remove(\mathbf{x}_i, \pi, p)$ là toán tử loại bỏ phần tử $\pi_p$ trong vector vị trí của cá thể $\mathbf{x}_i = (\pi_1, \pi_2, \ldots, \pi_n)$. Sau khi thực hiện toán tử này ta có vector mới là $(\pi_1, \pi_2, \ldots, \pi_{p-1}, \pi_{p+1}, \ldots, \pi_n)$. Gọi $insert(\mathbf{x}_i, \pi, p, S_i)$ 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 $(\pi_1, \pi_2, \ldots, \pi_{p-1}, S_i, \pi_p, \ldots, \pi_n)$. Một toán tử lân cận Best Exchange (BE) hoán đổi vị trí của hai máy chủ tại vị trí $p_1$ và $p_2$ tùy theo năng lực của máy chủ được mô tả trong thuật toán 1.

Toán tử Best Swap (BS) tìm ra vị trí tốt nhất $p$ trong một cá thể $\mathbf{x}_i = (\pi_1, \pi_2, \ldots, \pi_n)$ 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. Toán tử BS được mô tả trong thuật toán 2.

## 4. Thuật toán đề xuất TSPSO

Tổng hợp những cải tiến nói trên, thuật toán đề xuất với tên gọi TSPSO được mô tả như trong thuật toán 4. 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 $\omega$) thì chứng tỏ quần thể đang hội tụ tại một cực trị địa phương. Khi đó thủ tục Tabu_Search (Thuật toán 3) 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.

Tiếp theo ta tìm hiểu 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 $O(M \times N)$, trong đó $M$ 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 $O(M \times SCT)$, trong đó $SCT$ là số cá thể trong quần thể; trong thực nghiệm chúng tôi sử dụng $SCT = 100$. 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) $\times O(M^2)$.

**Thuật toán 4:** Thuật toán TSPSO

**Dữ liệu vào:** Tập **T**, tập **S**, mảng $W[1 \times M]$, mảng $P[1 \times N]$, mảng $B[N \times N]$, mảng $D[M \times M]$, hằng số $K$, độ lệch $\omega$, và số cá thể $SCT$
**Dữ liệu ra:** Lời giải tốt nhất $gbest$

```
Khởi tạo ngẫu nhiên vector vị trí và vector dịch chuyển của cá thể i;
Khởi tạo bước lặp t ← 0;
while điều kiện lặp do
    for i = 1 to SCT do
        Tính vector vị trí x_i theo (3);
        M_i = Makespan của cá thể x_i;
    end
    for i = 1 to SCT do
        if pbest_i < M_i then
            pbest_i = M_i;
        end
    end
    gbest = min{M_i};
    for i = 1 to SCT do
        Cập nhật vector v_i^k theo (1) và (3);
        Tính x_i theo (2);
    end
    t ← t + 1;
    if sau K thế hệ mà độ lệch giữa các gbest không vượt quá ω then
        gbest = Tabu_Search(gbest);
    end
end
return gbest;
```

## V. KẾT QUẢ THỰC NGHIỆM

### 1. Phân nhóm dữ liệu thực nghiệm

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 như sau. Nhóm 1 là các luồng công việc ngẫu nhiên với sự khác nhau về hệ số $\alpha$ và nhóm 2 là các luồng công việc từ ứng dụng Montage. Hệ số $\alpha$ được xác định như sau:

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

Tham số $\alpha$ cho biết đồ thị **G** phân thành bao nhiêu cấp, mỗi cấp có nhiều hay ít tác vụ, nói cách khác $\alpha$ 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 $\alpha$ lần lượt thay đổi như trong hình 4.

### 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 của các máy chủ $P_i \in [1; 250]$ (MI/s); khối lượng dữ liệu giữa các tác vụ $D_{ij} \in [1; 10.000]$ (Mb); băng thông giữa các máy chủ $B(S_i, S_j) \in [10; 100]$ (Mb/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 $SCT = 25$; độ lệch $\epsilon = 0,21$; hệ số $\alpha \in [0,2; 0,7]$.

### 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 [6] 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 [6] được sử dụng để thực hiện các phương thức tối ưu bày đàn. Đối tượng so sánh là các thuật toán PSO_H [12], EGA [18] và 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 4 GB, và hệ điều hành Windows 7 Ultimate. Thực nghiệm được tiến hành một cách độc lập 30 lần trên mỗi bộ dữ liệu thực nghiệm.

### 4. Kết quả thực nghiệm

Hình 4 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

Bảng II
KẾT QUẢ THỰC HIỆN THUẬT TOÁN VỚI CÁC BỘ DỮ LIỆU NGẪU NHIÊN

| Ký 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 III
KẾT QUẢ THỰC HIỆN THUẬT TOÁN VỚI CÁC BỘ DỮ LIỆU TỪ ỨNG DỤNG MONTAGE

| Ký 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 |

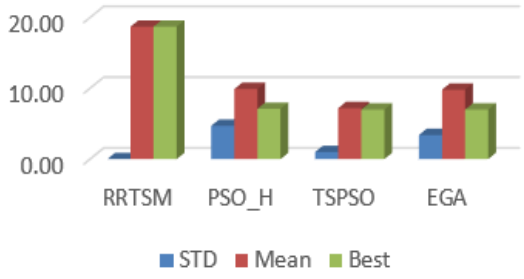
công việc từ ứng dụng Montage. Kết quả thực nghiệm được trình bày chi tiết trong các bảng II và III và hình 4. Kết quả so sánh giữa giá trị trung bình tính được bởi thuật toán TSPSO với các thuật toán đối sánh, trong hầu hết các trường hợp thuật toán TSPSO đều cho kết quả tốt hơn các thuật toán đối sánh, 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%.

Các hình này 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 *rand*om 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%. 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.

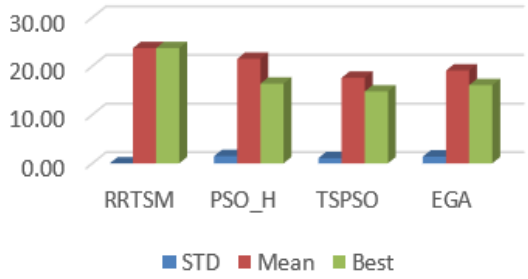
## VI. KẾT LUẬN

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:

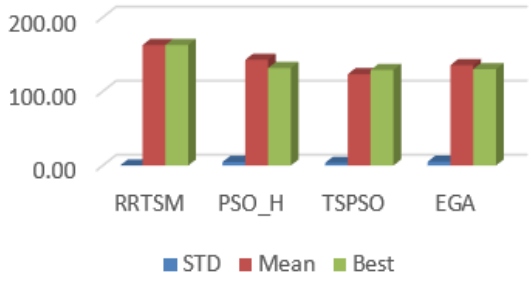
- Đề xuất một phương thức mới để cập nhật vị trí của cá thể bằng cách ánh xạ một giá trị thực tới máy chủ có tốc độ tính toán và băng thông gần với giá trị đó nhất.
- Đề xuất công thức tính vector dịch chuyển của cá thể thứ $i$ theo giá trị *gbest* và $pbest_i$.
- Đề xuất hai toán tử lân cận mới cho 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.




(a) T532




(b) T1051




(c) M2032

Hình 4. So sánh các thuật toán với các bộ dữ liệu khác nhau: T532, T1051 và M2032.

- 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] J. D. Ullman, "NP-complete scheduling problems," *Journal of Computer and System sciences*, vol. 10, no. 3, pp. 384–393, 1975.

[2] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman *et al.*, "Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand," in *Optimizing Scientific Return for Astronomy through Information Technologies*, vol. 5493, 2004, pp. 221–232.

[3] P. Maechling, E. Deelman, L. Zhao, R. Graves, G. Mehta, N. Gupta *et al.*, "SCEC CyberShake workflows—automating probabilistic seismic hazard analysis calculations," in *Workflows for e-Science*. Springer, 2007, pp. 143–163.

[4] USC Epigenome Center. [Online]. Available: http://epigenome.usc.edu.

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

[6] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in *IEEE International Conference on High Performance Computing & Simulation*, 2009, pp. 1–11.

[7] K. Dubey, M. Kumar, and S. Sharma, "Modified HEFT algorithm for task scheduling in cloud environment," *Procedia Computer Science*, vol. 125, pp. 725–732, 2018.

[8] A. M. Manasrah and H. Ba Ali, "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing," *Wireless Communications and Mobile Computing*, vol. 2018, no. 1934784, pp. 1–16, 2018.

[9] N. Grigoreva, "Branch and bound method for scheduling precedence constrained tasks on parallel identical processors," in *Proceedings of The World Congress on Engineering*, 2014, pp. 832–836.

[10] R. Rajavel and T. Mala, "Achieving service level agreement in cloud environment using job prioritization in hierarchical scheduling," in *International Conf. on Information Systems Design and Intelligent Applications*, 2012, pp. 547–554.

[11] S. Singh and M. Kalra, "Task scheduling optimization of independent tasks in cloud computing using enhanced genetic algorithm," *Int'l J. Application or Innovation in Engineering & Management*, vol. 3, no. 7, pp. 2319–4847, 2014.

[12] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 400–407.

[13] Q. Cao, Z.-B. Wei, and W.-M. Gong, "An optimized algorithm for task scheduling based on activity based costing in cloud computing," in *IEEE International Conference on Bioinformatics and Biomedical Engineering*, 2009, pp. 1–3.

[14] F. Glover, "Tabu search– part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[15] D. de Werra and A. Hertz, "Tabu search techniques: A tutorial and an application to neural networks," *Operations Research Spektrum*, vol. 11, no. 3, pp. 131–141, 1989.

[16] J. Vliet and F. Paganelli, *Programming Amazon EC2*. O'Reilly Media, Inc, 2011.

[17] Q. Jiang, Y. C. Lee, M. Arenaz, L. M. Leslie, and A. Y. Zomaya, "Optimizing scientific workflows in the cloud: A Montage example," in *IEEE International Conference on Utility and Cloud Computing*, 2014, pp. 517–522.

[18] R. Kaur and S. Kinger, "Enhanced genetic algorithm based task scheduling in cloud computing," *International Journal of Computer Applications*, vol. 101, no. 14, pp. 1–6, 2014.

[19] E. Upfal, *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

**Phan Thanh Toàn** sinh năm 1974 tại Thái Nguyên, tốt nghiệp Đại học và Thạc sĩ tại Trường Đại học Bách khoa Hà Nội và nhận bằng Tiến sĩ năm 2018 tại Học viện Khoa học Công nghệ Quân sự. Tác giả hiện đang công tác tại trường Đại học Sư phạm Hà Nội. Lĩnh vực nghiên cứu bao gồm phương pháp tiến hóa, tối ưu, xử lý song song và phân tán.

**Đặng Quốc Hữu** tốt nghiệp Đại học và Thạc sĩ tại Khoa Công nghệ Thông tin, Đại học Quốc gia Hà Nội năm 2006 và 2015. Tác giả hiện đang công tác tại Trường Đại học Thương mại, đồng thời là nghiên cứu sinh tại Viện Khoa học và Công nghệ Quân sự từ năm 2017. Lĩnh vực nghiên cứu bao gồm mạng máy tính và truyền thông, xử lý song song và phân tán.

**Nguyễn Thế Lộc** tốt nghiệp Đại học tại Khoa Toán–Tin, Trường Đại học Sư phạm Hà Nội năm 1993, Thạc sĩ Công nghệ Thông tin tại Trường Đại học Bách khoa Hà Nội, nhận bằng Tiến sĩ tại Viện Nghiên cứu Khoa học Công nghệ Nhật Bản (JAIST) năm 2007. Tác giả hiện đang công tác tại Trường Đại học Sư phạm Hà Nội. Lĩnh vực nghiên cứu bao gồm mạng máy tính và truyền thông, xử lý song song và phân tán.