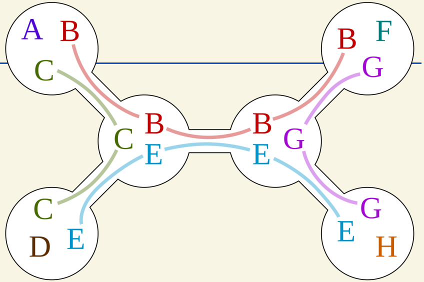
Treewidth I: Pathwidth

DM898: Parameterized Algorithms
Lars Rohwedder

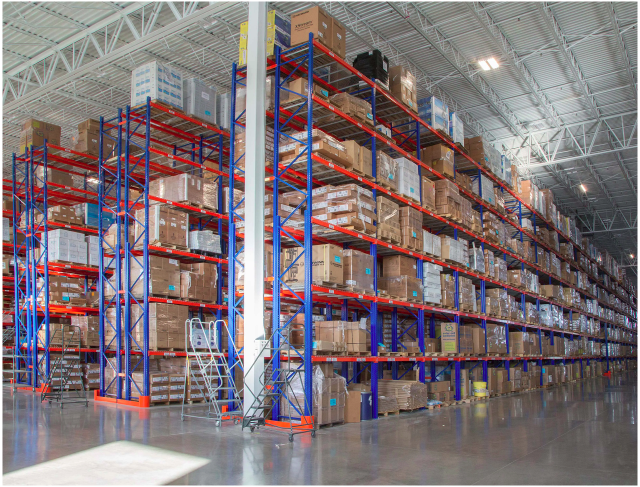


Today's lecture

- Dynamic programming over paths
- Path decomposition
- Maximum Weight Independent Set
- Order Picking

Motivating case

Order Picking



Source: [1]

- **Setting:** picker makes a tour through a warehouse and picks up a given set of orders
- Most commonly modelled as a TSP problem where we minimize the length of the trip
- Important problem in Operations Research: Order Picking makes up 55% of warehouse operational costs according to some estimates[2]

[1]: rebstorage.com/articles-white-papers/how-to-choose-your-industrial-warehouse-racking/

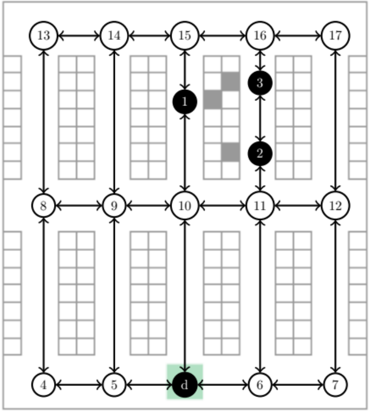
[2]: **Facilities planning. Tompkins, White, Bozer, Tanchoco. 2010.**

Complexity of Order Picking

TSP is NP-hard, so is Order Picking hopeless to solve efficiently?

Complexity of Order Picking

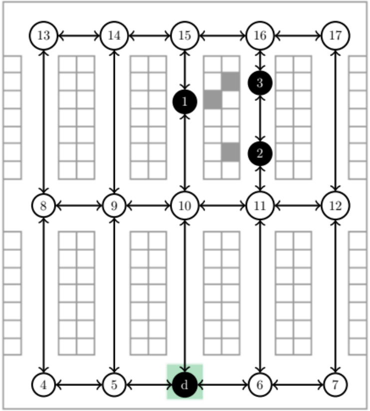
TSP is NP-hard, so is Order Picking hopeless to solve efficiently?



Source: `https://arxiv.org/abs/1703.00699`

Complexity of Order Picking

TSP is NP-hard, so is Order Picking hopeless to solve efficiently?



Source: `https://arxiv.org/abs/1703.00699`

Warehouse graphs from Order Picking are highly structured. NP-hardness does not necessarily hold there. (We have not formalized this class of graphs yet.)

Dynamic programming over paths

Maximum Weight Independent Set

Many problems become (computationally) easy if restricted to paths. Example:

Maximum Weight Independent Set

- Input: Graph $G = (V, E)$, weights $w : V \to \mathbb{Z}_{\geq 0}$
- Output: Vertex set $I \subseteq V$ with $(u, v) \notin E$ for each $u, v \in I$ where $\sum_{v \in I} w(v)$ is maximized

Maximum Weight Independent Set

Many problems become (computationally) easy if restricted to paths. Example:

Maximum Weight Independent Set

- Input: Graph $G = (V, E)$, weights $w : V \to \mathbb{Z}_{\geq 0}$
- Output: Vertex set $I \subseteq V$ with $(u, v) \notin E$ for each $u, v \in I$ where $\sum_{v \in I} w(v)$ is maximized

Dynamic program if G is a path

- Order vertices $\{v_1, \ldots, v_n\} = V$ such that $E = \{(v_i, v_{i+1}) \mid i \in \{1, 2, \ldots, n-1\}\}$

Maximum Weight Independent Set

Many problems become (computationally) easy if restricted to paths. Example:

Maximum Weight Independent Set

- Input: Graph $G = (V, E)$, weights $w : V \to \mathbb{Z}_{\geq 0}$
- Output: Vertex set $I \subseteq V$ with $(u, v) \notin E$ for each $u, v \in I$ where $\sum_{v \in I} w(v)$ is maximized

Dynamic program if G is a path

- Order vertices $\{v_1, \ldots, v_n\} = V$ such that $E = \{(v_i, v_{i+1}) \mid i \in \{1, 2, \ldots, n-1\}\}$
- Dynamic table: for each $i \in \{1, 2, \ldots, n\}$:

$$D[i] = \text{maximum weight of independent set in } \{v_1, \ldots, v_i\}$$

- Base cases: $D[1] = w(v_1)$, $D[2] = \max\{w(v_1), w(v_2)\}$
- Recurrence for $i \geq 3$: $D[i] = \max\{w(v_i) + D[i-2], D[i-1]\}$
- Proving correctness by induction is straight-forward
- Optimum in $D[n]$, solution can be output by easy modification

Maximum Weight Independent Set

Many problems become (computationally) easy if restricted to paths. Example:

Maximum Weight Independent Set

- Input: Graph $G = (V, E)$, weights $w : V \to \mathbb{Z}_{\geq 0}$
- Output: Vertex set $I \subseteq V$ with $(u, v) \notin E$ for each $u, v \in I$ where $\sum_{v \in I} w(v)$ is maximized

Dynamic program if G is a path

- Order vertices $\{v_1, \ldots, v_n\} = V$ such that $E = \{(v_i, v_{i+1}) \mid i \in \{1, 2, \ldots, n-1\}\}$
- Dynamic table: for each $i \in \{1, 2, \ldots, n\}$:

$$D[i] = \text{maximum weight of independent set in } \{v_1, \ldots, v_i\}$$

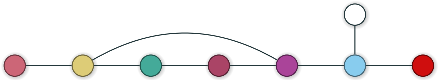
- Base cases: $D[1] = w(v_1)$, $D[2] = \max\{w(v_1), w(v_2)\}$
- Recurrence for $i \geq 3$: $D[i] = \max\{w(v_i) + D[i-2], D[i-1]\}$
- Proving correctness by induction is straight-forward
- Optimum in $D[n]$, solution can be output by easy modification

Can similar ideas transfer to more general classes of graphs? e.g. graphs that are almost paths?

Pathwidth

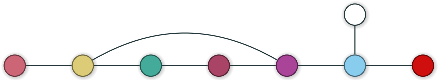
Pathwidth and path decomposition

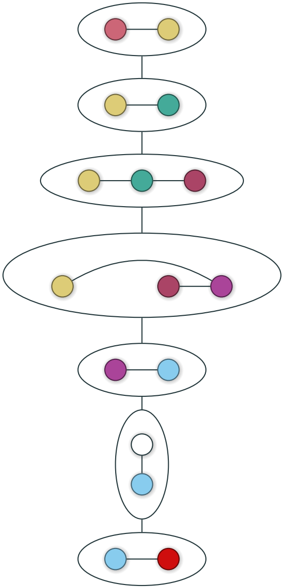
When is a graph **almost** a path?



Pathwidth and path decomposition

When is a graph **almost** a path?





Path decomposition

A path decomposition of a graph $G = (V, E)$ is a path with vertices (called bags) $X_1, \ldots, X_r$ and edges between (X_i, X_{i+1}) for all $i = 1, 2, \ldots, r-1$ such that

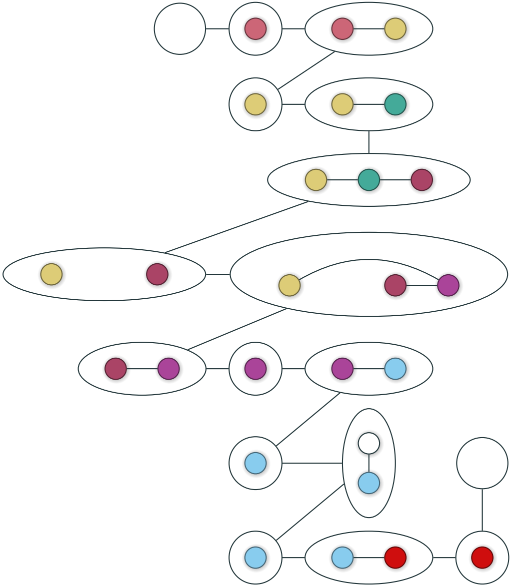
- $X_i \subseteq V$ for all i and $\bigcup_{i=1}^{r} X_i = V$
- For each $(u, v) \in E$ there is some i with $\{u, v\} \subseteq X_i$
- For every $v \in V$, $i < j < k$ with $v \in X_i$ and $v \in X_k$, we also have $v \in X_j$
- The **width** of the decomposition is $\max\{|X_1|, \ldots, |X_r|\} - 1$
- The **pathwidth** of the graph, $\text{pw}(G)$ is the smallest width over any decomposition. If G is a path itself, $\text{pw}(G) = 1$. $\text{pw}(G)$ is a popular parameter for algorithms for "path-like" graphs

Nice path decomposition

Nice path decomposition

A path decomposition $X_1, \ldots, X_r$ is **nice** if $X_1 = X_r = \emptyset$ and for every $i = 1, 2, \ldots, r-1$ either

- $X_{i+1} = X_i \cup \{v\}$ for some $v \in V \setminus X_i$ (**introduce bag**) or
- $X_{i+1} = X_i \setminus \{v\}$ for some $v \in X_i$ (**forget bag**)

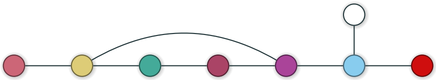


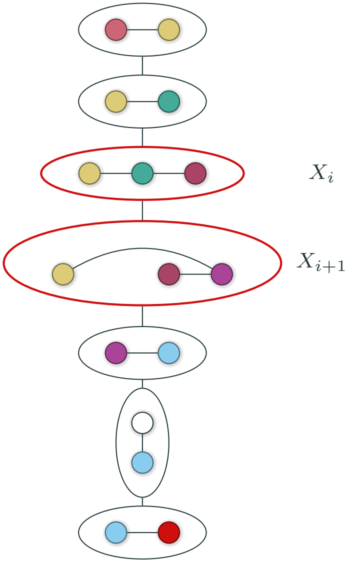
- We can in polynomial time transform a path decomposition of width w to a nice path decomposition of the same width
- A nice path decomposition is easier to work with in dynamic programming
- When devising FPT algorithms in $\mathrm{pw}(G)$ we assume that a path decomposition is given

Separation

Lemma

Let $X_1, \ldots, X_r$ be a path decomposition of graph G. For any bag X_i, there is no edge between $(X_1 \cup \cdots \cup X_i) \setminus (X_i \cap X_{i+1})$ and $(X_{i+1} \cup \cdots \cup X_r) \setminus (X_i \cap X_{i+1})$. We say, $X_i \cap X_{i+1}$ **separates** $X_1 \cup \cdots \cup X_i$ and $X_{i+1} \cup \cdots \cup X_r$.

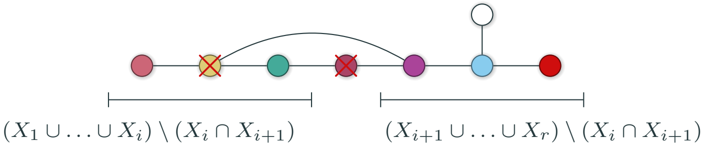


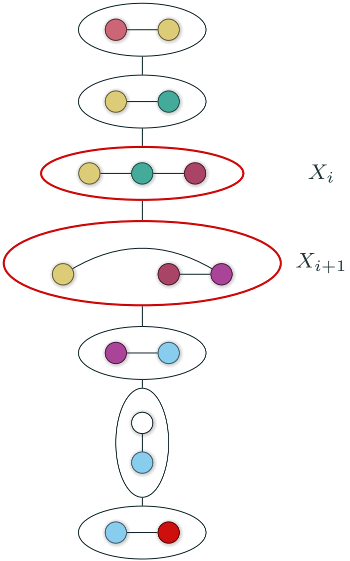


Separation

Lemma

Let $X_1, \ldots, X_r$ be a path decomposition of graph G. For any bag X_i, there is no edge between $(X_1 \cup \cdots \cup X_i) \setminus (X_i \cap X_{i+1})$ and $(X_{i+1} \cup \cdots \cup X_r) \setminus (X_i \cap X_{i+1})$. We say, $X_i \cap X_{i+1}$ **separates** $X_1 \cup \cdots \cup X_i$ and $X_{i+1} \cup \cdots \cup X_r$.

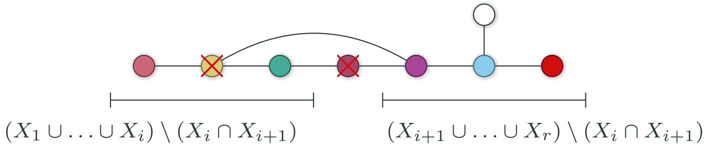


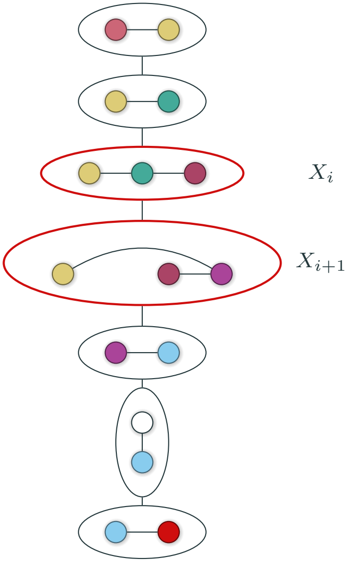


Separation

Lemma

Let $X_1, \ldots, X_r$ be a path decomposition of graph G. For any bag X_i, there is no edge between $(X_1 \cup \cdots \cup X_i) \setminus (X_i \cap X_{i+1})$ and $(X_{i+1} \cup \cdots \cup X_r) \setminus (X_i \cap X_{i+1})$. We say, $X_i \cap X_{i+1}$ **separates** $X_1 \cup \cdots \cup X_i$ and $X_{i+1} \cup \cdots \cup X_r$.





Proof. Let $u \in (X_1 \cup \cdots \cup X_i) \setminus (X_i \cap X_{i+1})$ and $v \in (X_{i+1} \cup \cdots \cup X_r) \setminus (X_i \cap X_{i+1})$.

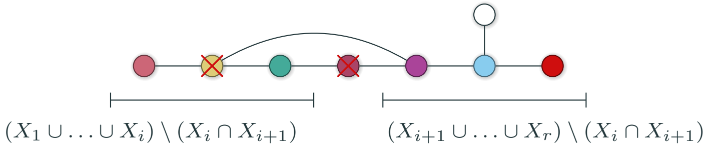
- $u \notin X_i \cap X_{i+1} \Rightarrow u \notin X_{i+1} \cup \cdots \cup X_r$
- $v \notin X_i \cap X_{i+1} \Rightarrow v \notin X_1 \cup \cdots \cup X_i$

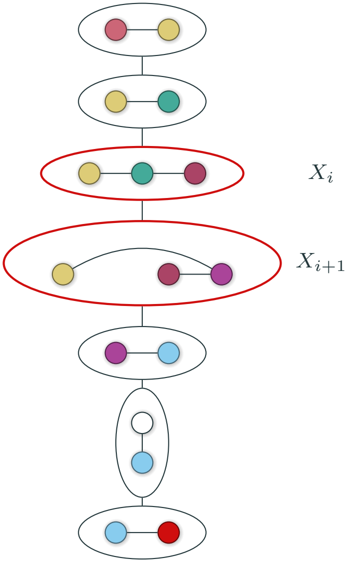
$\Rightarrow$ There is no X_j with $u, v \in X_j \Rightarrow (u, v) \notin E$

Separation

Lemma

Let $X_1, \ldots, X_r$ be a path decomposition of graph G. For any bag X_i, there is no edge between $(X_1 \cup \cdots \cup X_i) \setminus (X_i \cap X_{i+1})$ and $(X_{i+1} \cup \cdots \cup X_r) \setminus (X_i \cap X_{i+1})$. We say, $X_i \cap X_{i+1}$ **separates** $X_1 \cup \cdots \cup X_i$ and $X_{i+1} \cup \cdots \cup X_r$.





Proof. Let $u \in (X_1 \cup \cdots \cup X_i) \setminus (X_i \cap X_{i+1})$ and $v \in (X_{i+1} \cup \cdots \cup X_r) \setminus (X_i \cap X_{i+1})$.

- $u \notin X_i \cap X_{i+1} \Rightarrow u \notin X_{i+1} \cup \cdots \cup X_r$
- $v \notin X_i \cap X_{i+1} \Rightarrow v \notin X_1 \cup \cdots \cup X_i$

$\Rightarrow$ There is no X_j with $u, v \in X_j \Rightarrow (u, v) \notin E$

By fixing the choices in $X_i \cap X_{i+1}$ a problem usually splits into two independent subproblems

Dynamic programming over path decomposition

Maximum Weight Independent Set

Let $X_1, \ldots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \ldots, r\}$ and $S \subseteq X_i$ let

$$D[i, S] = \text{max. weight of independent set } I \subseteq X_1 \cup \cdots \cup X_i \text{ where } I \cap X_i = S \text{ or } -\infty \text{ if it does not exist}$$

Maximum Weight Independent Set

Let $X_1, \ldots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \ldots, r\}$ and $S \subseteq X_i$ let

$D[i, S] =$ max. weight of independent set $I \subseteq X_1 \cup \cdots \cup X_i$ where $I \cap X_i = S$ or $-\infty$ if it does not exist

We compute $D[i, S]$ based on the following case distinction.

Base case: $i = 1$. Then $X_1 \cup \cdots \cup X_i = \emptyset$ and $S = \emptyset$. Thus, $D[i, S] = 0$

Maximum Weight Independent Set

Let $X_1, \dots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \dots, r\}$ and $S \subseteq X_i$ let

$$D[i, S] = \text{max. weight of independent set } I \subseteq X_1 \cup \dots \cup X_i \text{ where } I \cap X_i = S \text{ or } -\infty \text{ if it does not exist}$$

We compute $D[i, S]$ based on the following case distinction.

Base case: $i = 1$**.** Then $X_1 \cup \dots \cup X_i = \emptyset$ and $S = \emptyset$. Thus, $D[i, S] = 0$

Introduce bag: $X_i = X_{i-1} \cup \{v\}$**.**

$$D[i, S] = \begin{cases} -\infty & \text{if } S \text{ is not independent,} \\ D[i-1, S \setminus \{v\}] + w(v) & \text{if } S \text{ independent and } v \in S, \\ D[i-1, S] & \text{otherwise.} \end{cases}$$

Maximum Weight Independent Set

Let $X_1, \dots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \dots, r\}$ and $S \subseteq X_i$ let

$$D[i, S] = \text{max. weight of independent set } I \subseteq X_1 \cup \cdots \cup X_i \text{ where } I \cap X_i = S \text{ or } -\infty \text{ if it does not exist}$$

We compute $D[i, S]$ based on the following case distinction.

Base case: $i = 1$. Then $X_1 \cup \cdots \cup X_i = \emptyset$ and $S = \emptyset$. Thus, $D[i, S] = 0$

Introduce bag: $X_i = X_{i-1} \cup \{v\}$.

$$D[i, S] = \begin{cases} -\infty & \text{if } S \text{ is not independent,} \\ D[i-1, S \setminus \{v\}] + w(v) & \text{if } S \text{ independent and } v \in S, \\ D[i-1, S] & \text{otherwise.} \end{cases}$$

Forget bag: $X_i = X_{i-1} \setminus \{v\}$. Then

$$D[i, S] = \begin{cases} -\infty & \text{if } S \text{ is not independent} \\ \max\{D[i-1, S], D[i-1, S \cup \{v\}]\} & \text{otherwise.} \end{cases}$$

Maximum Weight Independent Set

Let $X_1, \ldots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \ldots, r\}$ and $S \subseteq X_i$ let

$D[i, S] =$ max. weight of independent set $I \subseteq X_1 \cup \cdots \cup X_i$ where $I \cap X_i = S$ or $-\infty$ if it does not exist

We compute $D[i, S]$ based on the following case distinction.

Base case: $i = 1$**.** Then $X_1 \cup \cdots \cup X_i = \emptyset$ and $S = \emptyset$. Thus, $D[i, S] = 0$

Introduce bag: $X_i = X_{i-1} \cup \{v\}$**.**

$$D[i, S] = \begin{cases} -\infty & \text{if } S \text{ is not independent,} \\ D[i-1, S \setminus \{v\}] + w(v) & \text{if } S \text{ independent and } v \in S, \\ D[i-1, S] & \text{otherwise.} \end{cases}$$

Forget bag: $X_i = X_{i-1} \setminus \{v\}$**.** Then

$$D[i, S] = \begin{cases} -\infty & \text{if } S \text{ is not independent} \\ \max\{D[i-1, S], D[i-1, S \cup \{v\}]\} & \text{otherwise.} \end{cases}$$

Running time: $2^k \cdot \text{poly}(n, k)$

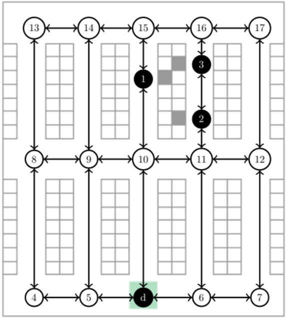
Correctness: blackboard

Order Picking

Warehouse graph

For the Order Picking problem we consider the following class of graphs:

- There are cross aisles $i = 1, 2, \ldots, h$ that form disjoint paths $(v_1^{(i)}, \ldots, v_k^{(i)})$. Usually $h \leq 3$
- For each $i = 1, 2, \ldots, h-1$ and $j = 1, 2, \ldots, k$ the vertices $v_j^{(i)}$ and $v_j^{(i+1)}$ are connected by a path (an "aisle") where the inner vertices of the path correspond to pick-up locations and are disjoint from each other and from the cross-aisles
- There is one depot vertex $d \in V$ at which the tour of the order picker starts and ends

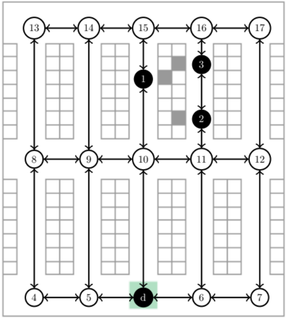


Source: https://arxiv.org/abs/1703.00699

Warehouse graph

For the Order Picking problem we consider the following class of graphs:

- There are cross aisles $i = 1, 2, \ldots, h$ that form disjoint paths $(v_1^{(i)}, \ldots, v_k^{(i)})$. Usually $h \leq 3$
- For each $i = 1, 2, \ldots, h-1$ and $j = 1, 2, \ldots, k$ the vertices $v_j^{(i)}$ and $v_j^{(i+1)}$ are connected by a path (an "aisle") where the inner vertices of the path correspond to pick-up locations and are disjoint from each other and from the cross-aisles
- There is one depot vertex $d \in V$ at which the tour of the order picker starts and ends



Source: https://arxiv.org/abs/1703.00699

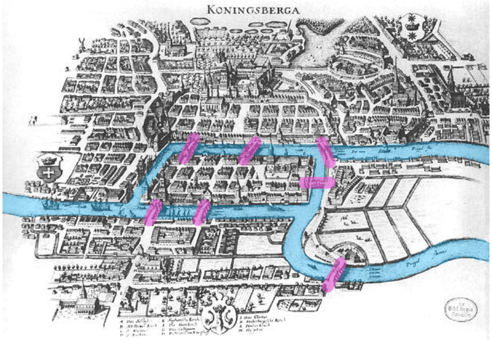
The pathwidth of a warehouse graph is at most $h + 1$ (see blackboard)

Order Picking problem

Given a warehouse graph $G = (V, E)$, edge lengths $w : E \to \mathbb{Z}_{\geq 0}$, depot $d \in V$, and pick-up locations $P \subseteq V$, find a tour of minimal length that visits $P \cup \{d\}$. The tour may cross vertices and edges several times.

Order Picking problem

Given a warehouse graph $G = (V, E)$, edge lengths $w : E \to \mathbb{Z}_{\geq 0}$, depot $d \in V$, and pick-up locations $P \subseteq V$, find a tour of minimal length that visits $P \cup \{d\}$. The tour may cross vertices and edges several times.



Source:

`https://commons.wikimedia.org/wiki/File:Bridges_of_Konigsberg.png`

Order Picking problem

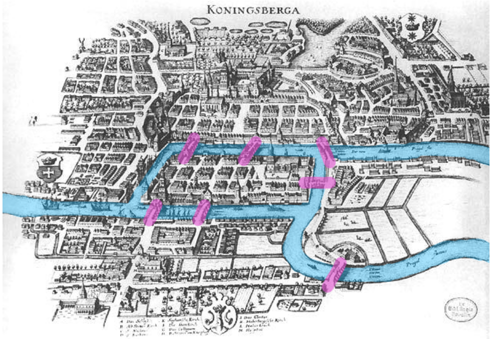
Given a warehouse graph $G = (V, E)$, edge lengths $w : E \to \mathbb{Z}_{\geq 0}$, depot $d \in V$, and pick-up locations $P \subseteq V$, find a tour of minimal length that visits $P \cup \{d\}$. The tour may cross vertices and edges several times.

Equivalent formulation

A multiset of edges F corresponds to the edges crossed in some tour visiting exactly the vertices U if and only if $\deg_F(u)$ is even and non-zero for each $u \in U$ and the graph (U, F) is connected. See **Eulerian tour** for reference.

Equivalent to Order Picking: Find a multiset of edges F of minimal total length such that $\deg_F(v)$ is even for each $v \in V$, $\deg_F(v) \neq 0$ for each $v \in P \cup \{d\}$, and (U, F) is connected, where $U = \{v \in V \mid \deg_F(v) \neq 0\}$.

In an optimal multiset F, each edge appears zero times, once or twice.



Source:

https://commons.wikimedia.org/wiki/File:Bridges_of_Konigsberg.png

Dynamic program for Order Picking

Let $X_1, \ldots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \ldots, r\}$, partition $\mathcal{P}$ of X_i ($\mathcal{P} = \emptyset$ if $X_i = \emptyset$), and $p \in \{\text{zero}, \text{odd}, \text{even}\}^{X_i}$ let $D[i, \mathcal{P}, p]$ be the minimum total distance of an edge multi-set F s.t.

- For each $v \in X_i$, $\deg_F(v)$ is zero if $p_v = \text{zero}$, odd if $p_v = \text{odd}$ and even and non-zero if $p_v = \text{even}$
- For each $v \in (X_1 \cup \cdots \cup X_i) \setminus X_i$ we have $\deg_F(v)$ is even; if $v \in P \cup \{d\}$ then $\deg_F(v)$ is non-zero
- For each $S \in \mathcal{P}$ it holds that S is connected in (V, F)
- If $i < r$ then each connected component in (V, F) contains a vertex $v \in X_i$; if $i = r$ then (V, F) contains a single connected component

Dynamic program for Order Picking

Let $X_1, \ldots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \ldots, r\}$, partition $\mathcal{P}$ of X_i ($\mathcal{P} = \emptyset$ if $X_i = \emptyset$), and $p \in \{\text{zero}, \text{odd}, \text{even}\}^{X_i}$ let $D[i, \mathcal{P}, p]$ be the minimum total distance of an edge multi-set F s.t.

- For each $v \in X_i$, $\deg_F(v)$ is zero if $p_v = \text{zero}$, odd if $p_v = \text{odd}$ and even and non-zero if $p_v = \text{even}$
- For each $v \in (X_1 \cup \cdots \cup X_i) \setminus X_i$ we have $\deg_F(v)$ is even; if $v \in P \cup \{d\}$ then $\deg_F(v)$ is non-zero
- For each $S \in \mathcal{P}$ it holds that S is connected in (V, F)
- If $i < r$ then each connected component in (V, F) contains a vertex $v \in X_i$; if $i = r$ then (V, F) contains a single connected component

We compute $D[i, \mathcal{P}, p]$ based on the following case distinction.

Base case: $i = 1$. Then $X_1 \cup \cdots \cup X_i = \emptyset$. Thus, $D[i, \mathcal{P}, p] = 0$

Dynamic program for Order Picking

Let $X_1, \ldots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \ldots, r\}$, partition $\mathcal{P}$ of X_i ($\mathcal{P} = \emptyset$ if $X_i = \emptyset$), and $p \in \{\text{zero}, \text{odd}, \text{even}\}^{X_i}$ let $D[i, \mathcal{P}, p]$ be the minimum total distance of an edge multi-set F s.t.

- For each $v \in X_i$, $\deg_F(v)$ is zero if $p_v = \text{zero}$, odd if $p_v = \text{odd}$ and even and non-zero if $p_v = \text{even}$
- For each $v \in (X_1 \cup \cdots \cup X_i) \setminus X_i$ we have $\deg_F(v)$ is even; if $v \in P \cup \{d\}$ then $\deg_F(v)$ is non-zero
- For each $S \in \mathcal{P}$ it holds that S is connected in (V, F)
- If $i < r$ then each connected component in (V, F) contains a vertex $v \in X_i$; if $i = r$ then (V, F) contains a single connected component

We compute $D[i, \mathcal{P}, p]$ based on the following case distinction.

Base case: $i = 1$. Then $X_1 \cup \cdots \cup X_i = \emptyset$. Thus, $D[i, \mathcal{P}, p] = 0$

Introduce bag: $X_i = X_{i-1} \cup \{v\}$. Then $D[i, \mathcal{P}, p] = \min_{F', \mathcal{P}', p'} \{D[i-1, \mathcal{P}', p'] + \sum_{e \in F'} w(e)\}$ where the minimum is over all multisets F' of edges between v and $X_i \setminus \{v\}$, partitions $\mathcal{P}'$ of X_{i-1} and $p' \in \{\text{zero}, \text{odd}, \text{even}\}^{X_{i-1}}$ with

- $p_u = p'_u + \deg_{F'}(u)$ (with the natural operation on $\text{zero}, \text{odd}, \text{even}$) for each $u \in X_{i-1}$,
- p_v consistent with $\deg_{F'}(v)$,
- let $S'' \subseteq X_i$ be the union of $\{v\}$ and all $S' \in \mathcal{P}'$ with $(w, v) \in F$ for some $w \in S'$. Then for each $S \in \mathcal{P}$ either $S \subseteq S''$ or there exists $S' \subseteq \mathcal{P}'$ with $S \subseteq S'$.
- each edge $e \in E$ occurs at most twice in F

Dynamic program for Order Picking

Let $X_1, \ldots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \ldots, r\}$, partition $\mathcal{P}$ of X_i ($\mathcal{P} = \emptyset$ if $X_i = \emptyset$), and $p \in \{\mathrm{zero}, \mathrm{odd}, \mathrm{even}\}^{X_i}$ let $D[i, \mathcal{P}, p]$ be the minimum total distance of an edge multi-set F s.t.

- For each $v \in X_i$, $\deg_F(v)$ is zero if $p_v = \mathrm{zero}$, odd if $p_v = \mathrm{odd}$ and even and non-zero if $p_v = \mathrm{even}$
- For each $v \in (X_1 \cup \cdots \cup X_i) \setminus X_i$ we have $\deg_F(v)$ is even; if $v \in P \cup \{d\}$ then $\deg_F(v)$ is non-zero
- For each $S \in \mathcal{P}$ it holds that S is connected in (V, F)
- If $i < r$ then each connected component in (V, F) contains a vertex $v \in X_i$; if $i = r$ then (V, F) contains a single connected component

We compute $D[i, \mathcal{P}, p]$ based on the following case distinction.

Base case: $i = 1$. Then $X_1 \cup \cdots \cup X_i = \emptyset$. Thus, $D[i, \mathcal{P}, p] = 0$

Introduce bag: $X_i = X_{i-1} \cup \{v\}$. Then $D[i, \mathcal{P}, p] = \min_{F', \mathcal{P}', p'} \{D[i-1, \mathcal{P}', p'] + \sum_{e \in F'} w(e)\}$

Forget bag: $X_i = X_{i-1} \setminus \{v\}$. Then $D[i, \mathcal{P}, p] = \min_{\mathcal{P}', p'} D[i-1, \mathcal{P}', p']$, where the minimum is over all partitions $\mathcal{P}'$ of X_{i-1} and $p' \in \{\mathrm{zero}, \mathrm{odd}, \mathrm{even}\}^{X_{i-1}}$ with

- for each $S \in P$ there exists $S' \in \mathcal{P}'$ with $S \subseteq S'$ and,
- $p'_u = p_u$ for each $u \in X_i$.
- if $v \in P \cup \{d\}$ then $p'_v \neq \mathrm{zero}$, *(v does not get isolated if it must be visited)*
- if $p'_v \neq \mathrm{zero}$ and $i < r$ then $\{v\} \notin \mathrm{P}'$, *(component of v does not get disconnected)*

Dynamic program for Order Picking

Let $X_1, \dots, X_r$ be nice path decomposition of width k. For each $i \in \{1, \dots, r\}$, partition $\mathcal{P}$ of X_i ($\mathcal{P} = \emptyset$ if $X_i = \emptyset$), and $p \in \{\mathrm{zero}, \mathrm{odd}, \mathrm{even}\}^{X_i}$ let $D[i, \mathcal{P}, p]$ be the minimum total distance of an edge multi-set F s.t.

- For each $v \in X_i$, $\deg_F(v)$ is zero if $p_v = \mathrm{zero}$, odd if $p_v = \mathrm{odd}$ and even and non-zero if $p_v = \mathrm{even}$
- For each $v \in (X_1 \cup \cdots \cup X_i) \setminus X_i$ we have $\deg_F(v)$ is even; if $v \in P \cup \{d\}$ then $\deg_F(v)$ is non-zero
- For each $S \in \mathcal{P}$ it holds that S is connected in (V, F)
- If $i < r$ then each connected component in (V, F) contains a vertex $v \in X_i$; if $i = r$ then (V, F) contains a single connected component

We compute $D[i, \mathcal{P}, p]$ based on the following case distinction.

Base case: $i = 1$. Then $X_1 \cup \cdots \cup X_i = \emptyset$. Thus, $D[i, \mathcal{P}, p] = 0$

Introduce bag: $X_i = X_{i-1} \cup \{v\}$. Then $D[i, \mathcal{P}, p] = \min_{F', \mathcal{P}', p'} \{D[i-1, \mathcal{P}', p'] + \sum_{e \in F'} w(e)\}$

Forget bag: $X_i = X_{i-1} \setminus \{v\}$. Then $D[i, \mathcal{P}, p] = \min_{\mathcal{P}', p'} D[i-1, \mathcal{P}', p']$,

Running time: $k^{O(k)} \cdot \mathrm{poly}(n)$, since number of partitions is always at most k^k

Correctness: ommitted here (can be checked by straight-forward, but tedious calculation)

Experimental results

An optimized version of this dynamic program has been implemented in[1]

SCFS+ and SCF+: Commercial solvers on different ILP formulations

PDYN: FPT algorithm based on dynamic programming

	Total	Storage policy		# aisles			# cross-aisles			# products		
		R	V	5	15	60	3	6	11	15	60	240
SCFS+	18	18	0	1	4	13	1	2	15	0	0	18
SCF+	136	88	48	19	34	83	51	41	44	0	26	110
PDYN	180	90	90	60	60	60	0	0	180	60	60	60
# instances	540	270	270	180	180	180	180	180	180	180	180	180

Table shows number of unsolved instances with different sizes after 30 minutes.

[1] Exact algorithms for the order picking problem. Pansart, Catusse , Cambazard. 2018.
https://arxiv.org/abs/1703.00699