Synchronized Pickup and Delivery Problems with Connecting FIFO Stack

M. Barbato[1] **A. Ceselli** **N. Facchinetti**

OptLab – Dept. of Computer Science
University of Milan
AD-COM Project[2]

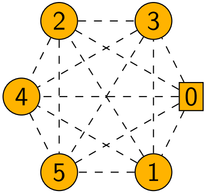
CTW 2020 – 18th Cologne-Twente Workshop on Graphs and Combinatorial Optimization

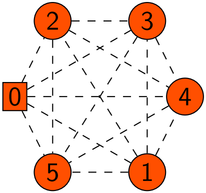
September 14-16, 2020

[1] michele.barbato@unimi.it
[2] https://ad-com.net/

Context

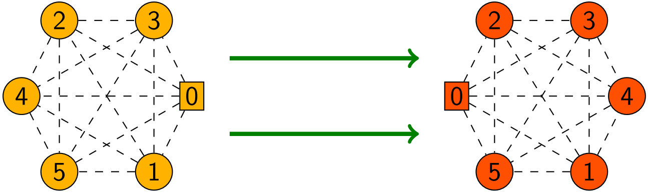
- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)





Context

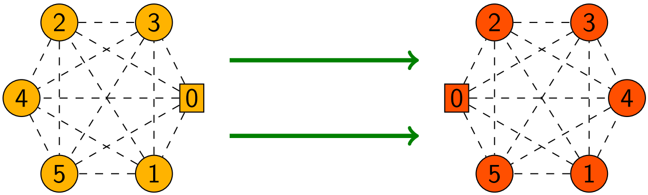
- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network



Context

- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

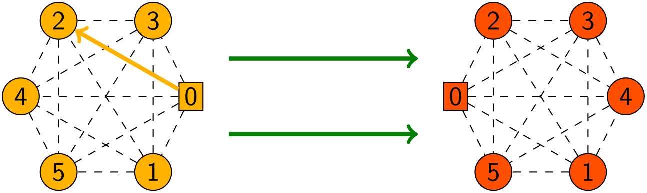
$$k_1 = k_2 = 3$$



Context

- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network $\rightarrow$ Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

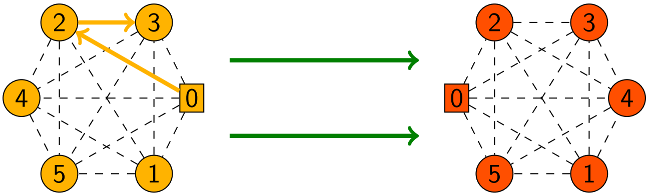
$$k_1 = k_2 = 3$$



Context

- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

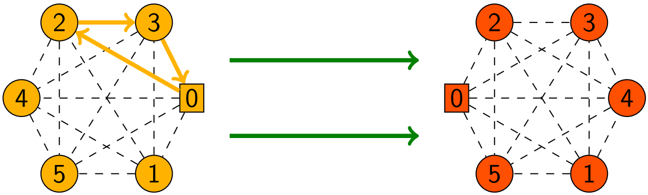
$k_1 = k_2 = 3$



Context

- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network $\rightarrow$ Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

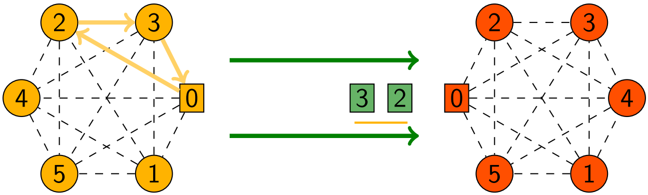
$k_1 = k_2 = 3$



Context

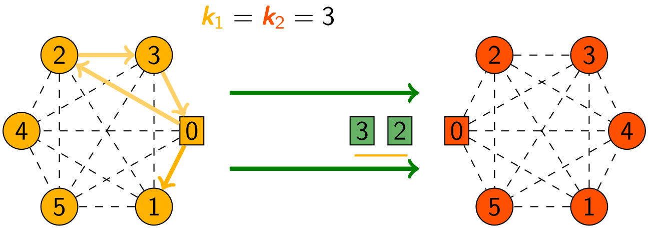
- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

$k_1 = k_2 = 3$



Context

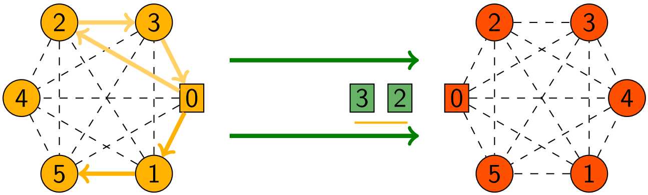
- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**



Context

- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network $\rightarrow$ Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

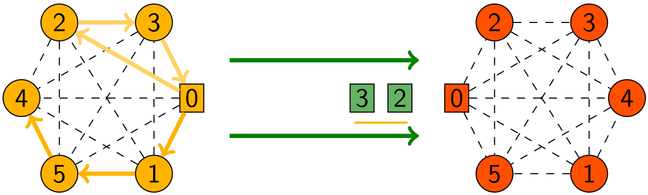
$k_1 = k_2 = 3$



Context

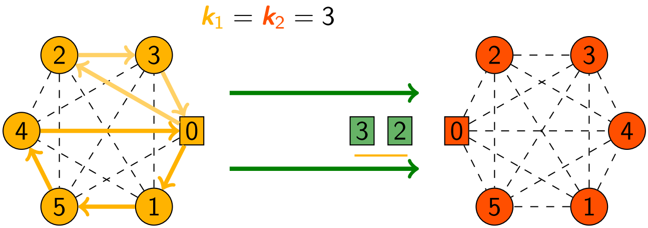
- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network $\rightarrow$ Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

$k_1 = k_2 = 3$



Context

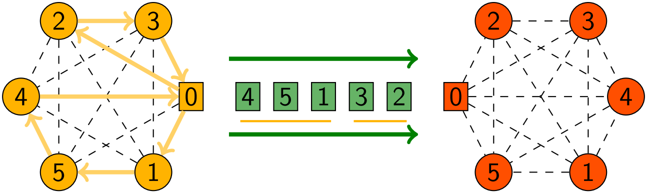
- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**



Context

- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

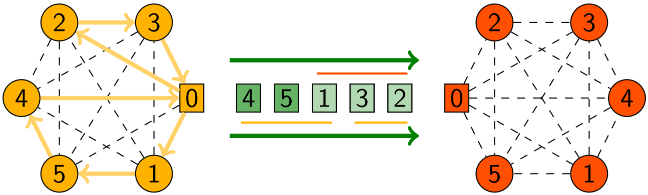
$k_1 = k_2 = 3$



Context

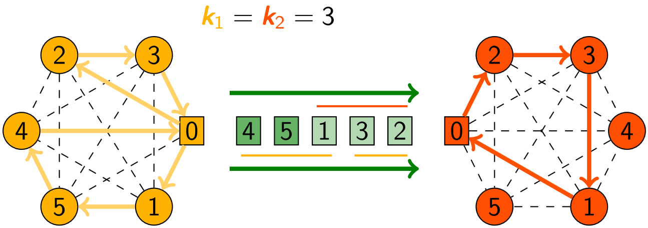
- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

$k_1 = k_2 = 3$



Context

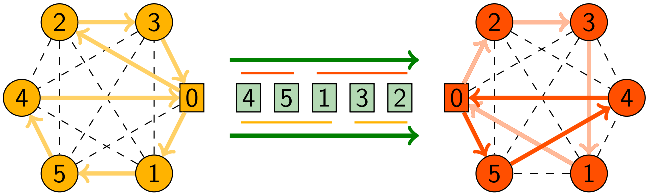
- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**



Context

- Research inspired by a collaboration with an **industrial partner**
- **Automated warehouse** characteristics:
  - **Pickup network** (item storage)
  - **Delivery network** (item processing)
  - **FIFO stack**: Pickup network → Delivery network
  - 2 vehicles of **small capacities** k_1 and k_2 idle at 0
  - **1-to-1 pickup and delivery** minimizing the **routing cost**

$k_1 = k_2 = 3$



Notation and Definitions

Basic data

- $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{A})$ complete digraph
- $\boldsymbol{c}^1, \boldsymbol{c}^2 \colon \boldsymbol{A} \to \mathbb{R}_+$ **cost** functions
- $\boldsymbol{k}_1, \boldsymbol{k}_2$ vehicle **capacities**

Notation and Definitions

Basic data

- $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{A})$ complete digraph
- $\boldsymbol{c}^1, \boldsymbol{c}^2 \colon \boldsymbol{A} \to \mathbb{R}_+$ **cost** functions
- $\boldsymbol{k}_1, \boldsymbol{k}_2$ vehicle **capacities**

Definitions

- **pickup network:** $\boldsymbol{D}^1 = (\boldsymbol{G}, \boldsymbol{c}^1)$
- **delivery network:** $\boldsymbol{D}^2 = (\boldsymbol{G}, \boldsymbol{c}^2)$

Notation and Definitions

Basic data

- $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{A})$ complete digraph
- $\boldsymbol{c}^1, \boldsymbol{c}^2 \colon \boldsymbol{A} \to \mathbb{R}_+$ **cost** functions
- $\boldsymbol{k}_1, \boldsymbol{k}_2$ vehicle **capacities**

Definitions

- **pickup network:** $\boldsymbol{D}^1 = (\boldsymbol{G}, \boldsymbol{c}^1)$
- **delivery network:** $\boldsymbol{D}^2 = (\boldsymbol{G}, \boldsymbol{c}^2)$
- **trip:** $\boldsymbol{t} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{\boldsymbol{k}})$ simple directed cycle (from/to 0)

Notation and Definitions

Basic data

- $G = (V, A)$ complete digraph
- $c^1, c^2 \colon A \to \mathbb{R}_+$ **cost** functions
- k_1, k_2 vehicle **capacities**

Definitions

- **pickup network:** $D^1 = (G, c^1)$
- **delivery network:** $D^2 = (G, c^2)$
- **trip:** $t = (v_1, v_2, \ldots, v_k)$ simple directed cycle (from/to 0)
    - $v_j \neq 0$ for all $j = 1, 2, \ldots, k$

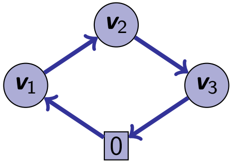
Notation and Definitions

Basic data

- $G = (V, A)$ complete digraph
- $c^1, c^2 \colon A \to \mathbb{R}_+$ **cost** functions
- k_1, k_2 vehicle **capacities**

Definitions

- **pickup network:** $D^1 = (G, c^1)$
- **delivery network:** $D^2 = (G, c^2)$
- **trip:** $t = (v_1, v_2, \ldots, v_k)$ simple directed cycle (from/to 0)
  - $v_j \neq 0$ for all $j = 1, 2, \ldots, k$
  - for a trip in D^i:
    - **feasibility:** $k \leq k_i$
    - **trip cost:** $c^i(t) = c^i(0, v_1) + \sum_{i=1}^{k-1} c^i(v_i, v_{i+1}) + c^i(v_k, 0)$



A General Problem Definition

A **Synchronized Pickup and Delivery Problem with FIFO stack** (**SPDP-FS**) is

$$
\begin{aligned}
\min \quad & \boldsymbol{c}^1(\boldsymbol{P}) + \boldsymbol{c}^2(\boldsymbol{D}) \\
\text{s. t.} \quad & \\
& \boldsymbol{P} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_\ell) \qquad \text{with } \boldsymbol{p_i} \text{ feasible trips partitioning } \boldsymbol{V} \setminus 0 \\
& \boldsymbol{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2, \dots, \boldsymbol{d_m}) \qquad \text{with } \boldsymbol{d_j} \text{ feasible trips partitioning } \boldsymbol{V} \setminus 0 \\
& (\boldsymbol{P}, \boldsymbol{D}) \text{ satisfies the } \textbf{FIFO}
\end{aligned}
$$

The No-Permutation Variant

No-Permutation Description:

- items on the **FIFO** stack respecting the **pickup order**
- items **delivered** in the order on the **FIFO** stack

The No-Permutation Variant

No-Permutation Description:

- items on the **FIFO** stack respecting the **pickup order**
- items **delivered** in the order on the **FIFO** stack

Definition

Let $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_\ell)$ be a sequence of **trips**.
The $\boldsymbol{T}$-sequence is the sequence of vertices in $\boldsymbol{V} \setminus 0$ in the order they appear in $\boldsymbol{T}$.

The No-Permutation Variant

No-Permutation Description:

- items on the **FIFO** stack respecting the **pickup order**
- items **delivered** in the order on the **FIFO** stack

Definition

Let $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_\ell)$ be a sequence of **trips**.
The $\boldsymbol{T}$-sequence is the sequence of vertices in $\boldsymbol{V} \setminus 0$ in the order they appear in $\boldsymbol{T}$.

Example

- $\boldsymbol{T} = ((2,3),(1,5,4))$
- **$\boldsymbol{T}$-sequence**$= (2,3,1,5,4)$

The No-Permutation Variant

No-Permutation Description:

- items on the **FIFO** stack respecting the **pickup order**
- items **delivered** in the order on the **FIFO** stack

Definition

Let $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_\ell)$ be a sequence of **trips**.
The $\boldsymbol{T}$-sequence is the sequence of vertices in $\boldsymbol{V} \setminus 0$ in the order they appear in $\boldsymbol{T}$.

No-Permutation SPDP-FS

- $(\boldsymbol{P}, \boldsymbol{D})$ **solution** $\Leftrightarrow$ $\boldsymbol{P}$**-sequence** $\equiv$ $\boldsymbol{D}$**-sequence**

The No-Permutation Variant

No-Permutation Description:

- items on the **FIFO** stack respecting the **pickup order**
- items **delivered** in the order on the **FIFO** stack

Definition

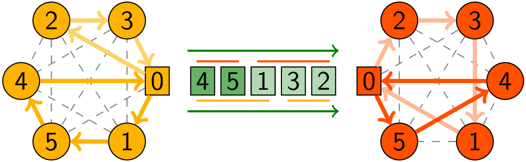
Let $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_\ell)$ be a sequence of trips.
The $\boldsymbol{T}$-sequence is the sequence of vertices in $\boldsymbol{V} \setminus 0$ in the order they appear in $\boldsymbol{T}$.

No-Permutation SPDP-FS

- $(\boldsymbol{P}, \boldsymbol{D})$ **solution** $\Leftrightarrow$ $\boldsymbol{P}$**-sequence** $\equiv$ $\boldsymbol{D}$**-sequence**

Example

$\boldsymbol{P} = ((2,3),(1,5,4))$
$\boldsymbol{D} = ((2,3,1),(5,4))$



The Permutation Variants

Permutation Description

- Each pickup trip unloads a **batch** of items on the **FIFO stack**
- Each delivery trip loads a **batch** of items from the **FIFO stack**
- The **order** inside **batches** is **arbitrary**
- The pickup and delivery batches satisfy the **FIFO**

The Permutation Variants

Permutation Description

- Each pickup trip unloads a **batch** of items on the **FIFO stack**
- Each delivery trip loads a **batch** of items from the **FIFO stack**
- The **order** inside **batches** is **arbitrary**
- The pickup and delivery batches satisfy the **FIFO**

Pickup-Permutation Description

- Each pickup trip unloads a **batch** of items on the **FIFO stack**
- The **order** inside **pickup batches** is **arbitrary**
- Items **delivered** in the order on the **FIFO** stack

The Permutation Variants

Permutation Description

- Each pickup trip unloads a **batch** of items on the **FIFO stack**
- Each delivery trip loads a **batch** of items from the **FIFO stack**
- The **order** inside **batches** is **arbitrary**
- The pickup and delivery batches satisfy the **FIFO**

Pickup-Permutation Description

- Each pickup trip unloads a **batch** of items on the **FIFO stack**
- The **order** inside **pickup batches** is **arbitrary**
- Items **delivered** in the order on the **FIFO** stack

Delivery-Permutation Description

- Items on the **FIFO** stack respecting the **pickup order**
- Each delivery trip loads a **batch** of items from the **FIFO stack**
- The **order** inside **delivery batches** is **arbitrary**

The No-Overlap Requirement

No-Overlap Description

- Each **delivery batch** is **contained** in one **pickup batch**
- The **other requirements** stay **valid**

The No-Overlap Requirement

No-Overlap Description

- Each **delivery batch** is **contained** in one **pickup batch**
- The **other requirements** stay **valid**

Definition

Let $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{\boldsymbol{k}})$ be a sequence of **trips**
We write $\boldsymbol{V}(\boldsymbol{t}_{\boldsymbol{i}})$ to indicate the vertices in $\boldsymbol{t}_{\boldsymbol{i}}$

The No-Overlap Requirement

No-Overlap Description

- Each **delivery batch** is **contained** in one **pickup batch**
- The **other requirements** stay **valid**

Definition

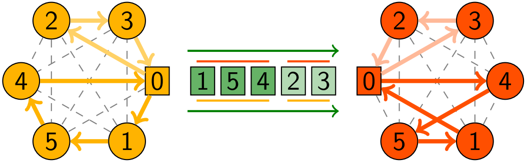
Let $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{\boldsymbol{k}})$ be a sequence of **trips**
We write $\boldsymbol{V}(\boldsymbol{t}_{\boldsymbol{i}})$ to indicate the vertices in $\boldsymbol{t}_{\boldsymbol{i}}$

SPDP-FS with No-Overlap

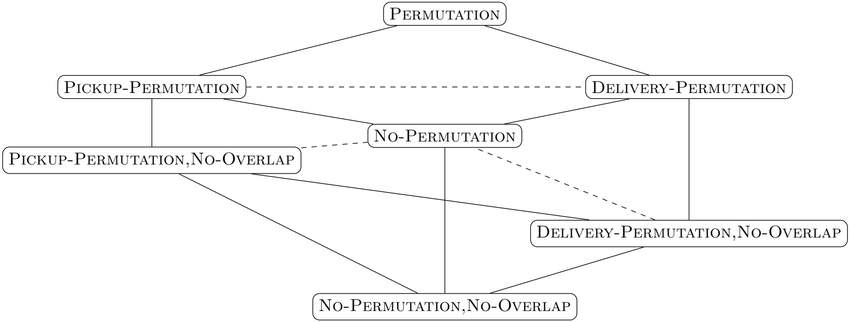
$(\boldsymbol{P}, \boldsymbol{D})$ satisfies the **requirement** **if** for all $\boldsymbol{i}$ there is $\boldsymbol{j}$ s.t. $\boldsymbol{V}(\boldsymbol{d}_{\boldsymbol{i}}) \subseteq \boldsymbol{V}(\boldsymbol{p}_{\boldsymbol{j}})$

Example

$\boldsymbol{P} = ((2, 3), (1, 5, 4))$
$\boldsymbol{D} = ((3, 2), (4, 5, 1))$



Variant Hierarchy



Complexity Results

Proposition. All **SPDP-FS** variants with **No-Overlap** requirement are solvable in **polynomial time** if $\boldsymbol{k}_1, \boldsymbol{k}_2 \in \{1, 2\}$.

Complexity Results

Proposition. All **SPDP-FS** variants with **No-Overlap** requirement are solvable in **polynomial time** if $k_1, k_2 \in \{1, 2\}$.

Proof. (Sketch)

- **Preprocess** all ways to pickup and deliver item **singletons** and item **pairs** and keep the best ones
- **Choose** the item **singletons** and **pairs** using a **perfect matching**

Complexity Results

Proposition. All **SPDP-FS** variants with **No-Overlap** requirement are solvable in **polynomial time** if $k_1, k_2 \in \{1, 2\}$.
Proof. (Sketch)

- **Preprocess** all ways to pickup and deliver item **singletons** and item **pairs** and keep the best ones
- **Choose** the item **singletons** and **pairs** using a **perfect matching**

Proposition. **All SPDP-FS** variants are **NP-hard** if k_1 and k_2 are part of the input.

Complexity Results

Proposition. All **SPDP-FS** variants with **No-Overlap** requirement are solvable in **polynomial time** if $k_1, k_2 \in \{1, 2\}$.
Proof. (Sketch)

- **Preprocess** all ways to pickup and deliver item **singletons** and item **pairs** and keep the best ones
- **Choose** the item **singletons** and **pairs** using a **perfect matching**

Proposition. **All SPDP-FS** variants are **NP-hard** if k_1 and k_2 are part of the input.
Proof. (Sketch)

- Let $n = |V \setminus 0|$, and choose $k_1 = n$, $k_2 = 1$ and $c^2 \equiv 0$.
- If c^1 is **metric** then **SPDP-FS** solves the **Euclidean-TSP** on $D = (G, c^1)$

No-Permutation Variants: the Splitting Subproblem

A solution to the **SPDP-FS** is **completely** described by

- ***P*** **pickup trip sequence**
- ***D*** **delivery trip sequence**
- ***F*** **item ordering** on the **FIFO stack**

No-Permutation Variants: the Splitting Subproblem

A solution to the **SPDP-FS** is **completely** described by

- ***P*** **pickup trip sequence**
- ***D*** **delivery trip sequence**
- ***F*** **item ordering** on the **FIFO stack**

In **No-Permutation** variants ***F*** partially describes ***P*** and ***D***

$\boldsymbol{F} = (1\ 2\ 3\ 4\ 5)$

No-Permutation Variants: the Splitting Subproblem

A solution to the **SPDP-FS** is **completely** described by

- ***P*** **pickup trip sequence**
- ***D*** **delivery trip sequence**
- ***F*** **item ordering** on the **FIFO stack**

In **No-Permutation** variants $\boldsymbol{F}$ partially describes $\boldsymbol{P}$ and $\boldsymbol{D}$

$\boldsymbol{F} = (1\ 2|3\ 4|5)$

$\boldsymbol{P} = ((1, 2), (3, 4), (5))$

No-Permutation Variants: the Splitting Subproblem

A solution to the **SPDP-FS** is **completely** described by

- ***P*** **pickup trip sequence**
- ***D*** **delivery trip sequence**
- ***F*** **item ordering** on the **FIFO stack**

In **No-Permutation** variants $\boldsymbol{F}$ partially describes $\boldsymbol{P}$ and $\boldsymbol{D}$

$\boldsymbol{F} = (1\ 2\ 3 | 4\ 5)$

$\boldsymbol{P} = ((1, 2), (3, 4), (5))$

$\boldsymbol{D} = ((1, 2, 3), (4, 5))$

No-Permutation Variants: the Splitting Subproblem

A solution to the **SPDP-FS** is **completely** described by

- ***P*** **pickup trip sequence**
- ***D*** **delivery trip sequence**
- ***F*** **item ordering** on the **FIFO stack**

In **No-Permutation** variants ***F*** partially describes ***P*** and ***D***

Definition

Sequences obtained from ***F*** as before are said **splitting**

No-Permutation Variants: the Splitting Subproblem

A solution to the **SPDP-FS** is **completely** described by

- $\boldsymbol{P}$ **pickup trip sequence**
- $\boldsymbol{D}$ **delivery trip sequence**
- $\boldsymbol{F}$ **item ordering** on the **FIFO stack**

In **No-Permutation** variants $\boldsymbol{F}$ partially describes $\boldsymbol{P}$ and $\boldsymbol{D}$

Definition

Sequences obtained from $\boldsymbol{F}$ as before are said **splitting**

Splitting subproblem. Given $\boldsymbol{F}$ find its pair of splittings $(\boldsymbol{P}, \boldsymbol{D})$ **minimizing** $\boldsymbol{c}^1(\boldsymbol{P}) + \boldsymbol{c}^2(\boldsymbol{D})$.

No-Permutation Variants: the Splitting Subproblem

A solution to the **SPDP-FS** is **completely** described by

- ***P*** **pickup trip sequence**
- ***D*** **delivery trip sequence**
- ***F*** **item ordering** on the **FIFO stack**

In **No-Permutation** variants $\boldsymbol{F}$ partially describes $\boldsymbol{P}$ and $\boldsymbol{D}$

Definition

Sequences obtained from $\boldsymbol{F}$ as before are said **splitting**

Splitting subproblem. Given $\boldsymbol{F}$ find its pair of splittings $(\boldsymbol{P}, \boldsymbol{D})$ **minimizing** $\boldsymbol{c}^1(\boldsymbol{P}) + \boldsymbol{c}^2(\boldsymbol{D})$.

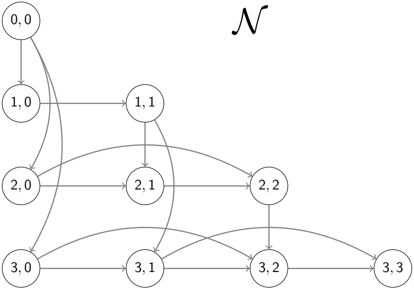
Relevance: embedding in a **2-opt heuristic** (see later)

Polynomial Algorithm for the Splitting Subproblem

No-Permutation,No-Overlap case.

Assume (wlog) $\boldsymbol{F} = (1, 2, \ldots, \boldsymbol{n})$

Approach: $(0, 0) - (\boldsymbol{n}, \boldsymbol{n})$ shortest-path in $\mathcal{N}$

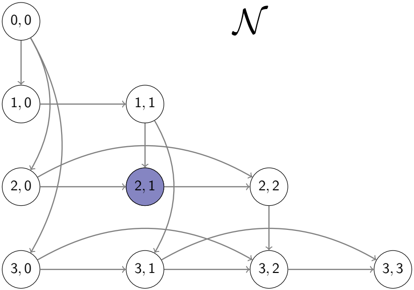


Polynomial Algorithm for the Splitting Subproblem

No-Permutation,No-Overlap case.

Assume (wlog) $\boldsymbol{F} = (1, 2, \ldots, \boldsymbol{n})$

Approach: $(0, 0) - (\boldsymbol{n}, \boldsymbol{n})$ shortest-path in $\mathcal{N}$



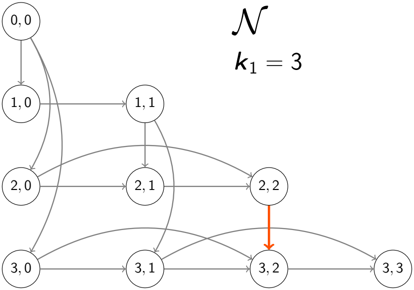
- $(\boldsymbol{i}, \boldsymbol{j})$**:** first $\boldsymbol{i}$ items picked-up and first $\boldsymbol{j}$ delivered $(\boldsymbol{i} \geq \boldsymbol{j})$

Polynomial Algorithm for the Splitting Subproblem

No-Permutation,No-Overlap case.

Assume (wlog) $F = (1, 2, \ldots, n)$

Approach: $(0, 0) - (n, n)$ shortest-path in $\mathcal{N}$



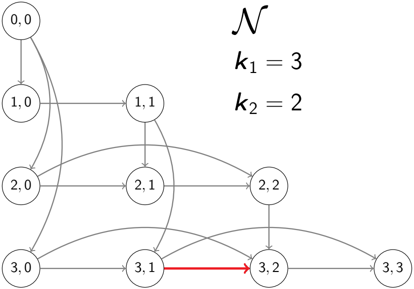
- (i, j): first i items picked-up and first j delivered ($i \geq j$)
- $(j, j) \rightarrow (i, j)$: extend P with trip $(j + 1, j + 2, \ldots, i)$

Polynomial Algorithm for the Splitting Subproblem

No-Permutation,No-Overlap case.

Assume (wlog) $F = (1, 2, \ldots, n)$

Approach: $(0, 0) - (n, n)$ shortest-path in $\mathcal{N}$



$\mathcal{N}$

$k_1 = 3$

$k_2 = 2$

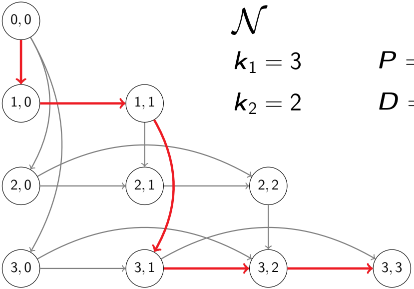
- (i, j): first i items picked-up and first j delivered $(i \geq j)$
- $(j, j) \rightarrow (i, j)$: extend P with trip $(j + 1, j + 2, \ldots, i)$
- $(i, j_1) \rightarrow (i, j_2)$: extend D with trip $(j_1 + 1, j_1 + 2 \ldots, j_2)$

Polynomial Algorithm for the Splitting Subproblem

No-Permutation,No-Overlap case.

Assume (wlog) $F = (1, 2, \ldots, n)$

Approach: $(0, 0) - (n, n)$ shortest-path in $\mathcal{N}$



$\mathcal{N}$

$k_1 = 3 \qquad P = ((1), (2, 3))$

$k_2 = 2 \qquad D = ((1), (2), (3))$

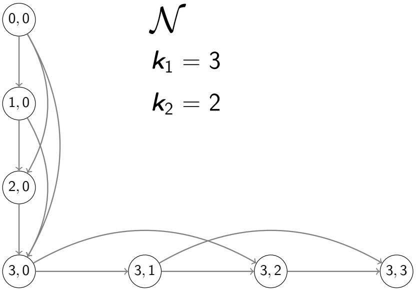
- (i, j): first i items picked-up and first j delivered ($i \geq j$)
- $(j, j) \rightarrow (i, j)$: extend P with trip $(j + 1, j + 2, \ldots, i)$
- $(i, j_1) \rightarrow (i, j_2)$: extend D with trip $(j_1 + 1, j_1 + 2 \ldots, j_2)$
- costs preprocessed in polynomial time

Polynomial Algorithm for the Splitting Subproblem

No-Permutation,Overlap case.

Assume (wlog) $\boldsymbol{F} = (1, 2, \ldots, \boldsymbol{n})$

Approach: $(0, 0) - (\boldsymbol{n}, \boldsymbol{n})$ shortest-path in $\mathcal{N}$



$\mathcal{N}$

$\boldsymbol{k}_1 = 3$

$\boldsymbol{k}_2 = 2$

Polynomial Algorithm for the Splitting Subproblem

No-Permutation,Overlap case.

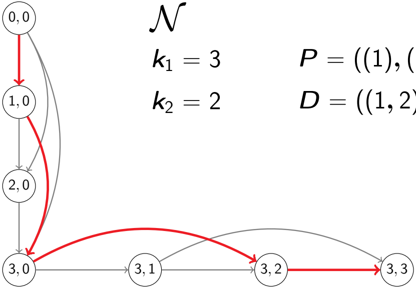
Assume (wlog) $F = (1, 2, \ldots, n)$

Approach: $(0, 0) - (n, n)$ shortest-path in $\mathcal{N}$

$\mathcal{N}$

$k_1 = 3$ $\quad P = ((1), (2, 3))$

$k_2 = 2$ $\quad D = ((1, 2), (3))$



No-Permutation Variants: A 2-Opt Heuristic

No-Permutation 2-Opt Heuristic

1) F: **TSP solution** on $D = (G, c)$ with $c(e) = c^1(e) + c^2(e)$
2) (P, D): optimal splittings of F
3) Generate the **2-opt neighborhood** of F, scored by **splitting value**
4) Choose the **best neighbor** and **repeat** 3) until no improvement

Computational Results: 2-Opt Performance

Instance Set:

- **11440 instances** adapted from the Double TSP with Multiple Stacks [PM09]
- **3 classes** of 10 instances with 33, 66, 132 items respectively
  - Classes 33/66:
    $k_1 \in \{3, 6, \ldots 33/66\}$
    $k_2 \in \{3, 6, \ldots, k_1\}$
  - Class 132:
    $k_1 \in \{6, 12, \ldots, 132\}$
    $k_2 \in \{6, 12, \ldots, k_1\}$

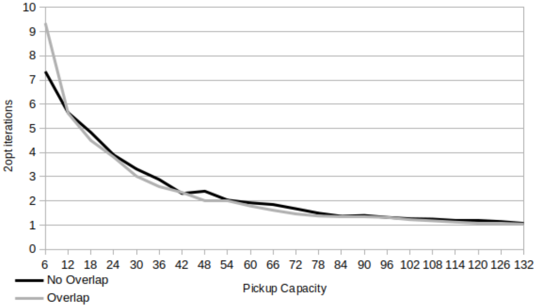
Computational Results: 2-Opt Performance

Instance Set:

- **11440 instances** adapted from the Double TSP with Multiple Stacks [PM09]
- **3 classes** of 10 instances with 33, 66, 132 items respectively
  - Classes 33/66:
    $k_1 \in \{3, 6, \ldots 33/66\}$
    $k_2 \in \{3, 6, \ldots, k_1\}$
  - Class 132:
    $k_1 \in \{6, 12, \ldots, 132\}$
    $k_2 \in \{6, 12, \ldots, k_1\}$

Specs:

- 1st TSP solved with CONCORDE [CON03]
- C++ compiled with `gcc 7.2 -O3`
- OS: `Linux`
- CPU: Intel `i7-3630QM` @2.40GHz



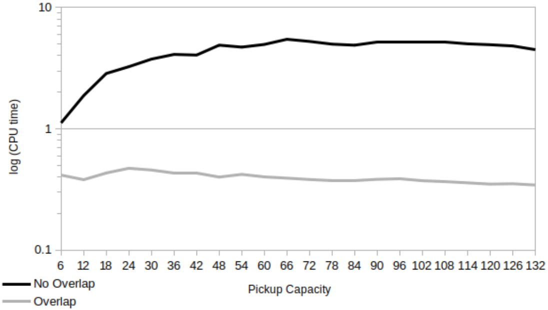
Computational Results: 2-Opt Performance

Instance Set:

- **11440 instances** adapted from the Double TSP with Multiple Stacks [PM09]
- **3 classes** of 10 instances with 33, 66, 132 items respectively
  - Classes 33/66:
    $k_1 \in \{3, 6, \ldots 33/66\}$
    $k_2 \in \{3, 6, \ldots, k_1\}$
  - Class 132:
    $k_1 \in \{6, 12, \ldots, 132\}$
    $k_2 \in \{6, 12, \ldots, k_1\}$

Specs:

- 1st TSP solved with CONCORDE [CON03]
- C++ compiled with `gcc 7.2 -O3`
- OS: `Linux`
- CPU: Intel `i7-3630QM` @2.40GHz



Computational Results: 2-Opt Quality

- IS: initial solution cost (1st splitting subproblem)
- FS: final solution cost (end of 2-opt heuristic)
- LB: $\mathbf{TSP}(\boldsymbol{D}^1)+\mathbf{TSP}(\boldsymbol{D}^2)$

Variant	Size	$(\mathrm{IS}-\mathrm{FS})/\mathrm{IS}$	$(\mathrm{FS}-\mathrm{LB})/\mathrm{LB}$
No-Overlap	33	0.84%	47.14%
	66	0.61%	54.27%
	132	0.41%	59.65%
Overlap	33	0.76%	45.09%
	66	0.52%	53.00%
	132	0.33%	59.07%

Table: Quality of heuristics and bounds. Results in average on all instances of the reported classes.

Conclusions and Perspectives

- **8 variants** of the SPDP-FS **formally characterized**
  - **Work in progress:** model the variants as MILPs
  - **Open:** consider other objectives (*e.g.*, completion time)
- Preliminary **complexity results** with fixed and non-fixed capacities
  - **Open:** extend the results to other capacity values and other variants

References I

[CON03] CONCORDE. D. L. Applegate, R. E. Bixby, V. Chvatal and W. J. Cook, 2003. http://www.math.uwaterloo.ca/tsp/concorde.html .

[PM09] Hanne L Petersen and Oli BG Madsen. The double travelling salesman problem with multiple stacks–formulation and heuristic solution approaches. *European Journal of Operational Research*, 198(1):139–147, 2009.

Appendix — Fixed Capacity Complexity

Proposition. The **SPDP-FS** variants

- **Permutation,No-Overlap**
- **No-Permutation,No-Overlap**

are solvable in **polynomial time** if $\boldsymbol{k}_1, \boldsymbol{k}_2 \in \{1, 2\}$.

Appendix — Fixed Capacity Complexity

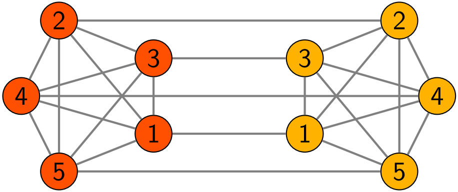
Proposition. The **SPDP-FS** variants

- **Permutation,No-Overlap**
- **No-Permutation,No-Overlap**

are solvable in **polynomial time** if $k_1, k_2 \in \{1, 2\}$.

Proof.

- 2 copies v', v'' for all $v \in V \setminus 0$



Appendix — Fixed Capacity Complexity

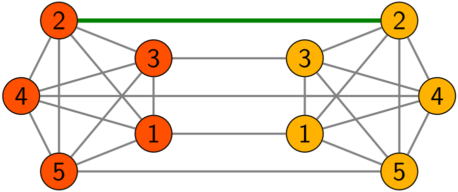
Proposition. The **SPDP-FS** variants

- **Permutation,No-Overlap**
- **No-Permutation,No-Overlap**

are solvable in **polynomial time** if $k_1, k_2 \in \{1, 2\}$.

Proof.

- 2 copies v', v'' for all $v \in V \setminus 0$
- edge (v', v'')=**trips** to collect and delivery v
- $c[v', v'']$ =cost to **collect and deliver** v



$P = \ldots(2)\ldots$
$D = \ldots(2)\ldots$

Appendix — Fixed Capacity Complexity

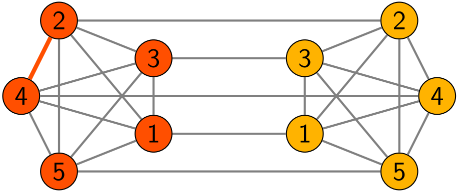
Proposition. The **SPDP-FS** variants

- **Permutation,No-Overlap**
- **No-Permutation,No-Overlap**

are solvable in **polynomial time** if $\boldsymbol{k}_1, \boldsymbol{k}_2 \in \{1, 2\}$.

Proof.

- 2 copies $\boldsymbol{v}', \boldsymbol{v}''$ for all $\boldsymbol{v} \in \boldsymbol{V} \setminus 0$
- edge $(\boldsymbol{v}', \boldsymbol{v}'')$=**trips** to collect and delivery $\boldsymbol{v}$
- $\boldsymbol{c}[\boldsymbol{v}', \boldsymbol{v}'']$ =cost to **collect and deliver** $\boldsymbol{v}$
- edge $(\boldsymbol{v}', \boldsymbol{w}')$=**best trips** to collect and deliver $\boldsymbol{v}, \boldsymbol{w}$
- $\boldsymbol{c}[\boldsymbol{v}', \boldsymbol{w}']$ =min-cost to **collect and deliver** $\boldsymbol{v}, \boldsymbol{w}$



$\boldsymbol{P} = \ldots (2, 4) \ldots$

$\boldsymbol{D} = \ldots (4, 2) \ldots$

Appendix — Fixed Capacity Complexity

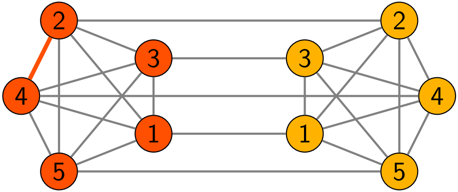
Proposition. The **SPDP-FS** variants

- **Permutation,No-Overlap**
- **No-Permutation,No-Overlap**

are solvable in **polynomial time** if $\boldsymbol{k}_1, \boldsymbol{k}_2 \in \{1, 2\}$.

Proof.

- 2 copies $\boldsymbol{v}', \boldsymbol{v}''$ for all $\boldsymbol{v} \in \boldsymbol{V} \setminus 0$
- edge $(\boldsymbol{v}', \boldsymbol{v}'')$=**trips** to collect and delivery $\boldsymbol{v}$
- $\boldsymbol{c}[\boldsymbol{v}', \boldsymbol{v}'']$ =cost to **collect and deliver** $\boldsymbol{v}$
- edge $(\boldsymbol{v}', \boldsymbol{w}')$=**best trips** to collect and deliver $\boldsymbol{v}, \boldsymbol{w}$
- $\boldsymbol{c}[\boldsymbol{v}', \boldsymbol{w}']$ =min-cost to **collect and deliver** $\boldsymbol{v}, \boldsymbol{w}$



$\boldsymbol{P} = \ldots (2, 4) \ldots$

$\boldsymbol{D} = \ldots (2), (4) \ldots$

Appendix — Fixed Capacity Complexity

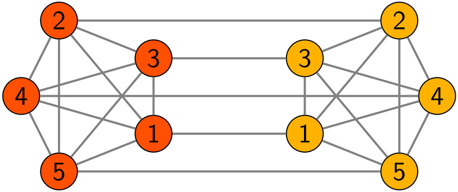
Proposition. The **SPDP-FS** variants

- **Permutation,No-Overlap**
- **No-Permutation,No-Overlap**

are solvable in **polynomial time** if $k_1, k_2 \in \{1, 2\}$.

Proof.

- 2 copies v', v'' for all $v \in V \setminus 0$
- edge (v', v'')=**trips** to collect and delivery v
- $c[v', v'']$ =cost to **collect and deliver** v
- edge (v', w')=**best trips** to collect and deliver v, w
- $c[v', w']$ =min-cost to **collect and deliver** v, w
- $c[v'', w''] = 0$



Appendix — Fixed Capacity Complexity

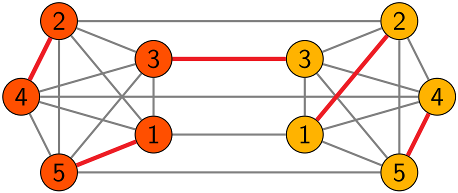
Proposition. The **SPDP-FS** variants

- **Permutation,No-Overlap**
- **No-Permutation,No-Overlap**

are solvable in **polynomial time** if $k_1, k_2 \in \{1, 2\}$.

Proof.

- 2 copies v', v'' for all $v \in V \setminus 0$
- edge (v', v'')=**trips** to collect and delivery v
- $c[v', v'']$ =cost to **collect and deliver** v
- edge (v', w')=**best trips** to collect and deliver v, w
- $c[v', w']$ =min-cost to **collect and deliver** v, w
- $c[v'', w''] = 0$



Solution=**Perfect Matching**

$P = ((2, 4), (3), (1, 5))$

$D = ((4), (2), (3), (5, 1))$

Appendix — Permutation Variants Definitions

Definition

Let $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{\boldsymbol{k}})$ be a sequence of **trips**

We write $\boldsymbol{v} \prec_{\boldsymbol{T}} \boldsymbol{w}$ whenever $\boldsymbol{v} \in \boldsymbol{t}_{\boldsymbol{i}}$ and $\boldsymbol{w} \in \boldsymbol{t}_{\boldsymbol{j}}$ for some $1 \leq \boldsymbol{i} < \boldsymbol{j} \leq \boldsymbol{k}$

Otherwise we write $\boldsymbol{v} \not\prec_{\boldsymbol{T}} \boldsymbol{w}$

Appendix — Permutation Variants Definitions

Definition

Let $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t_k})$ be a sequence of **trips**

We write $\boldsymbol{v} \prec_{\boldsymbol{T}} \boldsymbol{w}$ whenever $\boldsymbol{v} \in \boldsymbol{t_i}$ and $\boldsymbol{w} \in \boldsymbol{t_j}$ for some $1 \leq \boldsymbol{i} < \boldsymbol{j} \leq \boldsymbol{k}$

Otherwise we write $\boldsymbol{v} \not\prec_{\boldsymbol{T}} \boldsymbol{w}$

Example

$\boldsymbol{T} = ((1, 3, 5), (4, 2))$. Then:

- $1 \prec_{\boldsymbol{T}} 4$
- $1 \not\prec_{\boldsymbol{T}} 3$
- $2 \not\prec_{\boldsymbol{T}} 5$

Appendix — Permutation Variants Definitions

Definition

Let $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t_k})$ be a sequence of **trips**
We write $\boldsymbol{v} \prec_{\boldsymbol{T}} \boldsymbol{w}$ whenever $\boldsymbol{v} \in \boldsymbol{t_i}$ and $\boldsymbol{w} \in \boldsymbol{t_j}$ for some $1 \leq \boldsymbol{i} < \boldsymbol{j} \leq \boldsymbol{k}$
Otherwise we write $\boldsymbol{v} \not\prec_{\boldsymbol{T}} \boldsymbol{w}$

PERMUTATION. The pair (P, D) is a feasible solution if and only for every $v, w \in V \setminus \{0\}$ such that $v \prec_P w$ it also holds $w \not\prec_D v$.

DELIVERY PERMUTATION. The pair (P, D) is a feasible solution if and only if:

- for every $j = 1, 2, \ldots, m$, $V(d_j)$ is a set of elements which are consecutive in the P-sequence;
- for every $v, w \in V \setminus \{0\}$, if $v \prec_D w$ then v precedes w in the P-sequence.

PICKUP PERMUTATION. The pair (P, D) is a feasible solution for every $v, w \in V \setminus \{0\}$ such that $v \prec_P w$ we also have that v precedes w in the D-sequence.

Appendix — Pickup vs. Delivery Permutation

Let $k_1 = k_2 = 3$ and $n = 5$.

- **Pickup Permutation OK, Delivery Permutation NOT:**
  - $P = ((1, 5, 4), (2, 3))$
  - $D = ((1), (5, 4, 3), (2))$

Appendix — Pickup vs. Delivery Permutation

Let $k_1 = k_2 = 3$ and $n = 5$.

- **Pickup Permutation OK, Delivery Permutation NOT**:
  - $P = ((1, 5, 4), (2, 3))$
  - $D = ((1), (5, 4, 3), (2))$
  - $F = (1, 5, 4, 3, 2)$ if pickup permutes

Appendix — Pickup vs. Delivery Permutation

Let $k_1 = k_2 = 3$ and $n = 5$.

- **Pickup Permutation OK, Delivery Permutation NOT**:
  - $P = ((1, 5, 4), (2, 3))$
  - $D = ((1), (5, 4, 3), (2))$
  - $F = (1, 5, 4, 3, 2)$ if pickup permutes
  - $F = (1, 5, 4, 2, 3)$ if pickup cannot permute

Appendix — Pickup vs. Delivery Permutation

Let $k_1 = k_2 = 3$ and $n = 5$.

- **Pickup Permutation OK, Delivery Permutation NOT:**
  - $P = ((1, 5, 4), (2, 3))$
  - $D = ((1), (5, 4, 3), (2))$
  - $F = (1, 5, 4, 3, 2)$ if pickup permutes
  - $F = (1, 5, 4, 2, 3)$ if pickup cannot permute
- **Delivery Permutation OK, Pickup Permutation NOT:**
  - $P = ((1, 5, 4), (2, 3))$
  - $D = ((1), (5, 2, 4), (3))$