JOURNAL OF SCIENCE OF HNUE DOI: 10.18173/2354-1059.2017-0011
Natural Sci. 2017, Vol. 62, No. 3, pp. 88-96
This paper is available online at http://stdb.hnue.edu.vn

ÁP DỤNG CHIẾN LƯỢC TIẾN HÓA VI PHÂN ĐỂ NÂNG CAO HIỆU SUẤT CỦA ĐIỆN TOÁN ĐÁM MÂY

Phan Thanh Toàn[1], Đặng Quốc Hữu[2], Nguyễn Thế Lộc[3] và Nguyễn Doãn Cường[4]
[1] *Khoa Sư phạm Kĩ thuật, Trường Đại học Sư Phạm Hà Nội*
[2]*Trung tâm Công nghệ Thông tin, Trường Đại học Thương Mại*
[3]*Khoa Công nghệ Thông tin, Trường Đại học Sư phạm Hà Nội*
[4]*Viện Công nghệ Thông tin, Viện Khoa học Công nghệ Quân Sự*

Tóm tắt. Trong thực tiễn và nghiên cứu khoa học có nhiều bài toán được biểu diễn dưới dạng mô hình luồng công việc như lập lịch cho dây chuyền sản xuất, lập lịch điều phối tài nguyên trong hệ điều hành, lập lịch thời khóa biểu. Lập lịch là hoạt động nhằm gán các tác vụ vào thực hiện trên các tài nguyên tính toán và thảo mãn các ràng buộc về thứ tự các tác vụ trong luồng công việc cũng như các giới hạn về tài nguyên. Đa số các bài toán thuộc họ lập lịch đã được chứng minh thuộc lớp NP-Khó [1], do vậy việc tìm ra các thuật toán lập lịch nhằm cực tiểu hóa thời gian hoàn thành luồng công việc là một lĩnh vực khó và đã thu hút được sự quan tâm của nhiều nhà khoa học. Bài báo này đề xuất một thuật toán lập lịch luồng công việc mới IODE nhằm cực tiểu hóa thời gian hoàn thành luồng công việc trong môi trường thực thi điện toán đám mây. Các thực nghiệm đã chỉ ra chất lượng lời giải của thuật toán IODE tốt hơn các thuật toán đối sánh là Random, PSO_H và EGA.

Từ khóa: Lập lịch luồng công việc, ứng dụng luồng công việc, điện toán đám mây, tiến hóa vi phân.

1. Mở đầu

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 như Montage [2], CyberShake [3], Epigenomics [4], LIGO [5],… 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 hoàn thành luồng công việc (makespan) 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 báo trình bày một số công trình liên quan đến bài toán lập lịch luồng công việc, mô tả bài toán và trình bày mô hình toán học sau đó phát biểu bài toán và chứng minh rằng nó thuộc lớp NP-Khó và giới thiệu thuật toán đề xuất IODE. Để kiểm chứng hiệu năng của thuật toán IODE bài báo cũng trình bày quá trình thực nghiệm trên một số luồng công việc khoa học mẫu trong môi trường đám mây thông qua công cụ mô phỏng CloudSim và gói thư viện Jswarm [6] đồng thời phân tích số liệu thu được, từ đó đưa ra những nhận xét so sánh.

Ngày nhận bài: 8/2/2017. Ngày nhận đăng: 23/3/2017.
Tác giả liên hệ: Phan Thanh Toàn, email: pttoan@hnue.edu.vn

2. Nội dung nghiên cứu

2.1. Những công trình nghiên cứu liên quan

Bài toán lập lịch luồng công việc đã được chứng minh là thuộc lớp NP-Khó [1] do vậy tìm ra một phương án lập lịch nhằm cực tiểu hóa thời gian hoàn thành luồng công việc thường rất phức tạp, đặc biệt với bài toán lập lịch luồng công việc cho các ứng dụng khoa học càng khó khăn hơn bởi các ứng dụng này thường phải xử lí một số lượng rất lớn các tác vụ cùng với khối lượng dữ liệu truyền qua lại giữa các tác vụ cũng rất lớn. Các thuật toán dựa trên hướng tiếp cận heuristic/metaheuristic đã được nhiều nhà khoa học nghiên cứu và đề xuất.

Trong công trình [7] các tác giả đã đề xuất thuật toán lập lịch dựa trên phương pháp di truyền là EGA (Enhanced Genetic Algorithm), nhóm tác giả đã sử dụng thuật toán Enhanced Max Min trong bước khởi tạo nhằm tìm ra các cá thể tốt cho quá trình tiến hóa, qua đó nâng cao chất lượng tìm kiếm của thuật toán.

Năm 2010, S. Pandey đã đề xuất thuật toán lập lịch luồng công việc PSO Heuristic (PSO_H) [8] trong môi trường điện toán đám mây nhằm cực tiểu hóa chi phí thực thi luồng công việc. Thuật toán PSO_H hoạt động theo phương pháp tối ưu bày đàn, tác giả đã tiến hành thực nghiệm thuật toán dựa trên số liệu dịch vụ được cung cấp bởi Amazon, và tác giả cũng đã chỉ ra chất lượng lời giải của thuật toán PSO_H tốt hơn so với các thuật toán Random và Round Robin.

Kết hợp giữa phương pháp tiến hóa vi phân và giải thuật di truyền tác giả M. Sridhar [4] đã đề xuất một thuật toán lai GAPSO nhằm cực tiểu hóa thời gian hoàn thành các tác vụ trong luồng công việc, trong công trình các tác giả đã đề xuất toán tử lựa chọn, đột biến và lai ghép mới nhằm cải thiện hiệu năng của thuật toán. Tác giả đã thực nghiệm thuật toán trên nhiều bộ dữ liệu khác nhau và chỉ ra chất lượng thuật toán GAPSO tốt hơn các thuật toán Max-Min và MCT (Minimum Execution Time).

R. Buyya đã trình bày tổng quan về các chức năng chính của phần mềm mô phỏng môi trường điện toán đám mây, CloudSim [9], đây là phần mềm mô phỏng được nhiều tác giả sử dụng để giả lập môi trường điện toán đám mây trong các công trình nghiên cứu.

L. Guo đã đề xuất một mô hình cho bài toán lập lịch luồng công việc và thuật toán lập lịch dựa trên phương pháp tối ưu bày đàn [10], trong công trình tác giả đã sử dụng luật SPV (Smallest Position Value) nhằm rời rạc hóa các giá trị thực của vector vị trí và vector chuyển động trong quá trình tiến hóa của thuật toán.

2.2. Mô hình toán học

2.2.1. Hệ thống tính toán

Giả thiết cho trước hệ thống tính toán bao gồm:

- Tập hợp N máy chủ trong môi trường điện toán đám mây $S = \{S_1, S_2, ... S_N\}$.
- Luồng công việc cần thực hiện biểu diễn bởi đồ thị có hướng, không có chu trình G=(V,E), mỗi đỉnh biểu thị một tác vụ, mỗi cạnh biểu diễn mối quan hệ cha-con giữa một cặp tác vụ.
- Tập các tác vụ $T=\{T_1, T_2, ... T_M\}$ với M là số lượng tác vụ.
- Khối lượng tính toán của tác vụ T_i kí hiệu là W_i , đo bằng đơn vị flop (floating point operations: phép tính trên số thực dấu phảy động).
- Tốc độ tính toán của máy tính, đo bằng đơn vị flop/s (số phép tính thực hiện được trên giây), kí hiệu P(), là hàm số được định nghĩa như sau: $P: S \rightarrow R^+$; $S_i \rightarrow P(S_i)$
- Mọi cặp máy chủ (S_i, S_k) bất kì đều có đường truyền để trao đổi dữ liệu với nhau $(1 \leq i, k \leq N)$.
- Băng thông của đường truyền, kí hiệu B(), là tốc độ truyền dữ liệu giữa các máy chủ, đo bằng đơn vị bit trên giây (bps), là hàm số được định nghĩa như sau: $B: S \times S \rightarrow R^+$; $(S_i, S_k) \rightarrow B(S_i, S_k)$

- Hàm băng thông B() tuân theo các ràng buộc sau:

  + $B(S_i,S_i) = \infty$: thời gian truyền từ một máy chủ tới chính nó bằng 0, nghĩa là nếu tác vụ cha và tác vụ con được bố trí trên cùng một máy chủ thì không mất thời gian để truyền dữ liệu giữa chúng.

  + $B(S_i,S_k) = B(S_k,S_i)$: kênh truyền hoạt động từ hai đầu với tốc độ tương đương nhau.

  Khối lượng dữ liệu cần truyền giữa hai tác vụ T_i và T_k, kí hiệu là D_{ik}, là các giá trị cho trước, $D_{ik} \neq 0$ khi và chỉ khi T_i là tác vụ cha của T_k, ngược lại $D_{ik} = 0$.

2.2.2. Khái niệm lịch biểu

Một phương án xếp lịch *F*, còn gọi là lịch biểu *F*, được xác định bởi hai hàm (t_s , *proc*) trong đó

- $t_s: T \rightarrow R^+$; $t_s(T_i)$ là thời điểm mà tác vụ $T_i \in T$ bắt đầu được thực hiện
- $proc: T \rightarrow S$; $proc(T_i)$ là máy tính được phân công thực hiện tác vụ $T_i \in T$

Từ các giả thiết trên suy ra:

Thời gian tính toán của tác vụ T_i là:

$$\frac{W_i}{P(proc(T_i))}, i = 1,2,...,M \quad (1)$$

Thời gian truyền dữ liệu giữa tác vụ T_i và tác vụ T_k là:

$$\frac{D_{ik}}{B(proc(T_i), proc(T_k))}, i,k = 1,2,...,M \quad (2)$$

Mục tiêu của bài toán

Makespan của lịch biểu *F* được biểu diễn theo công thức sau:

$$makespan(F) = max_{T_i \in T}\{t_f(T_i)\} - min_{T_i \in T}\{t_s(T_i)\} \quad (3)$$

với $t_f(T_i)$ là thời điểm kết thúc và $t_s(T_i)$ là thời điểm bắt đầu thực hiện của tác vụ T_i. Mục tiêu của bài toán - từ đây được kí hiệu là CLOS - là tìm lịch biểu *F* sao cho $makespan(F) \rightarrow min$.

Bổ đề: CLOS là bài toán thuộc lớp NP-Khó.

Chứng minh:

Sau đây chúng tôi sẽ chứng minh rằng bài toán CLOS thuộc lớp NP-Khó. Như Ullman [1] đã chỉ ra, cách làm thông dụng để chứng minh một bài toán A thuộc lớp NP là NP-Khó là tìm ra một bài toán B, trước đó đã được chứng minh là thuộc lớp NP-Khó, và có thể quy dẫn về bài toán A, kí hiệu là B∞A. Trong số các bài toán kinh điển thuộc họ bài toán Lập lịch, chúng tôi chọn bài toán SCHED, đã được O. Sinnen chứng minh là NP-Khó năm 2007 [11]. Chúng tôi sẽ quy dẫn bài toán SCHED về bài toán CLOS và qua đó chứng minh được CLOS cũng thuộc lớp NP- Khó.

Bài toán SCHED được O. Sinnen [11] phát biểu như sau:

Giả sử các tác vụ của công việc được biểu diễn bởi đồ thị G=(V,E) và chúng được thực hiện trên hệ thống tính toán P bao gồm những thành phần như dưới đây. Hãy tìm ra lịch biểu S sao cho thời gian thực hiện S trên P là nhỏ nhất.

- Một tập hợp gồm hữu hạn máy tính có năng lực tính toán ngang nhau và chỉ phục vụ bài toán SCHED mà không được dùng vào việc gì khác. Tại một thời điểm mỗi máy tính chỉ có thể xử lí tối đa một tác vụ. Nếu tác vụ cha và tác tác vụ con được thực hiện trên cùng một máy thì thời gian truyền dữ liệu giữa chúng bằng không.
- Các máy tính chỉ phụ trách tính toán chứ không phải điều khiển hệ thống mạng kết nối.

- Việc truyền thông giữa các máy tính có thể được thực hiện đồng thời. Mọi cặp hai máy tính bất kì đều được kết nối bởi một đường truyền có tốc độ như nhau.

Bài toán SCHED được công nhận là "*quy dẫn được*" về bài toán CLOS khi thỏa mãn điều kiện sau: nếu có thuật toán thời gian đa thức để giải bài toán CLOS thì cũng có thuật toán thời gian đa thức để giải bài toán SCHED.

Giả sử tìm được thuật toán A để xây dựng lịch biểu tối ưu S cho bài toán CLOS. Ta sẽ chứng minh rằng A cũng là thuật toán để giải bài toán SCHED, và như vậy điều kiện trên đã được thỏa mãn.

Rõ ràng bài toán SCHED là một trường hợp riêng của bài toán CLOS khi bổ sung thêm hai ràng buộc sau:

- Tốc độ tính toán của mọi máy tính là như nhau: $P(S_i) = P(S_j)$ $(\forall i,j = 1, \dots, N)$
- Mọi tuyến kết nối đều có tốc độ đường truyền như nhau: $B(S_i, S_k)=B(S_u, S_v)$ $(\forall\ i,k,u,v = 1, \dots, N)$

Như vậy, để tìm lịch biểu tối ưu cho bài toán SCHED đầu tiên ta thay đổi hệ thống tính toán của bài toán CLOS bằng cách gán một hằng số cho các giá trị $P(S_i)$ và một hằng số khác cho các giá trị $B(S_i, S_k)$, bằng cách đó ta đã thỏa mãn ràng buộc của bài toán SCHED. Áp dụng thuật toán A trên hệ thống tính toán vừa xây dựng, chúng ta thu được lịch biểu S_1. Theo giả thiết thuật toán A đảm bảo tìm được lịch biểu tối ưu trên hệ thống tính toán tổng quát nên khi áp dụng thuật toán A cho hệ thống tính toán của bài toán SCHED (là một trường hợp riêng của hệ thống tổng quát) thì lịch biểu tìm được S_1 là tối ưu. Như vậy thuật toán A là thuật toán để giải bài toán SCHED, suy ra bài toán CLOS cũng thuộc lớp NP-đầy đủ.

2.3. Giải pháp đề xuất

2.3.1. Khái niệm đối xứng

Định nghĩa 1: giả sử $x \in [a,b]$; $x \in R$, khi đó phần tử đối xứng của x kí hiệu là $\bar{x}$ và được tính như sau:

$$\bar{x} = a + b - x \tag{4}$$

Định nghĩa 2: gọi $P(x_1, x_2,..,x_D)$ là một véc tơ D-chiều, với $x_i \in [a_i, b_i]$; i=1,2,…,D. Khi đó véc tơ đối xứng của P kí hiệu là $\bar{P} = (\overline{x_1}, \overline{x_2}, \dots, \overline{x_D})$ và được tính như sau:

$$\overline{x_i} = a_i + b_i - x_i \tag{5}$$

2.3.2. Biểu diễn cá thể

Mỗi cá thể p trong quần thể được biểu diễn bởi một vector $p = (p_1, p_2, \dots, p_M)$; $p_i \in \{S_1, S_2, \dots, S_N\}$.

Ví dụ: xét luồng công việc với 5 tác vụ $T=\{T_1, T_2, \dots, T_5\}$, tập máy chủ gồm 3 máy $S=\{S_1, S_2, S_3\}$. Khi đó cá thể $x_i = (1,2,1,3,2)$ được biểu diễn như sau:

T_1	T_2	T_3	T_4	T_5
S_1	S_2	S_1	S_3	S_2

2.3.3. Phương pháp tính đối xứng cho cá thể

Phương pháp ODE [12] yêu cầu phải tìm cá thể đối xứng cho mỗi cá thể, trong bài báo này chúng tôi đề xuất phương pháp tìm cá thể đối xứng như sau:

Gọi $a = \text{Max}\{P(S_i)\}$ và $b = \text{Min}\{P(S_i)\}$; $\forall i=1,2,..,N$;

$P(S_i)$ là tốc độ tính toán của máy chủ S_i.

Giả sử ta có cá thể $x_i = (S_{i\pi(1)}, S_{i\pi(2)}, \dots, S_{i\pi(M)})$; $S_{i\pi(j)} \in S$, $\forall j=1,2,..,M$; cá thể đối xứng của x_i kí hiệu là $\overline{x_i}$ và được tính theo công thức (6)

$$\overline{x_\iota} = \left(\overline{S_{\iota\pi(1)}}, \overline{S_{\iota\pi(2)}}, \dots, \overline{S_{\iota\pi(M)}}\right); \ \overline{S_{\iota\pi(J)}} = a + b - S_{i\pi(j)}; \ \forall j = 1,2,..,M \quad (6)$$

Sau đó sẽ gán các giá trị tương ứng với mỗi vị trí j trong véc tơ $\overline{x_\iota}$ bởi số hiệu máy chủ có tốc độ tính toán gần giá trị $\overline{S_{\iota\pi(J)}}$ nhất theo công thức (7):

$$\overline{x_{\iota J}} \leftarrow k \ nếu \left|P(S_k) - \overline{S_{\iota\pi(J)}}\right| \leq \left|P(S_r) - S_{i\pi(j)}\right| \forall S_r \quad (7)$$

2.3.4. Phương pháp bánh xe quay vòng dựa trên hạng cá thể

Bánh xe quay vòng dựa trên hạng cá thể [13] là một phương pháp lựa chọn cá thể cho các thế hệ kế tiếp. Trong phương pháp này mỗi cá thể được xếp hạng theo một hàm xác định, sau đó sẽ tính sắc xuất lựa chọn các cá thể theo hạng của chúng. Trong bài báo này chúng tôi đề xuất hàm tính hạng cho các cá thể như sau:

$$rank(pos) = 2 - SP + \left(2 \times (SP - 1) \times \frac{pos-1}{PopSize-1}\right) \quad (8)$$

trong đó: $1.0 \leq SP \leq 2.0$; pos: vị trí cá thể cần tính hạng.

2.3.5. Thuật toán IODE

Kết hợp các nội dung trên chúng tôi đề xuất thuật toán lập lịch luồng công việc IODE trong môi trường điện toán đám mây dựa trên thông tin đối xứng, thuật toán hoạt động theo phương pháp tiến hóa vi phân kết hợp với thông tin đối xứng của các cá thể trong quần thể. Chi tiết thuật toán như sau:

Algorithm **IODE** ( )

Input: T, S, khối lượng công việc cần thực hiện W[1×*M*], P[1×*N*], B[*N*×*N*], D[*M*×*M*], hằng số K, độ lệch chấp nhận được ε, số lượng cá thể *NoP*

Output: lời giải tốt nhất *gbest*

1. Khởi tạo quần thể P gồm PopSize cá thể một cách ngẫu nhiên
2. OP ← OP_Algorithm ; tính quần thể đối xứng của quần thể hiện tại
3. Chọn PopSize cá thể tốt nhất từ P ∪ OP
4. while(Điều kiện lặp)do
5. for i=1 to PopSize do
6. Lựa chọn cá thể p_1 theo thuật toán RBRWS
7. Lựa chọn cá thể p_2 theo thuật toán RBRWS
8. F ← 0.5
9. K ← 0.5
10. $v_i = x_i + K\times(x_{best} - x_i)+F\times(x_{r1} - x_{r2})$
11. Gán số hiệu máy chủ cho mỗi vị trí j của véc tơ v_i theo (7)
12. $rand_{i,j}$ ← Random(0,1)
13. I_{rand} ← random(1,M)
14. $u_{i,j} = \begin{cases} v_{i,j} \ nếu \ rand_{i,j} \leq CR \ \ hoặc \ i = I_{rand} \\ x_{i,j} \ nếu \ rand_{i,j} \geq CR \ hoặc \ i \neq I_{rand} \end{cases}$
15. if (makespan(u_i) < makespan(x_i))
16. $x_i \leftarrow u_i$
17. end if
18. end for
19. rand ← Random(0,1)
20. if(rand < J_r)
21. OP ← OP_Algorithm
22. Lựa chọn PopSize cá thể tốt nhất tử P ∪ OP

23. end if
24. End while
25. Return *gbest;*

Bước khởi tạo; thuật toán sinh ra quần thể P gồm Popsize cá thể một cách ngẫu nhiên, sau đó tìm quần thể đối xứng của P và chọn lọc Popsize cá thể tốt nhất từ quần thể ban đầu và quần thể đối xứng.

Véc tơ đột biến của mỗi cá thể i được thực hiện theo chiến lược current to best /1; với công thức:

$$v_i = x_i + K\times(x_{best} - x_i)+F\times(x_{r1} - x_{r2});$$

Trong đó: $x_{r1} - x_{r2}$ là hai cá thể được chọn theo phương pháp bánh xe quay vòng dựa trên hạng các cá thể, x_{best} là cá thể tốt nhất hiện tại.

Sau bước đột biến toán tử trao đổi chéo được áp dụng cho mỗi cá thể x_i để sinh ra cá thể mới u_i

$$u_{i,j} = \begin{cases} v_{i,j}\ nếu\ rand_{i,j} \le CR\ \ hoặc\ i = I_{rand} \\ x_{i,j}\ nếu\ rand_{i,j} \ge CR\ hoặc\ i \ne I_{rand} \end{cases}$$

Trong đó: $rand_{i,j} \in [0,1]$, Irand $\in [1,M]$, CR $\in [0,1]$.

Toán tử lựa chọn được áp dụng để quyết định cá thể nào được sống sót cho thế hệ kế tiếp

$$x_i = \begin{Bmatrix} u_i, if\ makespan(u_i) < makespan(x_i) \\ x_i, otherwise \end{Bmatrix}$$

Sau quá trình trao đổi chéo và lựa chọn, thuật toán sẽ tính quần thể đối xứng OP của P theo công thức (6), (7). Cuối cùng sẽ lựa chọn ra PopSize cá thể tốt nhất từ P $\cup$ OP.

Trong quá trình tiến hóa thuật toán sẽ tính toán và lưu lại giá trị *gbest* là giá trị tốt nhất theo hàm mục tiêu (Makespan).

2.3.6. Kết quả thực nghiệm

Để kiểm chứng hiệu năng của thuật toán đề xuất IODE chúng tôi đã tiến hành cài đặt thuật toán bằng ngôn ngữ lập trình Java, môi trường điện toán đám mây được giả lập bằng phần mềm mô phỏng CloudSim [6]. Đối tượng so sánh là thuật toán tiến hóa mạnh như PSO Heuristic [8], thuật toán Random [15], và EGA [7]. Các chương trình mô phỏng được chạy trên bộ vi xử lí Intel Core i5 2.2 GHz, RAM 4GB, hệ điều hành Windows 7 Ultimate.

Dữ liệu thực nghiệm

Luồng công việc trong dữ liệu thực nghiệm được lấy từ các luồng công việc ngẫu nhiên với sự đa dạng về cấu trúc của luồng công việc và hai ứng dụng thực tiễn là Montage [2] và Epigenomics [4]. Các tham số về tốc độ tính toán và băng thông giữa các máy chủ được lấy từ nhà cung cấp dịch vụ điện toán đám mây Amazon [15].

Tham số cấu hình

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 vi sai F=0.5; K=0.5; hệ số trao đổi chéo CR=0.9; J_r=0.3; *PopSize*=50

Quá trình tiến hành thực nghiệm

Để đánh giá hiệu năng của thuật toán đề xuất IODE chúng tôi đã tiến hành thực nghiệm và so sánh kết quả của IODE với các thuật toán tiến hóa mạnh hiện nay như PSO_H, Random và EGA. Dữ liệu thực nghiệm được lấy từ ứng dụng Montage và Epigenomics, các tham số băng thông, tốc độ tính toán của máy chủ được thiết lập thống nhất cho tất cả các thực nghiệm. Với mỗi bộ dữ liệu chúng tôi tiến hành chạy chương trình 30 lần độc lập. Kết quả thực nghiệm được trình bày chi tiết

trong Bảng 1, 2 và các Hình 1-4. Kết quả thực nghiệm đã chỉ ra chất lượng lời giải của thuật toán IODE luôn tốt hơn các thuật toán Random, PSO_H và EGA ở cả 3 tham số là độ lệch chuẩn, giá trị trung bình và giá trị tốt nhất. Giá trị trung bình tìm được bởi thuật toán IODE nhỏ hơn giá trị trung bình tìm được bởi thuật toán PSO_H từ 8% - 29% và nhỏ hơn giá trị trung bình tìm được bởi thuật toán EGA từ 6% - 25%.

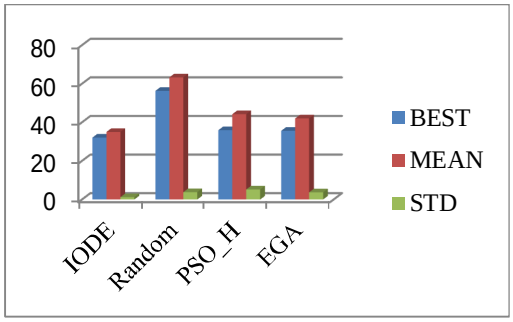
Các Hình 1-4 cũng so sánh giữa giá trị tốt nhất tìm được bởi thuật toán IODE với các thuật toán đối sánh, qua đó ta thấy giá trị tốt nhất tìm được bởi IODE tốt hơn giá trị tốt nhất tìm được bởi PSO_H từ 3% - 19%, giá trị tốt nhất tìm được bởi thuật toán IODE nhỏ hơn giá trị tốt nhất tìm được bởi EGA từ 1% - 18% và giá trị tốt nhất tìm được bởi IODE tốt hơn giá trị tốt nhất tìm được bởi Random từ 20% - 40%. Các Hình 1-4 chỉ ra kết quả so sánh giữa độ lệch chuẩn tìm được bởi thuật toán IODE với các thuật toán đối sánh, giá trị độ lệch chuẩn của IODE đều nhỏ hơn so với độ lệch chuẩn tìm được bởi các thuật toán Random, PSO_H và EGA, điều đó chứng tỏ thuật toán IODE có chất lượng lời giải tốt hơn các thuật toán đối sánh và độ ổn định trong các lần chạy cũng tốt hơn.

Bảng 1. Kết quả thực nghiệm với luồng công việc Montage

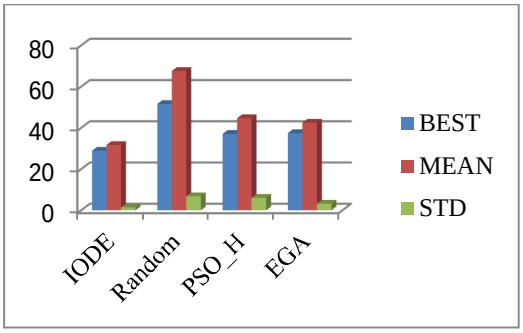
M	**N**	**IODE**			**PSO_H**			**RANDOM**			**EGA**		
		Best	***Mean***	***STD***	***Best***	***Mean***	***STD***	***Best***	***Mean***	***STD***	***Best***	***Mean***	***STD***
20	8	32.1	35.0	1.2	35.90	44.2	5.2	56.3	63.3	3.8	35.6	42.0	3.7
20	8	28.9	31.7	1.4	37.1	44.7	6.1	51.6	67.6	6.8	37.5	42.5	3.2
25	8	219.0	219.7	1.5	225.0	239.0	8.3	238.0	304.9	33.0	222.4	235.5	4.7
50	8	82.4	87.3	3.1	95.0	108.0	6.3	110.5	196.8	32.8	86.4	103.5	3.3

Bảng 2. Kết quả thực nghiệm với luồng công việc Epigenomics

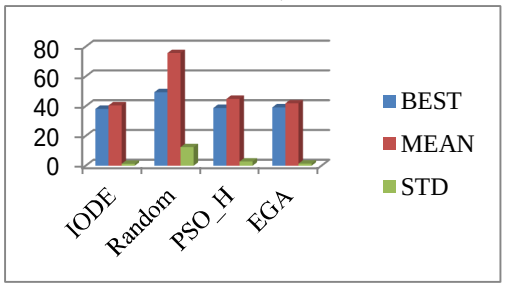
M	**N**	**IODE**			**PSO_H**			**RANDOM**			**EGA**		
		Best	***Mean***	***STD***	***Best***	***Mean***	***STD***	***Best***	***Mean***	***STD***	***Best***	***Mean***	***STD***
100	8	36.6	42.7	1.7	36.6	45.7	4.2	53.4	76.7	11.7	37.8	44.8	4.1
100	8	13.8	17.8	1.2	18.2	21.2	2.6	22.8	50.9	14.0	18.8	21.6	3.8
100	10	38.2	40.5	1.3	38.8	44.9	2.8	49.4	75.7	12.5	39.2	41.9	1.4
100	20	24.5	28.2	1.9	34.1	40.5	3.4	42.9	73.1	14.9	34.4	37.7	1.9
100	20	13.1	16.9	1.2	17.8	26.4	3.5	33.0	63.6	15.8	21.0	26.3	2.6



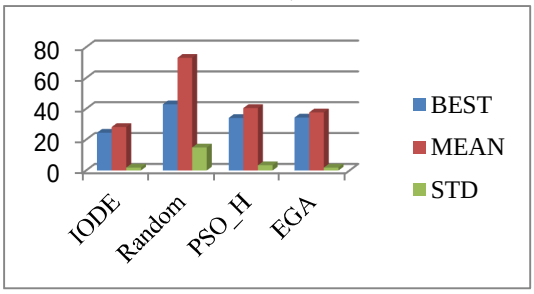
Hình 1. So sánh các thuật toán với M = 20, N = 8



Hình 2. So sánh các thuật toán với M = 25, N = 8



Hình 3. So sánh các thuật toán với M = 100, N = 10



Hình 4. So sánh các thuật toán với M = 100, N = 20

3. Kết luận

Lập lịch luồng công việc là một vấn đề quan trọng có nhiều ứng dụng trong thực tiễn và nghiên cứu khoa học, đặc biệt trong các môi trường phân tán như tính toán lưới hay điện toán đám mây thì hiệu năng làm việc của hệ thống sẽ phụ thuộc nhiều vào hiệu quả của các thuật toán lập lịch điều phối tài nguyên. Bài báo đã trình bày những kết quả chính sau đây:

- Đề xuất một mô hình lí thuyết cho bài toán lập lịch luồng công việc trong môi trường điện toán đám mây và chứng minh bài toán đề xuất thuộc lớp NP-Khó.

- Kết hợp giữa phương pháp tiến hóa vi phân, phương pháp đối xứng và phương pháp bánh xe quay vòng dựa trên hạng cá thể chúng tôi đã đề xuất thuật toán lập lịch luồng công việc mới IODE, các kết quả kiểm chứng đã chỉ ra chất lượng của thuật toán đề xuất tốt hơn các thuật toán đối sánh là Random, PSO_H và EGA.

TÀI LIỆU THAM KHẢO

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

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

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

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

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

[6] R. N. Calheiros, R. Ranjan, A. Beloglazov, Cesar A. F. De Rose, and R. Buyya, 2011. *CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms.* Software: Practice and Experience, volume 41, Number 1, Pages: 23-50, Wiley Press, USA.

[7] S. Singh, M.Kalra, 2014. *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.

[8] S. Pandey, L. Wu, S. M. Guru, R. Buyya, 2010. *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.

[9] Dr. M. Sridhar, 2015. *Hybrid Genetic Swarm Scheduling for Cloud Computing.* Global Journal of Computer Science and Technology, v.15, issue 3.

[10] L. Guo, 2012. *Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm.* Journal of networks, v.7, No.3, pp.547-552.

[11] O. Sinnen, 2007. *Task Scheduling for Parallel Systems.* John Wiley & Sons, pp. 83.

[12] R. Storn and K. Price, 1997. *Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces.* Journal of Global Optimization, pp. 341-359.

[13] N.M.Razali, J.Geraghty, 2011. *Genetic Algorithm Performance with Different Selection Strategies in Solving TSP*. Proceedings of the World Congress on Engineering.

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

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

ABSTRACT

Differential evolution strategy for improving the performance of the Cloud Computing

Phan Thanh Toan[1], Dang Quoc Huu[2], Nguyen The Loc[3] and Nguyen Doan Cuong[4]

[1]*Faculty of Psychology & Pedagogy, Hanoi National University of Education*

[2]*Centre of Information Technology, Vietnam University of Commerce*

[3]*Faculty of Information Technology, Hanoi National University of Education*

[4]*Information Technology Institute, Military Institute of Science and Technology*

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. Proper scheduling can have significant impact on the performance of the Cloud. Workflow scheduling, the most important affair which the cloud controllers deal with, is the factor which determines the performance of the cloud's services. The general scheduling problem was proven to be NP-hard long time ago [1]. In this paper we build a framework for workflow scheduling in Cloud environment. Based on the framework we propose a new heuristic algorithm for the mentioned problem so that execution time of the workflow (makespan) is minimized.

Keywords: Workflow scheduling, workflow applications, cloud computing, differential Evolution.