Lời cảm ơn

Với những kiến thức tích lũy được trong suốt thời gian học tập, nghiên cứu và thực hiện bài tập lớn tại Phòng thí nghiệm bộ môn Khoa học Máy tính, Đại học Bách Khoa Hà Nội, chúng em đã nhận được sự quan tâm giúp đỡ nhiệt tình của các thầy cô, các anh chị nghiên cứu sinh và sinh viên trong phòng thí nghiệm. Đến nay, nhóm em đã hoàn thành Báo cáo của mình. Với sự trân trọng chúng em xin chân thành cảm ơn đến PGS. TS. Huỳnh Thị Thanh Bình - Giảng viên cao cấp Bộ môn Khoa học Máy tính, Phó viện trưởng Viện Công nghệ thông tin và Truyền thông, trường Đại học Bách Khoa Hà Nội - người đã trực tiếp hướng dẫn chúng em trong suốt thời gian hoàn thành đồ án. Chúng em cũng xin gửi lời cảm ơn chân thành đến nghiên cứu sinh Phan Đình Thành và các anh chị nghiên cứu sinh, sinh viên trong phòng thí nghiệm Bộ môn Khoa học Máy tính đã tận tình cung cấp tài liệu cần thiết và đóng góp nhiều ý kiến quý báu để chúng em hoàn thành bài tập lớn này.

Một lần nữa chúng em xin chân thành cảm ơn và kính chúc PSG. TS. Huỳnh Thị Thanh Bình được dồi dao sức khỏe và công tác tốt. Kính chúc Quý Bộ môn đạt được nhiều thành công trong công tác nghiên cứu và giảng dạy.

Lời nói đầu

Trong khoa học máy tính, tối ưu hóa bầy kiến (Ant Colony Optimization) là một kĩ thuật dựa trên xác suất dùng để giải các bài toán tối ưu tổ hợp mà có thể giản lược được nhiều phương án để tìm ra đường đi tốt trong đồ thị. Những thuật toán này thuộc họ thuật toán bầy kiến (Ant Colony Algorithm) và là một trong các phương pháp trí tuệ bầy đàn (Swarm Intelligence) và chúng tạo nên một số các tối ưu hóa metaheuristic. Các thuật toán ACO được áp dụng vào giải rất nhiều bài toán tối ưu hóa tổ hợp, trải dài từ bài toán quy hoạch toàn phương (Quadratic Assignment Problems) cho tới các bài toán định tuyến phương tiện giao thông (routing vehicles). Hơn nữa, một số lượng lớn các phương pháp dẫn xuất của nó được điều chỉnh để giải các bài toán với các số thực, các bài toán có yếu tố ngẫu nhiên, bài toán triển khai song song đa mục tiêu. Các thuật toán tối ưu hóa bầy kiến này có nhiều ưu điểm vượt trội hơn so với phương pháp tiếp cận theo mô phỏng luyện kim (Simulated Annealing) hay giải thuật di truyền (Genetic Algorithm) trong một số bài toán tìm kiếm trên đồ thị khi đồ thị thay đổi liên tục theo thời gian thực. Sự thay đổi này thích hợp trong các bài toán định tuyến trong mạng và hệ thống giao thông.

Năm 1992, khởi đầu với sự đề xuất với hệ thống các con kiến (Ant System) bởi Marco Dorigo trong luận án tiến sĩ của mình, sau đó ông và cộng sự của mình – Gambardella – đã phát triển tiếp thuật toán bầy kiến (Ant Colony System) áp dụng vào giải bài toán người du lịch (Travelling Salesman Problem) vào năm 1997. Điều này đã đặt một dấu mốc quan trọng cho sự phát triển của lớp các thuật toán tối ưu hóa bầy kiến được cải tiến và chỉnh sửa về sau. Sau đó, có rất nhiều bài toán tối ưu hóa đã được giải để tìm lời giải gần đúng, đủ tốt trong khoảng thời gian chấp nhận được bằng các thuật toán này, có thể kể đến như các thuật toán xử lý ảnh số, tìm biên của vật thể trong ảnh – vốn được sử dụng bởi các phần mềm thị giác máy tính trong trí tuệ nhân tạo.

Để tiếp cận với lớp giải thuật khá mới mẻ này, chúng em quyết định nghiên cứu và tìm hiểu công bố khoa học của Dorigo và Gambardella vào năm 1997 có tựa đề “Ant colonies for the travelling salesman problem” với giải thuật cơ bản nhất Ant Colony System (ACS) trong lớp các thuật toán này. Tuy nhiên, trong công bố của tác giả, không nêu ra một chỉ dẫn hay một cách cài đặt thuật toán cụ thể để giải một bài toán cụ thể. Do đó, chúng em tìm hiểu và tham khảo thêm về cách cài đặt thuật toán ACS ở cuốn sách “Clever Algorithms Nature-Inspired Programming Recipes” của Jason Brownlee như một chỉ dẫn căn bản dễ hiểu nhất cho người mới làm quen với thuật toán ACS nói riêng và các thuật toán dựa trên tiến hóa, tự nhiên nói chung.

Do vậy, báo cáo của chúng em trình bày sẽ bao gồm hai nội dung chính:

Phần 1, chúng em nêu lên nội dung tóm tắt về công bố khoa học năm 1997 của tác giả về ACS.

Phần 2, chúng em nêu lên cách thức cài đặt, thực thi và tổng hợp kết quả thực nghiệm, nhận xét, kết luận và hướng phát triển tiếp theo trong lĩnh vực này.

Chương 1. Công bố khoa học

1.1. Đôi nét về tác giả

Marco Rodigo (sinh ngày 26 tháng tám năm 1961 tại Milan, Ý) hiện là Giám đốc Quỹ đầu tư cho Nghiên cứu học Bỉ và là giảm đốc của IRIDIA, phòng thí nghiệm trí tuệ nhân tạo của Đại học Tự do Bruc-xen (Université Libre de Bruxelles), Bỉ. Ông là người đề xuất giải thuật tối ưu hóa bầy kiến (xem trong cuốn sách của ông được xuất bản bởi nhà xuất bản MIT Press năm 2004) và là một trong các nhà sáng lập của ngành nghiên cứu về giải thuật trí tuệ bầy đàn. Rodigo cũng tham gia vào nghiên cứu trong lĩnh vực robot bầy đàn: hiện ông cộng tác trong dự án Swarmbots được rót vốn bởi Chương trình Công nghệ của Tương lai và Phát minh thuộc Ủy ban Châu Âu EC trong việc triển khai các robot hình người có khả năng tự sản xuất và tự lắp ráp. Hơn nữa, Rodigo cũng là tổng biên tập cho tờ “Swarm Intelligence”, một tạp chí khoa học về lĩnh vực ông nghiên cứu để đóng góp các công bố khoa học cho các phát triển mới cũng trong lĩnh vực đa ngành này.

Với những đóng góp lớn lao của mình, vào năm 2003, Rodigo được trao giải thưởng Marie Curie cho sự xuất sắc về nghiên cứu bởi Ủy ban Châu Âu EC; ngày 22 tháng mười một năm 2005, ông được đề cử giải thưởng Dr A. De Leeuw-Damry-Bourlart trong lĩnh vực khoa học ứng dụng bởi nhà vua Bỉ Albert II; năm 2007, ông nhận được giải thưởng quốc tế Cajastur trong lĩnh vực phần mềm tính toán, được trao bởi trung tâm Châu Âu về tính toán mềm; và trong năm 2015, ông tiếp tục nhận giải thưởng IEEE Frank Rosenblatt.

1.2. Nội dung bài báo

1.2.1. Tóm tắt

Nhóm tác giả mô tả một tổ kiến (bầy kiến) nhân tạo có khả năng giải bài toán người du lịch (Travelling Salesman Problem). Những con kiến của tổ kiến này có khả năng tạo ra các đường đi ngẵn và tối ưu bằng việc sử dụng các thông tin được tích lũy dưới dạng các vệt mùi của đồng loại để lại trên các cạnh của đồ thị TSP. Mô phỏng của mô hình này trên chương trình máy tính cho thấy tổ kiến nhân tạo này có thể tạo ra các lời giải đủ tốt cho cả các thể hiện của bài toán TSP đối xứng và bất đối xứng. Phương pháp tiếp cận này như một ví dụ cụ thể, cũng giống như mô phỏng luyện kim (simulated annealing), mạng nơ-ron nhân tạo và giải thuật di truyền đều là các sự áp dụng thành công của các hiện tượng trong tự nhiên vào để thiết kế môt giải thuật tối ưu hóa.

1.2.2. Mở đầu

Những con kiến trong tự nhiên có khả năng tìm ra một con đường đi ngắn nhất từ nguồn thức ăn tới tổ của chúng mà không cần sử dụng các thị giác hay các dấu hiệu trực quan. Hơn nữa, chúng cũng có thể thích nghi được với những thay đổi của môi trường, ví dụ như tìm một con đường ngắn nhất mới khi con đường tối ưu cũ bị chặn, không thể dùng được do có chướng ngại vật. Do đó, việc các loài côn trùng như kiến sử dụng mùi hương đồng loại làm phương tiện trao đổi và liên lạc trong bầy đàn là vô cùng phổ biến. Kiến nhả ra một lượng mùi nhất định trong khi di chuyển và mỗi con kiến trong số đó đều có một xác suất nào đó trong việc chọn một con đường có vệt mùi do đồng loại nhả ra. Hành vi nhỏ nhưng cơ bản này trong các bầy đàn côn trùng, đặc biệt là bầy kiến có thể được sử dụng để giải thích bằng cách nào mà chúng có thể tìm được một con đường khác cũng tối ưu kết nối lại thay cho con đường trước bị chặn bởi sự thay đổi của môi trường (vật cản không mong muốn xuất hiện đột ngột). Trong thực tế, một khi chướng ngại vật xuất hiện, những con kiến ở ngay trước vật cản không thể tiếp tục di chuyển theo vệt mùi nên chúng phải lựa chọn một hoặc rẽ phải hoặc rẽ trái để tránh vật cản. Trong tình huống này, chúng ta có thể giả sử rằng một nửa số con kiến sẽ đi sang phải và nửa kia đi hướng còn lại. Có một điều đáng chú ý rằng những con kiến mà may mắn chọn được đường đi ngắn hơn sẽ xây dựng lại được đường đi ngắn nhất nhanh hơn so với những cá thể kiến chọn đường đi còn lại. Do vậy, đường ngắn hơn sẽ nhận được lượng mùi lớn hơn trong cùng một đơn vị thời gian dẫn đến nhiều con kiến phía sau sẽ chọn con đường đó và tiếp tục nhả mùi trên đó nhiều hơn. Nhờ có quá trình tự xúc tác tích cực như vậy, tất cả các con kiến sẽ nhanh chóng chọn được con đường ngắn nhất. Một chi tiết thú vị nhất của quá trình tự xúc tác này là việc tìm ra đường đi ngắn nhất xung quanh chướng ngại vật là một đặc điểm quan trọng trong mối quan hệ giữa hình dạng vật thể và hành vi phân tán của kiến: cho dù tất cả các con kiến trong bầy đều di chuyển với một tốc độ xấp xỉ tương đương nhau và đều nhả ra một lượng mùi như nhau trong cùng một đơn vị thời gian, và hiển nhiên là sẽ mất nhiều thời gian hơn để đi vòng qua phần cạnh dài hơn của chướng ngại vật so với phần cạnh ngắn hơn – vốn được đi qua nhanh hơn với lượng tích lũy mùi đậm đặc hơn. Sự lựa chọn của những con kiến vào những cạnh ngắn hơn này làm cho cạnh đó ngày càng tích lũy được nhiều hơn nữa. Nhóm tác giả lúc này sẽ trình bày làm thể nào để một quá trình tương tự có thể hoạt động được trên chương trình máy tính mô phỏng lại bầy kiến và dùng nó để giải bài toán người du lịch.

Bài toán người du lịch (TSP) là bài toán tìm một đường đi có thứ tự sao cho (chu trình của đồ thị) trên đó, chi phí là nhỏ nhất. Trong công bố này, nhóm tác giả sẽ giới hạn các ràng buộc của bài toán TSP để dễ thử nghiệm. Đồ thị TSP có các đỉnh nằm trên cùng một mặt phẳng và giữa mỗi cặp đỉnh phân biệt đôi một với nhau, đều có một cạnh vô hướng nối giữa chúng (đồ thị TSP là liên thông đầy đủ, mạnh).

1.2.3. Các cá thể kiến nhân tạo

Trong bài báo, một cá thể kiến nhân tạo là một tác tử mà nó có thể di chuyển từ thành phố này sang thành phố khác trong một đồ thị TSP. Nó chọn thành phố tiếp theo để di chuyển tới bằng cách sử dụng một hàm xác suất liên quan tới cả lượng mùi được tích lũy trên đường đi và một giá trị heuristic, giá trị đó được tính từ độ dài của cạnh đó. Những con kiến sẽ lựa chọn với xác suất cao những thành phố mà có cạnh nối tới nó ngắn hơn mà tích lũy được lượng mùi đậm hơn. Khởi đầu, m con kiến như vậy sẽ được đặt một cách ngẫu nhiên vào các thành phố trong đồ thị TSP. Với mỗi bước di chuyển, chúng di chuyển tới một thành phố mới mà chúng chưa đi qua và thay đổi lượng mùi trên cạnh mà chúng vừa sử dụng. Đó gọi là cập nhật mùi cục bộ (local trail updating). Còn khi tất cả các con kiến đã hoàn thành chu trình đường đi của mình thì chu trình ngắn nhất sẽ được con kiến mà nó tìm ra thay đổi giá trị mùi trên chu trình đó. Thao tác đó được gọi là cập nhật mùi toàn cục (global trail updating) bằng cách thêm một lượng mùi bằng nghịch đảo giá trị của tổng khoảng cách đường đi ngắn nhất đó.

Có ba ý tưởng chính mà tác giả sẽ áp dụng vào tổ kiến nhân tạo: (i) sự ưu tiên chọn đường đi với lượng mùi cao, (ii) độ tăng nhanh của mùi trêm các cạnh ngắn và (iii) sự giao tiếp của những cá thể thông qua vệt mùi đồng loại. Những con kiến nhân tạo trong mô hình này còn có thêm một vài khả năng như là chúng không có một đối trọng trong tự nhiên nhưng để thích hợp cho việc giải bài toán TSP thỏa mãn điều kiện rằng mỗi thành phố chỉ đi qua một lần: chúng có thể nhận biết được khoảng cách của thành phố tiếp theo và được trang bị một trí nhớ Mk dùng để nhớ các thành phố mà chúng đã đi qua (trí nhớ sẽ được làm rỗng mỗi khi chúng được thả vào đồ thị và sẽ được cập nhật thêm một thành phố mỗi khi chúng đi qua nó).

Có rất nhiều cách để có thể chuyển các quy tắc trên vào một hệ thống tính toán với nhiệm vụ giải bài toán TSP. Nhưng ở đây, với ACS, một con kiến thứ k tại thành phố r chọn thành phố s để đi tiếp trong số những thành phố mà không thuộc trí nhớ Mk bằng cách áp dụng công thức xác suất sau:

< công thức (1)>

Với τ(r, u) là lượng mùi trên cạnh (r, u), η(r, u) là hàm heuristic được tính dựa trên nghịch đảo khoảng cách giữa hai thành phố r và u, β là một tham số biểu diễn sự tương quan giữa lượng mùi và khoảng cách, q là một than số thực được chọn ngẫu nhiên trong đoạn [0, 1], q0 (0 ≤ q0 ≤ 1) là tham số heuristic tự chọn và S là một biến ngẫu nhiên được chọn từ phân phối xác suất sau, các cạnh ngắn hơn và có nhiều mùi hơn sẽ có xác suất được chọn cao hơn:

<công thức (2)>

Với pk(r, s) là xác suất mà con kiến thứ k ở thành phố r chọn thành phố s để di chuyển tới.

Lượng mùi được thay đổi cả cục bộ lẫn toán cục. Cập nhật toàn cục với mục đích gây sự chú ý của cạnh ngắn nhất tìm được tới các con kiến khác. Một khi tất cả các con kiến trong bầy đã hoàn thành chu trình di chuyển của mình, con kiến với đường đi ngắn nhất sẽ nhà một lượng mùi lên trên các cạnh mà nó đi qua; những cạnh mà thuộc vào chu trình của nó. (Các cạnh khác không thay đổi.) Lượng mùi được thêm vào Δτ(r, u) được nhả ra trên mỗi cạnh (r, s) bởi con kiến có chu trình ngắn nhất được tính theo nghịch đảo của độ dài chu trình: độ dài chu trình càng ngắn thì lượng mùi được thêm vào càng lớn. Quá trình cập nhật mùi cục bộ có công thức như sau: <ảnh 3>, với Δτ(r, u) = (chi phí ngắn nhất)­-1.

Quá trình cập nhật cục bộ với mục đích để tránh một cạnh rất đậm mùi được chọn bởi tất cả các con kiến: mỗi một cạnh khi được chọn để đi qua thì lượng mùi trên đó được thay đổi bằng cách áp dụng công thức sau: <ảnh 4> với τ0 là một tham số. Việc câp nhật cục bộ này được mô phỏng dựa trên việc bay hơi mùi theo thời gian của bầy kiến trong tự nhiên.

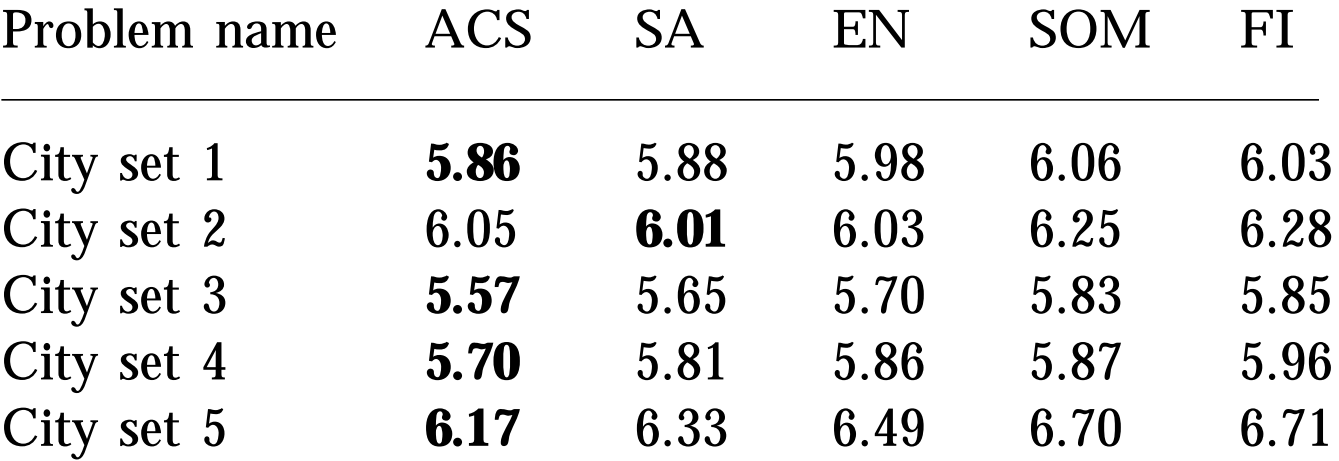
1.2.4. Kết quả

Nhóm tác giả đã áp dụng ACS vào giải các bài toán TSP đối xứng và bất đối xứng được liệt kê ở các bảng từ 1 đến 4 và bảng 7. Các bài toán giải thử nghiệm được chọn bởi vì chúng có dữ liệu sẵn có trong các tài liệu nghiên cứu liên quan để tiện so sánh các kết quả khi giải TSP đối xứng với các thuật toán dựa trên tự nhiên khác lẫn việc thể hiện khả năng giải các thể hiện khó của bài toán TSP bất đối xứng.

Với việc sử dụng các bài toán thử nghiệm trong các bảng 1 – 3, hiệu năng của ACS đã được so sánh với hiệu năng của các thuật toán tối ưu hóa dựa trên tự nhiên khác như: mô phỏng luyện kim (SA), các mạng nơ-ron (NNs), ở đây là mạng đàn hồi (EN) và bản đồ tự tổ chức (SOM), tính toán tiến hóa (EC), ở đây là áp dụng giải thuật di truyền (GA), lập trình tiến hóa (EP) và một phiên bản kết hợp giữa mô phỏng tôi luyện và giải thuật di truyền (AG); hơn nữa, nhóm tác giả cũng so sánh ACS với thuật toán heuristic chèn xa nhất (FI). Các số liệu thực nghiệm đã được thực hiện với ACS và FI, còn đối với số liệu cho các thuật toán còn lại thì được lấy ở các nghiên cứu liên quan trước đó. Các tham số cho ACS được đặt với các giá trị như sau: m =10, beta = 2, alpha = 0.1, q0 = 0.9, t0 = (n \* Lnn)^-1, ở đó với Lnn là độ dài của chu trình ngắn nhất tại thời điểm cập nhật và n là số lượng thành phố trong bài toán. Trong một số thử nghiệm (bảng 2), lời giải tối ưu được tìm ra bởi các giá trị heuristic tối ưu cục bộ 3-opt. Bảng kết quả cũng chỉ ra rằng ACS tìm được các kết quả mà ít nhất cũng tốt bằng, và thường là tốt hơn những kết quả được tìm ra bởi các thuật toán khác. Đương nhiên, các lời giải tối ưu được tìm bởi ACS trong bảng 2 được áp dụng phương pháp tối ưu cục bộ với 3-opt.

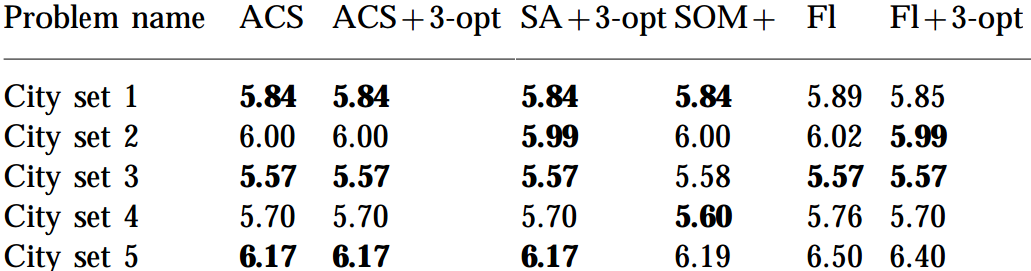
Nhóm tác giả cũng thực thi thuật toán trên một vài bài toán TSP có kích thước lớn hơn để tìm hiểu thêm về kết quả nếu ta áp dụng vào bộ dữ liệu lớn. Khi áp dụng, họ cũng triển khai một kiểu ACS có một chút sửa đổi. Họ thêm vào một cấu trúc dữ liệu có tên là danh sách ứng viên, một cấu trúc dữ liệu thường được sử dụng khi giải các bài toán TSP lớn (Reinelt, 1994; Johnson và McGeoch, 1997). Một danh sách ứng viên là một danh sách các thành phố được ưu tiên đi qua; nó là một cấu trúc dữ liệu tĩnh chứa các thông tin như: cho thành phố I, các thành phố cl được coi là các thành phố gần nhất. Trong thực thế, một con kiến trong ACS với một danh sách ứng viên sẽ ưu tiên chọn thành phố để di chuyển tới một trong số. Trừ khi tất cả các thành phố trong danh sách ứng viên không còn có thể đến được nữa thì các con kiến mới cân nhắc di chuyển tới các thành phố còn lại. Ở bảng 5 và 6, hiệu năng của ACS khi áp dụng với các kích cỡ khác nhau của danh sách ứng viên (ACS không sử dụng danh sách ứng viên tương ứng với ACS có sử dụng danh sách thành viên có kích cỡ cl = n). Kết quả có được khi áp dụng vào hai bài toán Eil51 và Pcb442 (cả hai bài toán này đều được lấy trên TSPLIB) chứng tỏ rằng một danh sách thành viên nhỏ có thể nâng cao cả hiệu năng trung bình và hiệu năng tốt nhất của ACS; hơn nữa, sử dụng một danh sách thành viên nhỏ cũng tốn ít thời gian của CPU để xây dựng một chu trình hơn là một danh sách dài. Những kết quả có trong bảng 4 được thực hiện trên danh sách có kích cỡ cl = 20. Hứa hẹn hơn nữa là các kết quả có được từ việc áp dụng ACS vào giải các bài toán TSP bất đối xứng (bảng 7). Ví dụ, ACS đã có thể tìm ra, trong vòng 220 giây với một chiếc máy tính cá nhân Pentium, lời giải tối ưu cho một bài toán TPS bất đối ứng với 43 thành phố tên là 42X3. Bài toán tương tự đã không thể giải được để tìm lời giải tối ưu trong khoảng thời gian 32 giờ tính toán trên máy trạm bởi mã nguồn tốt nhất dành cho các bài toán TSP bất đối xứng với một thuật toán chính xác dựa trên việc cắt đa diện (tư tưởng cắt tỉa nhánh-cận).

Bảng 1. Sự so sánh giữa ACS và các thuật toán được mô phỏng hiện tượng tự nhiên khi giải các thể hiện ngẫu nhiên của bài toán TSP



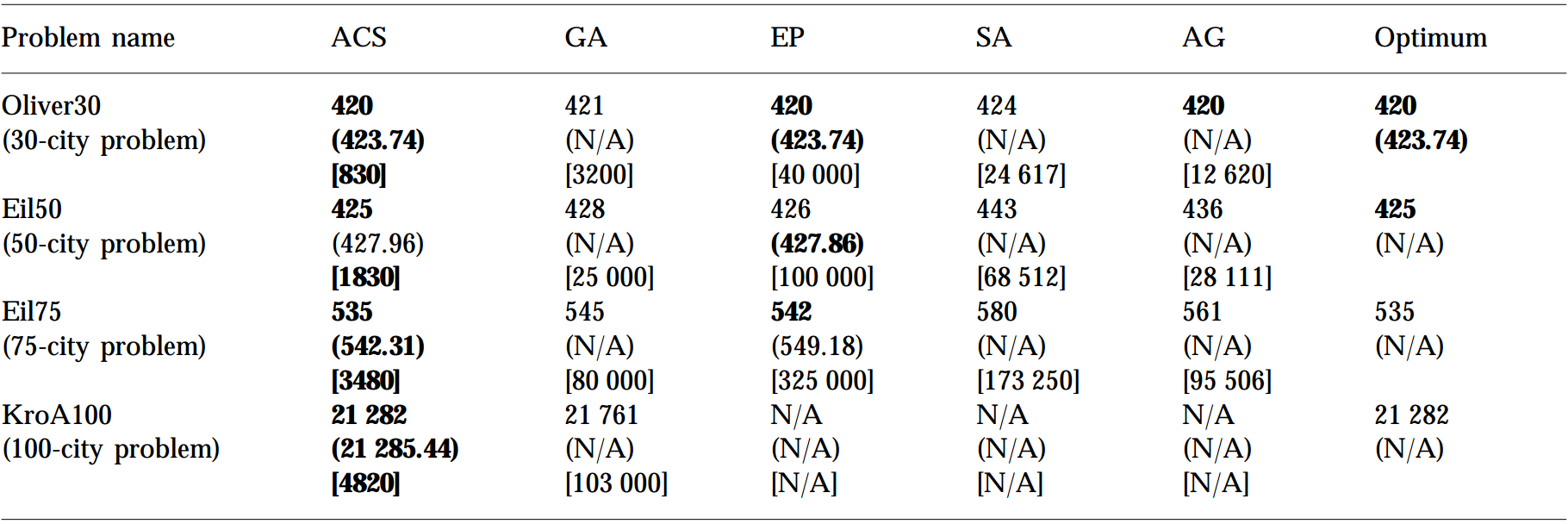
So sánh các độ dài chu trình trung bình của các bài toán có 50 thành phố. Kết quả của SA, EN và SOM được lấy của Durbin và Willshaw (1987) và Potvin (1993). Các kết quả của FI được lấy trung bình trên 15 lần thực thi, xuất phát từ các thành phố ngẫu nhiên. ACS được thực thi với 1250 thế hệ kiến, sử dụng m = 20 con kiến cho mỗi thế hệ và các kết quả được tính trung bình cho 15 lần thực thi của mỗi bài toán. Độ dài chu trình tốt nhất tính theo trung bình cho mỗi bài toán được in đậm.

Bảng 2. So sánh giữa ACS và các thuật toán mô phỏng tự nhiên trên các thể hiện ngẫu nhiên của bài toán TSP đối xứng



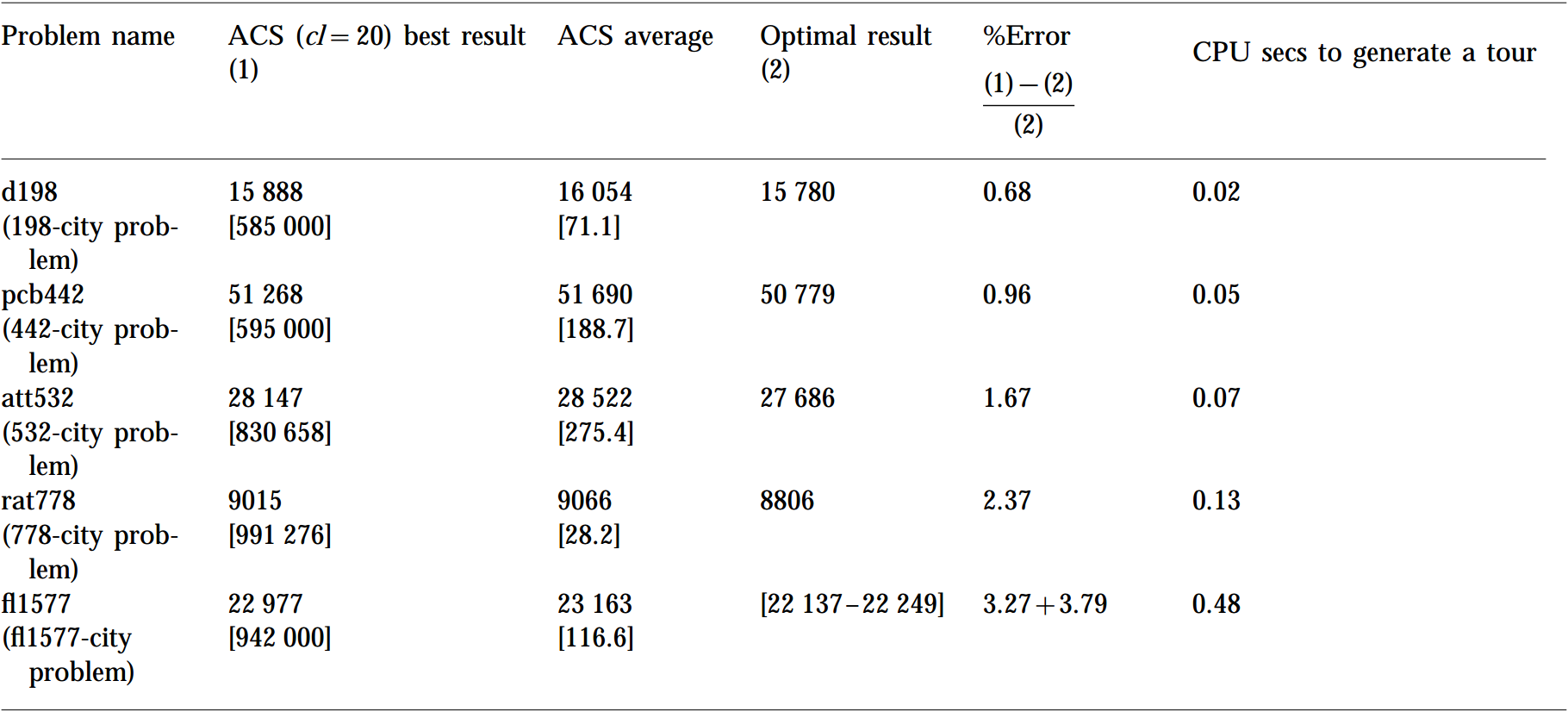
Các so sánh giữa độ dài đường đi ngắn nhất tìm được bởi SA+3-opt (độ đường đi ngắn nhất tìm bởi SA với các lần thực thi riêng biệt có nhúng 3-opt), SOM+ (độ dài đường đi tốt nhất tìm được bởi SOM với 4000 lần thực thi riêng biệt (bằng cách xử lý các thành phố với các thứ tự khác nhau)), FI, FI+3-opt (độ dài đường đi ngắn nhất tìm được bởi FI đã được tối ưu cục bộ bằng 3-opt) và ÁC với có và không có các tối ưu cục bộ 3-opt. Các heuristic 3-opt sử dụng kết quả của ACS và FI khi xâu dựng các cấu hình ban đầu cho tối ưu hóa. Các kết quả của SA+3-opt và SOM+ được lấy từ Durbin và Willshaw (1987) và Potvin (1993). ACS đã được thực thi với 1250 thế hệ kiến sử dụng m = 20 con kiến cho mỗi thế hệ và đường đi tốt nhất được lấy ra từ 15 thực thi. Độ dài đường đi tốt nhất cho mỗi bài toán được in đậm.

Bảng 3. So sánh giữa ACS với GA, EP, SA và AG (Lin et al., 1993)



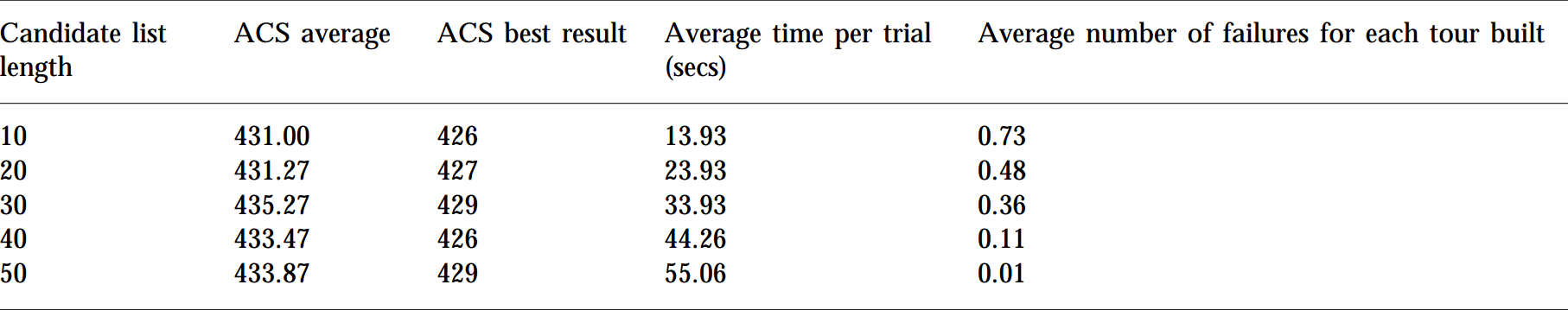
Tác giả đưa ra đường đi ngắn nhất bằng số nguyên, đường đi ngắn nhất với số thực (trong ngoặc tròn) và số đường đi phải duyệt để tìm được đường đi ngắn nhất theo số nguyên (ngoặc vuông). Các kết quả của EP được lấy từ Fogel (1993) và các kết quả của GA được lấy từ Bersini et al. (1995) cho các bài toán KroA100 và từ Whitley et al. (1989) cho các bài toán Oliver30, EilS0 và Eil75. Các kết quả sử dụng SA và AG được lấy từ Lin et al. (1993). Bài toán Oliver30 được lấy từ Oliver et al. (1987), Eil50, Eil75 được lấy từ Eilon et al. (1969) đều có trên TSPLIB với thêm một thành phố nữa với bài toán mới Eil51.tsp và Eil76.tsp. KroA100 cũng ở trong TSPLIB. Các kết quả tốt nhất của mỗi bài toán được in đậm. Có một điều đáng chú ý rằng độ phức tạp của thuật các thuật toán vào khoảng n2 \* t, trừ EP là n \* t (ở đây, n là số thành phố và t là số đường đi được sinh ra). Do vậy, hiển nhiên rằng ACS và EP sẽ vượt trội hơn hẳn GA, SA và AG. TSPLIB: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html (được duy trì bởi G. Reinelt).

Bảng 4. Hiệu năng của ACS khi giải một số bài toán TSP lớn hơn



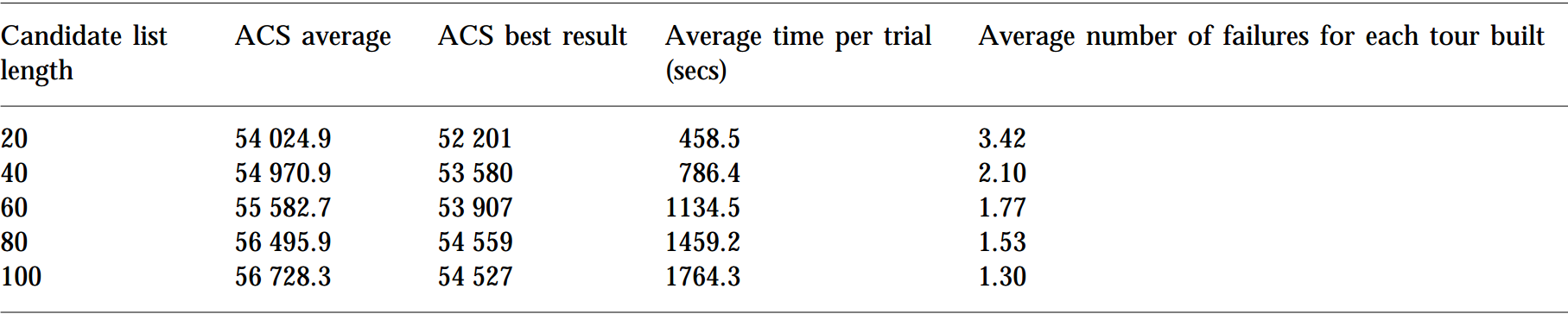
Cột thứ nhất: kết quả tốt nhất có được sau khi chạy ACS 15 lần; tác giả đưa ra độ dài chu trình ngắn nhất với số nguyên và số lượng đường đi đã được sinh ra trước khi có được lời giải tốt nhất đó (trong ngoặc vuông). Cột thứ 2: giá trị trung bình của ACS sau 15 lần thực thi và độ lệch chuẩn của nó trong ngoặc vuông. Cột thứ 3: giá trị tối ưu (cho bài toán fl1577, trong ngoặc vuông, cận trên và cận dưới đã biết, biểu thị rằng giá trị tối ưu không thể tính được chính xác, chỉ biết nó nằm trong đoạn đó). Cột thứ 4: phần trăm lỗi, phép đo cho chất lượng của lời giải tốt nhất được tìm bởi ACS. Cột thứ 5: thời gian cần thiết để sinh ra một lời giải trên máy chủ Sun Sparc (50 MHz). Tất cả các bài toán được giải đều có trên TSPLIB.

Bảng 5. So sánh giữa kích cỡ của các danh sách ứng viên



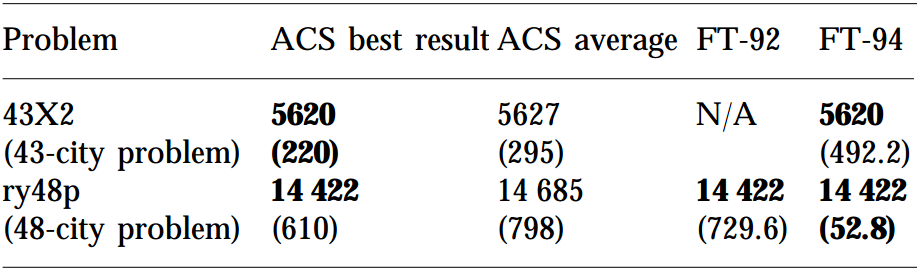
Bài toán: Eil51. Với mỗi độ dài của danh sách ứng viên, các giá trị trung bình được tính qua 15 lần chạy. Mỗi lần chạy sinh ra 500 lời giải.

Bảng 6. So sánh giữa kích cỡ của các danh sách ứng viên



Bài toán: Pcb442. Với mỗi độ dài của danh sách ứng viên, các giá trị trung bình được tính qua 10 lần chạy. Mỗi lần chạy sinh ra 20 000 lời giải.

Bảng 7. So sánh giữa các thuật toán chính xác và ACS khi giải các bài toán TSP bất đối xứng



Lời giải chính xác là mã nguồn xác định được công bố để giải các bài toán TSP bất đối xứng; các kết quả được lấy từ Fischetti và Toth (1992) và Fischetti và Toth (1994). Bài toán ry48p ở trong TSPLIB, và bài toán 43X2 được lấy từ Balas et al. (1993). Tác giả đưa ra độ dài đường đi và thời gian tính toán của CPU (trong ngoặc tròn) dùng để tìm ra lời giải (thử nghiệm chạy trên một máy tính cá nhân Pentium). ACS đã sử dụng 10 con kiến với 1500 thế hệ và kết quả có được sau 15 lần thực thi riêng biệt. Kết quả tốt nhất được tô đậm.

Kết luận

Điểm mất chốt cho việc áp dụng ACS vào các bài toán mới là tìm ra một cách mã hóa bài toán thích hợp (biểu diễn dưới dạng một đồ thị được tìm kiếm bởi các cá thể kiến nhân tạo) và một tập các giá trị heuristic để biểu diễn sự tương quan của trọng số giữa hai nút trên đồ thị. Sau đó, các tương tác xác suất giữa các cá thể kiến trong bầy sẽ được thực hiện dựa trên vệt mùi đồng loại trên các cạnh của đồ thị. Chúng sẽ tạo ra các lời giải tốt, thường là tối ưu cho bài toán. Có rất nhiều cách để cải tiến thuật toán ACS này giúp cho số lượng chu trình cần đạt sẽ giảm đi và hiệu năng tương ứng sẽ tăng lên, từ đó có thể áp dụng dễ dàng vào các thể hiện lớn hơn của bài toán. Trước hết, một tối ưu hóa cục bộ nào đó như 2-opt, 3-opt hoặc Lin-Kernighan có thể được nhúng vào trong thuật toán ACS (đây là một hướng tiếp cận chuẩn cho việc tăng hiệu năng của các thuật toán đa mục đích như EC, SA, NNs như đã được đề cập trong Johnson và McGeoch năm 1997). Trong các thử nghiệm này, tối ưu hóa cục bộ chỉ được thêm vào để làm cho các kết quả tốt nhất của nhiều thuật toán khác trở nên tốt hơn. Trái lại, mỗi con kiến có thể đạt được lời giải tối ưu cục bộ ngay trước khi xảy ra việc cập nhật toàn cục. Thứ hai, thuật toán có khả năng được thực thi song song hiệu quả, có thể nâng cao mạnh mẽ hiệu năng cho việc tìm lời giải tốt, đặc biệt là các bài toán có tập dữ liệu đầu vào lớn. Quá trình song song hóa có thể được triển khai ngay đó là việc phân tán các con kiến một cách độc lập trên các bộ xử lý khác nhau: cùng một bài toán TSP có thể được giải bởi một số lượng kiến nhất định trên mỗi bộ xử lý và chu trình tốt nhất được tìm thấy sẽ được trao đổi một cách bất đồng bộ giữa các bộ xử lý. Một triển khai trước đó trong cùng tư tưởng song song như vậy đã có trên một mạng các máy tính lớn, điều này chỉ ra rằng độ phức tạp của thuật toán phần lớn độc lập với số lượng cá thể kiến trong bầy. Thứ ba, hướng tiếp cận với ACS là rất rộng mở cho các phát triển trong tương lai, như là sự bắt đầu cho các họ kiến đặc biệt, các kết nối, quan hệ chặt chẽ hơn với các phương pháp học tăng cường và bắt đầu cho một số các hàm heuristic được đặc biệt hóa cho việc tìm kiếm.

Chương 2: Cài đặt giải thuật

2.1. Giới thiệu:

Trong quá trình tìm hiểu và cài đặt, thuật toán ACS đòi hỏi phải có một nỗ lực đáng kể để tìm kiếm các nguồn tài liệu liên quan (như sách, báo, công bố khoa học và các mã nguồn đã triển khai), thông hiểu và diễn giải kĩ thuật, cuối cùng mới là ghép nối các thông tin lại để được một bản cài đặt hoàn chỉnh. Do đó, sự liên hệ giữa các thuật toán trong lĩnh vực tính toán tiến hóa rõ ràng là một vấn đề nan giải. Thông thường, các mô tả thuật toán trong các công bố khoa học có các đặc điểm sau:

Không hoàn chỉnh: nhiều thuật toán được mô tả rất nhập nhằng, mô tả nửa vời hoặc chẳng mô tả gì.

Không nhất quán: một kĩ thuật đề xuất có thể được mô tả bằng cách sử dụng nhiều các phương pháp chuyên môn hoặc bán chuyên môn mà những phương pháp đó có thể sử dụng nhiều kĩ thuật khác, làm giới hạn việc chuyển sang giai đoạn cài đặt thuật toán (công thức toán học, giả mã, mã nguồn, mô tả bằng ngôn ngữ tự nhiên) khi đọc giả đọc nó. Một cách biểu diễn không nhất quán cho các kĩ thuật như vậy làm cho các kĩ năng dùng để thông hiểu một thuật toán nào đó không dùng để hiểu thuật toán còn lại được, thậm chí là phiên bản mở rộng của chính nó.

Phân tán: các chi tiết về cấu trúc dữ liệu, các phép toán và các tham số của một kĩ thuật nào đó có thể nằm rải rác ở các bài toán, công bố, sách và mã nguồn từ năm này tới năm khác, sự tiếp cận tới những nguồn tri thức này là tương đối hạn chế và khó khăn cho sinh viên mới làm quen với việc nghiên cứu có thể nắm bắt được.

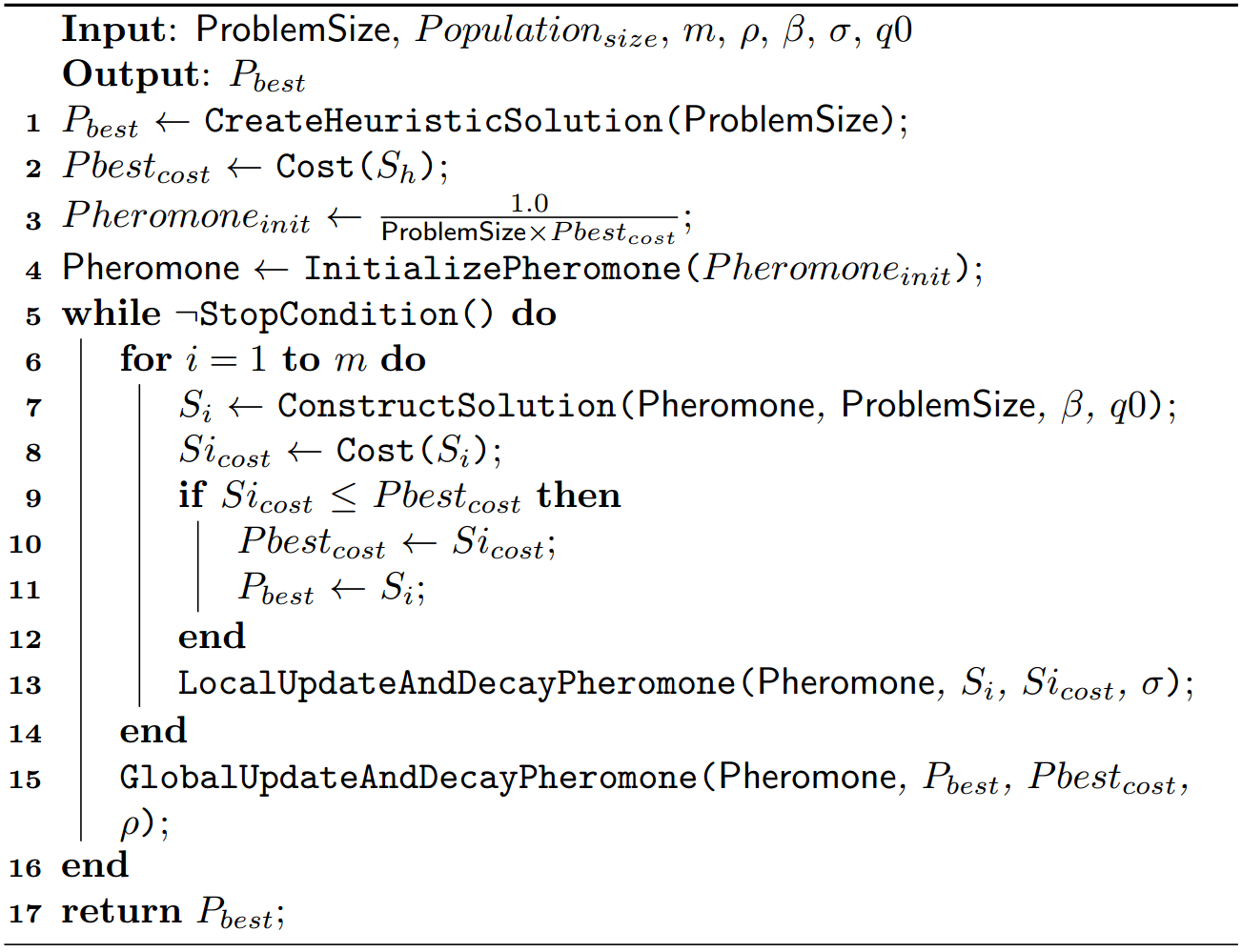
Các thuật toán trong cuốn sách này rất thực tiễn, thú vị và mục đích của nó là làm cho việc tiếp cận các thuật toán trong lĩnh vực tính toán tiến hóa dễ dàng hơn, hữu dụng hơn và dễ hiểu hơn.

2.2. Ý tưởng của chương trình

2.2.1. Chiến lược chung

Mục đích của chiến lược là khai thác các thông tin lịch sử và heuristic để xây dựng các lời giải và đóng góp vào các thông tin đã thu thập được từ việc xây dựng các lời giải trước đó. Các lời giải đó được xây dựng từng phần tử một, rời rạc theo phong cách từng-bước-một. Xác suất để lựa chọn một phần tử của lời giải được tính toán dựa trên các đóng góp heuristic về tổng chi phí và chất lượng của các lời giải trước đó. Lịch sử được cập nhật nghịch đảo của chất lượng lời giải tốt nhất tại thời điểm đó.

2.2.2. Thuật toán của chương trình



Đầu vào: dữ liệu về bài toán và các tham số heuristic.

Đầu ra: Lời giải tối ưu

Các bước thực hiện:

Bước một: tạo ra một lời giải heuristic làm lời giải tối ưu

Bước hai: tính chi phí cho lời giải tối ưu vừa gán

Bước ba: khởi tạo giá trị mùi ban đầu

Bước bốn: khởi tạo ma trận mùi cho các cạnh của đồ thị TSP

Bước năm: bắt đầu một thế hệ kiến mới

Bước sáu: xây dựng một lời giải cho một con kiến trong bầy

Bước bảy: tính chi phí cho lời giải của con kiến đó

Bước tám: chọn lời giải của con kiến đó làm lời giải tối ưu nếu nó tốt hơn lời giải tối ưu hiện tại

Bước chín: cập nhật mùi cục bộ

Bước mười: quay lại bước sáu nếu vẫn còn con kiến chưa xây dựng lời giải, ngược lại làm tiếp bước mười một

Bước mười một: cập nhật mùi toàn cục

Bước mười hai: quay lại bước năm nếu chưa tạo đủ số thế hệ đặt ra, ngược lại làm tiếp bước mười ba

Bước mười ba: đưa ra lời giải tối ưu

2.2.3. Cấu trúc dữ liệu của chương trình