Tạp chí Tin học và Điều khiển học, T.25, S.1 (2009), 79–87

TỐI ƯU ĐA MỤC TIÊU TRONG VIỆC LẬP LỊCH CHO HỆ THỐNG TÍNH TOÁN LƯỚI

TRỊNH THỊ THÚY GIANG[1], LÊ TRỌNG VĨNH[1], HOÀNG CHÍ THÀNH[1], NGUYỄN THANH THỦY[2]

[1]*Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội*
[2]*Đại học Bách khoa Hà Nội*

Abstract. In this paper, we study the problem of scheduling in grid computing system and propose a genetic algorithm for it. We propose a new fitness function which considers simultaneously two critical factors in the probem of scheduling, that are: makespan factor and flowtime factor. In addition, the new fitness function considers not only the load of systems and jobs but also the cost of data transfer for scheduling. Experimental resutls show that our algorithm can archive a scheduling method that is better than the simulated annealing-based algorithm do.

Tóm tắt. Trong bài báo này chúng tôi nghiên cứu bài toán lập lịch trong các hệ thống tính toán lưới và đưa ra một thuật toán di truyền để giải quyết nó. Chúng tôi đưa ra một hàm đo độ thích nghi (fitness function) mới, nhằm đồng thời tối ưu hóa hai yếu tố cơ bản của bài toán lập lịch trong lưới tính toán, đó là: khoảng thời gian để hoàn thành toàn bộ các công việc (makespan) và tổng thời gian thực hiện của các công việc (flowtime). Hơn nữa, ngoài việc quan tâm đến tải trọng tính toán của các tài nguyên và công việc, bài báo này cũng quan tâm đến giá của việc sử dụng các tài nguyên và giá của việc truyền dữ liệu đến các tài nguyên. Các kết quả thực nghiệm chỉ ra rằng thuật toán của chúng tôi đề xuất có phương án lập lịch tốt hơn thuật toán lập lịch truyền thống dựa trên thuật toán Simulated Annealing.

1. GIỚI THIỆU

Lưới tính toán (computational grid) - hay còn gọi là hệ thống tính toán lưới (grid computing system) - là một tập hợp rộng lớn và không đồng nhất của các hệ thống tự trị (autonomous systems) phân tán về địa lý được kết nối với nhau bởi liên mạng máy tính [1]. Lưới tính toán được sử dụng để giải quyết các vấn đề phức tạp thuộc nhiều lĩnh vực khác nhau như: tối ưu, mô phỏng, khám phá dược liệu, sinh học,... Trong một lưới tính toán các hệ thống tự trị phải chia sẻ công việc (job sharing) với nhau. Việc chia sẻ các công việc cho các hệ thống tự trị sao cho tài nguyên hệ thống được sử dụng hiệu quả và các công việc thực hiện được một cách tối ưu được gọi là lập lịch trong hệ thống tính toán lưới [1]. Bài toán lập lịch là một trong các tác vụ khó khăn chính của hệ thống tính toán lưới, vì vậy nó đã được rất nhiều nhà khoa học quan tâm. Không giống như các bài toán lập lịch trong hệ thống phân tán truyền thống, trong lưới tính toán việc lập lịch phức tạp hơn nhiều do các đặc trưng như: tính động của hệ thống, sự không đồng nhất về tài nguyên và công việc ... cần được quan

tâm khi xử lý. Bài toán lập lịch được xem xét trong cả hai trường hợp: tĩnh (static) và động (dynamic). Trong bài báo này chúng tôi chỉ xem xét bài toán ở dạng tĩnh.

Bài toán lập lịch đã được chứng minh là NP đầy đủ [1]. Do vậy, việc sử dụng các thuật toán heuristics là cách tiếp cận thực tế (de facto) để giải quyết nó. Trong những năm qua, nhiều tác giả đã sử dụng các thuật toán heuristics khác nhau để giải quyết bài toán lập lịch như: Ritchie [2] sử dụng thuật toán tìm kiếm cục bộ (local search), Yarkhan [3] sử dụng thuật toán Simulated Annealing, Abraham [4] sử dụng thuật toán Tabu Search, Marino [5] sử dụng thuật toán di truyền (genetic algorithms), Ritchie [6] sử dụng thuật toán lai giữa Ant Colony Optimization và Tabu Search ... Tuy nhiên, các cách tiếp cận này chỉ quan tâm đơn lẻ đến một trong hai yếu tố quan trọng nhất của công việc lập lịch trong một lưới tính toán đó là: cực tiểu makespan (khoảng thời gian hoàn thành toàn bộ các công việc) hoặc flowtime (tổng thời gian thực hiện của các công việc). Mặt khác, để đơn giản trong các mô hình mô phỏng các tác giả là thường chỉ quan tâm một cách tượng trưng đến các tài nguyên của hệ thống - khả năng tính toán của các hệ thống tự trị - cũng như yêu cầu tính toán của các công việc. Điều này dẫn đến các thuật toán được trình bày khó áp dụng trong các hệ thống tính toán lưới thực tế.

Thuật toán di truyền (Genetic Algorithm - GA) [7] ngoài ích lợi giảm không gian tìm kiếm, hội tụ về lời giải toàn cục, nó còn cho phép tối ưu hóa đa mục tiêu (multi-objectives). Các tác giả trong [9] đã sử dụng thuật toán di truyền đề đồng thời tối ưu hai yếu tố flowtime và makespan trong việc lập lịch cho lưới tính toán. Tuy nhiên, nó vẫn còn điểm yếu như đã nêu trên đó là các vấn đề về giá của tài nguyên và chi phí truyền tải dữ liệu chưa được quan tâm đến. Trong bài báo này, chúng tôi cũng sử dụng mô hình mô phỏng và thuật toán di truyền, tuy nhiên giá của các tài nguyên và phí truyền tải dữ liệu sẽ được quan tâm. Do vậy, thuật toán được đưa ra có nhiều khả năng ứng dụng trong các lưới tính toán thực hơn các thuật toán đã được trình bày.

Đóng góp chính của bài báo có thể tóm tắt như sau:

1) Phát biểu bài toán lập lịch tổng quát trong hệ thống lưới tính toán dưới dạng hình thức (mô hình toán học).

2) Giải quyết bài toán dựa vào thuật giải di truyền bằng việc đưa ra hàm đo độ thích nghi (fitness function) mới cho thuật toán di truyền không chỉ để quan tâm đồng thời đến cả hai yếu tố quan trọng của bài toán lập lịch trong lưới tính toán là makespan và flowtime mà còn quan tâm đến các yếu tố phụ khác như là giá sử dụng tài nguyên và truyền tải dữ liệu. Các tham số trong hàm đo độ thích nghi là các tham số thiết kế nên đối với các ứng dụng trong thực tế, chúng ta có thể tùy biến phù hợp với nhiều trường hợp khi các mục tiêu cần có sự ưu tiên khác nhau.

3) Một nghiên cứu sâu rộng bằng mô phỏng để so sánh các phương án lập lịch đưa ra bởi thuật toán di truyền và thuật toán Simulated Annealing truyền thống. Điều này được kiểm chứng thông qua các mô phỏng.

Phần còn lại của bài báo được tổ chức như sau. Mục 2 dành để mô hình hoá bài toán lập lịch trong lưới tính toán. Mục 3 đưa ra thuật toán lập lịch dựa trên thuật toán di truyền cho lưới tính toán. Mục 4 trình bày các kết quả thực nghiệm. Chúng tôi đưa ra các kết luận và

đề xuất các hướng phát triển trong Mục 5.

2. LẬP LỊCH TRONG CÁC HỆ THỐNG TÍNH TOÁN LƯỚI

Lập lịch trong hệ thống tính toán lưới được biết đến với nhiều cách thức khác nhau do độ phức tạp của hệ thống và tính phân tán của các ứng dụng. Tuy nhiên, trong bài báo này, chúng tôi chỉ quan tâm đến bài toán lập lịch tĩnh, trong đó các ứng dụng không phụ thuộc lẫn nhau. Mặc dù vậy, những vấn đề về trao đổi dữ liệu, tính kinh tế (giá) của việc sử dụng các tài nguyên cũng sẽ được quan tâm. Kiểu lập lịch này xuất hiện trong nhiều ứng dụng thực tế. Trong kiểu lập lịch này mỗi ứng dụng có thể được chia thành nhiều công việc độc lập, các công việc này được gửi (riêng rẽ) đến các hệ thống tự trị trong lưới để thực hiện và tạo ra các kết quả (bộ phận), kết quả cuối cùng là sự kết hợp của các kết quả bộ phận. Chúng tôi cũng chỉ quan tâm đến kịch bản, trong đó các ứng dụng được gửi tới lưới là độc lập và không có ưu tiên.

Để dễ dàng cho việc phát biểu bài toán và không mất tính tổng quát, ta giả sử rằng đã biết trước các thông tin sau: khả năng tính toán (cũng được biết là tải tính toán) của mỗi tài nguyên (hệ thống tự trị) trong lưới; ước lượng về tải tính toán (computational load) của mỗi công việc, tải tính toán đã sử dụng của mỗi tài nguyên. Một mô hình như vậy được gọi là mô hình ETC (Expected Time to Compute) [8]. Trong mô hình này, một ma trận ETC sẽ được xây dựng, trong đó mỗi phần tử ETC$[t][m]$ là thời gian mong muốn (expected time) để tính toán công việc t trong tài nguyên m. Giá trị này có thể được tính toán bằng cách chia khả năng tính toán của tài nguyên m cho tải tính toán của công việc t. Với mô hình ETC, chúng ta có thể phát biểu bài toán lập lịch trong lưới tính toán như sau. Giả sử có:
1) n công việc độc lập cần phải lập lịch với tải tính toán $\{J_1, J_2, ..., J_n\}$ thao tác trên tập dữ liệu kích thước $\{D_1, D_2, ..., D_n\}$. Chúng ta cũng giả sử là mỗi công việc sẽ được xử lý toàn bộ trong một tài nguyên duy nhất.
2) m tài nguyên (cũng được gọi là hệ thống tự trị - autonomous system) với khả năng tính toán $\{M_1, M_2, ..., M_m\}$ có giá trị kinh tế tương ứng là $\{v_1, v_2, ..., v_m\}$ cho mỗi một đơn vị tải tính toán và chi phí truyền thông trên một đơn vị dữ liệu gửi đến các tài nguyên này là $\{c_1, c_2, ..., c_m\}$.
3) $\{T_1, T_2, ..., T_m\}$ là thời gian mà tài nguyên i sẽ kết thúc công việc trước đó (chỉ ra tải trọng đã sử dụng trước đó).

Bài toán đặt ra là: Lập lịch để hệ thống tính toán thực hiện tất cả các công việc sao cho:
1) Flowtime là cực tiểu, nghĩa là:

$$T^f = \min_{M_{j(i)} \in \{M_1, M_2, ..., M_m\}} \sum_{i=1}^{n} \frac{M_{j(i)}}{J_i}. \tag{1}$$

2) Makespan cực tiểu, nghĩa là:

$$T^m = \min_{i=1..n} \{ \max_{M_{j(i)} \in \{M_1, M_2, ..., M_m\}} \{T_{j(i)} + \frac{M_{j(i)}}{J_i}\}\} \tag{2}$$

3) Giá sử dụng tài nguyên cực tiểu, nghĩa là:

$$V = \min\{\sum_{j(i)\in\{1..m\};i\in\{1..n\}} J_i \times v_{j(i)}\}. \quad (3)$$

4) Chi phí truyền tải dữ liệu cực tiểu, nghĩa là:

$$C = \min\{\sum_{j(i)\in\{1..m\};i\in\{1..n\}} D_i \times c_{j(i)}\}. \quad (4)$$

Một chú ý rất quan trọng là giá trị của biểu thức sau phải đủ lớn:

$$\min_{j(i)\in\{1..m\}} \frac{\sum_{i=1}^{n} J_i}{M_j}. \quad (5)$$

Điều này chính là bản chất của việc cần các hệ thống tính toán lưới. Nghĩa là tổng số lượng công việc khi thực hiện trên một hệ thống bất kỳ sẽ mất nhiều thời gian (dẫn đến kết quả đưa ra không có giá trị thực tế bởi tính thời gian của nó), vì vậy chúng ta phải giải quyết nó trên hệ thống tính toán lưới.

Việc định giá của sự truyền tải dữ liệu ở trên cũng rất cần nhấn mạnh đó là: vì trong tính toán lưới, các hệ thống tự trị thường được nối với nhau bằng các mạng đường trục tốc độ cao (như leased line), do vậy thời gian truyền tải dữ liệu (độ trễ - latency) có thể bỏ qua mà chỉ cần quan tâm đến chi phí đường truyền.

3. THUẬT TOÁN DI TRUYỀN CHO VIỆC LẬP LỊCH

3.1. Giới thiệu về thuật toán di truyền

Thuật toán di truyền [7] được xây dựng dựa trên quy luật tiến hóa sinh học hay phát triển tự nhiên của một quần thể sống. Các cá thể trải qua một quá trình phát triển và sinh sản để tạo ra những cá thể mới cho thế hệ tiếp theo. Trong quá trình tiến hóa những cá thể xấu (theo một tiêu chuẩn nào đó hay còn gọi là độ thích nghi với môi trường) sẽ bị đào thải. Ngược lại, những cá thể tốt sẽ được giữ lại. Liên quan đến giải thuật di truyền có các khái niệm sau:

- *Biểu diễn của cá thể*: Để áp dụng được thuật toán di truyền, việc đầu tiên là phải tìm được cách biểu diễn của các cá thể sao cho mỗi cá thể biểu diễn một giải pháp của bài toán đang được quan tâm.
- *Đánh giá độ thích nghi*: Độ thích nghi là khả năng phù hợp của mỗi cá thể (giải pháp) đối với môi trường (bài toán). Việc xây dựng độ thích nghi cũng là một bước quan trọng trong thuật toán di truyền. Để đánh giá được độ thích nghi của các cá thể giải thuật di truyền sử dụng một hàm đo độ thích nghi (Fitness Function).
- *Lai ghép*: Là quá trình tạo ra cá thể mới dựa trên nhiều cá thể đã có, gọi là các cá thể cha-mẹ. Hai cá thể con được tạo ra bằng cách hoán đổi các gen từ cá thể cha mẹ.

- *Đột biến*: Là quá trình tạo ra cá thể mới từ một cá thể ban đầu bằng cách thay đổi một số gen của nó.
- *Chọn lọc và thay thế*: Chọn lọc và thay thế (cũng được biết như là quá trình sinh sôi nảy nở - reproduction) là quá trình chọn những cá thể từ quần thể hiện tại để tạo ra thế hệ sau của nó. Trong quá trình này diễn ra sự đào thải những cá thể xấu, chỉ giữ lại những cá thể tốt. Những cá thể có độ thích nghi lớn hơn hoặc bằng với độ thích nghi tiêu chuẩn sẽ được giữ lại và độ thích nghi của các cá thể trong quần thể sẽ hoàn thiện hơn sau nhiều thế hệ.
- *Điều kiện dừng*: Thuật toán di truyền là một quá trình ngẫu nhiên, nên không thể đảm bảo chắc chắn thuật toán di truyền sẽ dừng sau hữu hạn bước. Vì vậy, để đảm bảo thuật toán di truyền sẽ kết thúc, người dùng thường phải định nghĩa điều kiện dừng cho thuật toán.

3.2. Thuật toán di truyền trong việc lập lịch

Như đã trình bày trong phần giới thiệu, thuật toán di truyền không chỉ giảm không gian tìm kiếm của bài toán mà còn cho phép tối ưu đa mục tiêu. Phần này tập trung trình bày thuật toán di truyền cho bài toán lập lịch trong lưới tính toán với bốn mục tiêu cần tối ưu hóa đồng thời như đã trình bày ở Mục 2.

Biểu diễn của các cá thể:

Để áp dụng thuật toán di truyền cho bài toán lập lịch, tiến hành mã hóa các cá thể như sau: mỗi cá thể là một véctơ, gọi là *schedule*, với kích thước n thành phần (n là số các công việc) với giá trị là các số nguyên trong khoảng từ $[1, m]$ (m là số lượng các tài nguyên) sao cho thành phần $schedule[i]$ chỉ ra công việc i được gửi cho tài nguyên (có số hiệu là giá trị) $schedule[i]$ thực hiện. Ví dụ, $schedule[3] = 5$ chỉ ra rằng: công việc thứ 3 sẽ được giao cho tài nguyên (hệ thống) số 5 thực hiện.

Khởi tạo quần thể ban đầu:

Sinh ra P (P là tham số thiết kế) cá thể ban đầu một cách ngẫu nhiên. Mỗi cá thể được sinh ngẫu nhiên bằng cách sinh ngẫu nhiên n lần các số nguyên trong khoảng từ 1 đến m và lần sinh thứ i ($i = 1...n$) tương ứng với các thành phần $schedule[i]$ của cá thể. Chú ý rằng sau khi sinh ngẫu nhiên cần kiểm tra tính hợp lệ của cá thể, nghĩa là tổng tải tính toán của các công việc giao cho một tài nguyên (hệ thống tự trị) nào đó không được vượt quá khả năng tính toán của tài nguyên đó.

Hàm đo độ thích nghi (Fitness Functions): Định nghĩa độ thích nghi của một cá thể k như sau:

$$F_k = \frac{1}{T_k^m + \alpha \times T_k^f + \varepsilon_1 \times V_k + \varepsilon_1 \times C_k}, \tag{6}$$

trong đó, $\alpha, \varepsilon_1, \varepsilon_2 \in (0, 1)$ là tham số thiết kế để điều chỉnh, T_k^f, T_k^m, V_k và C_k được tính như sau:

$$T_k^f = \sum_{i=1}^{n} ETC[i][schedule[i]], \tag{7}$$

$$T_k^m = \max_{i=1..n} (T_i + ETC[i][schedule[i]], \tag{8}$$

$$V_k = \sum_{i=1}^{n} J_i \times v_{schedule[i]}, \tag{9}$$

$$C_k = \sum_{i=1}^{n} D_i \times c_{schedule[i]}. \tag{10}$$

Phép toán lai ghép: Sử dụng phép toán lai ghép một điểm. Sinh ngẫu nhiên một số nguyên k trong khoảng từ $[2, n-1]$, k được gọi là điểm lai ghép và chia mỗi cá thể được chọn để lai ghép (gọi là cha mẹ) thành hai phần. Hai cá thể con được sinh ra bằng cách hoán đổi phần thứ hai (tính từ điểm lai ghép) của các cá thể cha mẹ. Chú ý rằng các cá thể con sinh ra cũng phải kiểm tra tính hợp lệ của chúng giống như trong quá trình khởi tạo ngẫu nhiên.

Quá trình lai ghép được thực hiện với mọi cặp có thể của các cá thể với một sắc xuất lai ghép là p_c (p_c là tham số thiết kế).

Phép toán đột biến: Sinh ngẫu nhiên một số nguyên k trong khoảng từ $[1, n]$, k được gọi là điểm đột biến. Sinh ngẫu nhiên một số nguyên j thay thế cho phần tử $schedule[k]$. Cá thể con được sinh ra cũng phải kiểm tra tính hợp lệ của chúng giống như trong quá trình khởi tạo ngẫu nhiên.

Quá trình đột biến được thực hiện với mọi cá thể với một sắc xuất đột biến là p_m(p_m là tham số thiết kế).

Chọn lọc các cá thể cho thế hệ tiếp: P cá thể có độ thích nghi cao nhất trong các cá thế của thế hệ trước, các cá thể được sinh ra bởi quá trình lai ghép và đột biến sẽ được lựa chọn cho thế hệ kế tiếp.

Điều kiện dừng: Thuật toán sẽ dừng sau G thế hệ (G là tham số thiết kế) hoặc giá trị trung bình về độ thích nghi của các cá thể không thay đổi.

Tóm lại, thuật toán di truyền cho việc lập lịch được viết như sau:

Bắt đầu

a. $t = 0$;

b. Khởi tạo quần thể ban đầu $P(t)$;

c. Tính độ thích nghi cho các cá thể thuộc $P(t)$;

d. Trong khi (điều kiện dừng chưa thoả mãn) lặp

i. $t = t + 1$;

ii. Lựa chọn $P(t)$ từ $P(t-1)$;

iii. Lai ghép trên $P(t)$ để nhận được $Q(t)$;

iv. Đột biến trên $P(t)$ để nhận được $R(t)$;

v. Chọn lọc từ $P(t-1) \cup Q(t) \cup R(t)$ để nhận được $P(t)$;

e. Hết lặp;

Kết thúc;

3.3. Độ phức tạp của thuật toán

Các tham số của thuật toán GA được ấn định trước khi chạy, vì vậy độ phức tạp của thuật toán có thể tính như sau.

- Quá trình khởi tạo quần thể ban đầu là: $P.O(n)$
- Tính toán các *flowtime* và *makespan* cho mỗi cá thể: $O(n.m) + O(P \log P)$
- Quá trình lai ghép: $P.(P-1).O(n) = P^2.O(n)$.
- Quá trình đột biến: $P.O(n)$.
- Quá trình chọn lọc: $O(P \log P)$ (vì chỉ cần sắp xếp)

Do vậy độ phức tạp sẽ là: $P.O(n)+G(P.(O(n.m)+O(P \log P))+P2.(O(n.m)+P \log P+O(n)) + P.O(n) + O(P \log P)) = G.P^2.O(n.m))$.

4. CÁC KẾT QUẢ THỬ NGHIỆM

Các kết quả thử nghiệm trình bày ở đây là kết quả trung bình của 20 lần chạy thử nghiệm độc lập. Chương tình mô phỏng được thực hiện bằng ngôn ngữ C^{++} chạy trên máy tính đơn có bộ vi xử lý Intel PenIV 2.7 Ghz, RAM 1Gb.

Kịch bản mô phỏng

Giả sử số công việc là $n = 200$ và số các tài nguyên của lưới tính toán $m = 15$. Chúng tôi cho sinh ngẫu nhiên tải tính toán cho mỗi công việc trong khoảng từ $[50, 200]$ và kích thước dữ liệu trong khoảng từ $[10, 30]$. Khả năng tính toán của các tài nguyên cũng được sinh ngẫu nhiên trong khoảng $[1000, 2000]$ với chi phí tính cho việc sử dụng mỗi đơn vị tài nguyên trong khoảng $[1, 10]$ và chi phí truyền dữ liệu trong khoảng từ $[1, 5]$. Sinh ngẫu nhiên tải tính toán đang sử dụng của mỗi tài nguyên trong khoảng từ $[500, 1200]$. Dữ liệu này được giữ nguyên cho tất cả các mô phỏng trong bài.

Các tham số của thuật toán di truyền

Trong các thí nghiệm của chúng tôi, các tham số p_c và pm được thiết lập theo khuyên cáo chung của thuật toán di truyền đó là $p_c = 0,02$ và $p_m = 0,6$ [7]. Chúng tôi chọn $P = 50$ (kích thước của quần thể) và $G = 50$ (số thế hệ tối đa trong quá trình tiến hoá) theo các tác giả trong [9].

Xác định tham số của hàm đo độ thích nghi

Như đã biết, makespan và flowtime là hai yếu tố quan trọng của bài toán lập lịch, tuy nhiên các nghiên cứu trước đây $[2-6]$ đều chỉ ra rằng makespan là quan trọng hơn flowtime. Vì vậy, chúng tôi chỉ chọn α trong khoảng $(0, 1)$ (do trọng số của makespan trong công thức (6) là 1). Hơn nữa trong thí nghiệm chúng tôi muốn ưu tiên các tham số makespan và flowtime do vậy giá tính toán trên tài nguyên và chi phí truyền tải dữ liệu được xem như các tham số phụ, cho nên chúng tôi chọn $\varepsilon_1 = \varepsilon_2 = 0,02$ và giữ nguyên trong mọi mô phỏng. Với giá trị của α thay đổi, kết quả mô phỏng được đưa ra trong Bảng 1.

Bảng 1. Hiệu quả của thuật toán khi giá trị a thay đổi

α	0,2	0,3	0,4	0,5	0,6
makespan	72,4	72,41	74,5	75,10	75,92
flowtime	67723	67726	67730	67869	67932

Dựa vào kết quả thực nghiệm, có thể thấy rằng với $\alpha < 0,4$ thuật toán đạt được hiệu quả tốt và ổn định. Do vậy trong các thí nghiệm so sánh với các thuật toán khác chúng tôi

chọn $\alpha = 0,3$.

So sánh hiệu quả của thuật toán di truyền với thuật toán khác

Các tác giả trong [3] chỉ ra rằng, thuật toán Simulated Annealing đạt được hiệu quả cao hơn các thuật toán Heuristics khác (khi chỉ quan tâm đến giá trị của makespan). Do vậy trong thử nghiệm này, chúng tôi chỉ so sánh thuật toán được đề xuất với thuật toán Simulated Annealing được mô tả trong [3]. Kết quả so sách được chỉ ra trong Bảng 2.

Bảng 2. So sánh hiệu quả của các thuật toán

	Simulated Annealing	Genetic Algorithm
Makespan	74,23	72,41
Flowtime	68154	67726

Kết quả so sánh trong Bảng 2 chỉ ra rằng thuật toán được nghiên cứu trong bài đưa ra phương án lập lịch tốt hơn thuật toán lập lịch dựa trên simulated annealing. Giá trị của Makespan không khác nhau nhiều, tuy nhiên sự khác nhau của Flowtime là rất đáng kể.

5. KẾT LUẬN VÀ ĐỀ XUẤT

Trong bài báo này, chúng tôi đã trình bày thuật toán di truyền cho bài toán lập lịch trong hệ thống tính toán lưới. Bằng việc đưa ra một hàm đo độ thích nghi mới, đồng thời quan tâm đến cả các yếu tố quan trọng: khoảng thời gian hoàn thành tất các công việc, tổng thời gian thực hiện của các công việc, giá sử dụng tài nguyên và truyền tải dữ liệu, thuật toán được đề xuất đã đưa ra các phương án lập lịch tốt hơn thuật toán được mô tả trong [3]. Điều này đã được kiểm chứng bằng các thử nghiệm. Tuy vậy, bài báo này cũng chưa quan tâm đến việc phát biểu hình thức cho các tham số sử dụng trong hàm đo độ thích nghi mới. Đây là phần việc sẽ được nghiên cứu tiếp tục trong tương lai.

TÀI LIỆU THAM KHẢO

[1] C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publisher Inc, 2002.

[2] G. Richie and J. Levine, "A fast, effective local search for scheduling independent jobs in heterogeneous computing environments", Technical report, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, 2003.

[3] A. Yarkhan and J. Dongarra, Experiments with scheduling using simulated annealing in a grid environment, *Proc. of 3rd International Workshop on Grid Computing*, USA, 2002 (232–242).

[4] A. Abraham, R. Buyya, and B. Nath, Natures heuristics for scheduling jobs on computational grids, *Proc. of the 8th IEEE International Conference on Advanced Computing and Communications*, India, 2000.

[5] V. D. Martino and M. Mililotti, Scheduling in a grid computing environment using genetic algorithms, *Proc of IPDPS,* USA, 2002.

[6] G. Ritchie and J. Levine, A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments, *Proc. of 23rd Workshop of the UK Planning and Scheduling Special Interest Group*, UK, 2004.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company Inc., 1997.

[8] M. Maheswaran, et al., Dynamic mapping of a class of independent tasks onto hetergeneous computing systems, *Journal of Parallel and Distributed Computing* **52** (2) (2005) 415–429.

[9] Trịnh Thị Thúy Giang, Lê Trọng Vĩnh, Hoàng Chí Thành, Nguyễn Thanh Thủy, Thuật toán di truyền cho việc lập lịch trong hệ thống tính toán lưới, *The Third National Symposium Fundamental and Applied Information Technology Research - FAIR2007*, Nha Trang, Aug. 2007.

Nhận bài ngày 1 - 4 - 2008
Nhận lại sau sửa ngày 8 - 12 -2008