Department of Statistics and Operational Research I
Universidad Complutense de Madrid. Spain

THE DOUBLE TSP WITH MULTIPLE STACKS: A VARIABLE NEIGHBORHOOD SEARCH APPROACH

Ángel Felipe Ortega
M. Teresa Ortuño Sánchez
Gregorio Tirado Domínguez

EURO XXII, Prague, 2007

Outline

1. Introduction
2. Mathematical Model
3. Neighborhood Structures
4. VNS approach
5. Results and Further Work

The Double TSP with Multiple Stacks (DTSPMS)

- Introduced by Hanne L. Petersen (2006)
  - real life application in Easy Cargo Systems A/S
- Extension of the well known TSP
  - pickups & deliveries → two separated graphs
  - sequencing constraints → multiple stacks
- Each order consists of a pickup location and a delivery location
- Load is organized in several rows in the container
  - different stacks obeying LIFO principle (Last-In-First-Out)

The Double TSP with Multiple Stacks (DTSPMS)

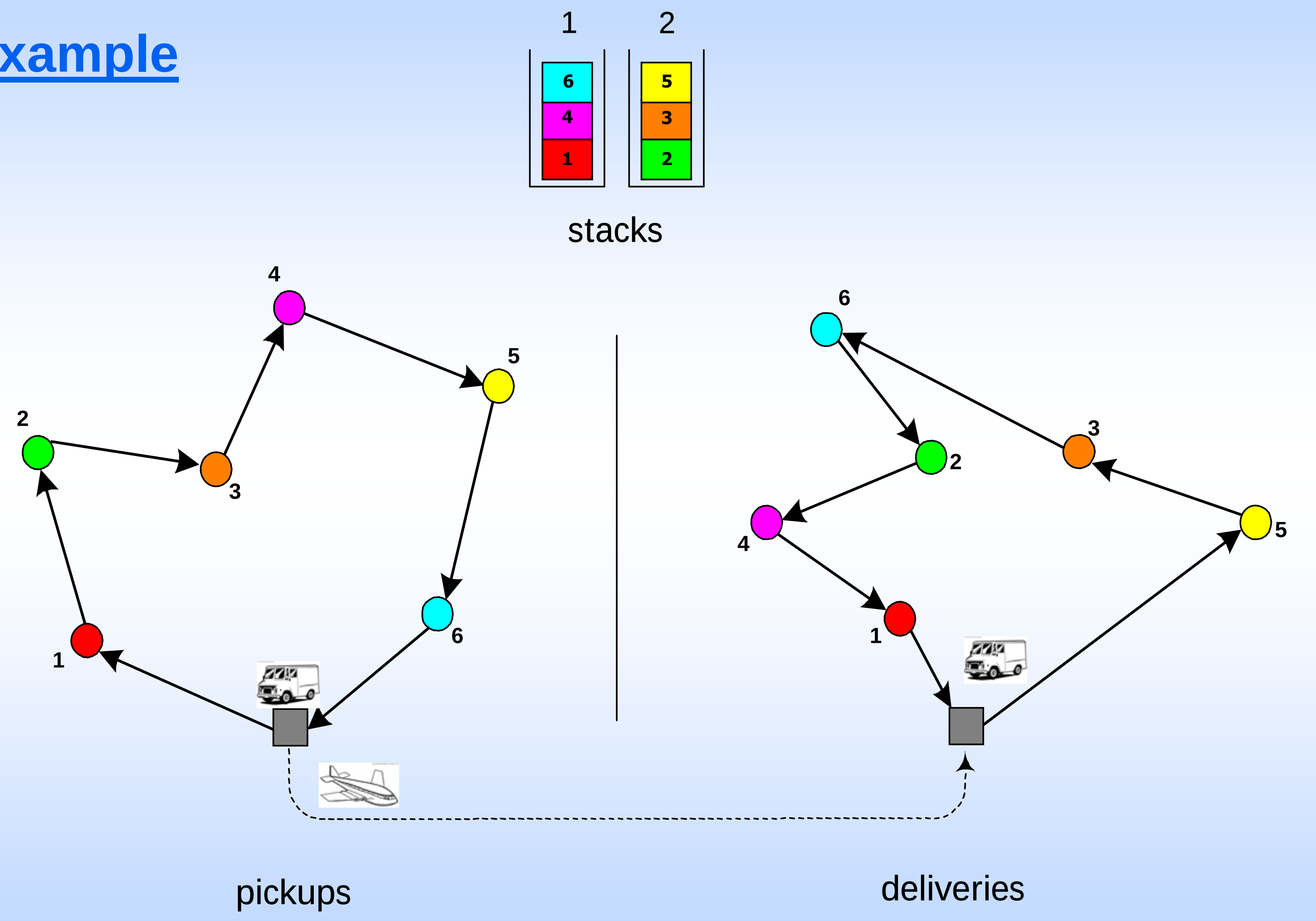
- Pickup & delivery problem with precedence constraints
- Two separated networks, one for pickups and one for deliveries
- All items are uniform
- No repacking allowed
- Transport between the depot of the pickup network and the depot of the delivery network (*longhaul transport*)

  ↳ Not part of the problem

The Double TSP with Multiple Stacks (DTSPMS)

- **Input:**
  - Set of orders (pickup location and delivery location)
- **Output:**
  - pickup route in the first graph
  - delivery route in the second graph
  - loading plan: how to store items
- **Objective:**
  - Minimize the sum of travelled distances

Example



Mathematical Model

$$D = \{1, \cdots, n\}$$ n orders

$$P = \{1, \cdots, m\}$$ m available stacks

maximum capacity: Q

$$N_*^\delta = \{1, \cdots, n\}$$ n nodes in each graph

$$\forall \delta \in \{1, 2\}$$

$$N^\delta = N_*^\delta \cup \{0\}$$

depot

Variables

- **Routing variables:**

$$x_{ij}^{\delta} = \begin{cases} 1 & \text{if } j \text{ follows } i \text{ in route } \delta \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j \in N^{\delta}$$

- **Precedence variables:**

$$y_{ij}^{\delta} = \begin{cases} 1 & \text{if } j \text{ is visited after } i \text{ in route } \delta \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j \in N_*^{\delta}$$

- **Loading variables:**

$$z_{ip} = \begin{cases} 1 & \text{if order } i \text{ is asigned to stack } p \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in D, \forall p \in P$$

Constraints

- **Flow balance:**

$$\sum_{i \in V^\delta} x_{ij}^\delta = 1 \qquad \forall j \in N^\delta,\ \forall \delta$$

$$\sum_{j \in V^\delta} x_{ij}^\delta = 1 \qquad \forall i \in N^\delta,\ \forall \delta$$

- **Precedence:**

$$y_{ij}^\delta + y_{ji}^\delta = 1 \qquad \forall i, j \in N_*^\delta,\ i \neq j,\ \forall \delta$$

$$y_{ik}^\delta + y_{kj}^\delta \leq y_{ij}^\delta + 1 \qquad \forall i, j, k \in N_*^\delta,\ \forall \delta$$

$$x_{ij}^\delta \leq y_{ij}^\delta \qquad \forall i, j \in N_*^\delta,\ \forall \delta$$

Constraints

- **LIFO principle:**

$$y_{ij}^{1} + z_{ir} + z_{jr} \leq 3 - y_{ij}^{2} \qquad \forall i, j \in N_{*}^{\delta},\ \forall p \in P$$

- **Loading plan:**

$$\sum_{p \in P} z_{ip} = 1 \qquad \forall i \in D$$

$$\sum_{i \in D} z_{ip} \leq Q \qquad \forall p \in P$$

- **Binary variables:**

$$x, y, z \in \{0, 1\}$$

Objective function

$$\min \sum_{\substack{i,j \in V^\delta \\ \delta \in \{1,2\}}} c_{ij}^\delta \cdot x_{ij}^\delta$$

Minimize the sum of travelled distance in both graphs

Exact solutions

- DTSPMS is a NP-hard problem, more difficult than TSP
- Mathematical model can be solved up to 12 orders in reasonable time

→ **HEURISTICS**

Representation of solutions

- **Before operator**

$$\boxed{S = (\pi_1, \pi_2, \lambda)}$$

$\pi_1 \Rightarrow$ route 1 (pickups) $\pi_2 \Rightarrow$ route 2 (deliveries)

$\lambda \Rightarrow$ loading plan

- **After operator**

$$\boxed{\hat{S} = (\hat{\pi}_1, \hat{\pi}_2, \hat{\lambda})}$$

$\hat{\pi}_1 \Rightarrow$ route 1 (pickups) $\hat{\pi}_2 \Rightarrow$ route 2 (deliveries)

$\hat{\lambda} \Rightarrow$ loading plan

Neighborhood structures

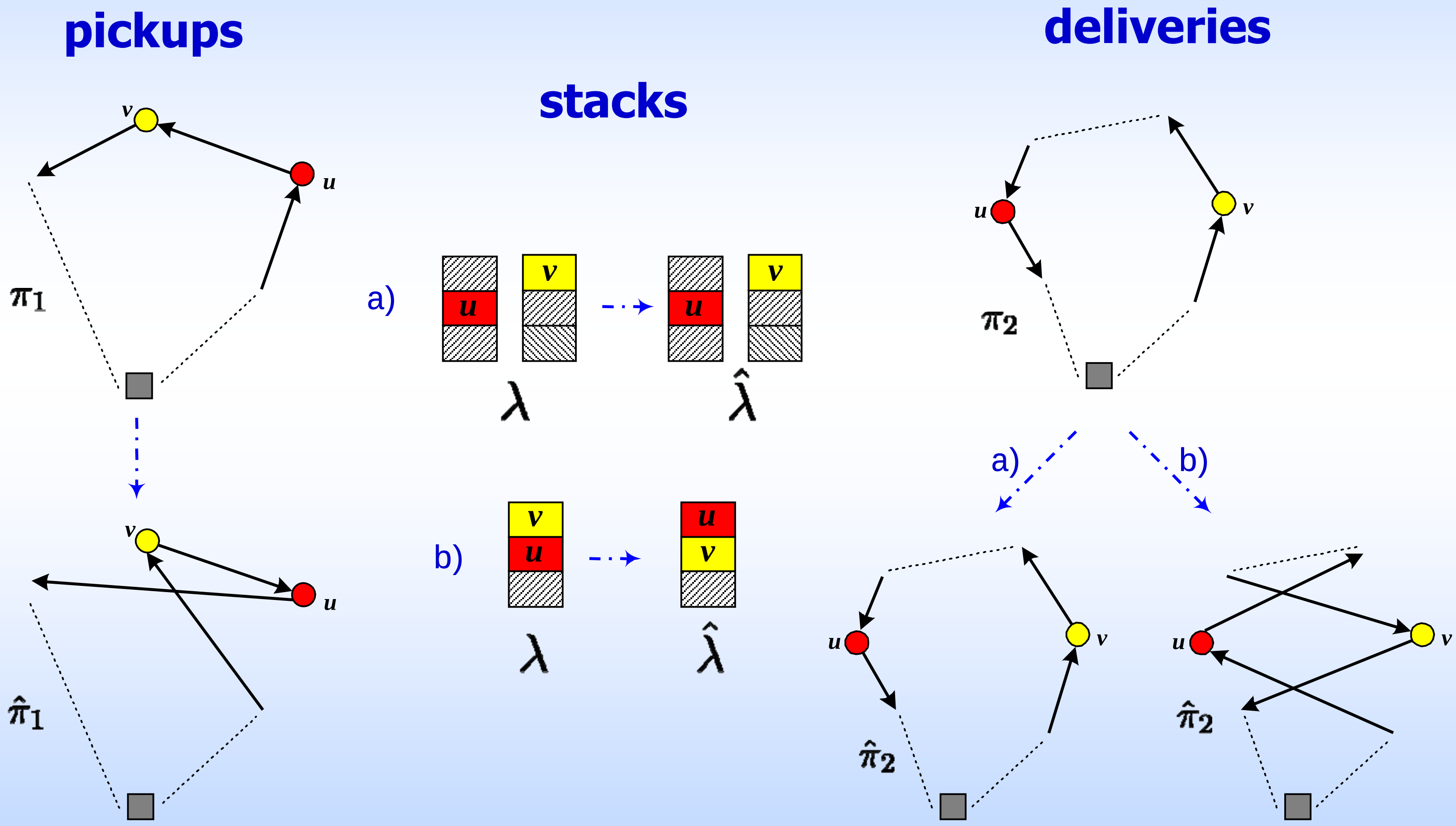
- **Route Swap (RS)**
- **Complete Swap (CS)**

2 operators borrowed from *Hanne L. Petersen (2006)* Tabu Search, Simulated Annealing

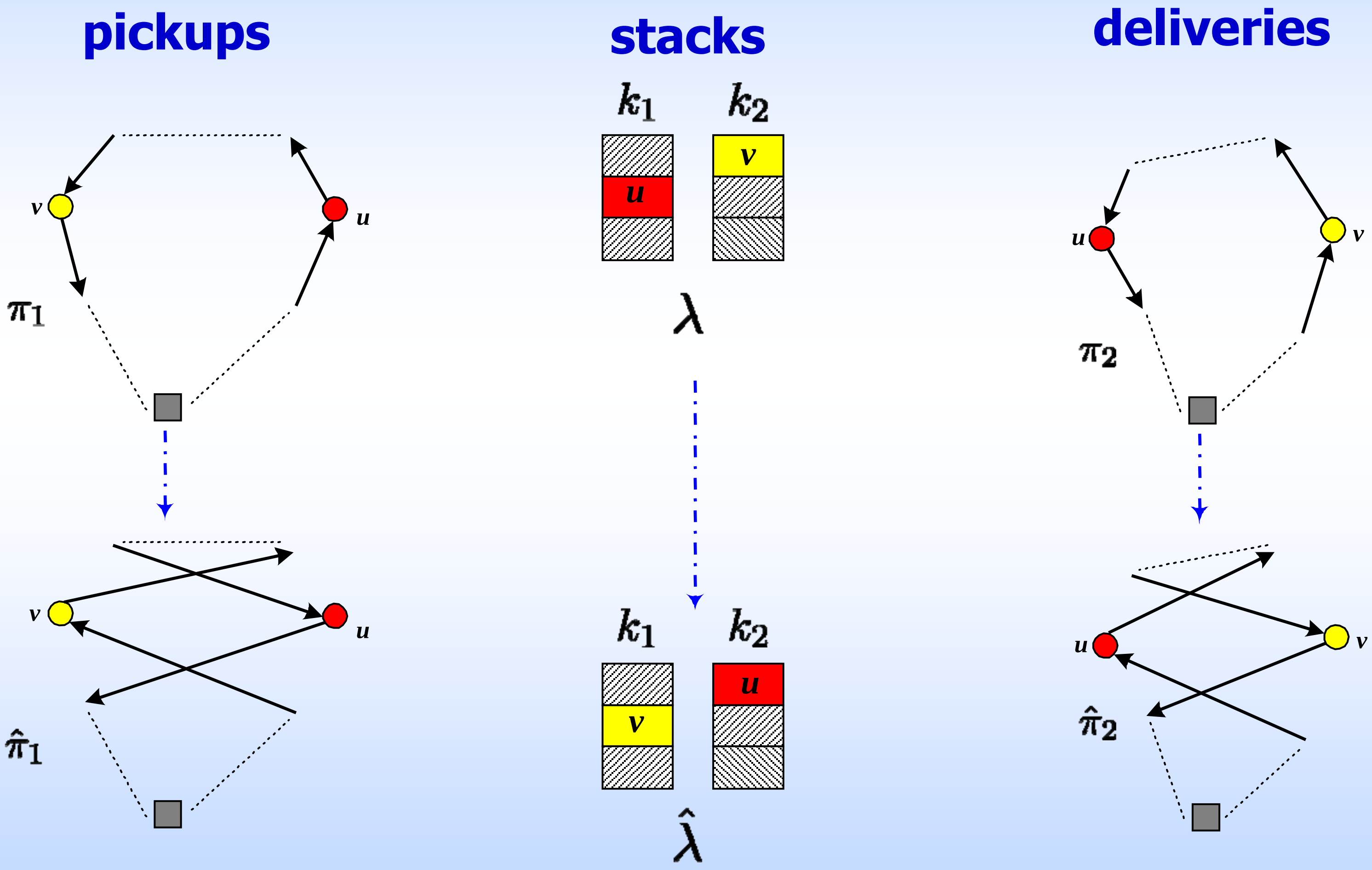
- **In-Stack Swap (ISS)**
- **Reinsertion (R)**
- ***k* – Route Permutation (*k*-RP)**
- ***k* – Stack Permutation (*k*-SP)**

4 new operators

Route Swap

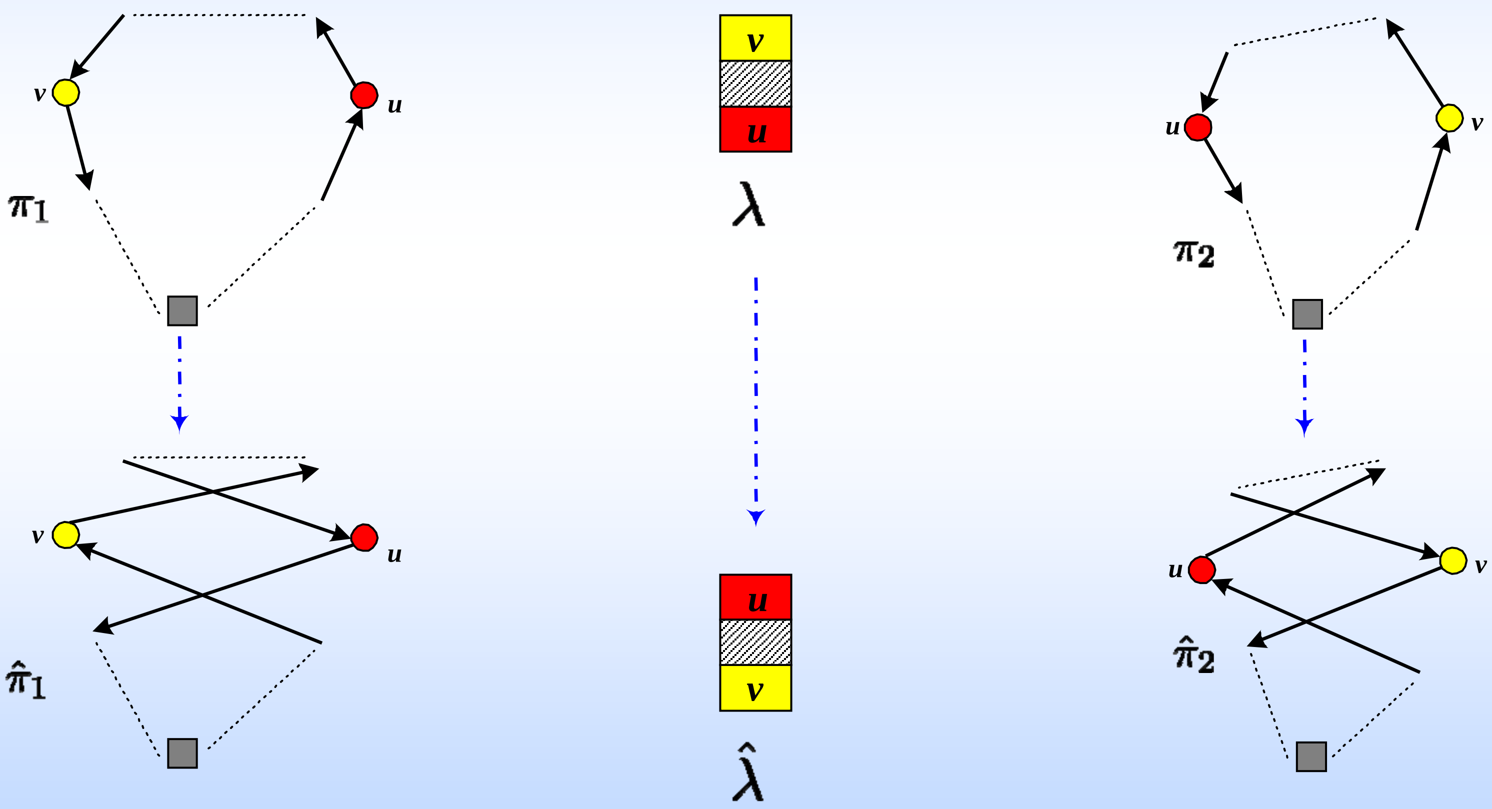

pickups
stacks
deliveries
v
u
π_1
$\hat{\pi}_1$
a)
b)
u
v
λ
$\hat{\lambda}$
π_2
$\hat{\pi}_2$

Complete Swap



In-Stack Swap

pickups **stacks** **deliveries**



Reinsertion

- **Choose:**
  - An order u
  - A stack k
  - A position i^* in route 1
  - A position j^* in route 2

Order u is assigned to stack k and moved to position i^* in route 1 and position j^* in route 2

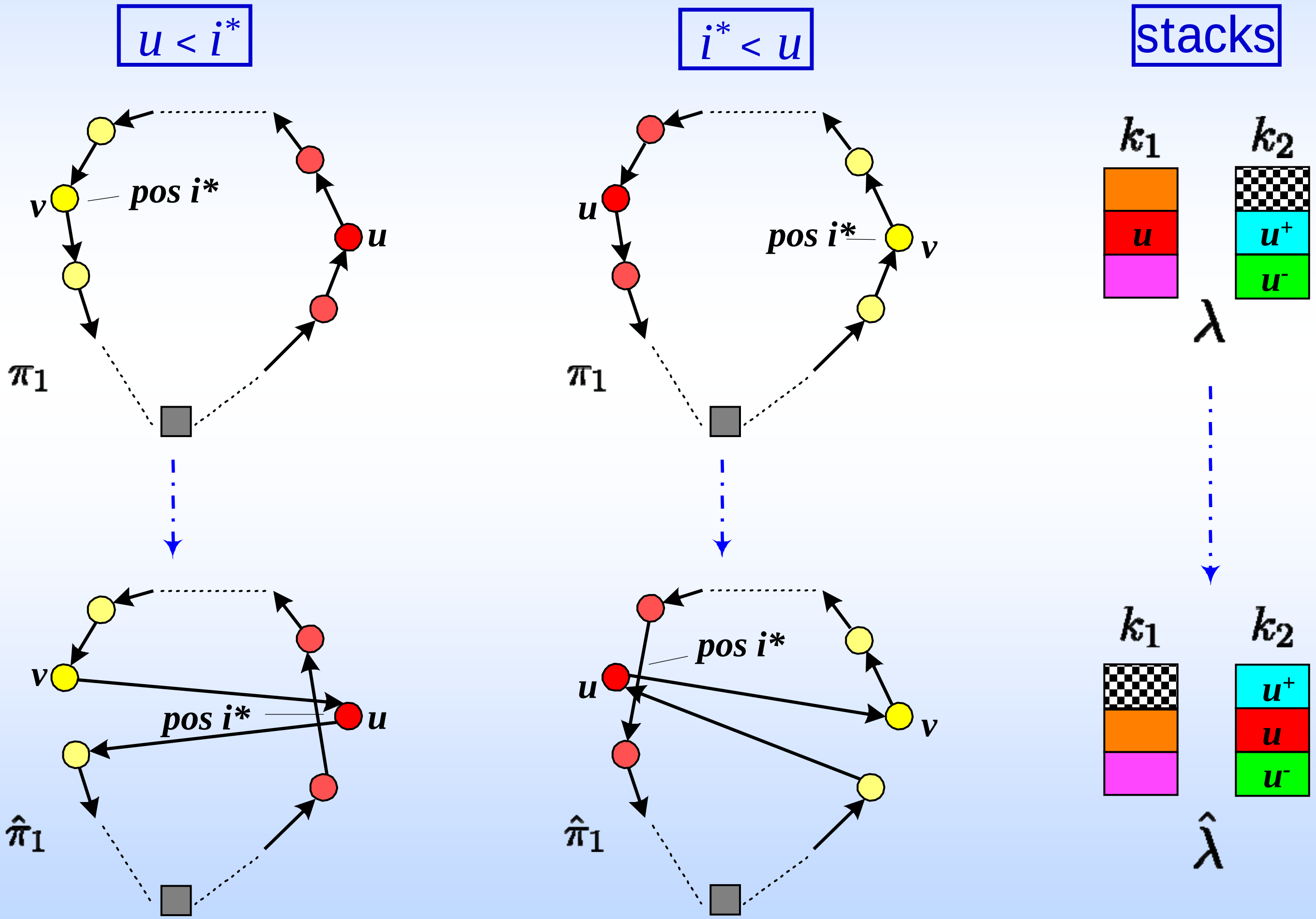
- Stack k must have room for new items
- Positions i^* and j^* have to be compatible
- The way reinsertion is implemented in route 1 (route 2) depends on the relative order between the position of u in the corresponding route and i^* (j^*).

Reinsertion

- Move order u to position i^* in route 1
- Reassign order u from stack k_1 to k_2
- Reinsertion in route 2 is done the same way

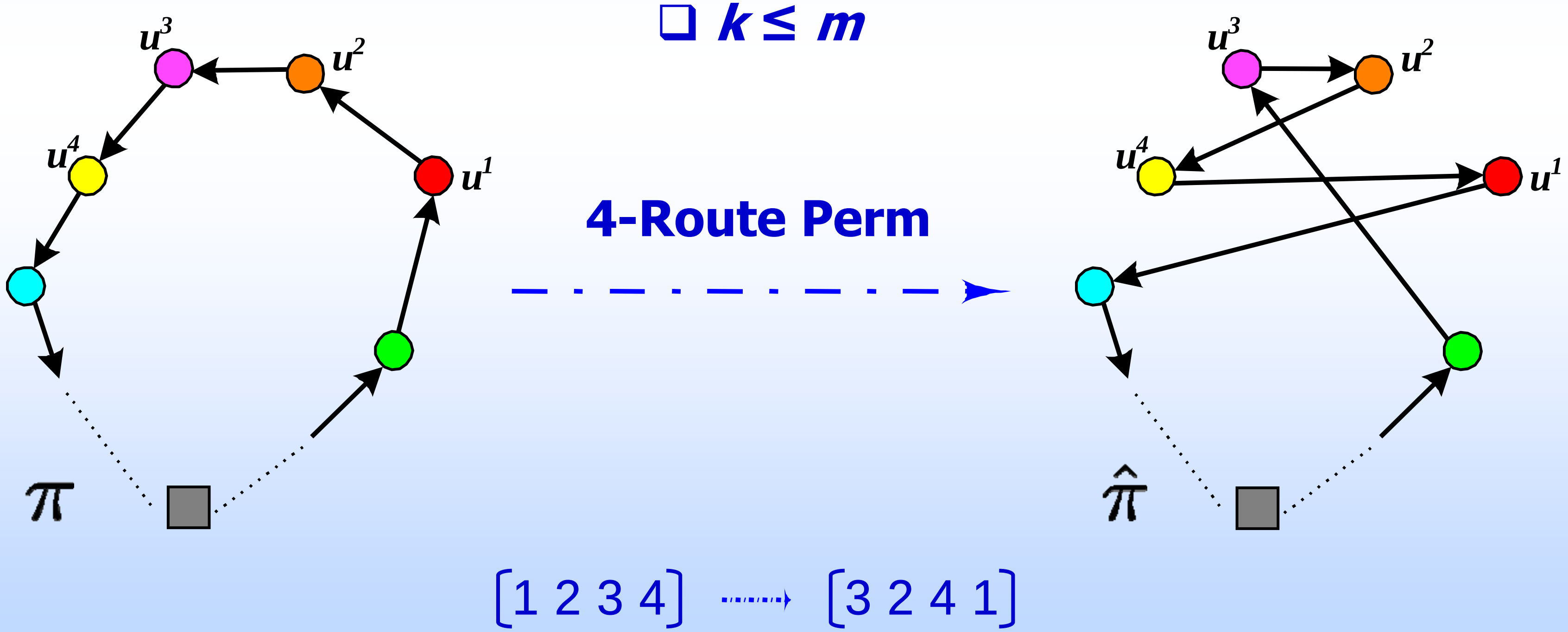
Reinsertion

- Move order u to position i^* in route 1
- Reassign order u from stack k_1 to k_2
- Reinsertion in route 2 is done the same way



k - Route Permutation

- Choose *k* **consecutive** orders in one route, **assigned to different stacks**, and permute them
- The other route and the loading plan do not change
- $k \leq m$

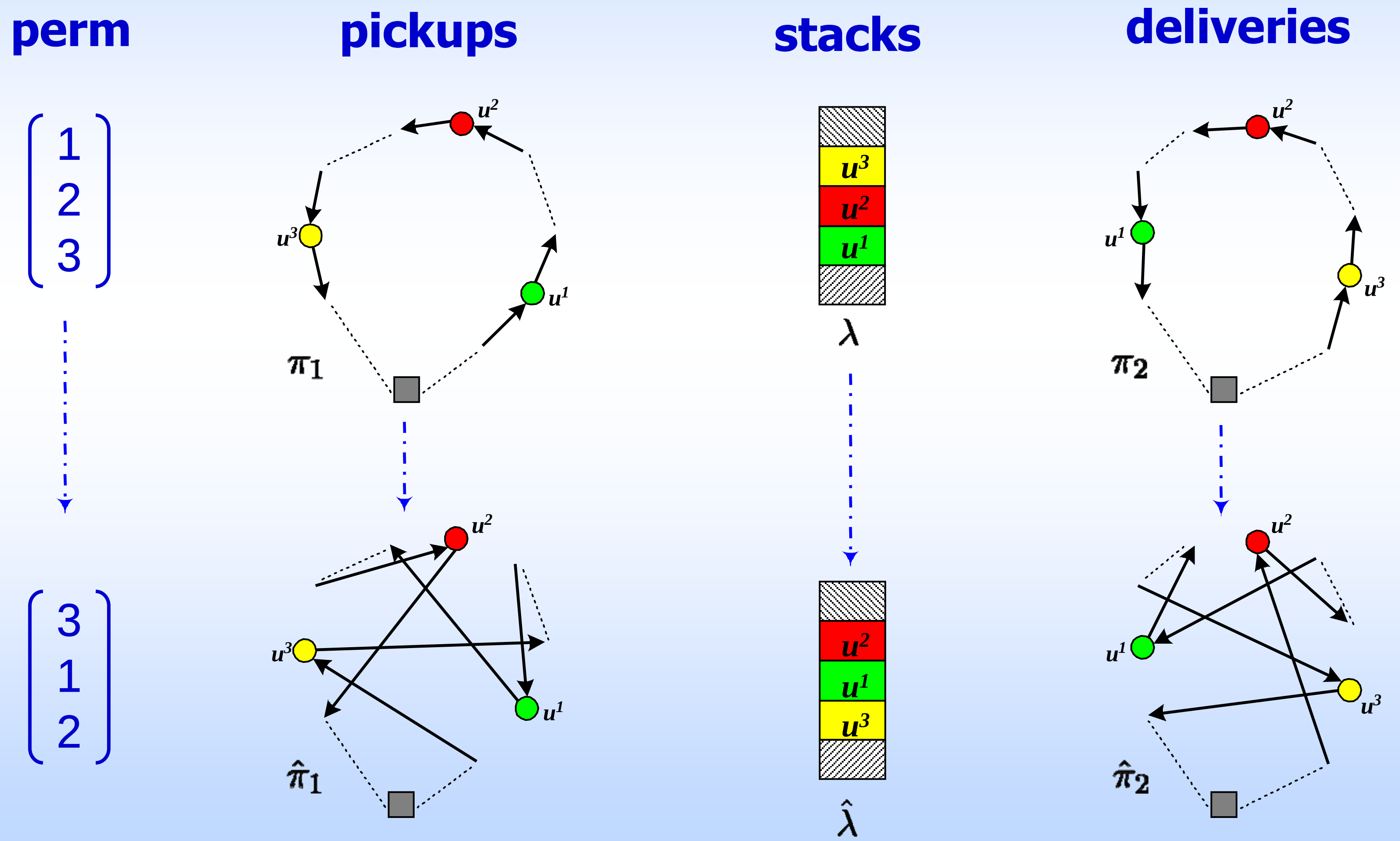


k - Stack Permutation

- Choose *k* **consecutive** orders in one stack and permute them

k - Stack Permutation

- Choose *k* **consecutive** orders in one stack and permute them



Variable Neighborhood Search (VNS)

- Metaheuristic introduced by Hansen y Mladenovic (1997)
- Main idea: use **different neighborhood structures** and **change between them** when finding a local optimum
- Local optimum with respect to one neighborhood structure is not necessarily local optimum with respect to another one
- Global optimum is local optimum with respect to every neighborhood structure

↳ Using different neighborhood structures usually provides local optima closer to the global optimum

Variable Neighborhood Descent (VND)

1. Initialization:
   1.1. Construct initial solution S
   1.2. $A = \{\Delta_1, \cdots, \Delta_{j_A}\} = \{RS, CS, ISS, R, k\text{-}RP, k\text{-}SP\}$
   1.3. $j \leftarrow 1$
2. Repeat until $j = j_A$:
   2.1. Local Search in Δ_j with initial solution $S \rightsquigarrow S'$
   2.2. If $z(S') < z(S) \implies S = S', j = 1$
   Otherwise $\implies j = j + 1$
3. S is a local minimum with respect to every neighborhood structure in A.

General Variable Neighborhood Search (GVNS)

1. Initialization:
   - 1.1. Construct initial solution S and define A for VND
   - 1.2. $B = \{\Omega_1, \cdots, \Omega_{j_B}\} = \{RS, CS, ISS, R\}$
   - 1.3. $n \leftarrow 1$
2. Repeat until $n = n_{max}$:
   - 2.1. $j \leftarrow 1$
   - 2.2. Perturbation: choose $S' \in \Omega_j(S)$ at random
   - 2.3. Perform VND with S' and $A \rightsquigarrow S''$
   - 2.4. If $z(S'') < z(S) \implies S = S''$, go to step 2.1.
   - 2.5. If $j < j_B \implies j = j + 1$, go to step 2.2.
   - 2.6. $n \leftarrow n + 1$

Test Problems

- Testing has been performed on two sets of 10 randomly generated problems with 33 orders, 3 available stacks with capacity of 11 units.

  Euro pallets

- Best results obtained using Simulated Annealing

(Hanne L. Petersen, 2006)

- Algorithm used: **GVNS**

  - 4 operators for Perturbation: RS, CS, ISS, R
  - 6 operators for VND : RS, CS, ISS, R, *k*-RS, *k*-SS

Set 0

$$\frac{Heur}{Best} \cdot 100$$

			10 seconds		3 minutes	
Problem	LB	Best	SA	GVNS	SA	GVNS
R00	914	1069	25	11.6	12	6.2
R01	875	1072	13	7.6	4	3.8
R02	935	1070	18	9.7	10	8.1
R03	961	1111	21	10.0	11	3.9
R04	933	1090	21	8.7	7	4.6
R05	898	1055	17	4.6	10	3.2
R06	998	1118	18	8.3	10	2.0
R07	962	1118	21	11.1	9	7.7
R08	976	1111	21	12.4	11	9.0
R09	982	1106	13	6.6	6	5.0
average			19	9.1	9	5.3

Set 1

$$\frac{Heur}{Best} \cdot 100$$

			10 seconds		3 minutes	
Problem	LB	Best	SA	GVNS	SA	GVNS
R10	901	1021	23	13.1	14	7.1
R11	892	1040	19	5.1	6	5.3
R12	984	1113	21	10.7	12	4.9
R13	956	1102	21	3.4	6	4.4
R14	879	1059	19	9.4	9	4.9
R15	985	1162	19	7.3	8	4.8
R16	967	1105	19	9.6	9	6.0
R17	946	1096	21	10.5	10	8.1
R18	1008	1180	15	4.5	8	2.1
R19	938	1123	14	6.9	8	3.9
average			19	8.1	9	5.1

Comparison between Operators

GVNS (% using all) − (% removing one)

	Time	All	RS	CS	ISS	R	k-RP	k-SP
Set 0	2 s	10.9%	-1.7	6.7	0.1	17.9	-1.3	0.1
	10 s	9.1%	-0.9	6.4	0.1	14.8	0.1	0.2
Set 1	2 s	9.7%	1.0	6.5	0.7	19.1	0.4	0.2
	10 s	8.1%	0.0	8.7	0.1	18.1	0.0	0.1

- **More operators** $\not\Longrightarrow$ **Better solution**
- **Reinsertion is the most effective neighborhood structure**

Further work

- Generate initial solutions using different methods
- Use different versions of Variable Neighborhood Search algorithms
- Try other metaheuristics: GRASP, Genetic algorithms, Ant Colony Systems, etc.
- Examine exact approaches

Department of Statistics and Operational Research I
Universidad Complutense de Madrid. Spain

THANK YOU FOR

YOUR ATTENTION

Gregorio Tirado Domínguez Email: *gregoriotd@mat.ucm.es*