TRÌNH BÀY THUẬT TOÁN

****

Hà Minh Hiếu

BTNT_BSO_HACO

I. Khởi tạo các tham số
- Set up pheromon = 1
- Set up result (kq tốt nhất) = 99999999.

II. Vòng lặp: For j in range (max_iteration):
- Set up current_best (lời giải tốt nhất) = 99999999.
- Vòng lặp: For i in range (sl con kiến):
    1. Con kiến xây dựng lời giải chấp nhận được của bài toán dựa trên pheromon
    2. Sau khi có lời giải, thực hiện IT (Injection Tree) để giảm số xe cần sử dụng. Nếu số xe ít hơn và hàm mục tiêu giảm thì thay thế lời giải hiện tại.
    3. Thực hiện Cross-Exchange trên lời giải để xem có cải thiện thành lời giải tốt hơn không. Nếu có thì thay.
    4. Nếu lời giải thu được tốt hơn result, cập nhập result. Sau đó thực hiện local_search trên result đó, nếu tốt hơn thì thay thế lời giải hiện tại.
    5. Cập nhập Local_Search ngẫu nhiên: Sinh một số ngẫu nhiên r, với xác suất p = p_min + (p_max – p_min) * (j/max_iteration) (cập nhập theo thời gian) (Lí do vì những con kiến về sau có những kiến thức tốt hơn lúc đầu nên cho xác suất local search cao hơn). Nếu r < p thì Local_Search nó.
    6. Nếu lời giải thu được tốt hơn current_best: cập nhập.
- Update các lời giả với IBSO. (Thoát Local Optimum)
- Thực hiện BTNT (nuôi dưỡng những con kiến tốt) (Update BTNT)
- Local_update_pheromon trên tất cả các lời giải của kiến.
- Global_update_pheromon trên lời giải của con kiến tốt nhất.

INJECTION TREE

I. Lựa chọn route có lượng khách hàng ít nhất (Trong trường hợp có nhiều route như vậy thì chọn route có độ dài lớn nhất). Kí hiệu là route_select.

II. For customer in route_select:
- Xác định các route target ứng viên: Lựa chọn các route sao cho khoảng cách giữa customer đến các customer của route target không vượt quá 0.3 lần khoảng cách lớn nhất giữa các khách hàng với nhau (Tiết kiệm chi phí tính toán)
- Chèn customer lần lượt vào các route target: Nếu tồn tại cách chèn sao cho không phải bỏ bớt customer nào của target thì thoả mãn. Nếu không thì di chuyển 1 đến 2

các khách hàng bên cạnh của customer sang route khác. (3 trường hợp: hoặc bên trái, hoặc bên phải, hoặc cả 2 bên)

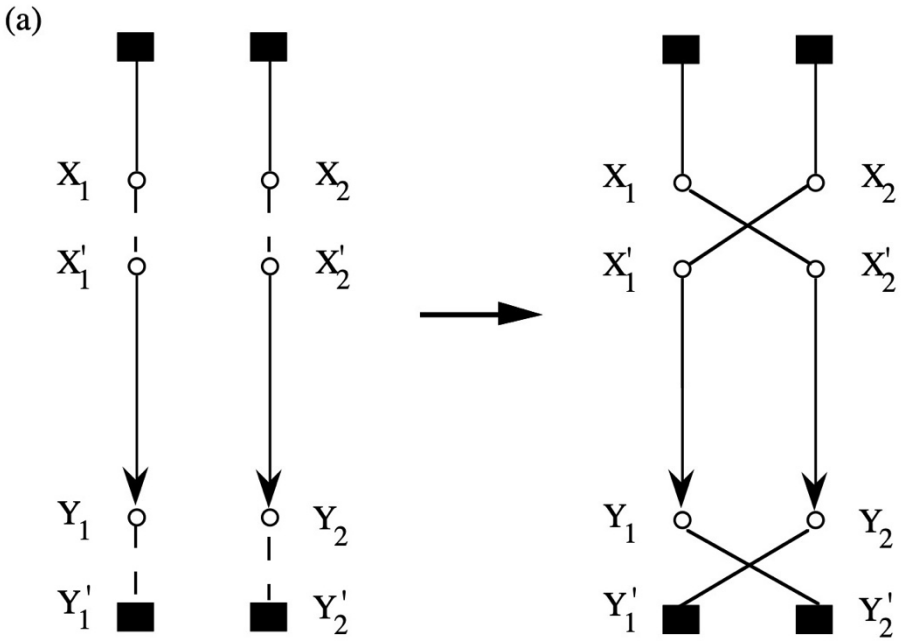
- Nếu không chèn customer vào được thì giữ nguyên customer đó.

III. Kiểm tra lời giải mới: Nếu tốt hơn thì thay thế, không thì giữ nguyên cái cũ.

CROSS-EXCHANGE

Chọn 2 route có độ dài đường đi lớn nhất để thực hiện (Hoặc có độ tương đồng lớn nhất). Các segment được tráo giữa 2 route là một số ngẫu nhiên từ 0 đến 4. Nếu lời giải tốt hơn thì thay thế lời giải hiện tại.

Thực hiện theo tham lam.



IBSO

Kí hiệu (Sm, Cm) đại diện cho lời giải của con kiến thứ m, trong đó S là lời giải còn C là cost.

I. Chia tập lời giải thành 2 nhóm A và B. Nhóm A bao gồm các lời giải có cost nhỏ hơn (Cmax + Cmin)/2. Nhóm B gồm các lời giải còn lại.

II. Với nhóm A, thay các lời giải thu được bằng 1 lời giải ngẫu nhiên. Lời giải ngẫu nhiên được sinh ra bằng việc khi xét xác suất lựa chọn khách hàng tiếp theo, ta không lấy xác suất cao nhất mà lấy theo phân phối mẫu.

III. Với nhóm B, sinh ngẫu nhiên một lời giải (S, C), và 1 số ngẫu nhiên r thuộc (0,1). Xét 1 tham số xác suất p. Nếu r < p, thay thế (Smin, Cmin) bằng (S,C) nếu (S,C) tốt hơn (Smin, Cmin). Nếu r > p, lựa chọn ngẫu nhiên (S,C) bất kì và thay thế bởi (S,C) nếu tốt hơn.

BTNT

Ở mỗi vòng lặp, ta thực hiện local_search cho lời giải ứng viên. Lời giải ứng viên ở đây có thể là lời giải tốt nhất hiện tại (curent_best). Thực hiện local_search cho đến khi lời giải không thể cải thiện thêm. Khi đó lời giải ứng viên sẽ chuyển cho con kiến có lời giải tốt nhất trong vòng lặp đó, đồng thời thực hiện update_pheromone_BTNT coi như là phần thưởng cho nỗ lực tìm kiếm của con kiến đó (Đồng thời cũng là cachs tăng xác suất cho những đường đi gần lời giải tối ưu, giúp những con kiến sau tìm đường đi tốt hơn => Điểm mới đột phá) . (Mỗi lần current_best được cập nhập thì lời giải ứng viên lập tức chuyển thành current_best) => Thực tế đây là trick để tránh lộ liễu việc lạm dụng local_search liên tục cho 1 lời giải.

Một số nhận xét ngoài lề:

- IBSO là 1 cách thoát khỏi Local_optimum rất hay, nhược điểm của các thuật toán bầy đàn => có cái để viết.
- Việc sử dụng BTNT là 1 điểm nhấn hay: Thứ nhất nó vừa thể hiện là các con kiến càng về sau sẽ có tiềm năng tốt hơn các con kiến lúc ban đầu, thứ hai là thực chất đây là việc áp dụng local_search nhiều lần cho lời giải (Local_search ảnh hưởng cực lớn đến chất lượng lời giải và chúng ta không muốn dùng nó liên tục nhiều lần => mất bản chất EA).
- Local_Search chạy vẫn hơi lâu, do lồng của 3 cái search vào nhau. Trên thực tế, 2 cái search đầu chạy tương đối nhanh (khoảng 2s), nhưng cái search thứ 3 thì lâu hơn. Và search 3 này chạy cực lâu trong trường hợp số route của lời giải ít.
- Thay vì sử dụng local_search để xét từng cặp route với nhau (rất lâu), đề xuất chọn ra những cặp route có độ tương đồng lớn (Ví dụ, khoảng cách giữa 2 điểm xa nhất trong cặp route đó là nhỏ nhất) => Giảm độ phức tạp đi nhiều, đồng thời thời gian cũng nhanh hơn, có cái để viết.. Ngoài ra, với những lời giải có lượng route nhỏ, có thể sử dụng cách search khác phù hợp hơn, nhanh hơn. => (Đã giải quyết bằng cách khi sử dụng local_search, ta cho nó 3 option: 1 là thực hiện ls trên 3 đường đi dài nhất, 2 là cho 3 đường đi có số khách hàng ít nhất, 3 là 2 đường đi có độ tương đồng lớn nhất)
- Ngoài ra, còn 1 vấn đề nữa, đó là nếu như những route được chọn để ls mà có lượng khách hàng trên đó rất lớn => độ phức tạp tính toán cao => Đã giải quyết bằng cách nếu những route đc chọn có lượng khách hàng lớn hơn 1 nửa tổng khách hàng => Chỉ chọn 2 route có độ tương đồng cao nhất => tiết kiệm chi phí tính toán.
- Những lời giải gần lời giải tối ưu => Local_Search hay bị kẹt, không cải thiện thêm
- Chạy những bộ 200,300 thì lâu.
- Thử tăng vòng lặp. (lâu)

SO SÁNH KẾT QUẢ

*****

DATASET	HACO with multiple Local Search	IT_BSO_HACO	BTNT_BSO_HACO	Optimal
C101	828.936 (*)	828.936	828.936	827.3
C102	1010.882	977.508	861.430	827.3
C201	591.556 (*)	591.556	591.556	589.1
C202	736.160	762.694	676.027	589.1
R101	1763.613	1736.324	1742.482	1637.7
R102	1564.273	1566.471	1573.574	1466.6
R201	1418.746	1332.872	1302.294	1143.2
R202	1308.220	1214.430	1172.733	1029.6
RC101	1925.767	1863.930	1792.409	1619.8
RC102	1723.499	1629.719	1621.800	1457.4

Nhiệm vụ cần triển khai để cải thiện thêm:

- Giải quyết dứt điểm hoàn toàn CVRP (Cân nhắc việc đánh đổi giữa giảm lượng route đi hay giảm tổng độ dài đường đi để phù hợp với hàm mục tiêu bài toán)
- Bắt đầu tìm hiểu về ML => Chia 2 nhóm: 1 nhóm tìm về ML cổ điển, không liên quan đến Neural Network, nhóm còn lại là về NN, hoặc 1 ng làm hết cũng được.
- Chạy và khảo sát tiếp các bộ dữ liệu của TW còn lại (trong docs), nghiên cứu tại sao những local_search hay bị kẹt tại điểm tối ưu địa phương bằng cách nhìn vào kết quả, đồ thị => Từ đó đề xuất cách tiếp cận local search khác để thoát được nghiệm địa phương. Không nên tập trung quá vào các kiểu local_search vét hết (tham lam) mà nên tìm các cách để đa dạng lời giải như IBSO để thoát local_optimum
- Cần thêm 1 bạn có chuyên môn về AI để trao đổi cho dễ.. (Nếu được).

Thiết kế model cho bài toán CVRP

******

- Vẫn áp dụng base ACO như trên, có thể cải tiến thêm ls để linh hoạt hơn
- Mỗi khi trong 1 con kiến, hoàn thành xong 1 phần lời giải (1 route) => Áp dụng 1 pretrained Model TSP cho route đó.
- Cách tạo model TSP: Dùng attention model để giải quyết bài toán TSP với n_customer (cố định, bằng tổng số khách hàng trong bài toán CVRP gốc). Khi áp dụng model vào route, ta duplicate cái depot lên nhiều lần và coi nó như khách hàng, cho đến khi số điểm trong route đó = n_customer. Có thể thấy rằng khi áp dụng model cho trường hợp duplicate như vậy => 1 route có thể tiến hoá thành nhiều route => Vẫn thoả mãn => Done