

# TỐI ƯU LẬP KẾ HOẠCH

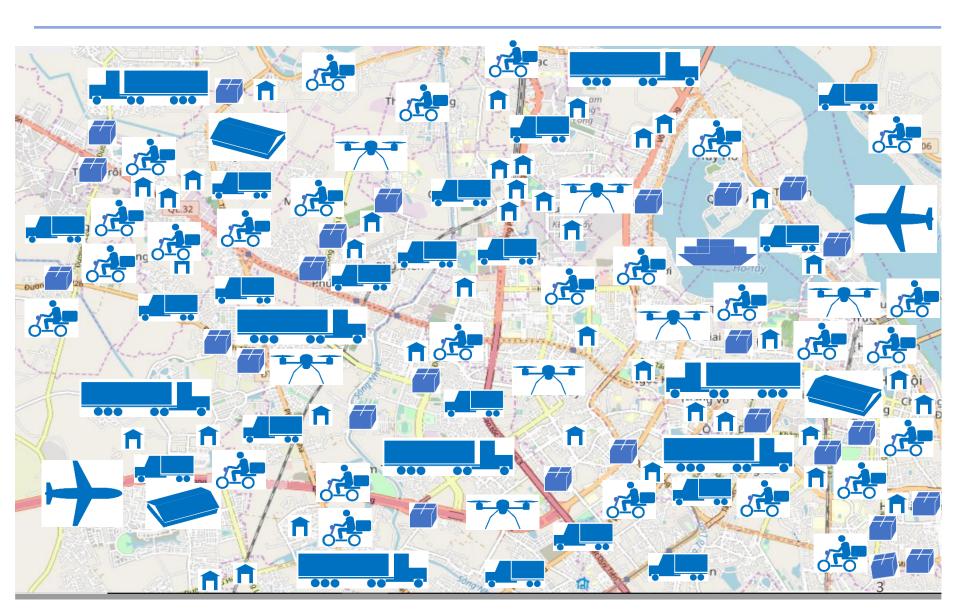
Bài toán lộ trình vận tải

#### Nội dung

- Tổng quan
- Quy hoạch nguyên tuyến tính
- Quy hoạch rang buộc
- Tìm kiếm cục bộ
- Thư viện CBLSVR



# Tổng quan



# Tổng quan

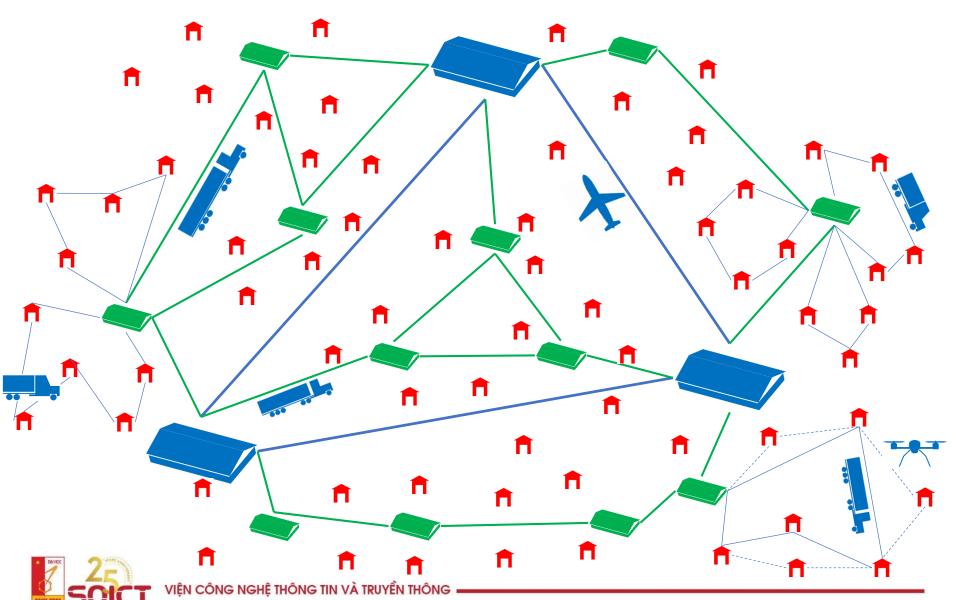
- Theo [Armstrong and associates, 2007]
  - Châu âu và các nước Bắc Mỹ: 8→11% GDP
  - Các nước đang phát triển: 12→21% GDP
- Theo báo cáo của CSCMP về logistics

Năm	Tổng chi phí logistics ở Mỹ (\$ tỉ)	
2016	1,392.64	
2017	1,494.70	

- Theo Hiệp hội doanh nghiệp dịch vụ Logistics Việt Nam (VLA)
  - Chi phí logistics của Việt Nam năm 2016 chiếm khoảng 20% GDP



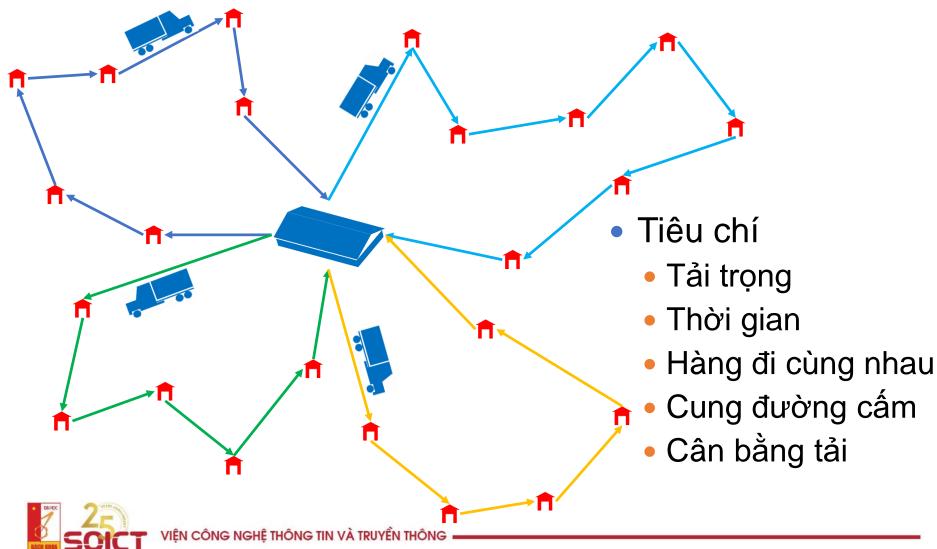
# Mạng lưới chuỗi cung ứng vận chuyển



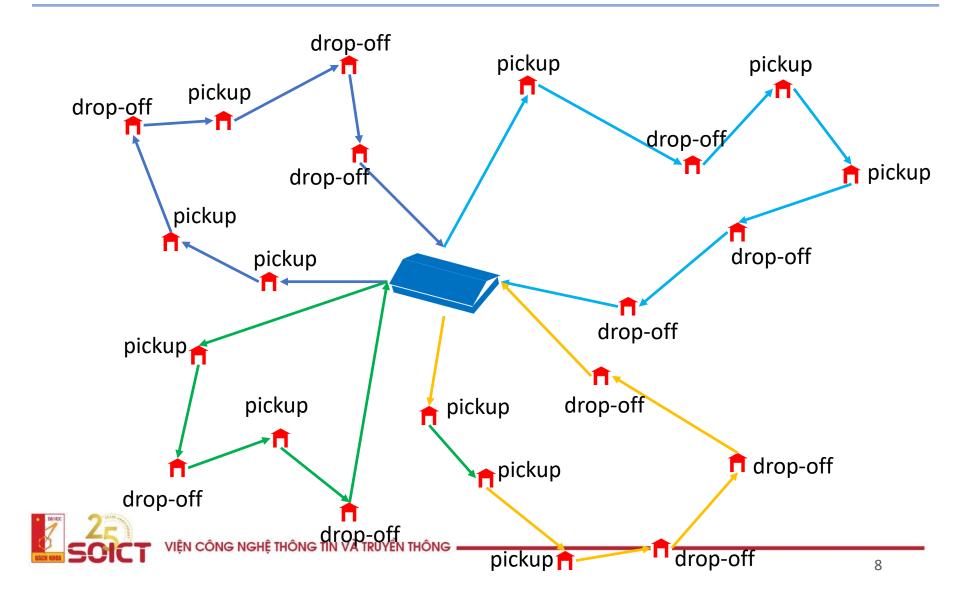
#### Bài toán lập lộ trình vận tải

- Giao hàng từ kho trung tâm
- Lộ trình lấy và trả hàng hóa
- Lộ trình vận chuyển container
- · Lộ trình đón, trả người kết hợp hàng hóa
- Giao hàng kết hợp xe tải và thiết bị bay

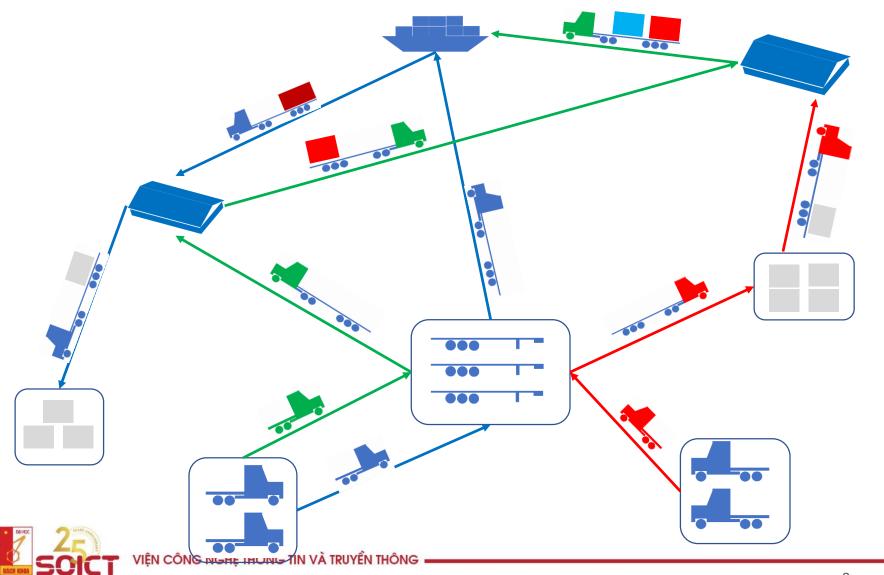




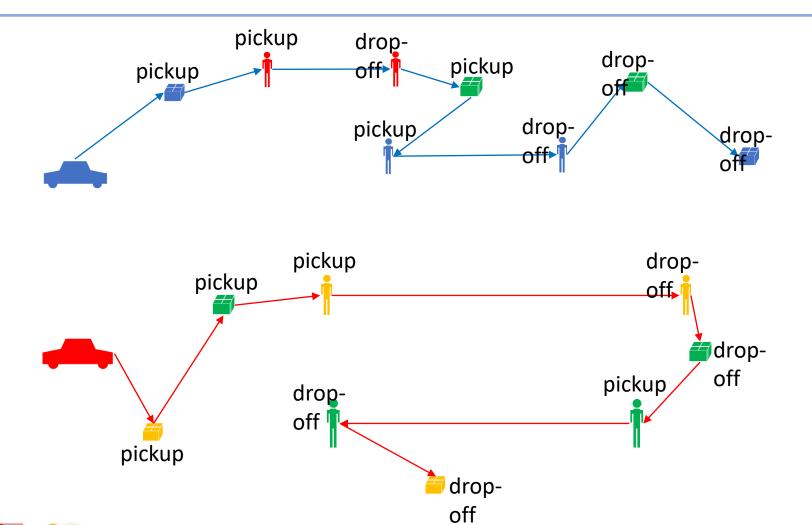
#### Lộ trình đón và trả



# Lộ trình vận chuyển container

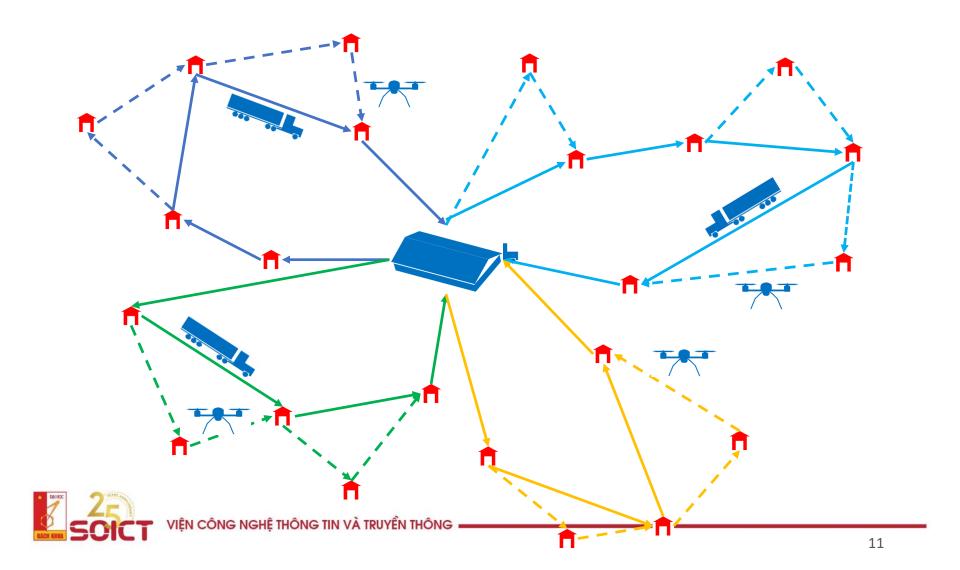


# Lộ trình đón trả người kết hợp hàng hóa



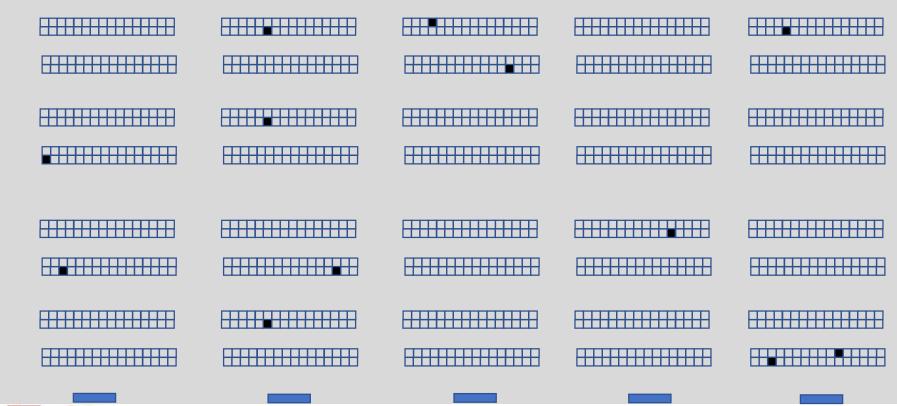


# Giao hàng kết hợp xe tải và thiết bị bay

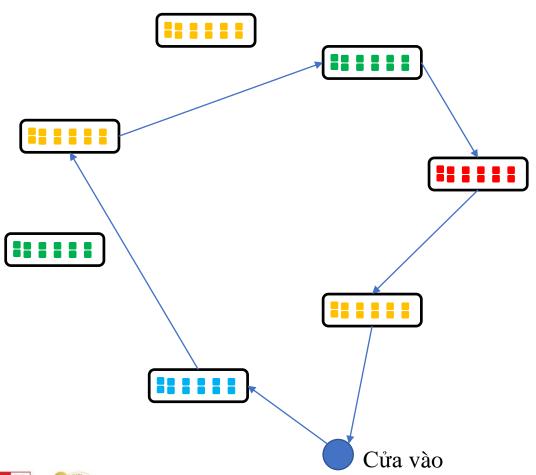


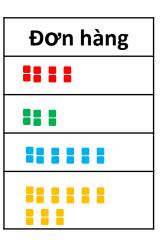
## Quản lý kho

• Lộ trình lấy hàng trong kho (order picking optimization)



# Lộ trình lấy hàng trong kho

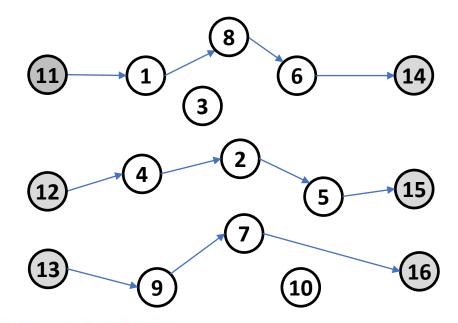






# Tổng quan bài toán VRP

- Mô hình hóa và giải bài toán lộ trình vận tải bằng tìm kiếm cục bộ
- Phương án
  - K lộ trình, mỗi lộ trình bắt đầu và kết thúc tại 2 điểm và thăm một số điểm khách hàng
  - Mỗi điểm nằm trên nhiều nhất 1 lộ trình





#### Nội dung

- Tổng quan
- Quy hoạch nguyên tuyến tính
- Quy hoạch ràng buộc
- Tìm kiếm cục bộ
- Thư viện CBLSVR



# Quy hoạch nguyên tuyến tính

#### Inputs:

- Kho hàng ở vị trí 0
- Tập khách hàng  $S=\{1,\dots,n\}$ , khoảng cách giữa 2 điểm (1 điểm có thể là khách hàng hoặc kho hàng) i,j là  $d_{ij}\in R^+$
- Đoàn gồm m xe tải giống nhau

#### Variables:

•  $x_{ij} = 1$  nếu 1 xe tải nào đó đi từ địa điểm i đến địa điểm j

#### Constraint:

- $\sum_{i=0}^{n} x_{ij} = \sum_{i=0}^{n} x_{ii} = 1$ ,  $\forall j \in \{1, ..., n\}$
- $\sum_{i=1}^{n} x_{0i} = \sum_{i=1}^{n} x_{i0}$
- $\sum_{i=1}^{n} x_{0i} \leq m$
- Ràng buộc chu trình con

#### Objectives:

• Minimize  $\sum_{i=0,j=0}^{n} x_{ij} d_{ij}$ 



• http://vrp.atd-lab.inf.puc-rio.br/index.php/en/



# Mở rộng

Capacitated Vehicle Routing Problem

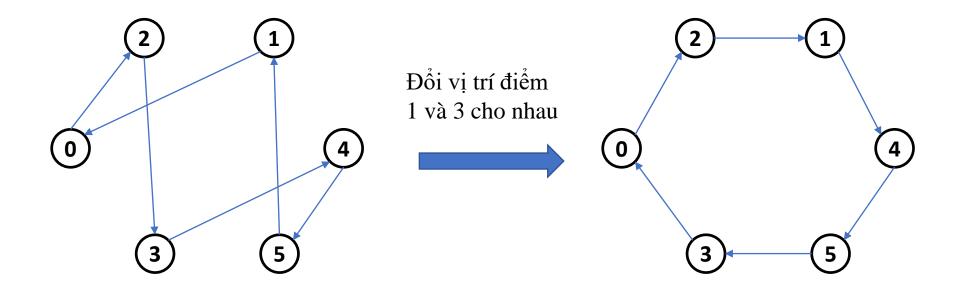


#### Nội dung

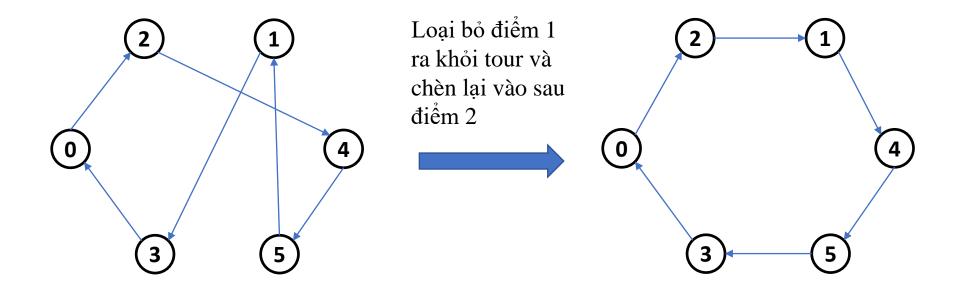
- Tổng quan
- Quy hoạch nguyên tuyến tính
- Quy hoạch ràng buộc
- Tìm kiếm cục bộ
- Thư viện CBLSVR



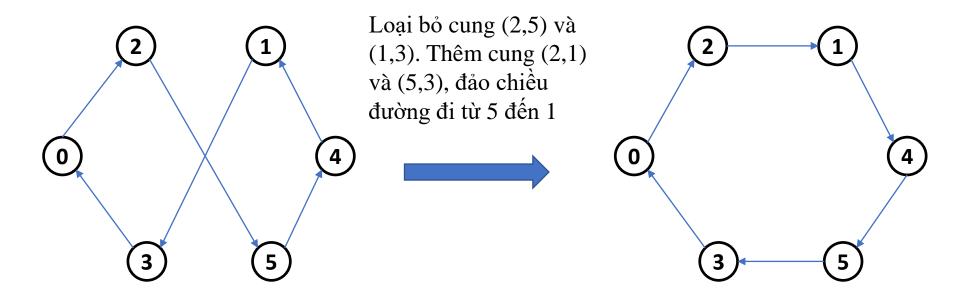
Toán tử láng giềng cho trường hợp 1 xe (TSP tour)



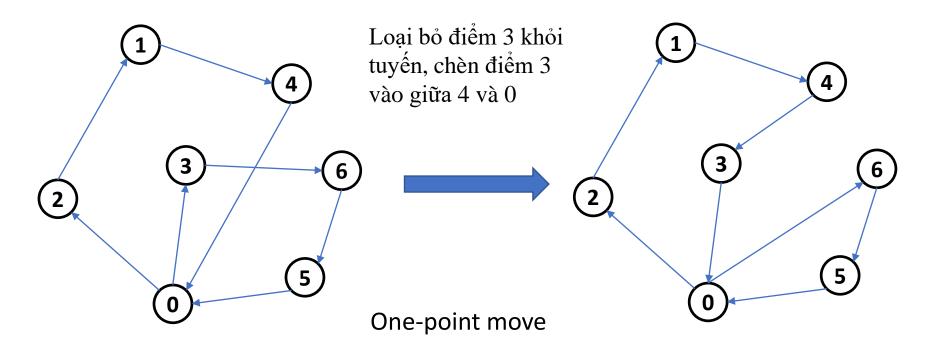
Toán tử láng giềng cho trường hợp 1 xe (TSP tour)



Toán tử láng giềng cho trường hợp 1 xe (TSP tour)

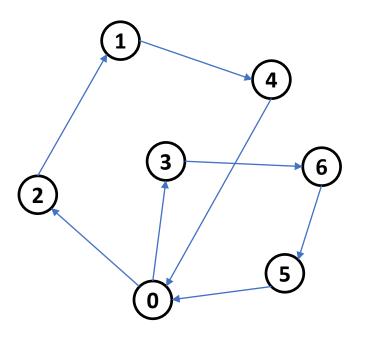


• Toán tử láng giềng cho trường hợp nhiều xe



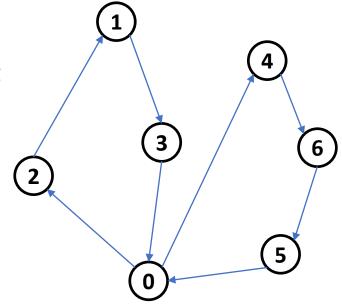


• Toán tử láng giềng cho trường hợp nhiều xe



Loại bỏ cung (1,4) và (3,6), thêm cung (1,3) và (4,6), đảo chiều các cung còn lại cho phù hợp

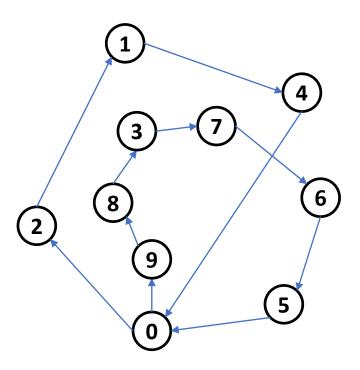




Two-opt move

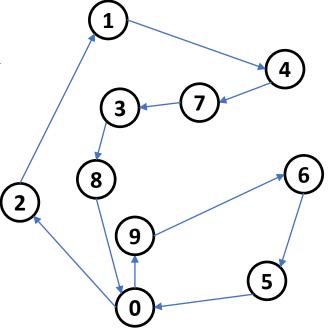


• Toán tử láng giềng cho trường hợp nhiều xe



Loại bỏ chuỗi (đường đi con) điểm 8,3,7 khỏi đường đi 2 và chèn chuỗi này vào sau điểm 4 trên đường đi 1, đảo chiều cung cho phù hợp

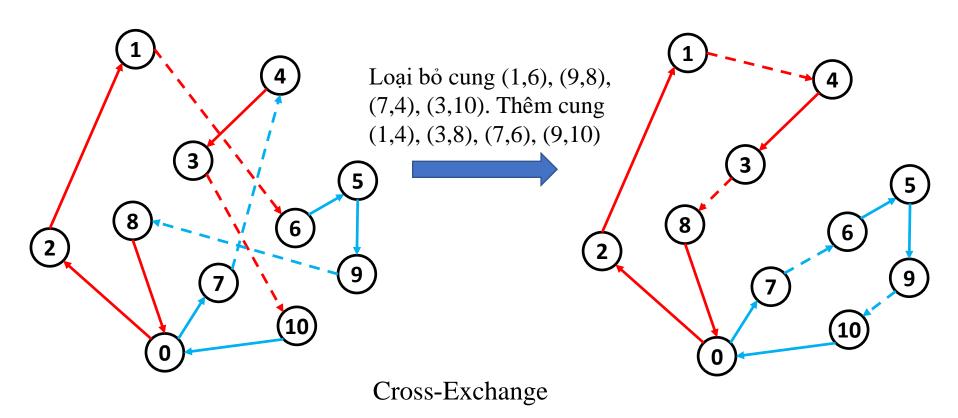




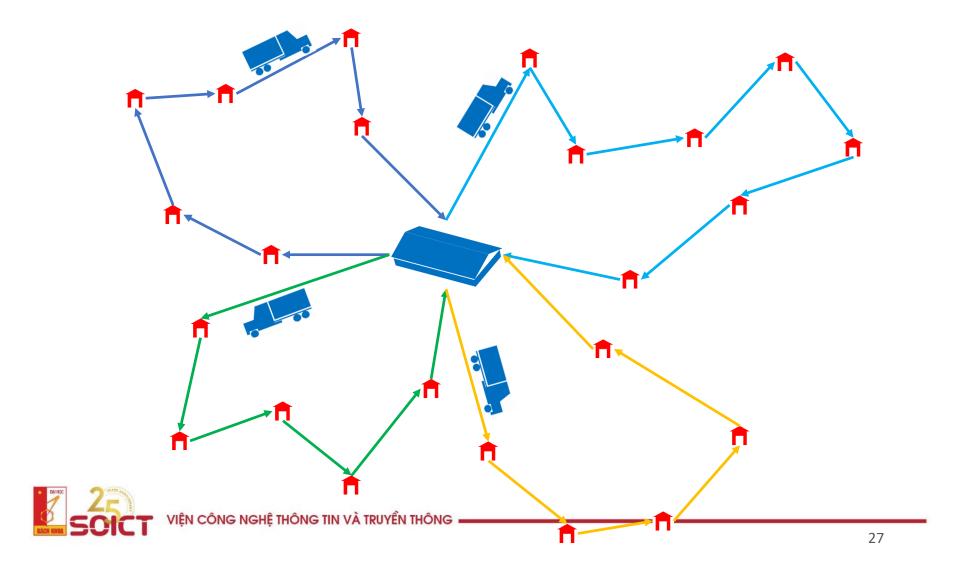
Or-opt move



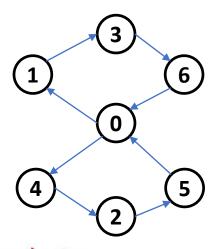
• Toán tử láng giềng cho trường hợp nhiều xe







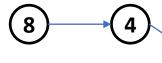
- Có N khách hàng 1, 2, ..., N. Khách hàng i ở điểm i và yêu cầu một lượng hàng w(i), với i = 1,..., N. Có K xe 1, 2, ..., K xuất phát từ kho (điểm 0). Xe k có tải trọng là c(k), với k = 1,2,..., K.
- d(i,j) là khoảng cách từ điểm i đến điểm j, với i,j = 0,1,2,..., N
- Cần lập kế hoạch vận chuyển cho K xe thỏa mãn ràng buộc về tải trọng và tối ưu tổng quãng đường của K xe.





- Điểm logic N+1, N+2, ..., N+2K trong đó N+i và N+K+i tương ứng là điểm xuất phát của route thứ k (mọi k = 1, 2, ..., K). Các điểm này tham chiếu đến điểm xuất phát (depot) vật lý
- Biến
  - X[i]: điểm tiếp theo của điểm i trên lộ trình, i = 1, 2, ..., N+K
  - IR[i]; chỉ số (index) của route đi qua điểm i, i = 1, 2, ..., N+2K
  - L[i]: khoảng cách tích luỹ của route chứa i từ điểm xuất phát đến điểm i, i = 1, ..., N + 2K
  - W[i]: trọng lượng tích luỹ của hàng hoá trên xe từ điểm xuất phát đến điểm i





0

- Ràng buộc
  - L[s] = 0, s = N+1, ..., N+K
  - W[s] = 0, s = N+1,... N+K
  - IR[N+i] = IR[N+K+i], i = 1, ..., K
  - $X[i] \neq X[j]$ , mọi  $i \neq j$ , i,j = 1,2,..., N+K
  - $X[i] \neq i$ , với mọi i = 1, 2, ..., N+K
  - $X[i] = j \Rightarrow IR[i] = IR[j]$ , với mọi i = 1,..., N+K, j = 1,..., N+2K
  - $X[i] = j \Rightarrow L[j] = L[i] + d(i,j)$ , với mọi i = 1,..., N+K, j = 1,..., N+2K
  - $X[i] = j \Rightarrow W[j] = W[i] + w_i$ , với mọi i = 1,..., N+K, j = 1,..., N+2K
  - $W[k] \le c(k), k = N+K+1,...N+2K$
- Hàm mục tiêu: *f* = *L*[*N*+*K*+1] + ... + *L*[*N*+2*K*] → min



- Cung cấp API mô hình hóa
  - Lời giải
  - Hàm, ràng buộc
- API cung cấp phương tiện để truy vấn các thuộc tính của bài toán, chất lượng các phương án láng giềng
- Người phát triển ứng dụng không cần thao tác, duy trì các cấu trúc dữ liệu phức tạp
  - Tập trung vào mô hình hóa
  - Phát triển các thuật toán tìm kiếm cục bộ hiệu quả
- Người phát triển ứng dụng có thể mở rộng
  - Thiết kế và cài đặt các hàm, ràng buộc để mô hình hóa các yếu tố mới của lớp bài toán VRP và tích hợp vào thư viện framework

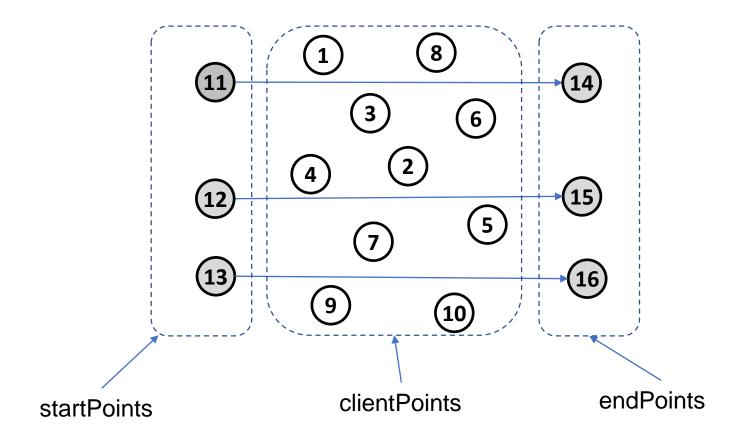


Mô hình hóa biến lộ trình

DS điểm bắt đầu của các lô trình

```
ArrayList<Point> startPoints;
                                          DS điểm kết thúc của các lộ trình
ArrayList<Point> endPoints;
ArrayList<Point> clientPoints;
                                        DS điểm cần đi qua của các lộ trình
VRManager mgr = new VRManager();
                                                  Đối tượng quản lý
VarRoutesVR XR = new VarRoutesVR(mgr):
                                                  Biến (nhiều) lộ trình
for(int i = 0; i < startPoints.size(); i++){</pre>
  Point s = startPoints.get(i);
  Point t = endPoints.get(i);
  XR.addRoute(s, t);
for(Point p: clientPoints) XR.addClientPoint(p);
```

Mô hình hóa biến lộ trình





• VarRoutesVR: biến lộ trình

Truy vấn	Mô tả
startPoint(int k)	Điểm xuất phát của lộ trình thứ k (1,2,)
endpoint(int k)	Điểm kết thúc của lộ trình thứ k (1,2,)
next(Point x)	Điểm tiếp theo của điểm x trên lộ trình
prev(Point x)	Điểm trước điểm x trên lộ trình

VarRoutesVR: biến lộ trình

Toán tử di chuyển cục bộ	Mô tả
performOnePointMove(Point x, Point y)	move of type a [Groer et al., 2010] move customer x to from route of x to route of y; insert x into the position between y and next[y] x and y are not depot remove (prev[x],x), (x, next[x]), (y,next[y]) insert (prev[x], next[x]), (y,x), (x, next[y])
performTwoPointsMove(Point x, Point y)	move of type b [Groer et al., 2010]  x and y are on the same route and are not the depot, y locates before x on the route remove (prev[x],x) and (x,next[x]) and (prev[y], y) and (y, next[y]) insert (x,prev[y]) and (next[y],x) and (next[x],y) and (y, prev[x])

• VarRoutesVR: biến lộ trình

Toán tử di chuyển cục bộ	Mô tả
performCrossExchangeMove(Point x1, Point y1, Point x2, Point y2)	move of type g [Groer et al., 2010] x1 and y1 are on the same route, x1 is before y1 x2 and y2 are on the same route, x2 is before y2 remove (x1,next[x1]) and (y1, next[y1]) remove (x2, next[x2]) and (y2, next[y2]) insert (x1, next[x2]) and (y2, next[y1]) insert (x2, next[x1]) and (y1, next[y2])
performTwoOptMove1(Point x, Point y)	move of type c [Groer et al., 2010] x and y are on different routes and are not the depots remove (x,next[x]) and (y,next[y]) insert (x,y) and (next[x],next[y])



## Thư viện CBLSVR

• IFunctionVR, IConstraintVR: interface mà các hàm, ràng buộc thi hành

Truy vấn	Mô tả
evaluateOnePointMove(Point x, Point y)	move of type a [Groer et al., 2010] move customer x to from route of x to route of y; insert x into the position between y and next[y] x and y are not depot remove (prev[x],x), (x, next[x]), (y,next[y]) insert (prev[x], next[x]), (y,x), (x, next[y])
evaluateTwoPointsMove(Point x, Point y)	move of type b [Groer et al., 2010] x and y are on the same route and are not the depot, y locates before x on the route remove (prev[x],x) and (x,next[x]) and (prev[y], y) and (y, next[y]) insert (x,prev[y]) and (next[y],x) and (next[x],y) and (y, prev[x])

## Thư viện CBLSVR

• IFunctionVR, IConstraintVR: interface mà các hàm, ràng buộc thi hành

Truy vấn	Mô tả
evaluateCrossExchangeMove(Point x1, Point y1, Point x2, Point y2)	move of type g [Groer et al., 2010] x1 and y1 are on the same route, x1 is before y1 x2 and y2 are on the same route, x2 is before y2 remove (x1,next[x1]) and (y1, next[y1]) remove (x2, next[x2]) and (y2, next[y2]) insert (x1, next[x2]) and (y2, next[y1]) insert (x2, next[x1]) and (y1, next[y2])
evaluateTwoOptMove1(Point x, Point y)	move of type c [Groer et al., 2010] x and y are on different routes and are not the depots remove (x,next[x]) and (y,next[y]) insert (x,y) and (next[x],next[y])



#### Thư viện CBLSVR

 Invariants: Duy trì các thuộc tính của bài toán, tự động cập nhật theo sự thay đổi của lộ trình trong quá trình tìm kiếm cục bộ

Bất biến	Mô tả
AccumulatedWeightNodesVR	Trọng số tích lũy trên các điểm. Hàm getSumWeights(Point x) trả về tổng trọng số trên các điểm trên lộ trình từ điểm xuất phát của lộ trình chứa x cho đến x
AccumulatedWeightEdgesVR	Trọng số tích lũy trên cạnh. Hàm getCostRight(Point x) trả về khoảng cách trên lộ trình từ điểm xuất phát của lộ trình chứa x đến điểm x

```
public class CVRP {
  int K = 2;// number of routes
  int N = 6;// number of clients
  int capacity = 11;
  int[] demand = {0,4,2,5,2,3,5};
  int[][] c = {
    \{0,3,2,1,4,3,7\},
    \{2,0,2,3,5,3,9\},
    \{1,3,0,2,4,2,4\},
    \{5,3,2,0,1,1,7\},
    {3,1,5,1,0,3,6},
    \{6,3,2,4,4,0,9\},
    {2,3,2,1,2,8,0}
  };
```



```
ArrayList<Point> start;
ArrayList<Point> end;
ArrayList<Point> clientPoints;
ArrayList<Point> allPoints;
ArcWeightsManager awm;// luu tru trong so tren canh noi giua cac point
NodeWeightsManager nwm;// luu tru trong so tren cac point
HashMap<Point, Integer> mapPoint2ID;
// modelling
VRManager mgr;
VarRoutesVR XR;// bien loi giai (luu tap cac route)
ConstraintSystemVR CS;
LexMultiFunctions F;
IFunctionVR obj;
IFunctionVR[] d;// d[k] la demand cua route k
IFunctionVR[] cost;// cost[k] la chieu dai cua route thu k
Random R = new Random();
```

```
public void mapping(){
  start = new ArrayList<Point>();
 end = new ArrayList<Point>();
  clientPoints = new ArrayList<Point>();
  allPoints = new ArrayList<Point>();
 mapPoint2ID = new HashMap<Point, Integer>();
 // khoi tao cac diem bat dau va ket thuc cua cac xe (route)
 for(int k = 1; k <= K; k++){
    Point s = new Point(0);
    Point t = new Point(0);
    start.add(s); end.add(t);
    allPoints.add(s); allPoints.add(t);
    mapPoint2ID.put(s, 0);
    mapPoint2ID.put(t,0);
```

```
for(int i = 1; i <= N; i++){
   Point p = new Point(i);
   clientPoints.add(p); allPoints.add(p); mapPoint2ID.put(p, i);
 }
 awm = new ArcWeightsManager(allPoints);
nwm = new NodeWeightsManager(allPoints);
for(Point p: allPoints){
   for(Point q: allPoints){
     int ip = mapPoint2ID.get(p); int iq = mapPoint2ID.get(q);
     awm.setWeight(p,q, c[ip][iq]);
   }
for(Point p: allPoints)
 nwm.setWeight(p, demand[mapPoint2ID.get(p)]);
```

```
public void stateModel(){
 mgr = new VRManager();
 XR = new VarRoutesVR(mgr);
 for(int i = 0; i < start.size(); i++){
   Point s = start.get(i);
    Point t = end.get(i);
   XR.addRoute(s, t);// them 1 route vao phuong an (s --> t)
  }
 for(Point p: clientPoints)
   XR.addClientPoint(p);// khai bao XR co the se di qua diem p
 // thiet lap rang buoc
 CS = new ConstraintSystemVR(mgr);
 AccumulatedWeightNodesVR accDemand = new AccumulatedWeightNodesVR(XR, nwm);
 AccumulatedWeightEdgesVR accW = new AccumulatedWeightEdgesVR(XR, awm);
```

```
d = new IFunctionVR[K];// demand on routes
for(int k = 1; k <= K; k++){
  Point tk = XR.endPoint(k);// diem cuoi cung cua route thu k
  d[k-1] = new AccumulatedNodeWeightsOnPathVR(accDemand, tk);
 CS.post(new Leq(d[k-1],capacity));
}
cost = new IFunctionVR[K];
for(int k = 1; k <= K; k++){}
  Point tk = XR.endPoint(k);
  cost[k-1] = new AccumulatedEdgeWeightsOnPathVR(accW, tk);
}
obj = new TotalCostVR(XR, awm);// tong khoang cach di chuyen cua K xe (route)
mgr.close();
```



```
public void initialSolution(){
 ArrayList<Point> listPoints = new ArrayList<Point>();
 for(int k = 1; k <= XR.getNbRoutes(); k++){</pre>
    listPoints.add(XR.startPoint(k));
 for(Point p: clientPoints){
   Point x = listPoints.get(R.nextInt(listPoints.size()));
   mgr.performAddOnePoint(p, x);
    System.out.println(XR.toString() + "violations = " + CS.violations()
      + ", cost = " + obj.getValue());
    listPoints.add(p);
```



```
public void exploreNeighborhood(ArrayList<Move> cand){
 cand.clear(); int minDeltaC = Integer.MAX VALUE; double minDeltaF = minDeltaC;
 for(int k = 1; k <= XR.getNbRoutes(); k++){</pre>
   for(Point y = XR.startPoint(k); y != XR.endPoint(k); y = XR.next(y)){
      for(Point x: clientPoints)if(x != y && x != XR.next(y)){
        int deltaC = CS.evaluateOnePointMove(x, y);
        double deltaF = obj.evaluateOnePointMove(x, y);
        if(!(deltaC < 0 | deltaC == 0 && deltaF < 0)) continue;
        if(deltaC < minDeltaC || deltaC == minDeltaC && deltaF < minDeltaF){</pre>
          cand.clear(); cand.add(new Move(x,y));
          minDeltaC = deltaC; minDeltaF = deltaF;
        }else if(deltaC == minDeltaC && deltaF == minDeltaF) cand.add(new Move(x,y));
```

```
public void search(int maxIter){
  initialSolution();
  int it = 0;
 ArrayList<Move> cand = new ArrayList<Move>();
 while(it < maxIter){</pre>
    exploreNeighborhood(cand);
    if(cand.size() <= 0)</pre>
      System.out.println("Reach local optimum"); break;
   Move m = cand.get(R.nextInt(cand.size()));
    mgr.performOnePointMove(m.x, m.y);
    System.out.println("Step " + it + ", XR = " + XR.toString() + "violations = "
            + CS.violations() + ", cost = " + obj.getValue());
    it++;
```

```
class Move{
 Point x;
 Point y;
 public Move(Point x, Point y){
    this.x = x; this.y = y;
public static void main(String[] args) {
 CVRP A = new CVRP();
 A.mapping();
 A.stateModel();
 A.search(100);
```

