Mục lục

Danh sách hình vẽ

Danh sách bảng

1 – Giới thiệu về giải thuật GRASP

1.1 Giới thiệu

GRAPS (Greedy Randomized Adaptive Search Procedure) là một metaheuristic đa khởi đầu để giải quyết các vấn đề tối ưu tổ hợp khó khăn. Mỗi lần lặp của GRASP bao gồm 2 giai đoạn: 1 giai đoạn xây dựng cấu trúc tham ăn thích nghi ngẫu nhiên và một giai đoạn tìm kiếm địa phương (local search). Bắt đầu từ các giải pháp khả thi được xây dựng từ giai đoạn tìm cấu trúc tham lam thích nghi ngẫu nhiên, giai đoạn tìm kiếm địa phương sẽ tìm kiếm các giải pháp hàng xóm của giải pháp được chọn cho đến khi tìm được một giải pháp tối ưu địa phương. Giải pháp tốt nhất được tìm thấy qua tất cả các lần lặp được là kết quả của quá trình.

GRAPS được giới thiệu lần đầu tiên vào năm 1989 bởi Thomas A. Feo và Mauricio G. C. Resende. Trong lần giới thiệu đầu tiên này GRASP đã được ứng dụng vào vấn đề tập phủ.

1.2 Bài toán và giải thuật

Xét bài toán tối ưu tổ hợp định nghĩa bởi một tập hữu hạn E = 1,...,n, một tập tất cả các giải pháp có thể $F \subseteq 2^E$ và một hàm mục tiêu f: $2^E \rightarrow R$. Trong phiên bản cực tiểu, chúng ta tìm một giải pháp $S^* \in F$ sao cho $f(S^*) \leq f(S), \forall S \in F$. Tập E, hàm giá f, tập các giải pháp khả thi F được định nghĩa cụ thể cho từng bài toán. Ví dụ, trong trường hợp của bài toán người đi du lịch, tập E là tập tất cả các cạnh

kết nối tất cả các thành phố cần được tới thăm. Hàm giá $f(S)$ là tổng giá trị của tất cả các cạnh $e \in S$, và F được hình thành bởi tất cả các tập cạnh xác định bởi chu trình Haminton.
GRAPS (Greedy Randomized Adaptive Search Procedure) là một metaheuristic đa khởi đầu để giải quyết các vấn đề tối ưu tổ hợp khó khăn. Mỗi lần lặp của GRASP bao gồm 2 giai đoạn: 1 giai đoạn xây dựng cấu trúc tham ăn thích nghi ngẫu nhiên và một giai đoạn tìm kiếm địa phương (local search). Bắt đầu từ các giải pháp khả thi được xây dựng từ giai đoạn tìm cấu trúc tham lam thích nghi ngẫu nhiên, giai đoạn tìm kiếm địa phương sẽ tìm kiếm các giải pháp hàng xóm của giải pháp được chọn cho đến khi tìm được một giải pháp tối ưu địa phương. Giải pháp tốt nhất được tìm thấy qua tất cả các lần lặp được là kết quả của quá trình. Giả mã trong hình 1.1 mô tả khối main của quá trình GRASP cho giá trị cực tiểu với Max_Iterations bước lặp được thực hiện và Seed được sử dụng như giá trị ban đầu cho hàm sinh số giả ngẫu nhiên.

procedure `GRASP(Max_Iterations, Seed)`

```
1: Read_Input();
2: for k = 1 to Max_Iterations do
3:   Solution ← Greedy_Randomized_Construction(Seed);
4:   Solution ← Local_Search(Solution);
5:   Update_Solution(Solution, Best_Solution);
6: end for
7: return Best_Solution;
```

end `GRASP`.

Bên dưới là giả mã của quá trình xây dựng cấu trúc tham lam thích nghi ngẫu nhiên. Với mỗi lần lặp lại giai đoạn này, tập các ứng cử viên được hình thành từ tất cả các phần tử có thể kết hợp với một giải pháp cụ thể mà không làm mất tính khả thi. Việc lựa chọn các phần tử tiếp theo để tổng hợp được xác định bởi việc đánh giá tất cả các ứng cử viên theo một hàm đánh giá tham lam. Hàm tham lam này thường đại diện cho chi phí do sự kết hợp của các ứng cử viên này vào các giải pháp được xây dựng. Việc đánh giá các phần tử này bằng hàm đánh giá dẫn đến việc tạo ra một danh sách ứng cử viên bị hạn chế (RLC viết tắt của Restricted Candidate List) bởi các phần tử tốt nhất. Các phần tử kết hợp với giải pháp cụ thể hiện tại cho giá

trị hàm giá tăng lên nhỏ nhất (đó là tính tham lam của thuật toán). Những phần tử này được lựa chọn ngẫu nhiên từ danh sách ứng cử viên hạn chế (đây là tính xác xuất hay ngẫu nhiên của thuật toán). Khi có một ứng cử viên được lựa chọn, danh sách ứng cử viên sẽ được cập nhật, giá trị gia tăng của hàm giá cũng được đánh giá lại (đây là tính thích nghi của thuật toán). Chiến lược này cũng tương tự như chiến lược bán tham lam của Hart và Shogan (1987), cũng là một giải pháp đa bắt đầu dựa trên cấu trúc tham lam ngẫu nhiên, nhưng không có tìm kiếm địa phương.

procedure `Greedy_Randomized_Contruction(Seed)`

1: `Solution` $\leftarrow \emptyset$;
2: Evaluate the incremental cost of candidate elements;
3: **while** `Solution` is not a complete solution **do**
4: Build the restricted candidate list (RCL);
5: Select an elment s from the RCL at random;
6: `Solution` $\leftarrow$ `Solution` $\cup\{s\}$;
7: Reevaluate the incremental costs;
8: **end while**
9: **return** `Solution`;

end `Greedy_Randomized_Contruction.`

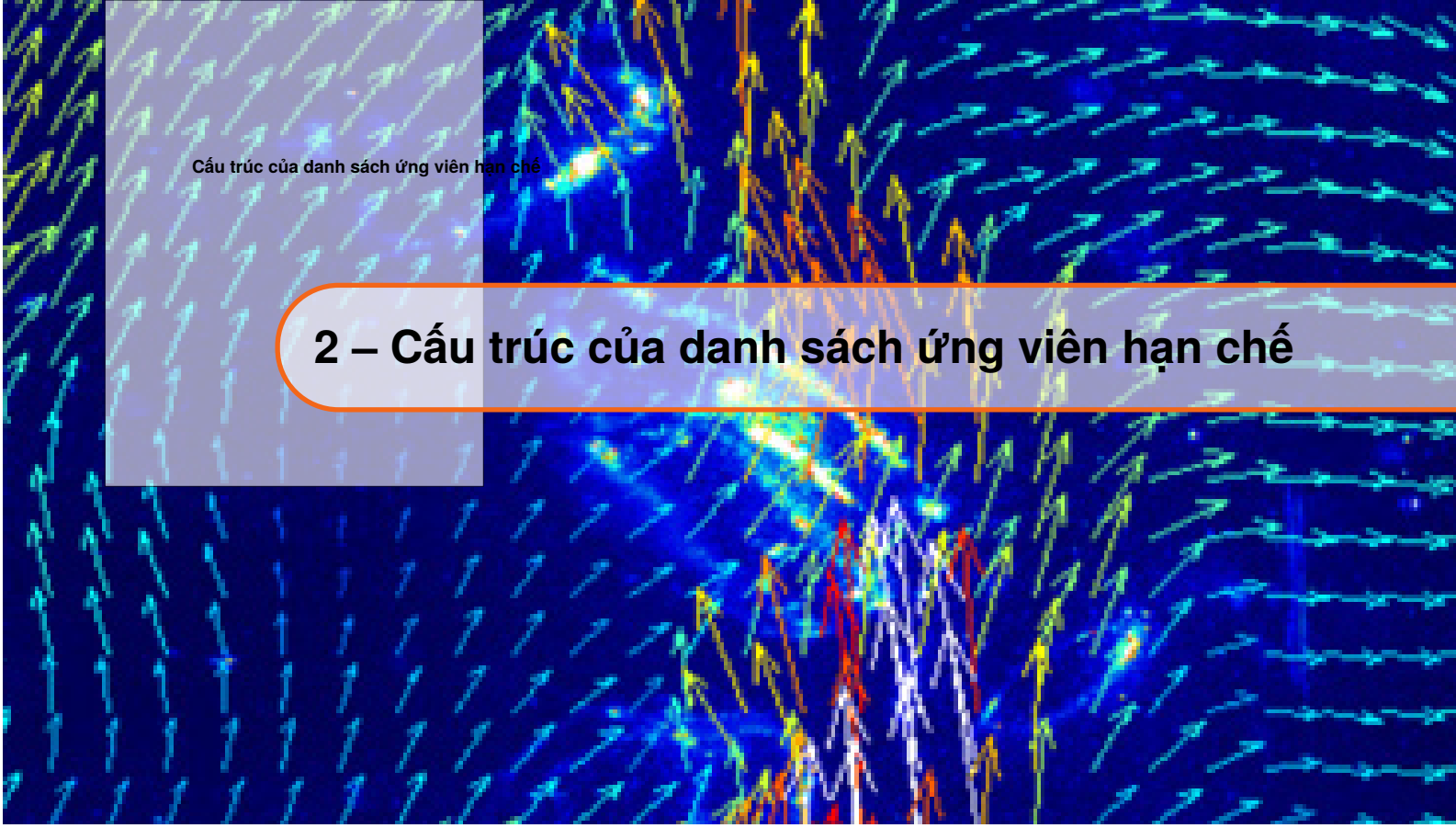
Các giải pháp được sinh ra từ cấu trúc tham lam ngẫu nhiên không nhất thiết phải tối ưu, thậm chí là tối ưu đơn giản. Giai đoạn tìm kiếm địa phương thường sẽ cải thiện lại giải pháp đó. Một giải thuật tìm kiếm địa phương làm việc trong một bước lặp bằng cách liên tục thay thế giải pháp hiện tại bằng giải pháp lân cận tốt hơn của giải pháp hiện tại. Giả mã của thuật toán tìm kiếm địa phương bắt đầu từ giải pháp Solution của giai đoạn đầu tiên và sử dụng một N - hàng xóm được giữ như giả mã ở trên.

procedure `Local_Search(Solution)`

1: **while** `Solution` is not locally optimal **do**
2: Build the restricted candidate list (RCL);
3: Find $s' \in N($`Solution`$)$ with $f(s') < f($`Solution`$)$;
4: `Solution` $\leftarrow s'$;
5: **end while**
6: **return** `Solution`;

end `Local_Search.`

Hiệu quả của một thủ tục tìm kiếm địa phương phụ thuộc vào một số khía cạnh như: cấu trúc của hàng xóm, kĩ thuật tìm kiếm hàng xóm, đánh giá nhanh của hàm giá của hàng xóm, và giải pháp bắt đầu của nó. Giai đoạn xây dựng giải pháp đóng vai trò quan trọng đối với các khía cạnh cuối cùng, xây dựng giải pháp bắt đầu có chất lượng cao cho tìm kiếm địa phương. Phương pháp hàng xóm đơn giản (simple neighborhoods) thường được sử dụng. Việc tìm kiếm hàng xóm có thể được thực hiện bằng cách sử dụng một cải tiến tốt nhất (best-improving) hoặc chiến lược cải tiến đầu tiên (first-improving). Trong trường hợp của chiến lược cải tiến tốt nhất, tất cả các hàng xóm được điều tra và hàng xóm tốt nhất được sử dụng. Trong trường hợp của chiến lược cải tiến đầu tiên, giải pháp hiện tại được chuyển tới giải pháp hàng xóm đầu tiên có giá trị hàm giá nhỏ hơn giá trị hiện tại. Trong thực hành, khi quan sát nhiều ứng dụng của cả hai chiến lược đều đưa tới cùng kết quả cuối cùng, nhưng chiến lược cải tiến đầu tiên thường cho thời gian tính toán nhỏ hơn.



2 – Cấu trúc của danh sách ứng viên hạn chế

2.1 Cấu trúc của danh sách ứng viên hạn chế

Một tính chất đặc biệt của GRASP là nó dễ dàng được thực hiện và cài đặt. Vài thông số cần phải được thiết lập và điều chỉnh. Vì vậy, phát triển thể tập trung thực hiện các cấu trúc dữ liệu hiệu quả để đảm bảo lặp lại nhanh chóng. GRASP có hai thông số chính: một liên quan đến các tiêu chí dừng và một liên quan tới chất lượng của các yếu tố trong danh sách ứng cử viên bị hạn chế. Tiêu chí để dừng giải thuật được sử dụng trong giả mã thủ tục GRASP ở chương 1 được xác định là số Max_Iterations các bước lặp. Mặc dù xác suất tìm được một giải pháp mới cải tiến tốt nhất với một số bước lặp, chất lượng của giải pháp tốt nhất được tìm thấy có thể chỉ cải thiện với các lần lặp sau. Từ khi thời gian tính toán qua từng bước lặp không thay đổi nhiều, tổng thời gian tính toán có thể dự doán được là tăng tuyến tính tới số lần lặp. Do đó, với một số lượng lớn các bước lặp, thời gian tính toán sẽ lớn hơn và kết quả tìm kiếm cũng tốt hơn.
Đối với việc xây dựng danh sách ứng viên hạn chế RCL được sử dụng trong giai đoạn đầu tiên, không làm mất tính tổng quát, một vấn đề cực tiểu như là một công thức trong mục 1.1. Ký hiệu $c(e)$ là giá trị gia tăng liên quan tới sự kết hợp của phần tử $e \in E$ tới cấu trúc của giải pháp. Tại mỗi bước lặp của GRASP, đặt c^min và c^max là giá trị gia tăng nhỏ nhất và lớn nhất.

Danh sách ứng viên hạn chế RCL được tạo thành từ các phần tử $e \in E$ với giá trị gia tăng tốt nhất (i.e., nhỏ nhất) $c(e)$. Danh sách này có thể bị giới hạn bởi số lượng

các phần tử (cardinality-based) hoặc bởi chất lượng của chúng (value-based). Trong trường hợp đầu tiên, danh sách RCL được tạo thành từ p phần tử với giá trị gia tăng giá tốt nhất, p ở đây là một tham số. Các danh sách RCL được liên kết tới một tham số ngưỡng $\alpha \in [0,1]$. Danh sách ứng viên hạn chế được hình thành từ tất cả các phần tử *einE* có thể thêm vào cấu trúc của giải pháp cụ thể thêm vào danh sách cụ thể mà không làm mất tính khả thi của giải pháp đó và có chất lượng vượt trội giá trị ngưỡng, i.e., $c(e) \in [c^m in, c^m in + \alpha(c^m ax - c^m in)]$. Trường hợp $\alpha = 0$ tương ứng với giải thuật thuần tham lam. Giả mã phía dưới là một sàng lọc của giả mã cấu trúc tham lam thích nghi ngẫu nhiên trong thủ tục Greedy_Randomized_Contruction ở chương 1 và chỉ ra rằng tham số α điều khiển độ tham lam và tính ngẫu nhiên của thuật toán.

procedure `Greedy_Randomized_Contruction(`α`, Seed)`

1: `Solution` $\leftarrow \emptyset$;
2: Initialize the candidate set: $C \leftarrow E$;
3: Evaluate the incremental cost $c(e)$ for all $e \in C$;
4: **while** $C \neq \emptyset$ **do**
5: $c^{\min} \leftarrow \min\{c(e) \,|\, e \in C\}$
6: $c^{\max} \leftarrow \max\{c(e) \,|\, e \in C\}$
7: **RCL** $\leftarrow \{e \in C \,|\, c(e) \leq c^{min} + \alpha(c^{\max} - c^{\min})\}$;
8: Select an element s from the RCL at random;
9: `Solution` $\leftarrow$ `Solution` $\cup\{s\}$;
10: Update the candidate set C;
11: Reevaluate the incremental costs $c(e)$ for all $e \in C$;
12: **end while**
13: **return** `Solution`;

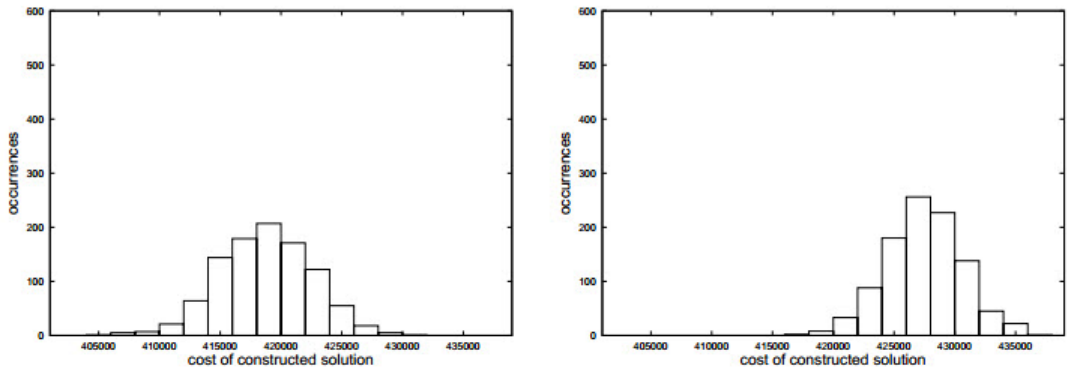
end `Greedy_Randomized_Contruction.`

GRASP có thể được xem như một kỹ thuật lấy mẫu lặp đi lặp lại. Mỗi bước lặp tạo ra một mẫu giải pháp từ một phân phối không biết, trung bình và phương sai là chức năng và hạn chế tự nhiên của danh sách RCL. Ví dụ, nếu RCL được hạn chế bởi một phần tử duy nhất, thì các giải pháp tương tự cũng sẽ được tạo ra ở tất cả các bước lặp. Phương sai của phân phối sẽ tới 0 và trung bình sẽ tới giá trị của giải pháp tham lam. Nếu danh sách RCL có nhiều phần tử hơn, nhiều giải pháp khác nhau sẽ được tạo ra, phương sai sẽ lớn hơn. Từ khi tham lam đóng vai trò nhỏ trong trường

hợp này, giá trị của giải pháp trung bình trở nên tồi hơn. Tuy nhiên, giá trị của giải pháp tốt nhất được tìm thấy nhanh hơn giá trị trung bình và thường là tối ưu. Biểu đồ trong hình 2.1 minh họa trường hợp này trong bài toán MAXSAT với 100 biến và 850 ràng buộc, kết quả thu được với 1000 cấu trúc đọc lập sử dụng quá trình đầu tiên của GRASP được mô tả ở Resende et al. (1997) và Resende et al. (2000). Từ đó, với trường hợp tìm cực đại, cấu trúc tham lam tương ứng với trường hợp $\alpha = 1$, trong khi cấu trúc ngẫu nhiên xảy ra với $\alpha = 0$. Chúng ta chú ý rằng whi giá trị của α là từ 0 tới 1, giá trị giải pháp trung bình tăng theo giá trị giải pháp tham lam khi phương sai dần tới 0.

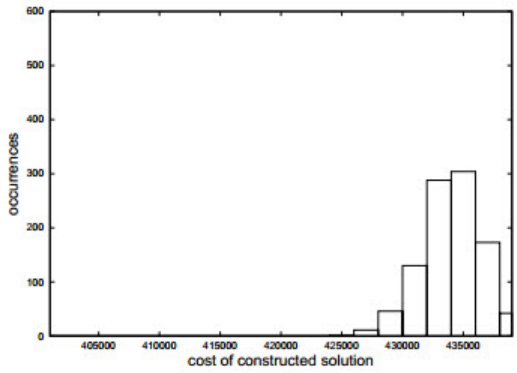
Với mỗi giá trị của α, hình 2.2 thể hiện kết quả thu được khi áp dụng tìm kiếm địa phương tới mỗi 1000 cấu trúc giải pháp. Hình 2.3 tóm tắt các kết quả của thí nghiệm này về sự đa dạng giải pháp, chất lượng và thời gian tính toán. Đầu tiên chúng ta nhận thấy rằng khi phương sai của các giá trị các giải pháp thu được trong gian đoạn đầu tiên càng lớn thì phương sai của tổng thể các giá trị giải pháp càng lớn, xem đồ thị trên. Đồ thị ở giữa minh họa mỗi quan hệ giữa phương sai của giá trị giải pháp và trung bình giá trị giải pháp, và điều này ảnh hưởng như thế nào tới giải pháp tốt nhất tìm được. Không thể chắc chắn rằng GRASP sẽ tìm được giải pháp tối ưu khi giá trị trung bình của giải pháp thấp, thận chí nếu có một phương sai lớn trong tổng thể các giá trị giải pháp, như trường hợp $\alpha = 0$. Trong trường hợp khác, khi giá trị phương sai nhỏ, GRASP cũng không chắc chắn có thể tìm được giá trị tối ưu, thậm chí khi giá trị trung bình của các giải pháp cao, như trường hợp $\alpha = 1$. Một giải pháp tốt thường có giá trị trung bình giải pháp cao trong một hiện diện phương sai tương đối lớn, ví dụ: $\alpha = 0.8$. Đồ thị này cũng chỉ ra rằng khoảnh cách giữa giá trị giải pháp trung bình và giá trị tốt nhất tìm được tăng khi tăng độ tham lam và tính ngẫu nhiên của quá trình xây dựng cấu trúc tham lam ngẫu nhiên thích nghi. Điều này làm cho thời gian chung bình dành cho việc tìm kiếm địa phương tăng lên. giống như đồ thị ở phía dưới. Bình thường, rất nhiều giải pháp GRASP được tạo ra trong cùng một lượng thời gian cần thiết cho thủ tục tối ưu địa phương hội tụ từ một vị trí bắt đầu ngẫu nhiên duy nhất.

Các kết quả được minh họa trong hình 2.4 và hình 2.5, về một trường hợp MAXSAT khác với 1000 bước lặp đã chạy. Với mỗi giá trị của α trong khoảng từ 0 tới 1, ta

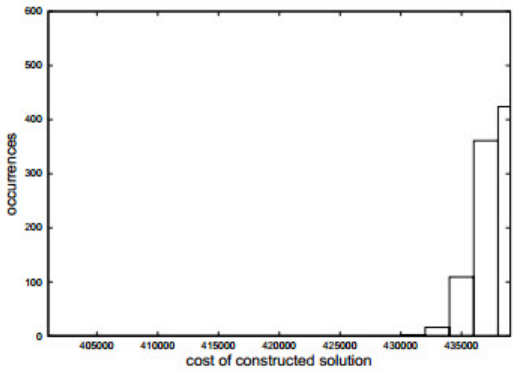


(a) RCL parameter alpha = 0.0 (random construction)

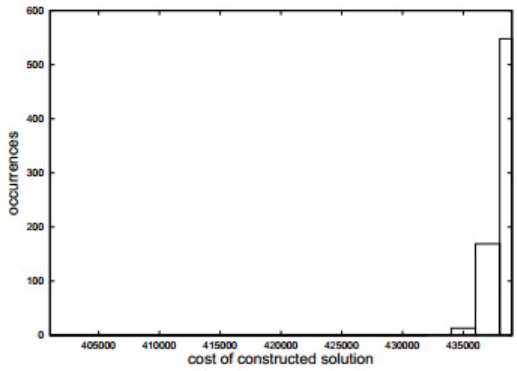
(b) RCL parameter alpha = 0.2



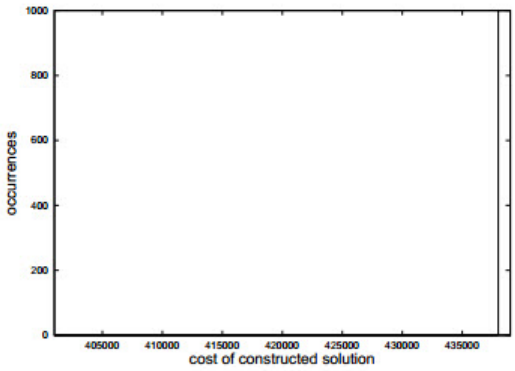
(c) RCL parameter alpha = 0.4



(d) RCL parameter alpha = 0.6

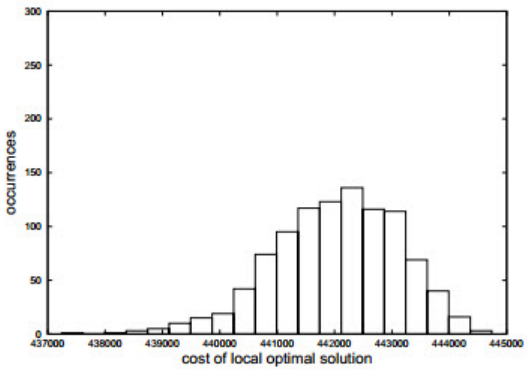


(e) RCL parameter alpha = 0.8

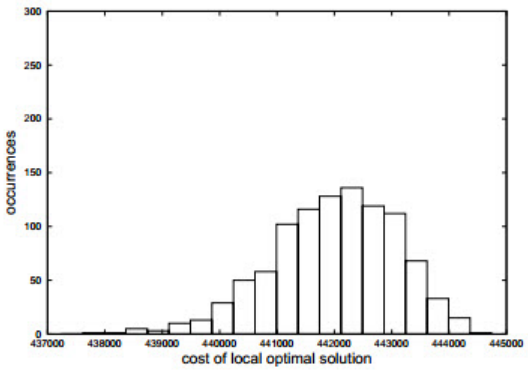


(f) RCL parameter alpha = 1.0 (greedy construction)

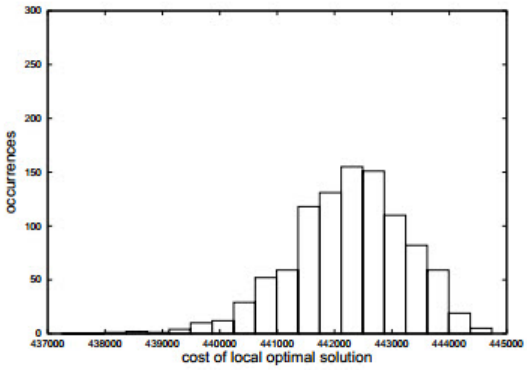
Hình 2.1: Phân phối giá trị giải pháp của quá trình cấu trúc như một hàm của tham số danh sách RCL α (1000 lần lặp cho mỗi giá trị α)



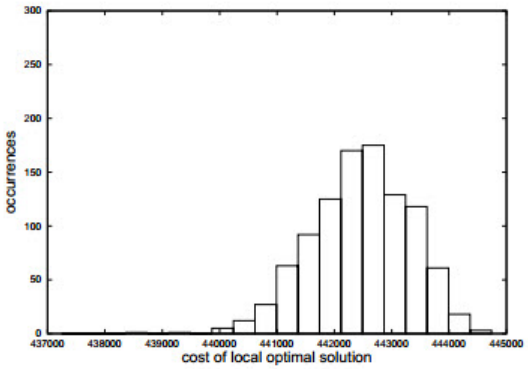
(a) RCL parameter alpha = 0.0 (random)



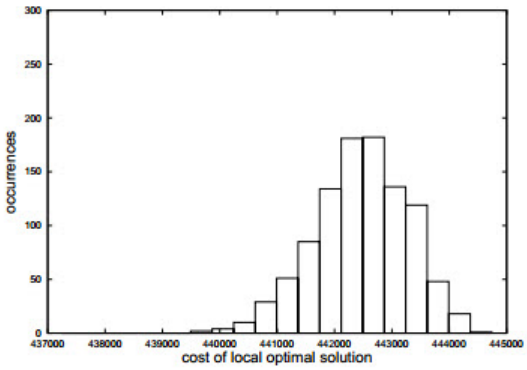
(b) RCL parameter alpha = 0.2



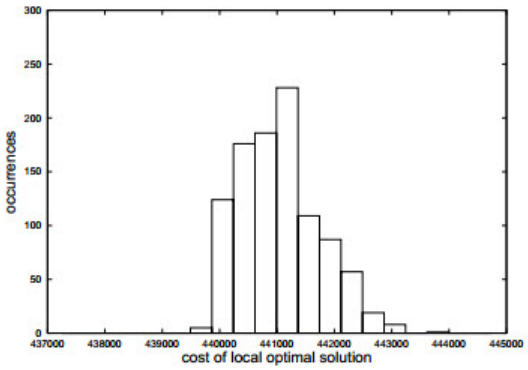
(c) RCL parameter alpha = 0.4



(d) RCL parameter alpha = 0.6

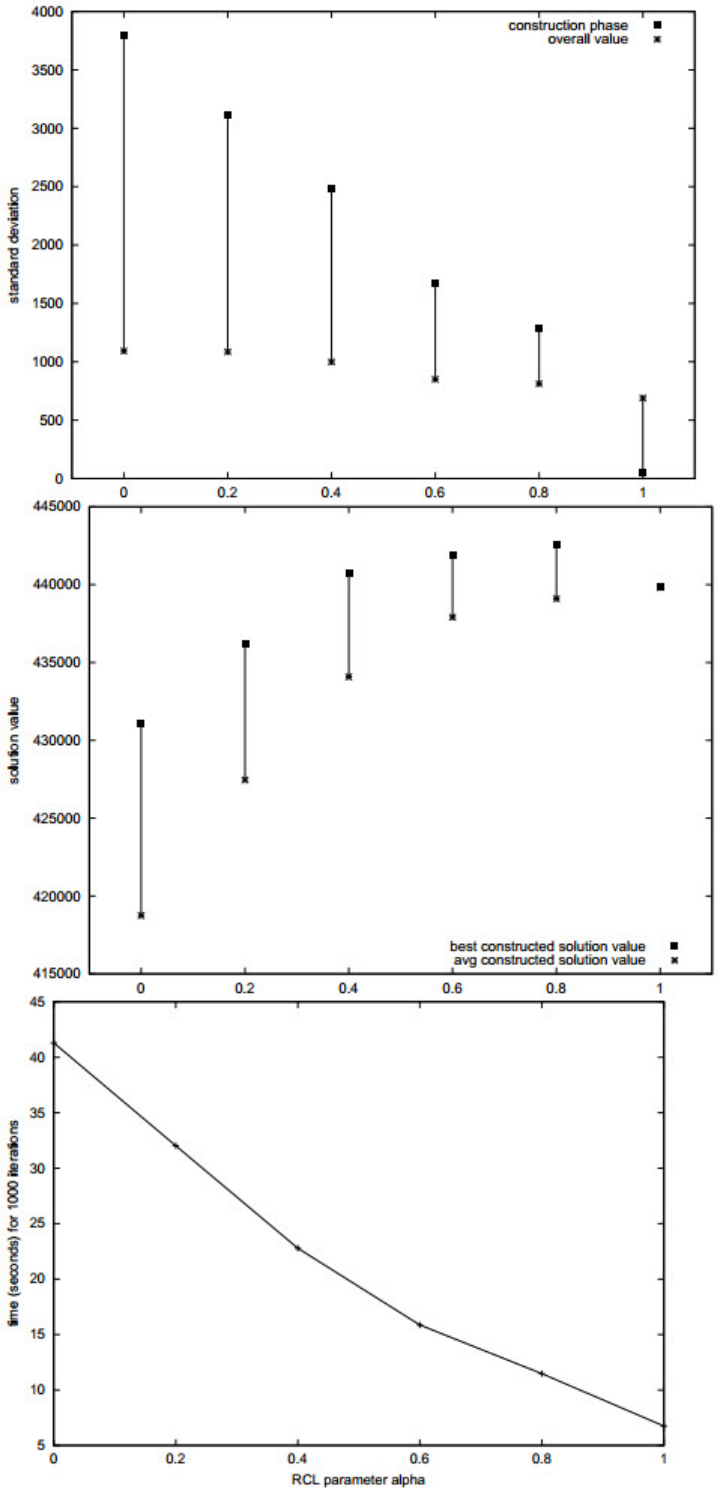


(e) RCL parameter alpha = 0.8



(f) RCL parameter alpha = 1.0 (greedy)

Hình 2.2: Phân phối giá trị giải pháp của quá trình tìm kiếm địa phương như một hàm của tham số danh sách RCL α (1000 lần lặp cho mỗi giá trị α)

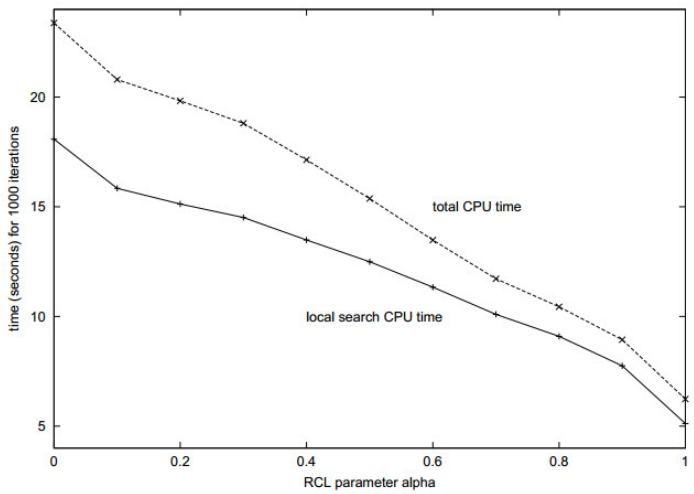


Hình 2.3: Độ lệch chuẩn của các giải pháp tìm được, giải pháp tốt nhất và giải phải trung bình tìm được, tổng thời gian tìm được như một hàm của tham số RCL α (1000 lần lặp với mỗi giá trị α)

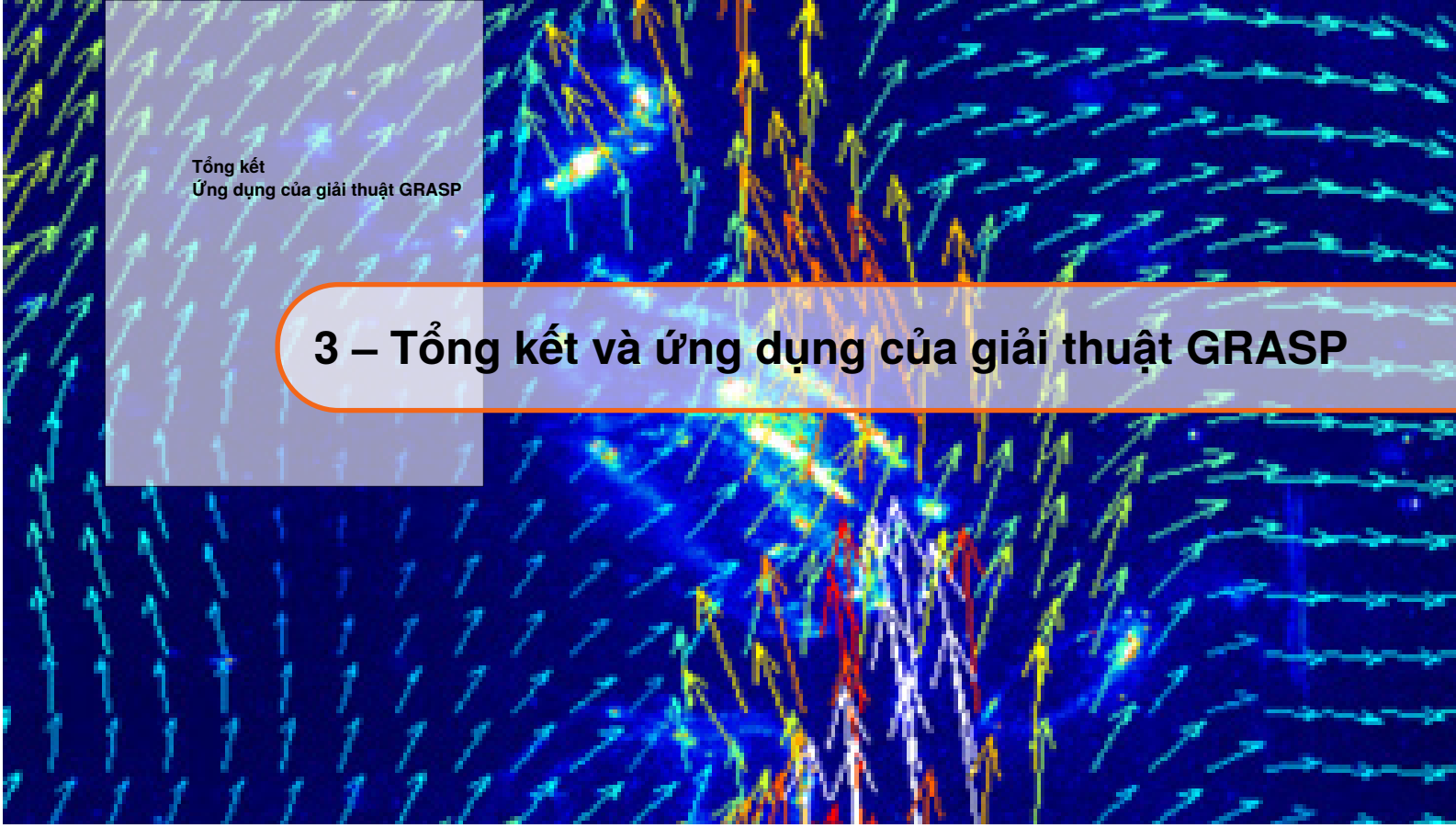
thu được hình 1.9, khoảng cách Hamming trung bình giữa mỗi giải pháp được xây dựng vào cuối giai đoạn đầu tiên và tương ứng với giá trị tối ưu địa phương thu được sau giai đoạn tìm kiếm địa phương. Số trung bình các bước chuyển từ đầu tiên tới cuối cùng. Hình 1.10 tóm tắt giá trị quan sát cho tổng thời gian xử lý và tổng thời gian tìm kiếm. Chú ý rằng, cả hai thời gian đều giảm khi α dần tới 1, tiếp cận tới lựa chọn tham lam. Đặc biệt, thời gian tìm kiếm địa phương khi $\alpha = 0$ gấp khoảng 2,5 lần trong trương hợp $\alpha = 0.9$. Lựa chọn giá trị của tham số RCL α rất quan trọng vì nó liên quan tới việc đạt được một sự cân bằng tốt giữa thời gian tính toán và chất lượng của giải pháp.

α	avg. distance	avg. moves	local search time (s)	total time (s)
0.0	12.487	12.373	18.083	23.378
0.1	10.787	10.709	15.842	20.801
0.2	10.242	10.166	15.127	19.830
0.3	9.777	9.721	14.511	18.806
0.4	9.003	8.957	13.489	17.139
0.5	8.241	8.189	12.494	15.375
0.6	7.389	7.341	11.338	13.482
0.7	6.452	6.436	10.098	11.720
0.8	5.667	5.643	9.094	10.441
0.9	4.697	4.691	7.753	8.941
1.0	2.733	2.733	5.118	6.235

Hình 2.4: Trung bình số bước di chuyển và thời gian tìm kiếm như một hàm của tham số α



Hình 2.5: Tổng thời gian CPU và thời gian CPU cho tìm kiếm địa phương như một hàm của tham số RCL α (1000 lần lặp với mỗi giá trị α)



3 – Tổng kết và ứng dụng của giải thuật GRASP

3.1 Tổng kết

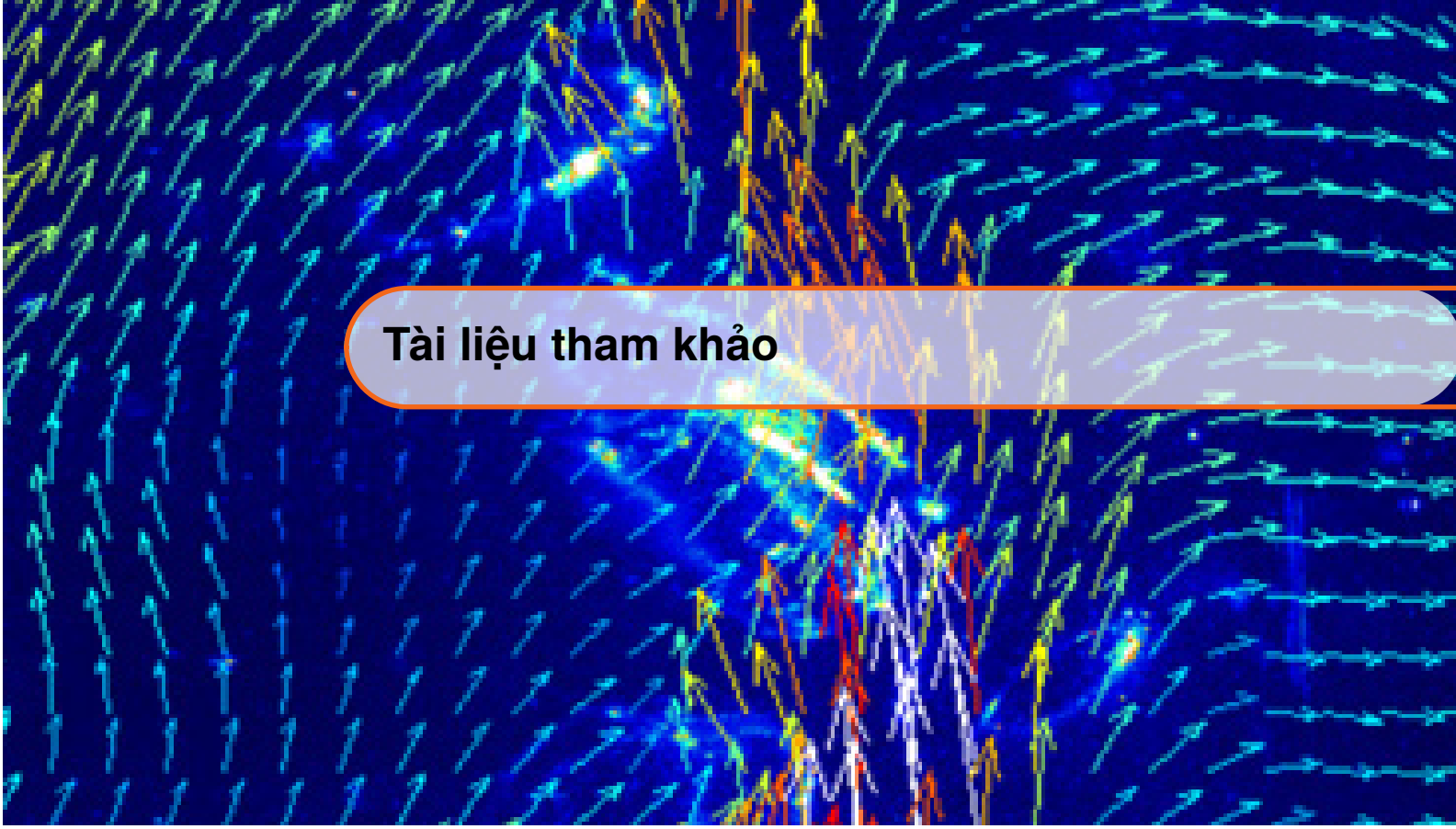
Các kết quả được mô tả ở chương trên phản ánh mô tả thành công của GRASP tới một số lượng lớn các lớp bài toán tối ưu tổ hợp cổ điển cũng như các vấn đề thực tiễn trong các lĩnh vực khác nhau của kinh doanh, khoa học và kỹ thuật.
Chương trình GRASP được xây dựng đơn giản bằng việc xây dựng hai khối đơn giản GRASP: thủ tục xây dựng giải pháp và phương pháp tìm kiếm địa phương, thường đã có sẵn. Trái với những gì đã xảy ra với các phương pháp metaheuristics khác như Tabu search, hoặc các giải thuật sinh học, với việc sử dụng một lượng lớn các tham số, phiên bản cơ bản của GRASP chỉ yêu cầu điều chỉnh duy nhất một tham số.
Gần đây, giải thuật GRASP cơ bản đã được cải tiến rất nhiều và cho phép cải thiện hơn nữa các giải pháp tìm được như các giải thuật React GRASP, tự động điều chỉnh tham số của danh sách ứng viên hạn chế, biến hàng xóm.

3.2 Ứng dụng của giải thuật GRASP

Ứng dụng đầu tiên của GRASP được mô tả trong các tài liệu về vấn đề tập phủ của Feo và Resende (1989). Một số ứng dụng chính của GRASP trong các lĩnh vực khác nhau:

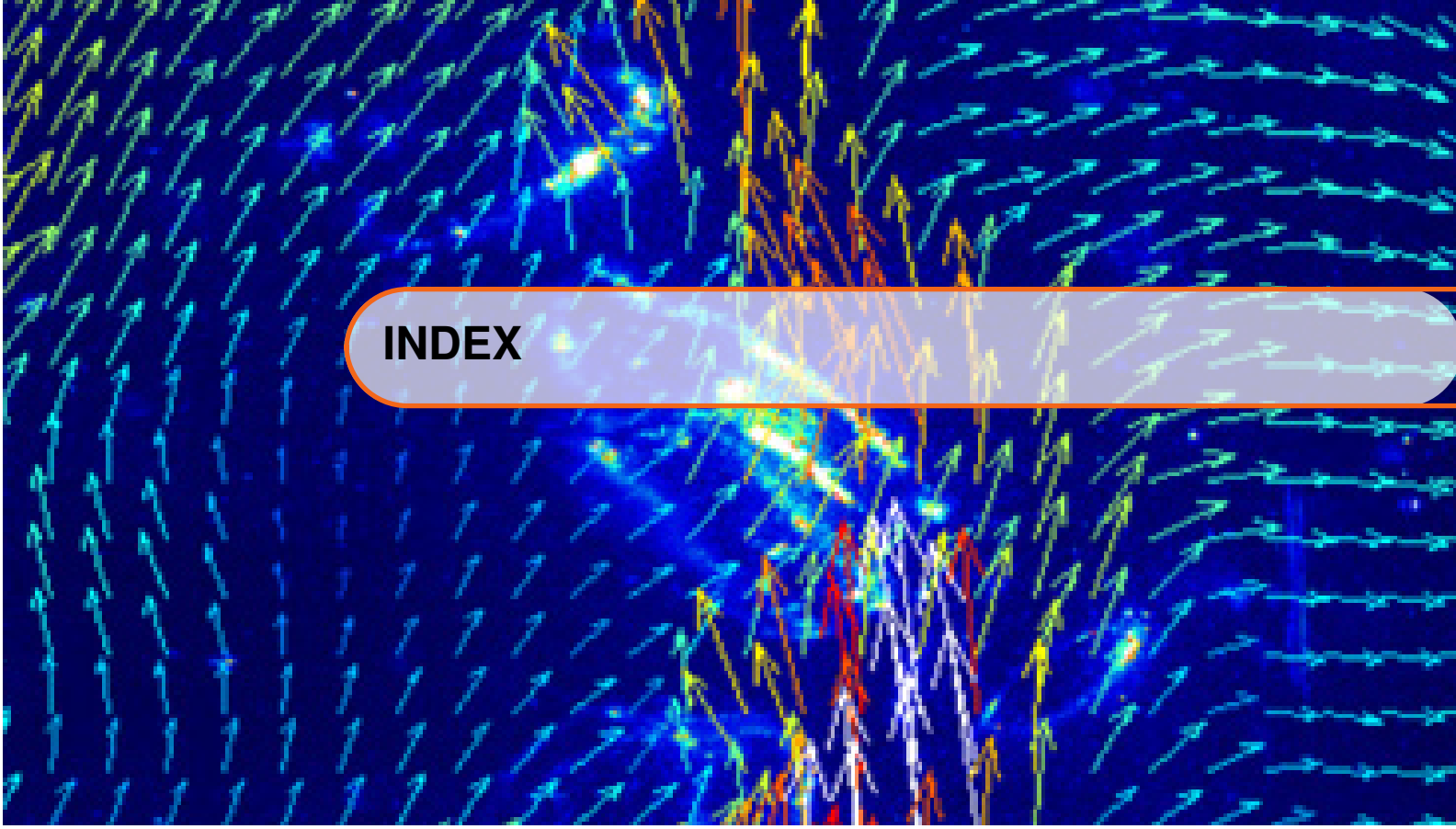
- Định tuyến: Arguello et al. (1997); Atkinson (1998); Bard et al. (1998); Carreto và Baker (2002); Kontoravdis and Bard (1995);

- Logic: Deshpande and Triantaphyllou (1998); Pardalos et al. (1996); Resende và Feo (1996); Resende et al. (1997);
- Phủ và phân vùng: Areibi and Vannelli (1997); Arguello et al. (1996); Feo và Resende (1989); Ghosh (1996); Hammer and Rader, Jr. (2001);
- Vị trí (location): Abdinnour-Helm and Hadley (2000); Delmaire et al. (1999); Klincewicz (1992); Urban (1998); Urban et al. (2000);
- Cực tiểu (minimum): Steiner tree Canuto et al. (1999); Martins et al. (1999; 2000; 1998); Ribeiro et al. (2002);
- Tối ưu đồ thị: Abello et al. (1999); Feo et al. (1994); Laguna et al. (1994); Pardalos et al. (1999); Resende (1998); Resende and Ribeiro (1997); Ribeiro and Resende (1999);



Tài liệu tham khảo

[1] Mauricio G. C. Resende, AT&T Labs – Research, Florham Park, USA “An introduction to GRASP”, 2007



INDEX