SOICT Hackathon 2023: TIKI Routing Optimization

Phan Trường Trí [1 2] **Trần Đình Khang** [1 2] **Võ Khánh An** [1 2] **Lê Nguyễn Minh Huy** [1 2] **Lê Gia Khang** [1 2]

1. Giới thiệu

The TIKI Routing Optimization track of SOICT Hackathon 2023 đã đặt ra một thử thách xuất phát từ bài toán Vehicle Routing Problem (VRP) (Toth & Vigo, 2002b; Braekers et al., 2016; Eksioglu et al., 2009). Bài toán của cuộc thi bao gồm việc phân phối có chiến lược các xe, mỗi xe sẽ được giao nhiệm vụ đóng gói lấy và giao các đơn hàng.

So với bài toán VRP gốc, mỗi tuyến đường trong bài toán này được giới hạn bởi một khung thời gian (time window) và ràng buộc sức chứa tối đa. Mục tiêu là đưa ra lời giải bao gồm những tuyến đường hiệu quả được ối ưu hóa đồng thời nhiều yếu tố về mặt vận hành. Những yếu tố này bao gồm số lượng đơn hàng được giao, số lượng xe sử dụng và tổng thời gian hoạt động của các xe. Những yếu tố này phải được tối ưu trong khi vẫn tuân thủ khung thời gian được xác định trước và các ràng buộc. Về mặt bản chất, bài toán này là sự kết hợp của bài toán Capacitated Vehicle Routing Problem (CVRP) (Toth & Vigo, 2002a; Ralphs et al., 2003), Vehicle Routing Problem with Time Windows (VRPTW) (Tan et al., 2001; Desrochers et al., 1992), và Pickup and Delivery Problem (PDP) (Savelsbergh & Sol, 1995).

2. Bài toán

2.1. Bài toán gốc

Bài toán có thể được biểu diễn bằng đồ thị $G = (V, E)$ với $V = \{1 \dots M\}$ là tập đỉnh (hub), và $E = \{(u, v) | v, u \in V\}$ là tập cạnh (đường đi). Cạnh (u, v) có độ dài là $d_{u,v}$, các cạnh của đồ thị thỏa mãn bất đẳng thức tam giác:

$$d_{u,v} \leq d_{u,w} + d_{w,v} \quad \forall (u, v, w) \in V$$

Có N yêu cầu vận chuyển, mỗi yêu cầu vận chuyển có một số ràng buộc nhất định. Yêu cầu vận chuyển thứ i $(i \in \{1 \dots N\})$có hàng cần được lấy ở đỉnh p_i trong khoảng thời gian $[ep_i, lp_i]$ và cần được dỡ hàng ở đỉnh d_i trong khoảng thời gian $[ed_i, ld_i]$ $(p_i, d_i \in V)$. Ngoài ra, đơn hàng sẽ mất sp_i giây để lấy hàng (tại đỉnh p_i), sd_i để dỡ hàng (tại đỉnh d_i) và có trọng lượng là cap_i, thể tích là vol_i. Lưu ý, chỉ cần bắt đầu lấy hàng và dỡ hàng trong khoảng thời gian cho phép, thời điểm kết thúc việc lấy, dỡ hàng có thể trễ hơn lp_i nếu là lấy hàng và trễ hơn ld_i nếu là dỡ hàng.

Có K xe tải thực hiện việc vận chuyển hàng hóa. Xe tải thứ i $(i \in \{1, \dots, K\})$ có vận tốc là vel_i, có thể chở được tối đa $mcap_i$ về cân nặng và $mvol_i$ về thể tích. Mỗi xe tải sẽ bắt đầu cũng như kết thúc việc vận chuyển ở đỉnh de_i và chỉ hoạt động được trong khoảng thời gian $[f_i, t_i]$. Tức là xe có thể xuất phát sớm nhất tại f_i và phải về lại đến de_i chậm nhất là t_i. Trong quá trình vận chuyển, tổng thể tích của các đơn hàng trên xe tại một thời điểm không vượt quá $mvol_i$ và tổng trọng lượng không vượt quá $mcap_i$. Thời gian xe tải thứ i di chuyển từ đỉnh u đến đỉnh v là:

$$time_i(u, v) = \left\lceil \frac{vel_i}{d(u, v)} \right\rceil$$

Với một lời giải hợp lệ S, gọi O_N là số đơn hàng xử lí thành công, O_K là số xe sử dụng, O_T là tổng thời gian hoạt động của các xe. Thời gian hoạt động của một xe được tính là thời điểm xe bắt đầu hoạt động đến lúc xe trở về điểm xuất phát lần cuối. Khi đó số điểm nhận được là:

$$\boldsymbol{F}(S) = 10^9 \times \frac{O_N}{N} - 10^6 \times \frac{O_K}{K} - \frac{O_T}{10^3} \tag{1}$$

2.2. Định nghĩa lại bài toán

Ta xem như các đỉnh $p_i, d_i, (i \in \{1 \dots N\}), de_j, (j \in \{1 \dots k\})$ là đôi một khác nhau. Nếu có nhiều đỉnh giống nhau thì ta chỉ việc tạo thêm các đỉnh giống như đỉnh bị trùng (có khoảng cách đến các đỉnh khác giống như đỉnh gốc, và có khoảng cách đến đỉnh gốc bằng 0).

Bài toán có thể biểu diễn lại bằng đồ thị $G'(V', E')$ như sau:

- Tập đỉnh $V' = \{\mathcal{P}, \mathcal{D}, \mathcal{V}\}$, trong đó:
  - $\mathcal{P} = \{p_1, p_2, \dots, p_N\}$ với p_i là đỉnh lấy hàng của đơn hàng thứ i.
  - $\mathcal{D} = \{d_1, d_2, \dots, d_N\}$ với d_i là đỉnh dỡ hàng của đơn hàng thứ i.
  - $\mathcal{T} = \{t_1, t_2, \dots, t_K\}$ với t_i là đỉnh bắt đầu và kết thúc của xe tải thứ i.

[1]Trường Đại học Công nghệ Thông tin, TP. Hồ Chí Minh, Việt Nam [2]Đại học Quốc gia TP. Hồ Chí Minh, TP. Hồ Chí Minh, Việt Nam. **AUTHORERR: Missing \icmlcorrespondingauthor.**

SOICT Hackathon, Hanoi, Vietnam. 2023.

- Tập cạnh $E' = V' \times V'$, các cạnh có độ dài giống như độ dài trên đồ thị gốc.
- Mỗi đỉnh i sẽ có thêm một số thông tin:
  - **Thời gian hoạt động** $[e_i, l_i]$: là khoang thời gian bốc hàng (nếu $i \in \mathcal{P}$), dỡ hàng (nếu $i \in \mathcal{D}$) hoặc thời gian hoạt động của xe tải (nếu $i \in \mathcal{T}$). Xe tải chỉ có thể hoạt động ở đỉnh i trong khoảng thời gian này.
  - **Thời gian phục vụ** s_i: Là thời gian bốc hàng, dỡ hàng hoặc bằng 0 lần lượt nếu i thuộc $\mathcal{P}, \mathcal{D}, \mathcal{T}$.
  - **Tải trọng** c: là trong lượng tải xe tải cần xử lí thêm: $c_{p_i} = cap_i$, $c_{d_i} = -cap_i$ và $c_{t_i} = 0$
  - **Thể tích** v: là lượng thể tích xe tải cần xử lí thêm: $v_{p_i} = vol_i$, $v_{d_i} = -vol_i$ và $v_{t_i} = 0$

2.3. Biểu diễn lời giải

Một lời giải cho bài toán là một lịch trình cho các xe, ta gọi $\pi^{(i)}$ là lịch trình cho xe thứ i, $\pi^{(i)}_j$ là đỉnh thứ i trong lịch trình của xe i. Một lời giải $S = \{\pi^{(1)}, \ldots, \pi^{(K)}\}$ gọi là hợp lệ nếu thỏa mãn tất cả các ràng buộc.

Ràng buộc về điểm bắt đầu và kết thúc:

$$\pi^{(i)}_1 = \pi^{(i)}_{|\pi^{(i)}|} = t_i$$

Ràng buộc về thứ tự xử lí đơn hàng: mỗi đơn hàng chỉ được xử lí bởi tối đa một xe, phải cùng được lấy hàng và dỡ hàng bởi một xe và phải được lấy trước khi dỡ hàng:

1. $p_i \in \pi^{(i)} \Rightarrow p_i \notin \pi^{(j)}, \quad \forall i \neq j$
2. $p_i \in \pi^{(i)} \Leftrightarrow d_i \in \pi^{(i)}$
3. $\begin{cases} \pi^{(i)}_j = p_x \\ \pi^{(i)}_k = d_x \end{cases} \Rightarrow j < k$

Ràng buộc về tải trọng và thể tích: tại mọi thời điểm, xe không được chứa nhiều hơn sức chứa tối đa:

1. $\forall i \leq K, j \leq |\pi^{(i)}| : \quad \sum_{k=1}^{j} c_{\pi^{(i)}_k} \leq mcap_i$
2. $\forall i \leq K, j \leq |\pi^{(i)}| : \quad \sum_{k=1}^{j} v_{\pi^{(i)}_k} \leq mvol_i$

Ràng buộc về thời gian: Gọi $ar^{(i)}_j$ là thời điểm sớm nhất mà xe thứ i thăm đỉnh thứ j trên hành trình của mình:

1. $ar^{(i)}_j \geq e_{\pi^{(i)}_j}$
2. $ar^{(i)}_j \leq l_{\pi^{(i)}_j}$
3. $ar^{(i)}_{j+1} \geq ar^{(i)}_j + s_{\pi^{(i)}_j} + time_i(\pi^{(i)}_j, \pi^{(i)}_{j+1})$

3. Phương pháp

Phương pháp chủ yếu của nhóm là sử dụng giải thuật tham lam và metaheuristic bằng cách duy trì một lời giải hợp lệ và cố gắng làm cho lời giải đó tốt hơn. Thuật toán gồm hai bước chính:

- Khởi tạo lời giải bằng giải thuật tham lam.
- Sử dụng metaheuristic để tối ưu lời giải

3.1. Khởi tạo

Nhóm đề xuất hai phương án khởi tạo dựa trên tham lam. Với mỗi xe, ta lần lượt thêm từng cặp (p_i, d_i) vào lịch trình của xe. Mỗi lần thêm sẽ chọn cặp (p_i, d_i) mà sau khi thêm vào thì penalty của lịch trình là thấp nhất. Nếu tất cả các cặp (p_i, d_i) đều không thể thêm vào lịch trình thì chuyển sang xe tiếp theo.

Penalty của một lịch trình được tính bằng thời điểm xe xử lí xong yêu cầu giao hàng cuối cùng, tức là thời điểm xe bắt đầu đi đến đỉnh kết thúc. Tất nhiên, nếu lịch trình không hợp lệ thì penalty được gán bằng ∞.

Điểm khác nhau giữa hai phương án khởi tạo là cách chọn vị trí để thêm p_i và d_i (dòng 6 của thuật toán).

- **Phương án 1:** Thêm d_i vào cuối lịch trình hiện tại, chọn vị trí thêm p_i để penalty nhỏ nhất
- **Phương án 2:** Thử tất cả các cặp vị trí có thể để thêm p_i và d_i. Chọn cặp vị trí tạo thành lịch trình có penalty nhỏ nhất.

Algorithm 1 Initial Solution

```
1:  P ← {1, 2, ..., N}
2:  for each vehicle do
3:      while True do
4:          bestPen ← ∞
5:          for each p ∈ P do
6:              newPen ← Penalty if insert p to vehicle
7:              bestPen ← min(bestPen, newPen)
8:          end for
9:          if bestPen < ∞ then
10:             Insert request i has lowest penalty
11:             Remove i from P
12:         else
13:             Break
14:         end if
15:     end while
16: end for
```

3.2. Large Neighborhood Search

Ý tưởng của Large Neighborhood Search (LNS) là tại mỗi vòng lặp, xóa bỏ q cặp (p_i, d_i) sau đó thêm lại các cặp

(p_i, d_i) này và cố gắng thêm các cặp (p_i, d_i) khác trong số các yêu cầu chưa được xử lí. Để giảm thời gian tính toán, ta cũng chỉ thử trong $maxTry$ cặp (p_i, d_i). Solution mới được tạo ra sẽ được xem xét lấy hay không theo simulated annealing.

Algorithm 2 Large Neighborhood Search

Require: solution S, parameter $q, maxTry$
$S_{best} \leftarrow S$
while not terminated **do**
  $S' \leftarrow S$
  $D \leftarrow q$ inserted request
  Remove requests from D of S
  Reinsert requests from D to S
  $E \leftarrow maxTry$ request not inserted not in D
  for each $e \in E$ **do**
    if can insert e to S **then**
      insert e to S
    end if
  end for
  if $F(S) > F(S_{best})$ **then**
    $S_{best} = S$
  end if
  if not accept(S, S') **then**
    $S \leftarrow S'$
  end if
end while

3.3. Guided Ejection Search

Thuật toán Guided Ejection Search (GES) sẽ remove tất cả request của một xe nào đó, sau đó cố gắng thêm các request chưa được dùng vào các xe khác. Nếu không thể thêm request vào các xe khác, xem xét xóa bỏ một số request hiện có để thêm request đó vào và perturbation solution hiện tại thông qua một số hàm pertubation cơ bản. Việc chọn cách xóa sẽ thông qua một hệ thông penalty h như sau:

- Ban đầu, $h_i = 0, (\forall i)$
- Khi được xem xét thêm vào lại solution, nếu không thể thêm trực tiếp request i vào bất kì xe tải nào thì tăng h_i lên 1
- Trong tất cả các cách xóa sẽ tạo ra solution mới có thể thêm request i vào ta sẽ chọn cách xóa có tổng h của các request bị xóa là nhỏ nhất.

4. Thực nghiệm

Thực nghiệm được chạy trên CPU I5-9300H 2.40GHz, thời gian chạy là 250 giây. Thuật toán LNS có paramater $q, maxTry$ được gán bằng 10

Algorithm 3 Guided Ejection Search

Require: solution S
$S_{best} \leftarrow S$
Remove all request of random truck r
$EP \leftarrow$ all non-inserted request
Initialize penalty $h_i = 1, \forall i \leq N$
while not terminated **do**
  $pd \leftarrow$ a request from EP
  if can insert pd to S **then**
    Insert pd to S
    Remove pd from EP
  else
    $h_{pd} = h_{pd} + 1$
    if can insert pd by remove some request **then**
      Remove requests have minimum penalty
      Insert these requests to EP
      Insert pd to S
      Remove pd from EP
    end if
    $S = pertubation(S)$
  end if
end while
Insert requests to truck r via Initial process

	LNS	**AGES**	**LNS + AGES**
Test 1	**538997935.9**	538997926.7	546997902.3
Test 2	**627997558.2**	606997606.5	605997611.3
Test 3	**662997213.6**	653997260.8	661997235.8
Test 4	**720996927.7**	714996942.0	719996905.8
Test 5	793996524.1	794996566.7	**802996532.5**
Total	**3344986159.5**	3309986302.7	3337986187.7

4.1. Thực nghiệm chỉ dùng một thuật toán

Sau khi khởi tạo solution ban đầu, chạy LNS hoặc AGES đến khi hết running time.

4.2. Thực nghiệm kết hợp hai thuật toán

Sau khi khởi tạo solution ban đầu, chạy LNS và AGES thay phiên nhau đến khi hết running time.

5. Kết luận

Trong cuộc thi lần này, nhóm đã thực nghiệm hai giải pháp là LNS, AEGS và phương pháp kết hợp cả LNS và AEGS. Việc kết quả cả LNS và AEGS đã đạt được hiệu năng tốt hơn việc chạy hai giải pháp đơn lẻ mà vẫn đảm bảo giới hạn thời gian mà ban tổ chức yêu cầu. Việc này thể hiện được hiệu năng và tính khả thi của giải pháp trong việc triển khai thực tế.

Tài liệu

Braekers, K., Ramaekers, K., and Nieuwenhuyse, I. V. The vehicle routing problem: State of the art classification and review. *Comput. Ind. Eng.*, 99:300–313, 2016. doi: 10.1016/J.CIE.2015.12.007. URL https://doi.org/10.1016/j.cie.2015.12.007.

Desrochers, M., Desrosiers, J., and Solomon, M. M. A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.*, 40(2):342–354, 1992. doi: 10.1287/OPRE.40.2.342. URL https://doi.org/10.1287/opre.40.2.342.

Eksioglu, B., Vural, A. V., and Reisman, A. The vehicle routing problem: A taxonomic review. *Comput. Ind. Eng.*, 57(4):1472–1483, 2009. doi: 10.1016/J.CIE.2009.05.009. URL https://doi.org/10.1016/j.cie.2009.05.009.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R., and Jr., L. E. T. On the capacitated vehicle routing problem. *Math. Program.*, 94(2-3):343–359, 2003. doi: 10.1007/S10107-002-0323-0. URL https://doi.org/10.1007/s10107-002-0323-0.

Savelsbergh, M. W. P. and Sol, M. The general pickup and delivery problem. *Transp. Sci.*, 29(1):17–29, 1995. doi: 10.1287/TRSC.29.1.17. URL https://doi.org/10.1287/trsc.29.1.17.

Tan, K. C., Lee, L. H., Zhu, K. Q., and Ou, K. Heuristic methods for vehicle routing problem with time windows. *Artif. Intell. Eng.*, 15(3):281–295, 2001. doi: 10.1016/S0954-1810(01)00005-X. URL https://doi.org/10.1016/S0954-1810(01)00005-X.

Toth, P. and Vigo, D. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discret. Appl. Math.*, 123(1-3):487–512, 2002a. doi: 10.1016/S0166-218X(01)00351-1. URL https://doi.org/10.1016/S0166-218X(01)00351-1.

Toth, P. and Vigo, D. (eds.). *The Vehicle Routing Problem*, volume 9 of *SIAM monographs on discrete mathematics and applications*. SIAM, 2002b. ISBN 978-0-89871-498-2. doi: 10.1137/1.9780898718515. URL https://doi.org/10.1137/1.9780898718515.