UED Journal of Sciences, Humanities & Education – ISSN 1859 - 4603

TẠP CHÍ KHOA HỌC XÃ HỘI, NHÂN VĂN VÀ GIÁO DỤC

SỬ DỤNG GIẢI THUẬT TỐI ƯU HÓA RỪNG CÂY RỜI RẠC CHO BÀI TOÁN LẬP LỊCH CÁC CÔNG VIỆC ĐỘC LẬP TRONG LƯỚI TÍNH TOÁN

Nhận bài:
12 – 01 – 2015
Chấp nhận đăng:
25 – 03 – 2015
http://jshe.ued.udn.vn/

Đỗ Vĩnh Trúc

Tóm tắt: Lưới tính toán (Computational Grid-CG) là bài toán mới xuất hiện gần đây. Việc lập lịch (scheduling) với các công việc độc lập (independent jobs) trên CG với mục tiêu cực tiểu makespan là bài toán khó nhưng hấp dẫn. Đóng góp mới nhất vào nhóm các giải thuật tiến hóa nổi tiếng như giải thuật di truyền (Genetic Algorithm-GA), tối ưu bầy đàn (Particle Swarm Optimization-PSO), giải thuật tối ưu hóa đàn kiến (Ant Colony Optimization-ACO)… để giải quyết bài toán này cũng như các bài toán trong lĩnh vực tối ưu hóa là tối ưu hóa rừng cây (Forest Optimization Algorithm-FOA) [1]. Đề tài này giới thiệu thuật toán FOA có hiệu chỉnh và áp dụng để giải quyết bài toán lập lịch các công việc độc lập trên lưới tính toán với mục tiêu cực tiểu hóa makespan. Kết quả cho thấy FOA có thể áp dụng tốt cho việc giải bài toán tối ưu hóa trên.

Từ khóa: giải thuật tối ưu hóa rừng cây; lưới tính toán; công việc độc lập; lập lịch; makespan.

1. Đặt vấn đề

Một CG là một hệ tính toán phân tán theo địa lý bao gồm một tập hợp các tài nguyên máy tính đa dạng, quy mô rộng lớn và độc lập [2][3][4][5], chúng được nối kết với nhau bởi các mạng băng thông cao [6]. Việc chia sẻ các công việc tính toán là một ứng dụng chính của tính toán lưới. Trong một CG, các nguồn tài nguyên năng động, đa dạng và có thể được thêm vào và rút ra bất kỳ lúc nào. CG được coi là một tiếp cận hiệu quả để giải quyết các ứng dụng của thế giới thực phân tán, quy mô lớn [7]. Lập điều độ trong môi trường CG, có nghĩa là phân bổ công việc cho các tài nguyên trên tính toán lưới, đó là công việc rất quan trọng và nhiệm vụ tính toán khó khăn thậm chí ngay cả khi các công việc độc lập nhau do yêu cầu thực tế [3].

Xét bài toán lập lịch trên tính toán lưới cho các công việc độc lập [7][8]. Với một số lượng n công việc (job) và m máy móc (machine) cho trước, mục tiêu là tìm ra giải pháp tối ưu để phân bổ công việc cho các máy nhằm cực tiểu hóa makespan Cmax=Max{Ci}, với Ci là thời gian hoàn thành của máy i=1,…m. Trong bài toán này một công việc chỉ có thể được xử lý chỉ trên một máy và một máy chỉ có thể xử lý một công việc tại một thời điểm nào đó. Các giả định là các công việc độc lập nhau và không có sự ưu tiên.

Braun và cộng sự [9] so sánh 11 heuristics cho lập lịch tính toán lưới (Grid Computating Scheduling- GCS) với các công việc độc lập. Một tập dữ liệu lớn [9] đã được phát triển dựa trên ma trận thời gian kỳ vọng cho tính toán (Expected Time to Compute-ETC) nhằm đánh giá hiệu suất của heuristics đã đề xuất. Từ thực nghiệm dựa trên thuật toán di truyền, [9] cung cấp hiệu suất tốt nhất trong hầu hết các trường hợp, mà trong đó heuristic Min-Min là một phương pháp tốt để nhanh chóng tạo ra một giải pháp với một makespan ngắn hợp lý. Hai tác giả Page và Naughton [10] đã đề xuất một phương pháp GA khác cho GCS có sử dụng một danh sách heuristic đã lập lịch để khởi tạo ra một tổng thể ngẫu nhiên ban đầu khá tốt. Các toán tử đột biến trong đề xuất này được chuyên biệt hóa nhằm cải thiện hiệu suất GA. Ritchie và Levine [11] đã phát triển giải thuật tối ưu hóa đàn kiến lai (Hybrid ACO) để lập lịch trên tính toán lưới. Tìm kiếm Tabu [12] được dùng để hoàn thiện các giải pháp đạt được bằng ACO. Các ACO lai tạo được đánh giá tốt hơn các phương pháp GA và heuristics khác. Thuật toán cellular memetic (Cellular Memetic Algorithm-CMA) của Xhafa và cộng sự [13] đã giảm thiểu cả makespan và flowtime. Heuristics địa

* Liên hệ tác giả
Đỗ Vĩnh Trúc
Trường Đại học Quốc tế, Đại học Quốc Gia TP. HCM
Email: dvtruc@hcmiu.edu.vn
Điện thoại: 0919067281

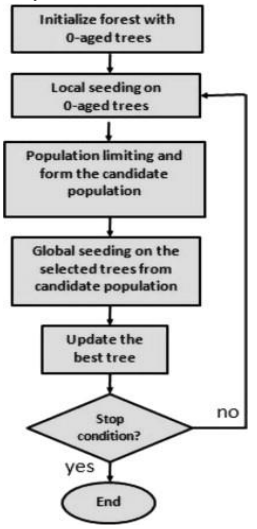
phương khác cũng đã được kiểm tra trong bài báo [13] này. Kết quả cho thấy CMA là một cách tiếp cận hiệu quả cho GCS. Xhafa và cộng sự [3] đã phát triển một phương pháp tìm kiếm Tabu mới cho GCS. Chiến lược đa dạng hóa được chuyên biệt khác cũng đã được xem xét trong phương pháp tìm kiếm Tabu nhằm nâng cao hiệu quả của nó. Phương pháp này không những nhanh hơn so với giải thuật như Tabu Search [12], ACO lai [11], và CMA [13] mà còn cung cấp các lời giải tốt hơn. Các heuristics khác cũng được đề xuất cho GCS như giải thuật luyện kim (Simulated Annealing-SA) [9][14], giải thuật di truyền đấu tranh (Struggle GA-SGA) [13], sufferage [15], GA lai [16].

Gần đây, một số phương pháp PSO [2][17] đã được đề xuất để giải bài toán GCS với công việc độc lập. Liu và các cộng sự [2] đề xuất một phương pháp PSO liên tục (Continuous PSO-CPSO) cho GCS. Trong phương pháp này, vị trí và tốc độ của một cá thể được biểu diễn như là ma trận số thực (n×m). Để xây dựng một lịch trình vị trí, đầu tiên ma trận phải được chuyển đổi thành một lịch trình bằng cách gán cho mỗi công việc vào máy mà có giá trị chuẩn hóa cao nhất trong cột tương ứng của công việc đó trong ma trận vị trí. Ma trận vị trí trong phương pháp này được coi là một ma trận mờ trong đó một yếu tố đại diện cho mức độ thành viên của công việc và máy tương ứng. Các thí nghiệm cho thấy rằng CPSO là tốt hơn so với SA và GA. Izakian cùng các cộng sự [17] đưa ra một cách tiếp cận PSO rời rạc (Discrete PSO-DPSO) để lập lịch lưới. Trong phương pháp [17], các cá thể được đại diện là một ma trận vận tốc số thực cho công việc và máy. Giá trị tại một vị trí công việc/máy được cho trước trong ma trận xác định công việc này có liên quan đến máy khác như thế nào. Vị trí của một cá thể là một mảng các số nguyên, ở đó mỗi vị trí trong danh sách là một công việc và các số nguyên ở vị trí đó là các máy mà công việc được phân công đến. Trong giai đoạn cập nhật cho cá thể này, vận tốc của cá thể được thay đổi hướng về phía vận tốc của cá thể tốt nhất toàn cục và tốc độ cá thể tốt nhất dựa trên các thành phần xã hội công nhận và tự nhận, tương ứng. Khi vận tốc được cập nhật, vị trí của cá thể được thiết lập để mỗi công việc được giao cho các máy có giá trị cao nhất trong cột đó của công việc của ma trận vận tốc. Thí nghiệm [17] cho thấy rằng DPSO là tốt hơn so với chẩn đoán khác như Min-Min, GA, và ACO lai [11].

2. Giải thuật Tối ưu hóa rừng cây- Forest Optimization Algorithm (FOA) [1]

2.1. Giới thiệu

Thuật toán FOA bao gồm ba giai đoạn chính:
1- Gieo mầm địa phương.
2- Giới hạn quần thể.
3- Gieo mầm toàn cục.
Sơ đồ của thuật toán như sau.

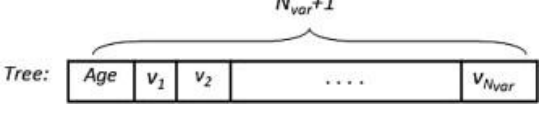


Hình 1. *Lưu đồ của FOA*

FOA bắt đầu với quần thể ban đầu của cây. Mỗi cây đại diện cho một giải pháp khả dĩ của bài toán. Một cây gồm có giá trị các biến và độ tuổi (Age). Độ tuổi của một cây ban đầu được gán bằng 0. Sau khi được khởi tạo, các cây con sẽ được nẩy mầm từ những cây có tuổi 0 và chúng được thêm vào rừng. Sau đó, tất cả các cây cũ tăng thêm 1 tuổi.

2.1.1. Khởi tạo cây

Tạo các cây có Age=0. Hình 2 cho thấy một cây có Nvar chiều, là các giá trị của các biến và Age là độ tuổi của cây.



Hình 2. *Biểu diễn lời giải của FOA*

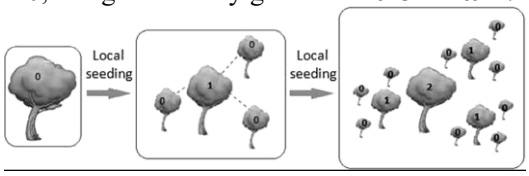
Một cây được coi là một mảng 1x(Nvar+1), với Nvar là số chiều của bài toán và Age đại diện cho tuổi của cây.

Tree=[Age,v1,v2,v3…,vnvar]

Tuổi tối đa cho phép của một cây là một tham số được xác định trước và được đặt tên là tuổi thọ (life time). Tuổi thọ được xác định vào lúc bắt đầu của thuật toán. Khi Age một cây đạt đến "tuổi thọ", cây bị loại khỏi rừng và được thêm vào danh sách cây ứng viên. Nếu tham số này lớn, mỗi lần lặp của thuật toán chỉ làm tăng độ tuổi của cây và rừng sẽ chứa nhiều cây già nua mà không tham gia vào các giai đoạn gieo mầm địa phương. Nếu tham số này nhỏ thì cây sẽ già đi rất sớm và chúng sẽ bị bỏ qua ở giai đoạn đầu của khởi tạo. Vì vậy, tham số này sẽ cung cấp một cơ hội tốt cho tìm kiếm địa phương.

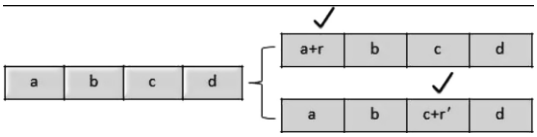
2.1.2. Gieo mầm địa phương

Gieo mầm địa phương của cây trong bài cố gắng mô phỏng quá trình tạo cây con của thiên nhiên. Chỉ có cây với Age= 0 mới cho nảy sinh cây non thành cây láng giềng và tạo thành rừng. Hai lần tạo cây non được minh họa như Hình 3. Sau một lần như vậy, cây có Age= 0 sẽ thành 1 và cây con có Age=0, trong khi đó cây già hơn sẽ thêm 1 tuổi.



Hình 3. *Ví dụ của gieo mầm địa phương trên một cây cho 2 lần lặp, với LSC=3*

Các cây vượt quá "tuổi thọ" sẽ được xem xét. Nếu một cây là đầy tiềm năng, Age của cây đó được gán về 0, và đưa nó thành láng giềng tốt vào rừng. Ngược lại, các cây không hứa hẹn sẽ chết. Số cây non được tạo ra từ 1 cây nào đó do tham số "Sự thay đổi gieo mầm địa phương" (Local Seeding Changes-LSC) quyết định. Giá trị của tham số LSC này là 3 trong như trong Hình 3. Kết quả, thực hiện gieo mầm địa phương trên một cây với Age 0 sẽ nảy mầm 3 cây non. Tham số này nên được xác định tùy theo kích thước của bài toán. Gieo mầm địa phương mô phỏng tìm kiếm cục bộ cho thuật toán này. Hình 4 minh họa một ví dụ về công đoạn gieo mầm địa phương cho bài toán thực trong không gian liên tục 4 chiều và ở đây giá trị của "LSC" được coi là 2. Nếu (a+r) hay (c+r') nằm ngoài giới hạn dưới và trên của biến liên quan nó sẽ được điều chỉnh để để thuộc trong giới hạn cho phép.



Hình 4. *Một ví dụ gieo mầm địa phương cho không gian liên tục, r và r' thuộc [-Δx, Δx]*

2.1.3. Giới hạn quần thể

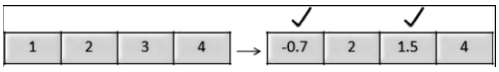
Số cây trong rừng phải được giới hạn để ngăn chặn sự mở rộng vô hạn của rừng bằng tham số "giới hạn diện tích" ("area limit"). Xếp hạng các cây từ tốt đến xấu, nếu số lượng cây lớn hơn "giới hạn diện tích", cây tốt sẽ được giữ lại, cây xấu hơn sẽ bị loại ra khỏi rừng và thêm vào nhóm cây ứng viên. Các cây được khởi tạo ban đầu bằng với "giới hạn diện tích". Sau khi hạn chế số cây của rừng, giai đoạn gieo mầm toàn cục được thực hiện trên tỷ lệ phần trăm của nhóm ứng viên sẽ được mô tả sau.

2.1.4. Gieo mầm toàn cục

Động vật, chim trong rừng ăn hạt và trái cây của những cây này và làm hạt giống của cây được phát tán trong toàn bộ khu rừng và kết quả là môi trường sống của cây trở nên rộng hơn. Đó là giai đoạn gieo mầm toàn cục. Giai đoạn này được thực hiện trên tỷ lệ phần trăm xác định qua một tham số có tên là "tốc độ lan truyền" (tranfer rate). Đầu tiên, các cây từ nhóm ứng viên được lựa chọn theo "tốc độ lan truyền". Sau đó, một số biến của mỗi cây được lựa chọn ngẫu nhiên. Giá trị của mỗi biến được lựa chọn sẽ được thay bằng một giá trị ngẫu nhiên. Bằng cách này, không gian tìm kiếm toàn cục được xem xét và không bị hạn chế. Kết quả là cây có tuổi 0 được thêm vào rừng. Số lượng các biến có giá trị sẽ bị thay đổi là một tham số của thuật toán và được đặt tên là "Thay đổi gieo mầm toàn cục" (Global Seeding Changes-GSC). Hình 5 là một ví dụ về thực hiện công đoạn gieo hạt toàn cầu cho một cây trong không gian liên tục. Trong Hình 6, GSC= 2, như vậy 2 biến được lựa chọn ngẫu nhiên và giá trị của chúng được thay bằng 2 giá trị được tạo ra ngẫu nhiên khác như r và r'.

Hình 5. *Ví dụ gieo mầm toàn cục trên 1 cây*

Ví dụ trong Hình 6 cho thấy giá trị của tham số GSC=2 và phạm vi là [-5,5]. Kết quả là, giá trị của 2 biến lựa chọn ngẫu nhiên thay bằng 2 giá trị khác trong phạm vi [-5, 5] như -0.7 và 1.5.



Hình 6. *Ví dụ bằng số của gieo mầm toàn cục với GCS=2*

2.1.5. Cập nhật giá trị tối ưu

Sau khi phân loại cây theo giá trị phù hợp của chúng, cây có giá trị phù hợp cao nhất được chọn làm cây tốt nhất, Age của cây tốt nhất sẽ được thiết lập về 0. Như vậy cây tốt nhất có thể đạt tối ưu hóa địa phương trong giai đoạn gieo mầm địa phương.

2.1.6. Điều kiện dừng

Ba điều kiện dừng có thể được áp dụng: 1) số bước lặp được xác định trước; 2) không có sự thay đổi trong giá trị tối ưu sau một số lần lặp; 3) đạt đến cấp độ nhất định chính xác. Các giai đoạn chính của FOA được hiển thị như mã giả ở phần sau.

2.2. Giải thuật FOA

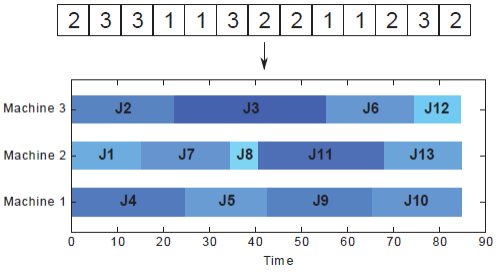
Giải thuật FOA(life time, LSC, GSC, tranfer rate, area limit)

Nhập: giá trị tối ưu hoặc gần tối ưu của hàm mục tiêu f(x).
Xuất: trả về giá trị gần tối ưu hay tối ưu của f(x).
1. Khởi tạo rừng với cây tạo ngẫu nhiên
    Mỗi cây là một vec tơ x có (n+1) chiều, ứng với bài toán n chiều, $x=(age,x_1,x_2,\ldots,x_n)$.
    Age=0 lúc khởi tạo.
2. Trong khi điều kiện dừng chưa thỏa
    2.1. Thực hiện gieo mầm cục bộ với cây có Age=0
        For i=1:LSC
            Chọn ngẫu nhiên 1 biến của cây đang xét.
            Thêm một giá trị dx thuộc [-Δx, Δx] vào biến ngẫu nhiên trên.
        Tăng tuổi của các cây lên 1 tuổi ngoại trừ cây con mới mọc.
    2.2. Giới hạn quần thể
        Loại bỏ cây có tuổi vượt quá "Life time" và đưa chúng vào nhóm cây ứng viên.
        Sắp xếp cây theo độ phù hợp fitness.
        Loại bỏ các cây nằm ngoài "area limit" có độ phù hợp thấp nhất và đưa chúng vào nhóm cây ứng viên.
    2.3. Gieo mầm toàn cục.
        Chọn tỉ lệ "tranfer rate" từ nhóm cây ứng viên.
        Với mỗi cây ứng viên trên
            Chọn ngẫu nhiên "GCS" biến
            Thay giá trị của mỗi biến với giá trị ngẫu nhiên trong miền giới hạn và tạo cây con với Age=0, sau đó đưa vào rừng.
    2.4. Cập nhật giá trị tối ưu
        Sắp xếp cây theo fitness.
        Gán trị Age=0 cho cây tốt nhất.
3. Trả về giá trị tốt nhất

3. Giải thuật đề nghị

3.1. Mô tả bài toán

Trong giải thuật đề nghị này, mỗi cây x=(Age,x1,…,xn) là một mảng số nguyên. Mỗi phần tử xj đại diện cho công việc j được gán cho một máy cụ thể nào đó. Bất kỳ công việc j nào cũng có thể được xử lý trên 1 máy nào đó tùy ý. Vì các công việc là độc lập và mục tiêu của bài toán là cực tiểu hóa makespan nên lời giải của bài toán là các công việc không bị ràng buộc theo một thứ tự nào. Xét một ví dụ như hình sau, x1=2, x2=3 có nghĩa công việc 1, 2 được xử lý trên máy 2, và 3 với thời gian xử lý là p12 và p23 và được biểu diễn như sơ đồ trong Hình 7.



Hình 7. *Biểu diễn lời giải của FOA đề xuất*

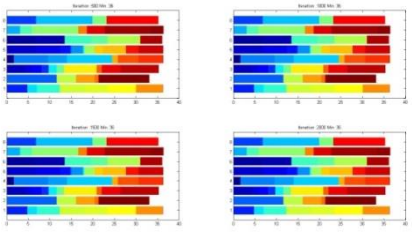
3.2. Giải thuật tổng thể

Khởi tạo rừng bằng các cây được tạo ngẫu nhiên
Mỗi cây là 1 vec tơ x có (d+1) chiều, $x=(age,x_1,x_2,\ldots,x_D)$
Gán age=0
Trong khi điều kiện dừng chưa thỏa
    Thực hiện gieo mầm cục bộ với cây có Age=0 bằng các số rời rạc.
    Thực hiện giới hạn quần thể.
    Gieo mầm toàn cục.
    Cập nhật cây tốt nhất.
Trả về cây tốt nhất

4. Thiết kế thực nghiệm và kết quả

Để minh họa, bài báo sử dụng dữ liệu được lấy từ [2]. Thực nghiệm bắt đầu với bài toán có 3 máy và 13 công việc, ký hiệu là (3,13). Tốc độ xử lý của 3 máy là 4,3,2 đơn vị thời gian và 13 công việc có thời gian xử lý là 6, 12, 16, 20, 24, 28, 30, 36, 40, 42, 48, 52, 60 đơn vị thời gian. Kết quả thực tế từ giải thuật này với 10 lần chạy là (46, 47, 46, 47, 47, 47, 47, 47, 47, 46). Giá trị nhỏ nhất và giá trị trung bình là 46 và 46.67. Thông số giải thuật cho bài toán với số chiều <5 là "life time"=15, LSC=1, GSC=1, "tranfer rate"=10, "area limit"=10. Còn với số chiều >=5 thì "life time"=15, LSC=20%*số chiều của bài toán, GSC=10%*số chiều của bài

toán, "tranfer rate"=10, "area limit"=10). Hình dưới đây cho thấy kết quả của bài toán (3,13).



Hình 8. *Kết quả bài toán (3,13)*

Với bài toán (5,100), thực hiện 10 lần thử nghiệm kết quả là (100, 100, 100, 101, 100, 100, 101, 100, 100, 100). Giá trị nhỏ nhất và giá trị trung bình là 100 và 100.2. Với bài toán (8,60), thực hiện 10 lần thử nghiệm kết quả là (40, 39, 40, 40, 39, 40, 40, 40, 40, 39). Giá trị nhỏ nhất và giá trị trung bình là 39 và 39.7. Với bài toán (10,50), thực hiện 10 lần thử nghiệm kết quả là (41, 41, 42, 40, 43, 41, 41, 41, 42, 42). Giá trị nhỏ nhất và giá trị trung bình là 40 và 41.4.

5. Kết luận

Bài báo này giới thiệu giải thuật mới, tối ưu hóa rừng cây để giải quyết các bài toán tối ưu liên tục. Sau đó chúng tôi đã chỉnh sửa để có thể áp dụng cho bài toán rời rạc. Ý tưởng chính của bài này dùng giải thuật tối ưu hóa rừng cây với các biến rời rạc để giải bài toán lập lịch lưới tính toán cho các công việc độc lập. Kết quả thực nghiệm cho thấy giải thuật cho kết quả tốt và nhanh chóng trên các bài toán (3,13), (5,100), (8,60) và (10,50). Với bài toán (3,13) thì kết quả cũng xấp xỉ như [8]. Các kết quả khác chưa có so sánh nhưng thời gian hội tụ là rất nhanh. Kết quả cho thấy giải thuật đạt đến kết quả nhanh chóng cho bài GCS. Hy vọng rằng FOA với không gian rời rạc sẽ được áp dụng cho các bài toán của lập lịch nói riêng hay các lĩnh vực khác nói chung.

Tài liệu tham khảo

[1] M. Ghaemi and M.-R. Feizi-Derakhshi (2014), "Forest Optimization Algorithm", Expert Systems with Applications, vol. 41, no. 15, pp. 6676–6687, Nov. 2014.

[2] H. Liu, A. Abrahamc, and A. E. Hassanien (2010), "Scheduling jobs on computational grids using a fuzzy particle swarm optimisation algorithm", Future Generation Computer Systems, vol. 26, pp. 1336–1343.

[3] F. Xhafa, J. Carretero, B. Dorronsoro, and E. Alba (2009), "A tabu search algorithm for scheduling independent jobs in computational grids", Computing and informatics, vol. 28, no. 2, pp. 237–250.

[4] I. Foster, C. Kesselman, and S. Tuecke (2001), "The anatomy of the grid," Berman et al.[2], pp. 171–197.

[5] F. Dong and S. G. Akl (2006), "Scheduling algorithms for grid computing: State of the art and open problems", Technical report.

[6] I. Foster and C. Kesselman (2003), The Grid 2: Blueprint for a New Computing Infrastructure. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[7] E.-G. Talbi and A. Y. Zomaya, Eds. (2007), "Wiley Series on Bioinformatics: Computational Techniques and Engineering", in Grid Computing for Bioinformatics and Computational Biology, John Wiley & Sons, Inc, pp. 393–393.

[8] S. B. Nguyen, M. Zhang, and others (2014), "A hybrid discrete particle swarm optimisation method for grid computation scheduling", in Evolutionary Computation (CEC), 2014 IEEE Congress on, pp. 483–490.

[9] Howard Jay Siegel Tracy D. Braun and N. Beck (2001), "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing, vol. 61, pp. 810–837.

[10] A. J. Page and T. J. Naughton (2005), "Framework for Task Scheduling in Heterogeneous Distributed Computing Using Genetic Algorithms", Artif Intell Rev, vol. 24, no. 3–4, pp. 415–429.

[11] G. Ritchie and J. Levine (2003), "A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments".

[12] J. L. Graham Ritchie, "A fast, effective local search for scheduling independent jobs" in heterogeneous computing environments.

[13] F. Xhafa, E. Alba, and B. Dorronsoro (2007), "Efficient Batch Job Scheduling in Grids using Cellular Memetic Algorithms," in Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, pp. 1–8.

[14] A. YarKhan and J. J. Dongarra (2002), "Experiments with Scheduling Using Simulated Annealing in a Grid Environment", in Grid Computing — GRID 2002, M. Parashar, Ed. Springer Berlin Heidelberg, pp. 232–242.

[15] H. Izakian and A. Abraham, Performance Comparison of Six Efficient Pure Heuristics for Scheduling Meta-Tasks on Heterogeneous Distributed Environments.
[16] A. Abraham, R. Buyya, and B. Nath (2000), "Nature's Heuristics for Scheduling Jobs on Computational Grids", in ieee international conference on advanced computing and communications, pp. 45–52.
[17] H. Izakian, B. T. Ladani, A. Abraham, and V´aclav Sn´aˇsel (2010), "A Discrete Particle Swarm Optimization Approach for Grid Job Scheduling," International Journal of Innovative Computing, Information and Control, vol. 6, no. 9.

USING THE DISCRETE FOREST OPTIMIZATION ALGORITHM FOR THE PROBLEM OF SCHEDULING INDEPENDENT JOBS ON COMPUTATIONAL GRIDS

Abstract: The Computational Grid (CG) is a new problem which has appeared recently. The scheduling of independent jobs on CG for the purpose of minimizing makespan is difficult but fascinating. To solve this problem as well as problems in the field of optimization, there has been the latest contribution to the group of well-known evolutionary algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Forest Optimization Algorithm (FOA) [1],... This paper introduces a revised FOA and applies it to the solution of the problem of independent job scheduling on CG with the goal of minimizingmakespan. The results show that FOA is also a good algorithm for solving the above optimization problem.

Key words: FOA; computational grid; independent job; scheduling; makespan.