UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Analyse des Kostensenkungspotenzials durch Energy Demand Side Management in elektrisch betriebenen Containerterminals

Allgemeine Betriebswirtschaftslehre und Produktionsmanagement

Prof. Dr. Jutta Geldermann

Sebastian Schär, M.Sc.

Erik Pohl, M.Sc.

Lehrstuhl für ABWL und Produktionsmanagement

Universität Duisburg-Essen

sebastian.schaer@uni-due.de

Analyse des Kostensenkungspotenzials durch Energy Demand Side Management in elektrisch betriebenen Containerterminals

1. Betrachtungsgegenstand Containerterminal
2. Problemformulierung
3. Vorstellung Lösungsverfahren
4. Ergebnisse
5. Fazit und Diskussion

Analyse des Kostensenkungspotenzials durch Energy Demand Side Management in elektrisch betriebenen Containerterminals

1 **Betrachtungsgegenstand Containerhafen**

2 **Problemformulierung**

3 **Vorstellung Lösungsverfahren**

4 **Ergebnisse**

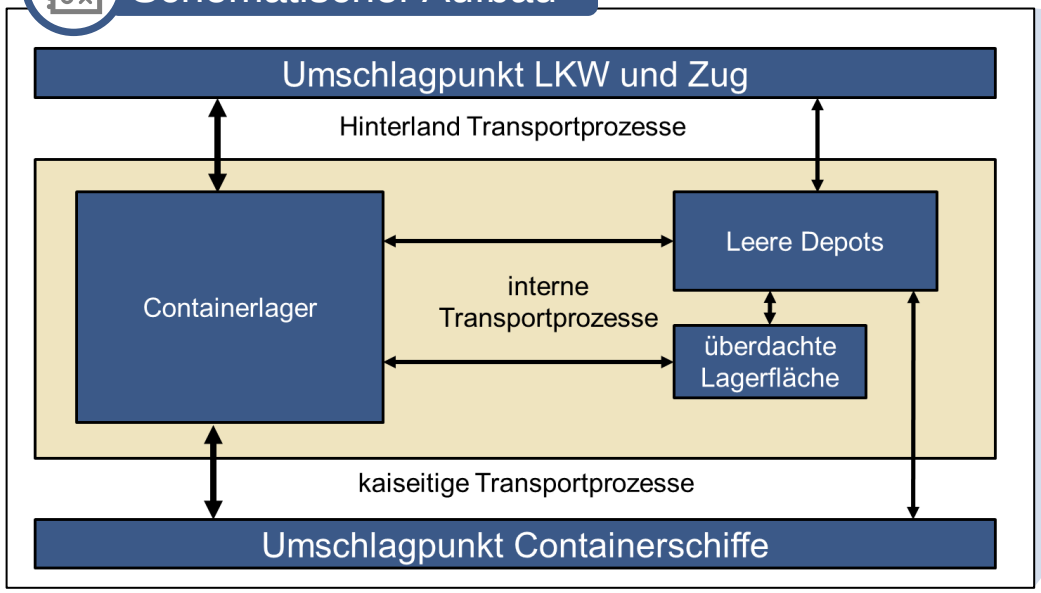
5 **Fazit und Diskussion**

Aufbau eines Containerterminals

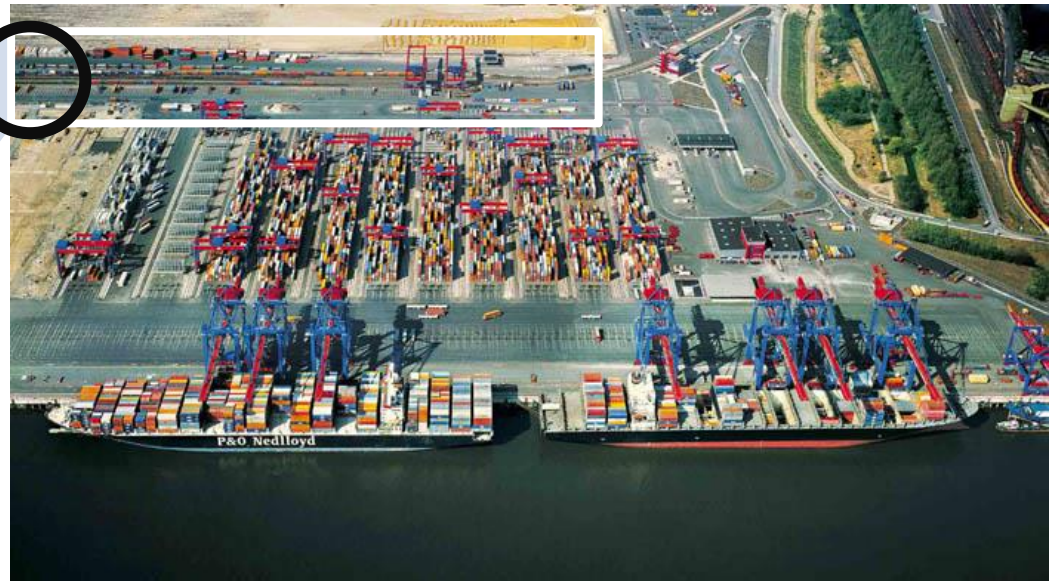
Allgemein

- Nahezu jedes per Seefracht transportierte Stückgut wird heutzutage per Container überführt
- Container bieten Schutz gegen Wetter und Plünderung und vereinfachen Be- und Entladeprozesse
- Abfertigung der Güterzüge mithilfe vollautomatischer, elektrisch betriebener Kräne

Schematischer Aufbau



Quelle: Steenken et al. 2004



Bildquelle: HHLA

Analyse des Kostensenkungspotenzials durch Energy Demand Side Management in elektrisch betriebenen Containerterminals

1 Betrachtungsgegenstand Containerhafen

2 **Problemformulierung**

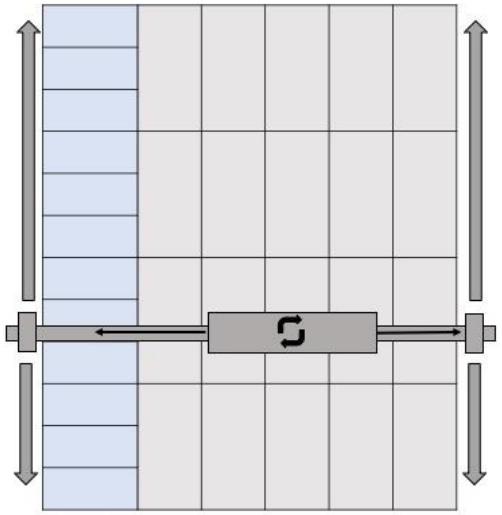
3 Vorstellung Lösungsverfahren

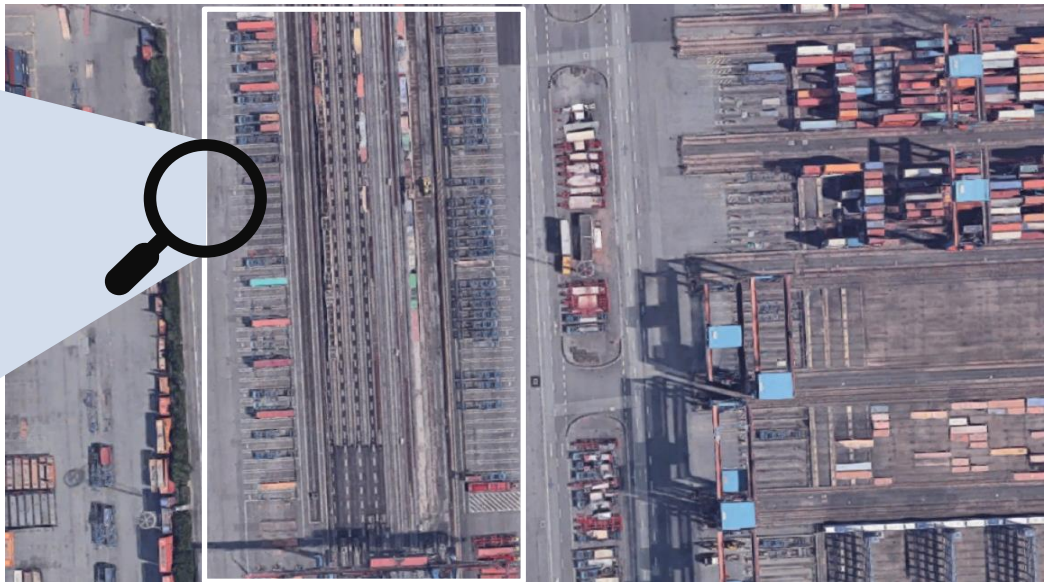
4 Ergebnisse

5 Fazit und Diskussion

Einsatz von Portalkränen im Umschlagbahnhof

Schematischer Aufbau





- Züge sind an festen Fahrplan gebunden ➡ Entstehung von Pufferzeiten

Bildquelle: Google Maps

Grundlagen von Demand Response Programmen

Demand Response (DR)

*Demand Response (DR) bezeichnet die **bewusste Beeinflussung der Konsummuster** von Endkunden am Strommarkt hinsichtlich des Zeitpunktes, des momentanem Nachfrageniveaus sowie der insgesamt verbrauchten Menge an elektrischem Strom **durch ein monetäres Anreizsignal**.*

Vorteile von DR Programmen

- geringere Energiekosten (bei Annahme volatiler Strompreise)
- Sicherstellung der Netzstabilität durch Lastanpassung
- bessere Einbindung erneuerbarer Energien in das Stromnetz

Ziel

Gezieltes Ausnutzen von Pufferzeiten der Containerkräne im Umschlagbahnhof durch Demand Response, um Spitzenlasten zu reduzieren und Kosten zu sparen.

Quellen: Albadi 2007, Strbac 2008, Schmidt et al. 2015

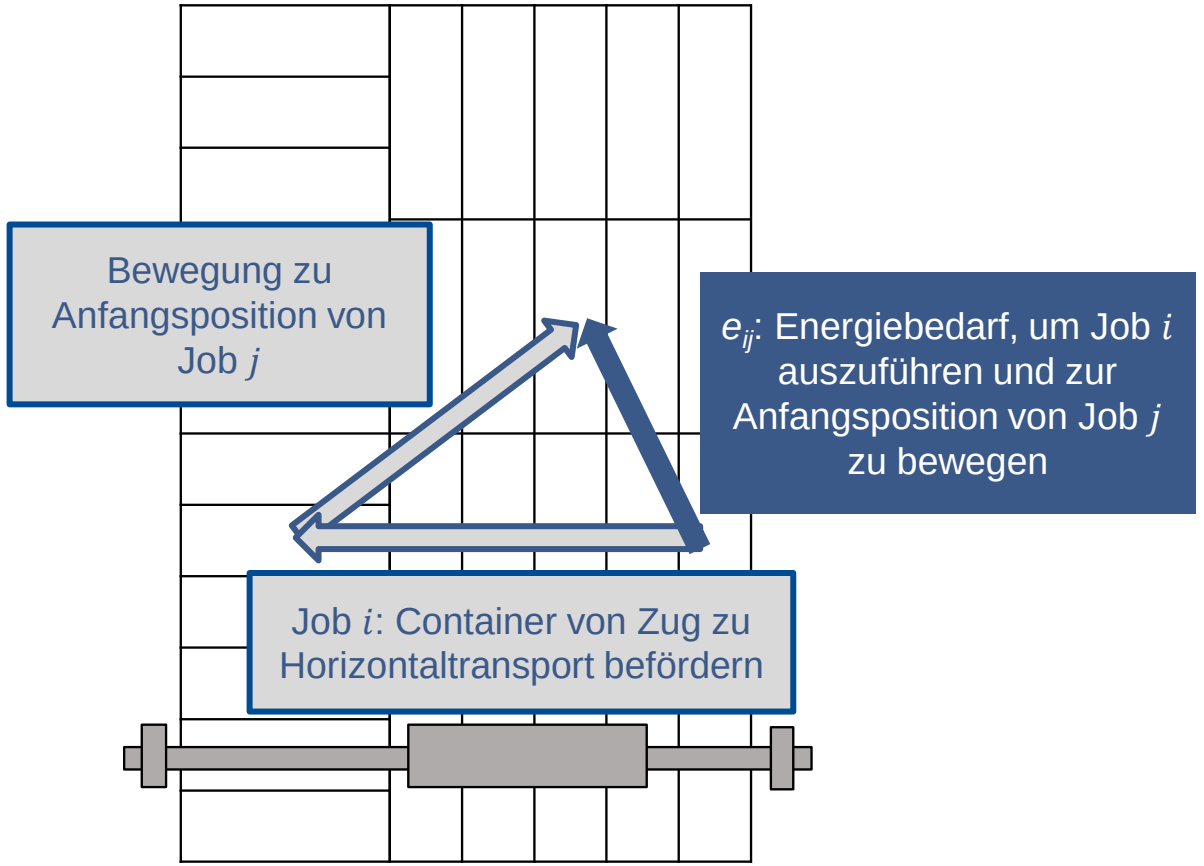
Ziel ist die Analyse des Kostensenkungspotenzials bei Teilnahme an DR-Programmen

- Anwendung des Travelling Salesman Problems auf den Betrachtungsgegenstand Containerhafen zur Bestimmung von Ablaufplänen.
- Verbindung mit Demand Response durch Berücksichtigung der Energiekosten bei volatilen Strompreisen und Ausnutzen von identifizierten Pufferzeiten.
- Das Modell kann vom Hafenbetreiber genutzt werden, um zu entscheiden ob die relative Kostenersparnis den praktischen Implementierungsaufwand zur Teilnahme am DR Progamm rechtfertigt.

Bestimmung des energieminimalen Ablaufplans am Containerbahnhof

„Job“

Entladen eines Containers vom Zug bzw. Beladen des Zuges mit einem Container



Die Bestimmung eines energieminimalen Ablaufplans für den Containerkran lässt sich als Travelling Salesman Problem mit Zeitfensterbeschränkungen (TSPTW) auffassen.

Erster Lösungsansatz: Implementierung in Optimierungssoftware

Erster Ansatz

Formulierung als MILP und Implementierung in der Optimierungssoftware *Gurobi*

GUROBI OPTIMIZATION

Ermittlung des optimalen Ablaufplans und optimaler Startzeiten

Jedoch nur für kleine Probleminstanzen in annehmbarer Laufzeit lösbar und somit nicht geeignet für Anwendung in der Praxis (NP-schwer)

Dekomposition und Lösung des TSPTW mithilfe von Heuristiken anstelle exakter Verfahren

Analyse des Kostensenkungspotenzials durch Energy Demand Side Management in elektrisch betriebenen Containerterminals

1 Betrachtungsgegenstand Containerhafen

2 Problemformulierung

3 Vorstellung Lösungsverfahren

4 Ergebnisse

5 Fazit und Diskussion

Vorstellung Lösungsverfahren

Allgemein

- Implementierung via Programmiersprache Python
- Über Heuristik generierte zulässige Auftragsreihenfolge wird anschließend als Input für das Optimierungsmodell (Schritt 2) in Gurobi verwendet

Lösung via Dekomposition

Schritt 1

Teilproblem TSPTW heuristisch lösen:

VNS Heuristik sucht zulässige Lösung

heuristische Verbesserung der zulässigen Lösung

Schritt 2

Lösung des TSPTW in Form einer zulässigen Auftragsreihenfolge als Input für Optimierungsmodell

Algorithm 5: GVNS nach Mladenovic

```
1  X = build feasible solution by VNS;
2  repeat
3      k = 1;
4      while k <= kmax do
5          X' = Shake(x, k);
6          X" = SeqVND(X');
7          k = k +1;
8          if X" is better than X then
9              X = X", k = 1;
10         end
11     end
12 until t <= tmax;
```

Quelle: Mladenovic et al. 2013
Bildquelle: Gurobi Optimization LLC

Vorstellung Lösungsverfahren

Allgemein

- Implementierung via Programmiersprache Python
- Über Heuristik generierte zulässige Auftragsreihenfolge wird anschließend als Input für das Optimierungsmodell (Schritt 2) in Gurobi verwendet

Lösung via Dekomposition

Schritt 1

Teilproblem TSPTW heuristisch lösen:

VNS Heuristik sucht zulässige Lösung

heuristische Verbesserung der zulässigen Lösung

Schritt 2

Lösung des TSPTW in Form einer zulässigen Auftragsreihenfolge als Input für Optimierungsmodell

Algorithm 5: GVNS nach Mladenovic

```
1  X = build feasible solution by VNS;
2  repeat
3      k = 1;
4      while k <= kmax do
5          X' = Shake(x, k);
6          X" = SeqVND(X');
7          k = k +1;
8          if X" is better than X then
9              X = X", k = 1;
10         end
11     end
12 until t <= tmax;
```

Quelle: Mladenovic et al. 2013
Bildquelle: Gurobi Optimization LLC

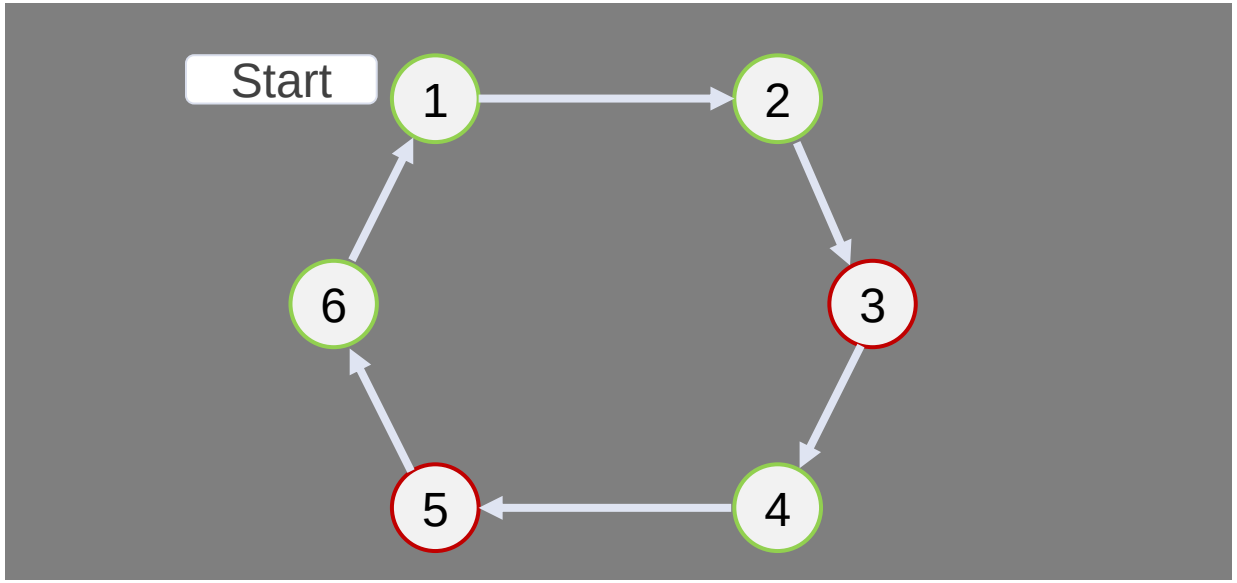
Schritt 1: VNS Heuristik sucht zulässige Lösung

Algorithmus

Suche einer zulässigen Lösung für das TSPTW via Variable Neighborhood Search (VNS)

Algorithm 3: VNS nach Da Silva und Urrutia

```
Data: X
Result: X
repeat
    level = 1;
    X = RandomSolution();
    X = Local1Shift(X);
    while X is infeasible and level < levelMax do
        X' = Perturbation(X, level);
        X' = Local1Shift(X');
        X = better(X, X');
        if X is equal to X' then
            level = 1
        else
            level ++
        end
    end
until X is feasible;
```

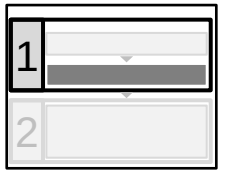


Local1Shift

1. nicht zulässige Jobs an frühere Stelle verschieben
2. zulässige Jobs an spätere Stelle verschieben
3. zulässige Jobs an frühere Stelle verschieben
4. unzulässige Jobs an spätere Stelle verschieben

Quelle: da Silva und Urrutia 2010

Schritt 1: Verbesserung einer zulässigen Lösung



Algorithmus

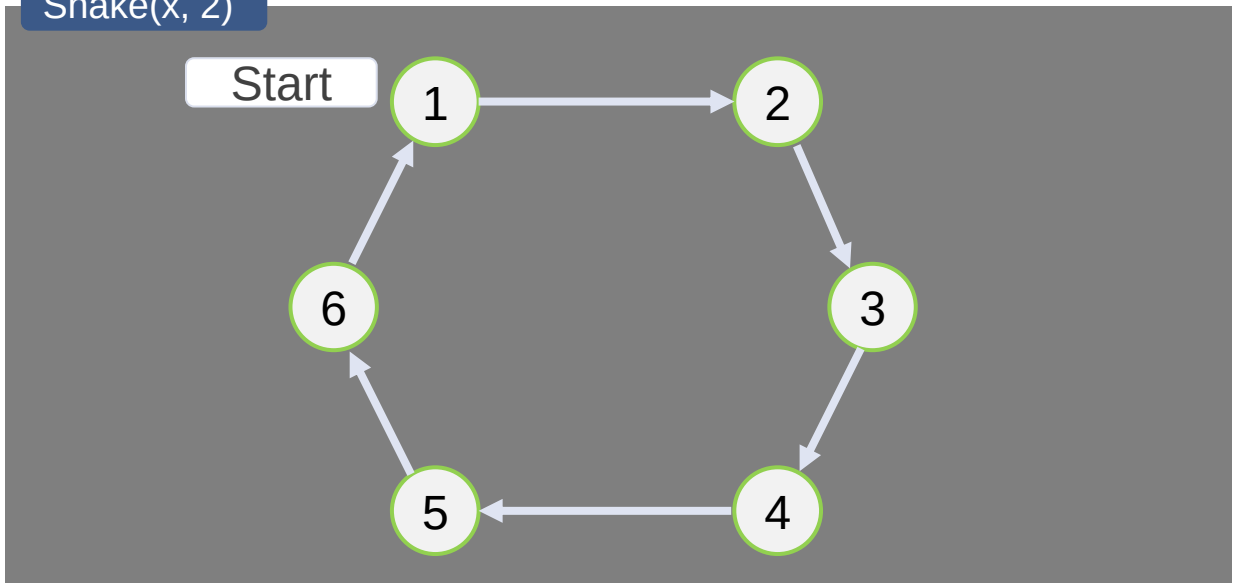
Verfahren der iterative Nachbarschaftssuche mit Perturbation

Algorithm 5: GVNS nach Mladenovic

```
1  X = build feasible solution by VNS;
2  repeat
3      k = 1;
4      while k <= kmax do
5          X' = Shake(x, k);
6          X" = SeqVND(X');
7          k = k +1;
8          if X" is better than X then
9              X = X", k = 1;
10         end
11     end
12 until t <= tmax;
```

Quelle: Mladenovic et al. 2013

Shake(x, 2)

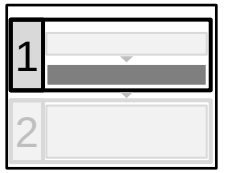


Shake

k zufällige, aber zulässige Verschiebungen von Jobs um eine Position in der aktuellen Lösung *x* ohne einen Teil der Lösung zu invertieren

➞ sog. 1-shifts um eine Position

Schritt 1: Verbesserung einer zulässigen Lösung



Algorithmus

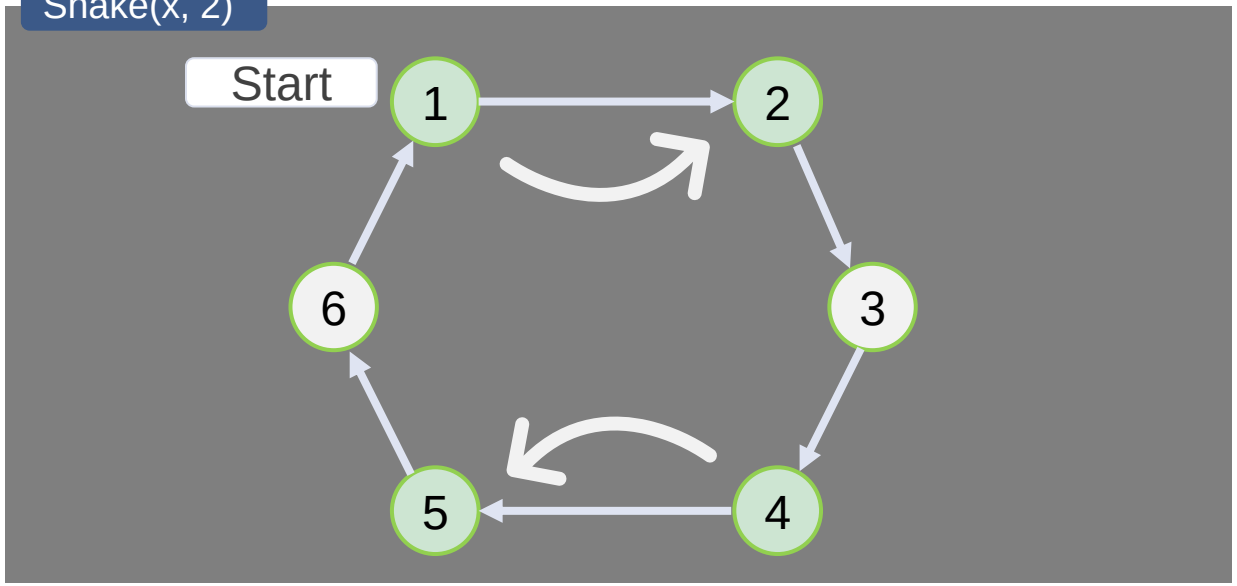
Verfahren der iterative Nachbarschaftssuche mit Perturbation

Algorithm 5: GVNS nach Mladenovic

```
1  X = build feasible solution by VNS;
2  repeat
3      k = 1;
4      while k <= kmax do
5          X' = Shake(x, k);
6          X" = SeqVND(X');
7          k = k +1;
8          if X" is better than X then
9              X = X", k = 1;
10         end
11     end
12 until t <= tmax;
```

Quelle: Mladenovic et al. 2013

Shake(x, 2)

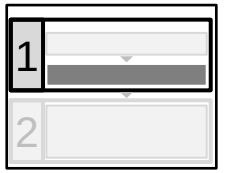


Shake

k zufällige, aber zulässige Verschiebungen von Jobs um eine Position in der aktuellen Lösung *x* ohne einen Teil der Lösung zu invertieren

➞ sog. 1-shifts um eine Position

Schritt 1: Verbesserung einer zulässigen Lösung



Algorithmus

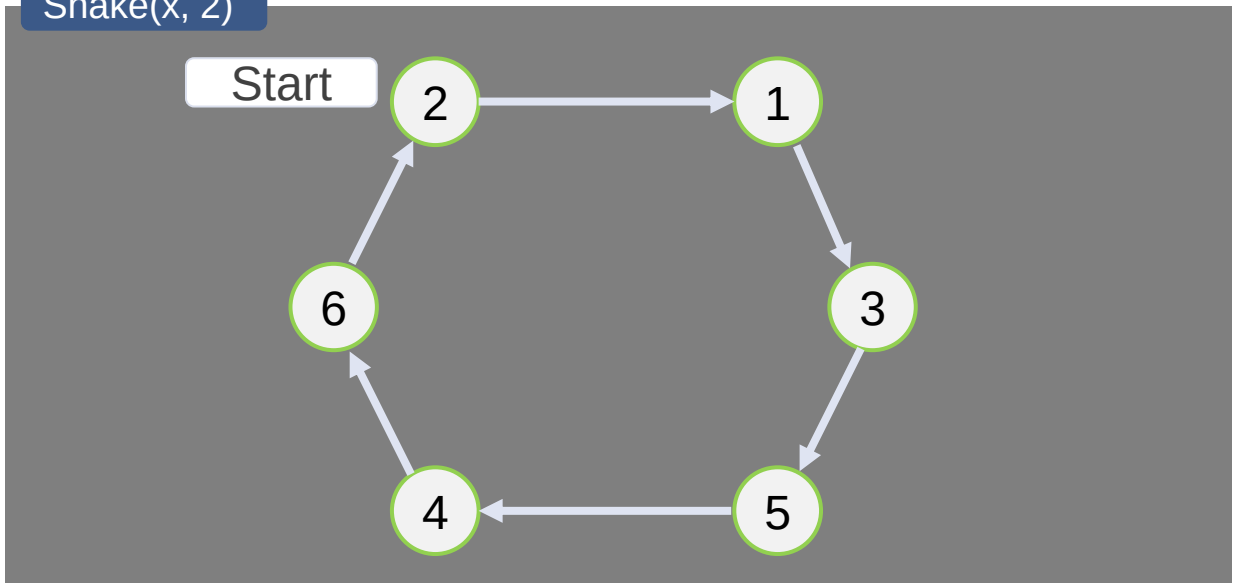
Verfahren der iterative Nachbarschaftssuche mit Perturbation

Algorithm 5: GVNS nach Mladenovic

```
1  X = build feasible solution by VNS;
2  repeat
3      k = 1;
4      while k <= kmax do
5          X' = Shake(x, k);
6          X" = SeqVND(X');
7          k = k +1;
8          if X" is better than X then
9              X = X", k = 1;
10         end
11     end
12 until t <= tmax;
```

Quelle: Mladenovic et al. 2013

Shake(x, 2)

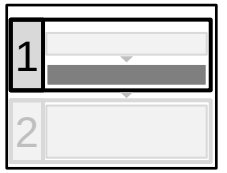


Shake

k zufällige, aber zulässige Verschiebungen von Jobs um eine Position in der aktuellen Lösung *x* ohne einen Teil der Lösung zu invertieren

➞ sog. 1-shifts um eine Position

Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



Algorithmus

Verfahren der iterative Nachbarschaftssuche mit Perturbation

Algorithm 5: GVNS nach Mladenovic

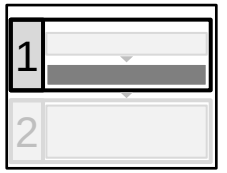
```
1  X = build feasible solution by VNS;
2  repeat
3  |  k = 1;
4  |  while k <= kmax do
5  |  |  X' = Shake(x, k);
6  |  |  X" = SeqVND(X');
7  |  |  k = k +1;
8  |  |  if X" is better than X then
9  |  |  |  X = X", k = 1;
10 |  |  end
11 |  end
12 until t <= tmax;
```

Quelle: Mladenovic et al. 2013

Sequential Variable Neighborhood Descent (VND)

- Verschiebung eines ausgewählten Jobs i solange Zulässigkeit der Lösung beibehalten wird
- Nachbarschaft der Lösung wird mit folgenden Suchverfahren durchsucht:
  1-opt, backward/forward 2-shift, backward/forward 1-shift, 2-opt
- „best improvement search“

Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



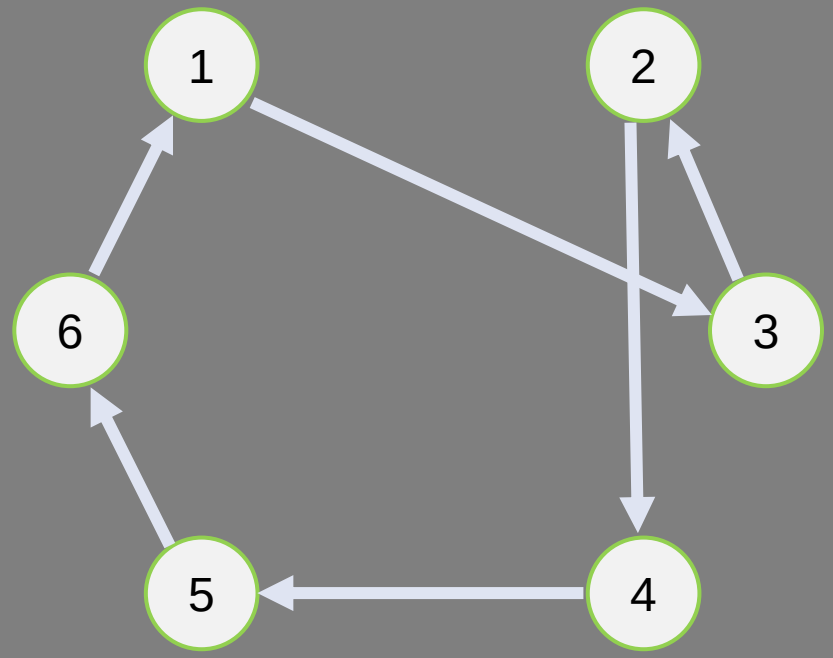
Vorgehen:

1-opt

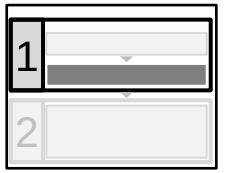
Die Kanten von 4 aufeinanderfolgenden Jobs werden aufgelöst und die mittleren Jobs werden invertiert

Auftragsfolge

X = [1, 3, 2, 4, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



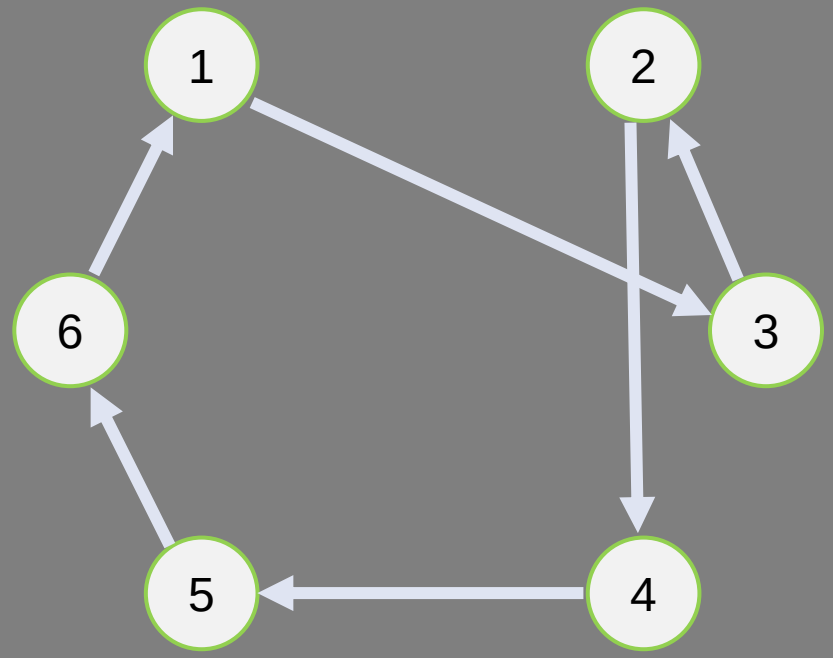
Vorgehen:

1-opt

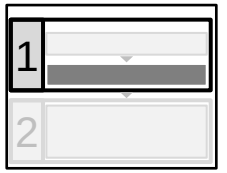
Die Kanten von **4 aufeinanderfolgenden Jobs** werden aufgelöst und die mittleren Jobs werden invertiert

Auftragsfolge

X = [**1, 3, 2, 4**, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



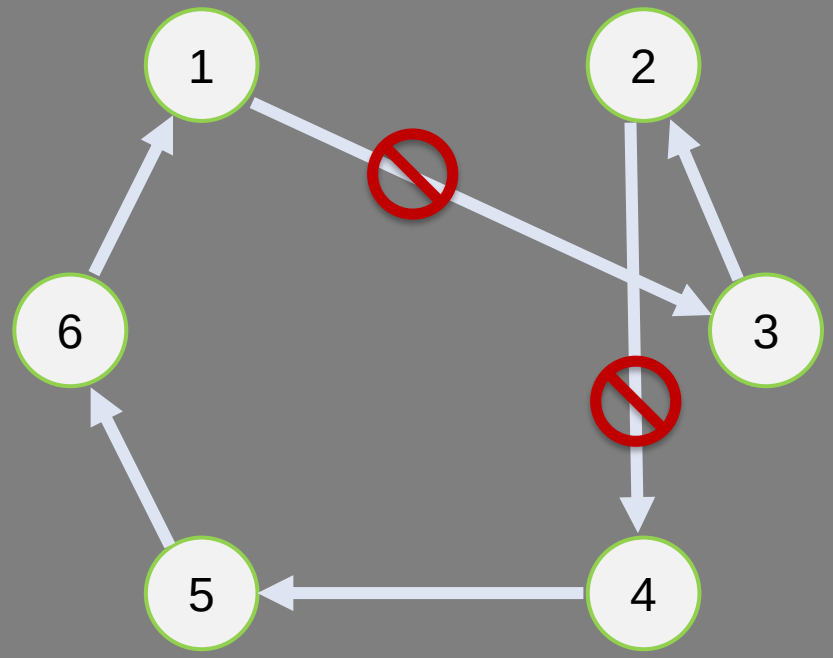
Vorgehen:

1-opt

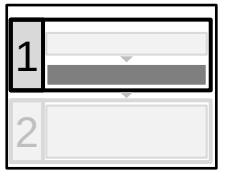
Die Kanten von **4 aufeinanderfolgenden Jobs** werden aufgelöst und die mittleren Jobs werden invertiert

Auftragsfolge

X = [**1, 3, 2, 4**, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



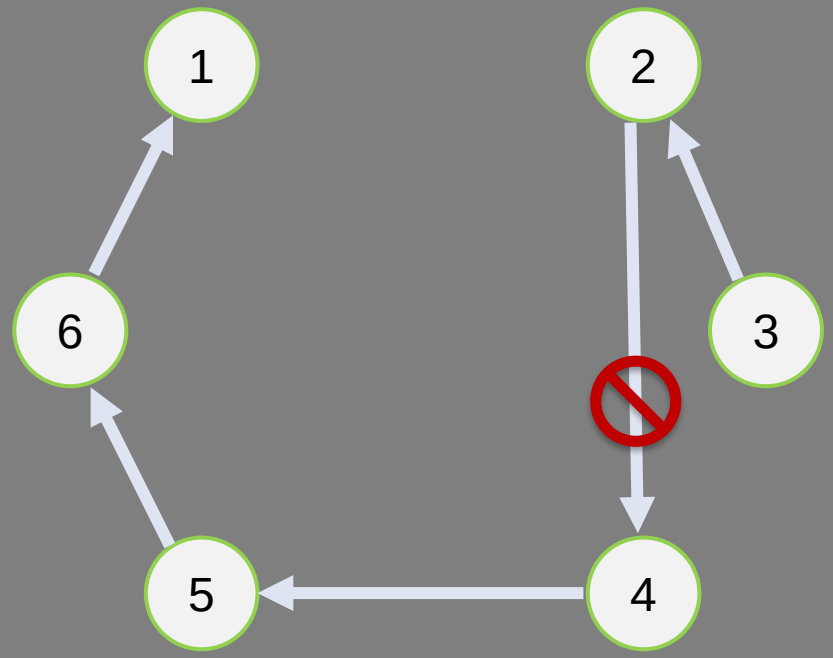
Vorgehen:

1-opt

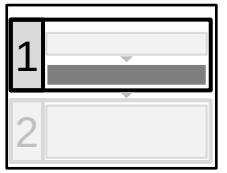
Die Kanten von **4 aufeinanderfolgenden Jobs** werden aufgelöst und die mittleren Jobs werden invertiert

Auftragsfolge

X = [**1, 3, 2, 4**, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



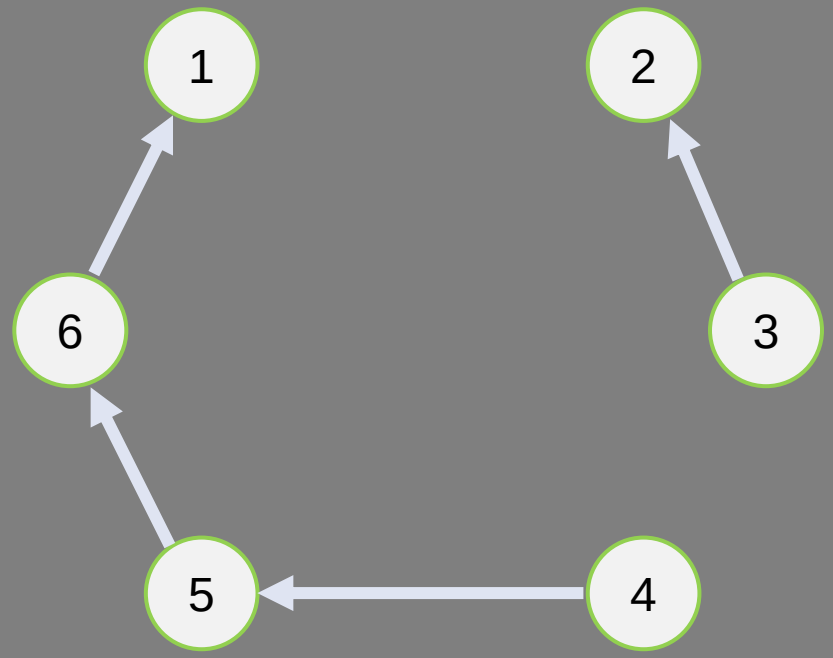
Vorgehen:

1-opt

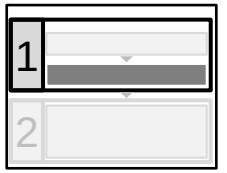
Die Kanten von **4 aufeinanderfolgenden Jobs** werden aufgelöst und die mittleren Jobs werden invertiert

Auftragsfolge

X = [**1, 3, 2, 4**, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



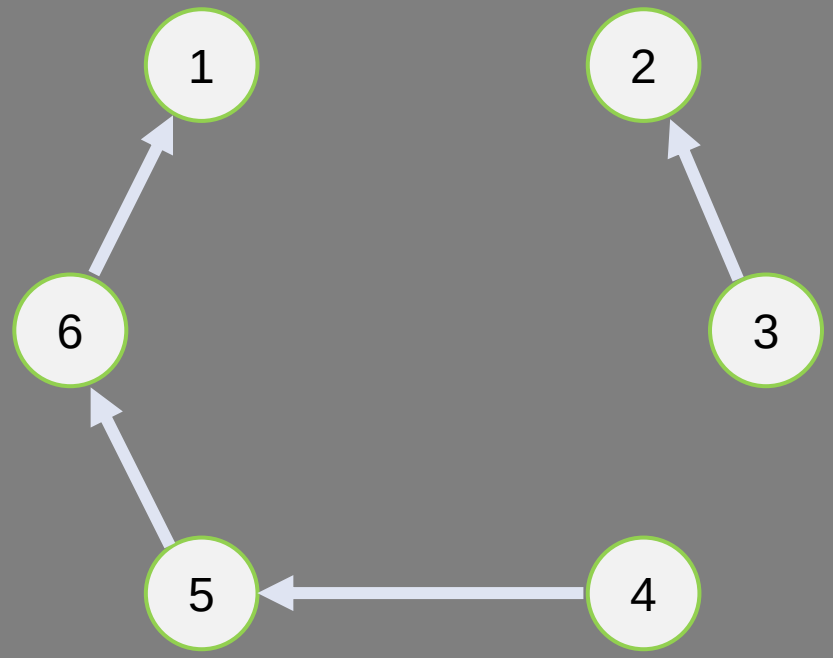
Vorgehen:

1-opt

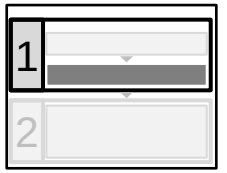
Die Kanten von 4 aufeinanderfolgenden Jobs werden aufgelöst und die **mittleren Jobs werden invertiert**

Auftragsfolge

X = [1, **3, 2,** 4, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



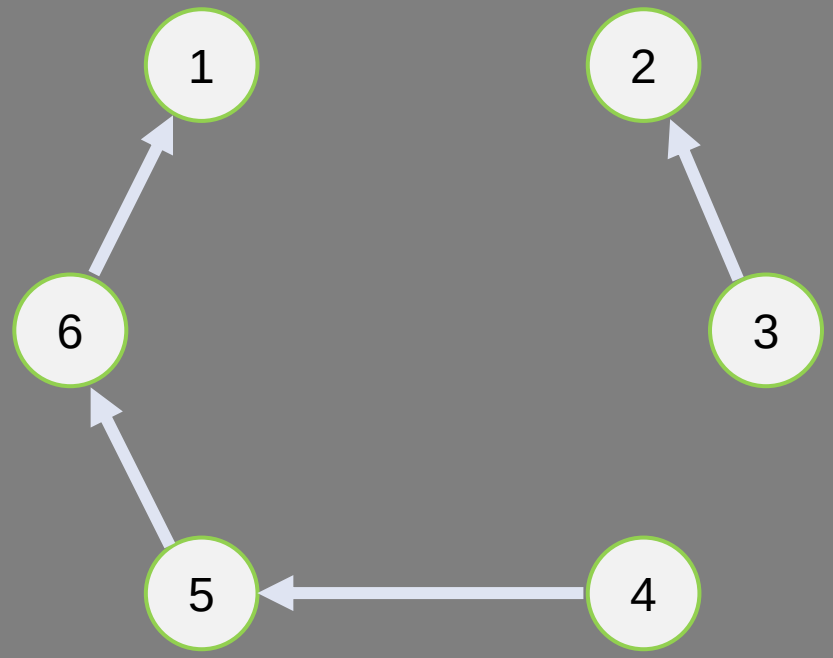
Vorgehen:

1-opt

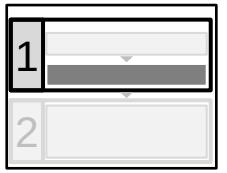
Die Kanten von 4 aufeinanderfolgenden Jobs werden aufgelöst und die **mittleren Jobs werden invertiert**

Auftragsfolge

X = [1, **3, 2,** 4, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



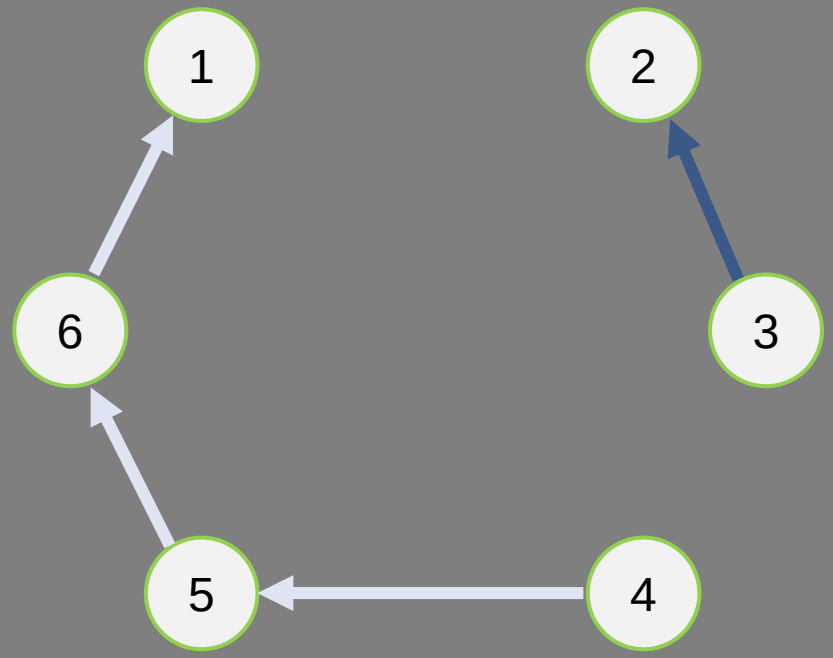
Vorgehen:

1-opt

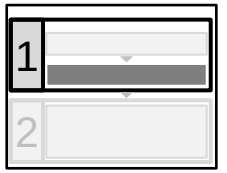
Die Kanten von 4 aufeinanderfolgenden Jobs werden aufgelöst und die **mittleren Jobs werden invertiert**

Auftragsfolge

X = [1, **3, 2,** 4, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



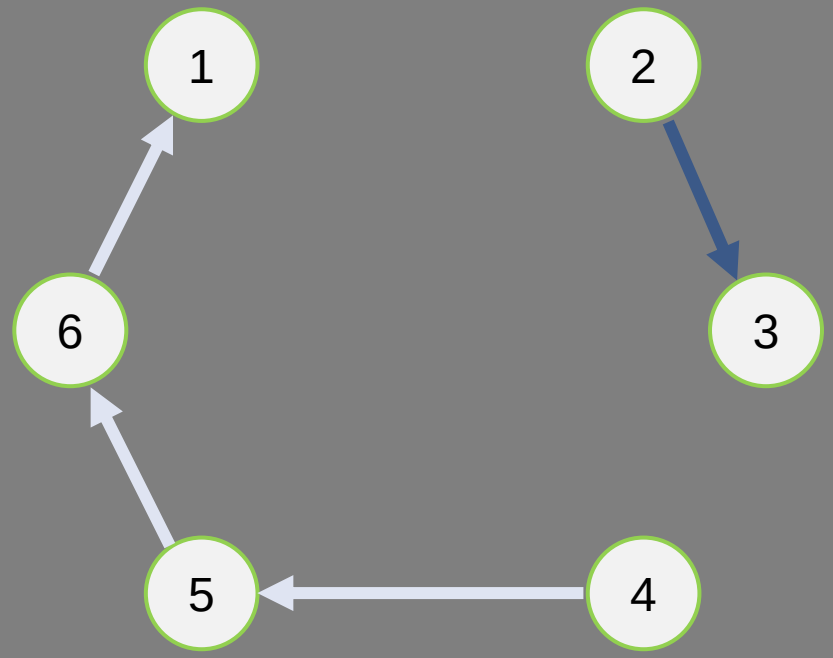
Vorgehen:

1-opt

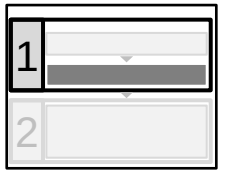
Die Kanten von 4 aufeinanderfolgenden Jobs werden aufgelöst und die **mittleren Jobs werden invertiert**

Auftragsfolge

X = [1, **2, 3,** 4, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



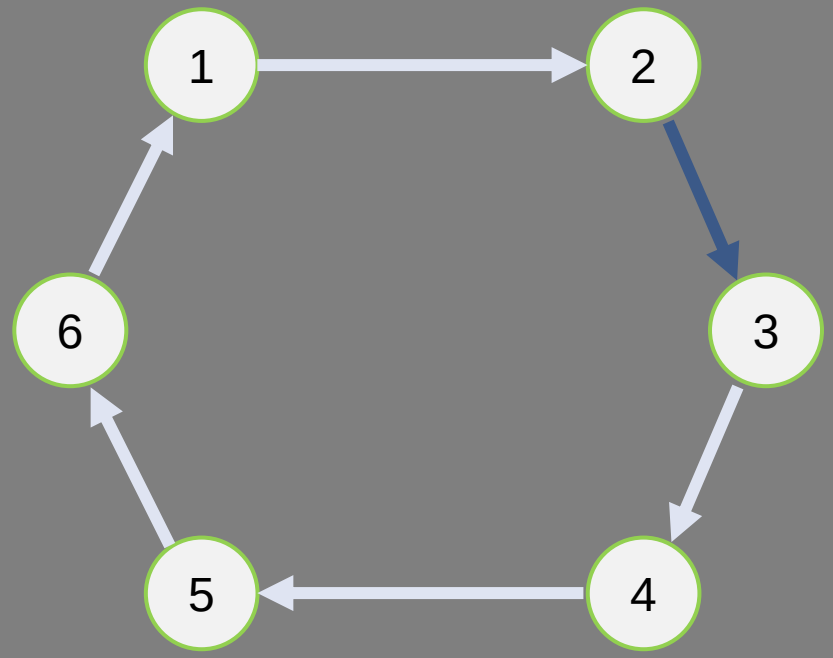
Vorgehen:

1-opt

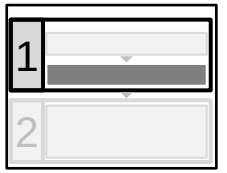
Die Kanten von 4 aufeinanderfolgenden Jobs werden aufgelöst und die **mittleren Jobs werden invertiert**

Auftragsfolge

X = [1, **2, 3,** 4, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



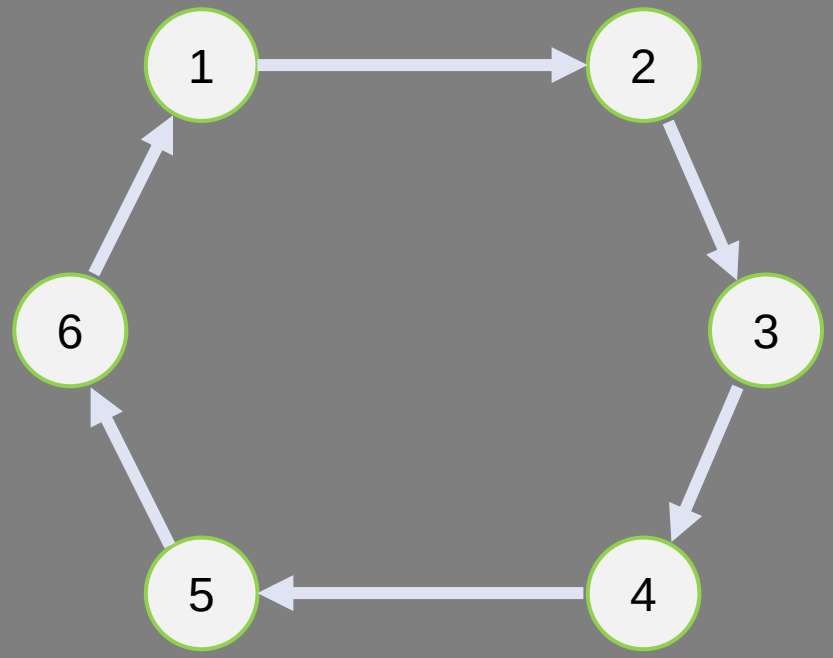
Vorgehen:

1-opt

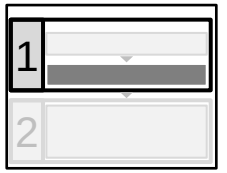
Die Kanten von 4 aufeinanderfolgenden Jobs werden aufgelöst und die mittleren Jobs werden invertiert

Auftragsfolge

X = [1, 2, 3, 4, 5, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



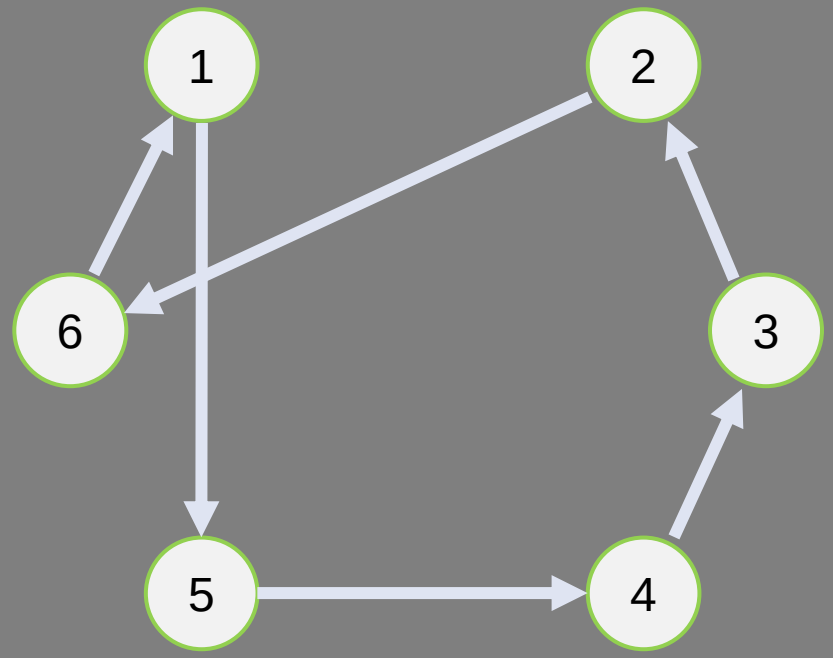
Vorgehen:

2-opt

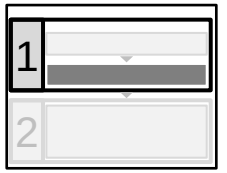
2 Kanten der derzeitigen Lösung werden aufgelöst und die zwischenliegende Jobreihenfolge wird invertiert

Auftragsfolge

X = [1, 5, 4, 3, 2, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



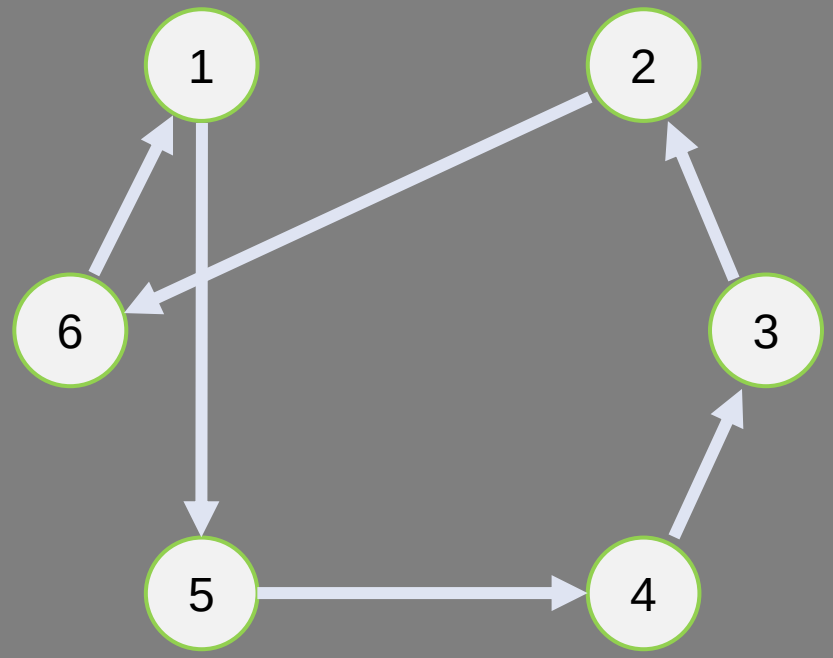
Vorgehen:

2-opt

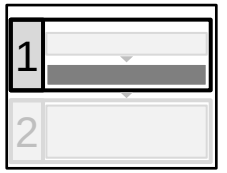
2 Kanten der derzeitigen Lösung werden aufgelöst und die zwischenliegende Jobreihenfolge wird invertiert

Auftragsfolge

X = [1, 5, 4, 3, 2, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



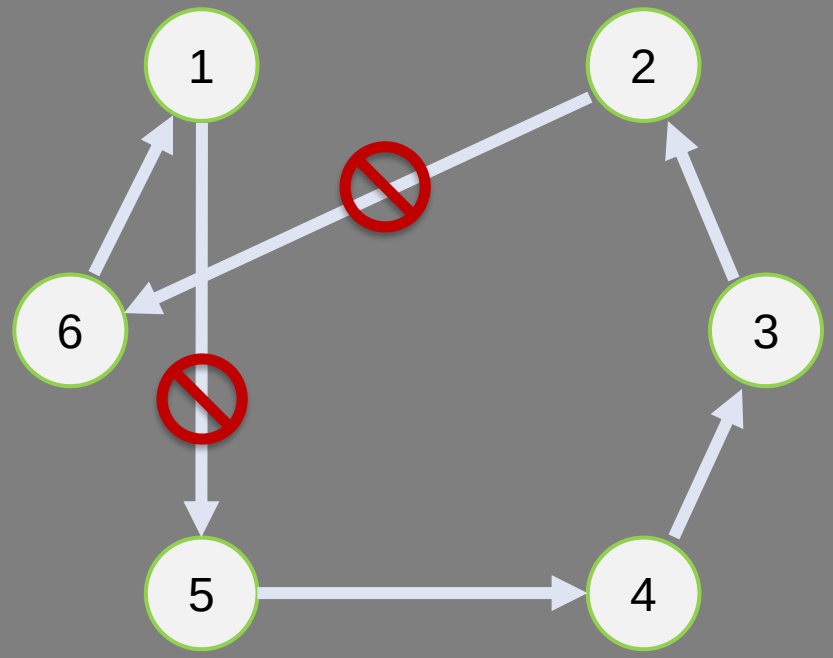
Vorgehen:

2-opt

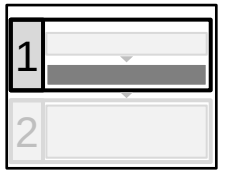
2 Kanten der derzeitigen Lösung werden aufgelöst und die zwischenliegende Jobreihenfolge wird invertiert

Auftragsfolge

X = [1, 5, 4, 3, 2, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



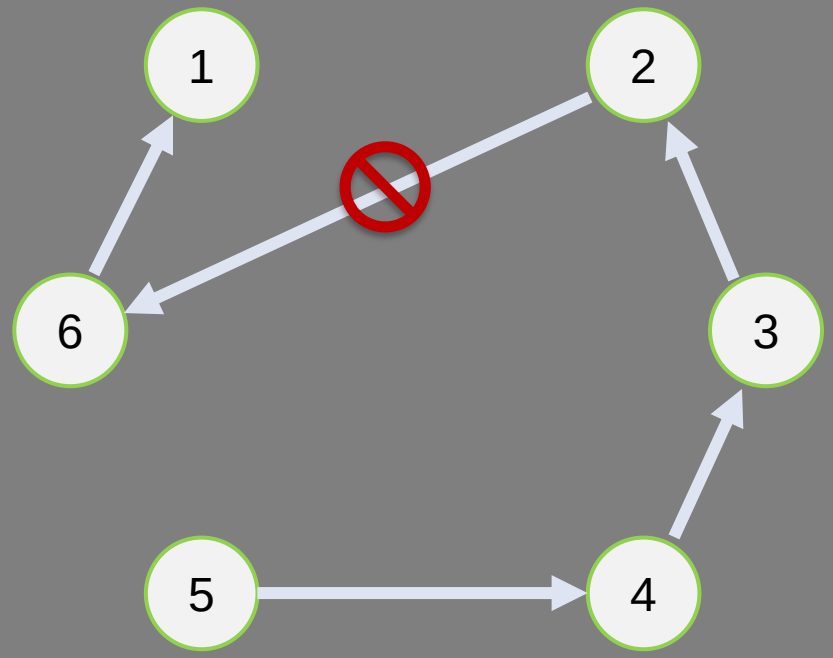
Vorgehen:

2-opt

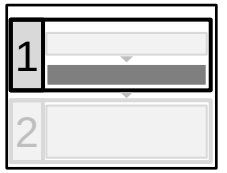
2 Kanten der derzeitigen Lösung werden aufgelöst und die zwischenliegende Jobreihenfolge wird invertiert

Auftragsfolge

X = [1, 5, 4, 3, 2, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



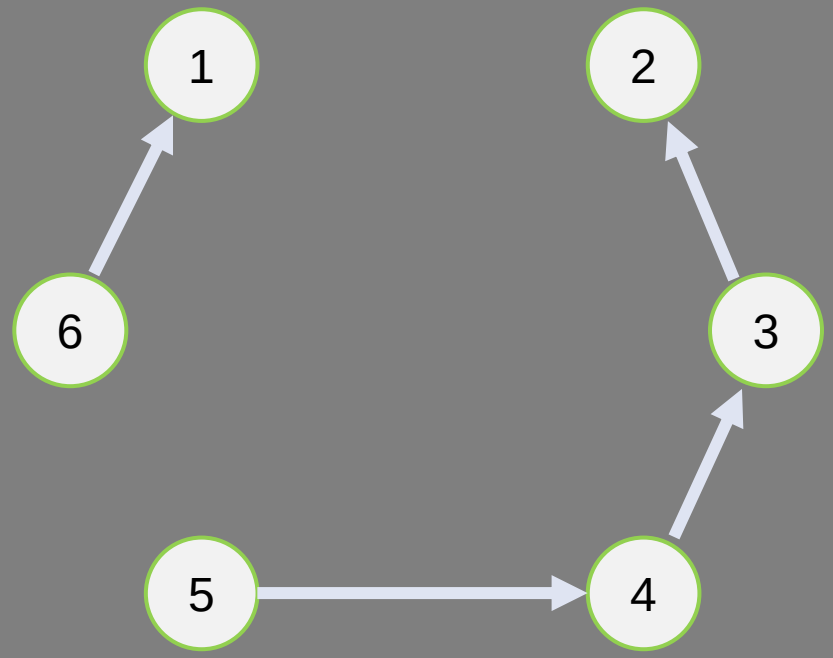
Vorgehen:

2-opt

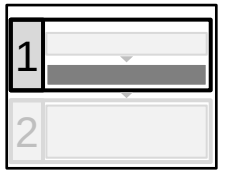
2 Kanten der derzeitigen Lösung werden aufgelöst und die zwischenliegende Jobreihenfolge wird invertiert

Auftragsfolge

X = [1, 5, 4, 3, 2, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



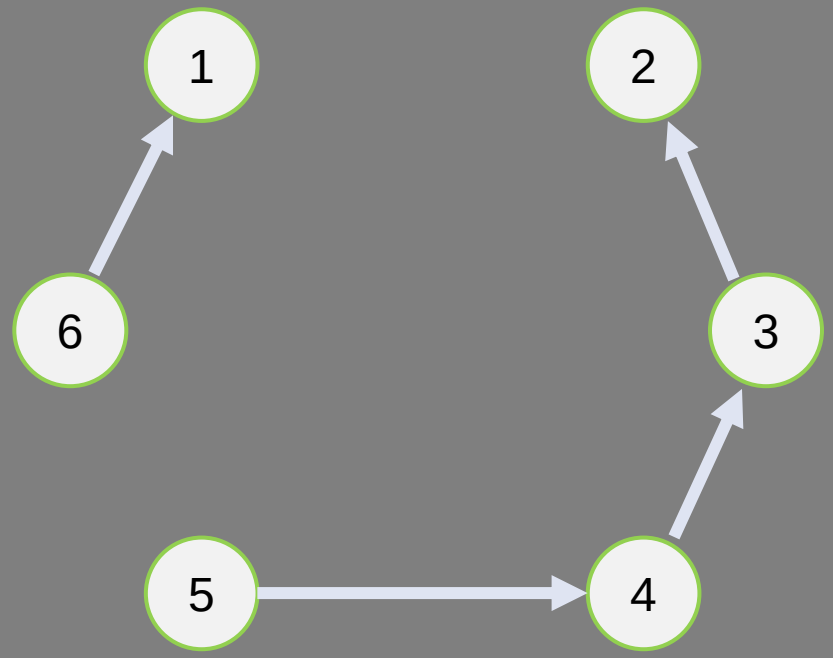
Vorgehen:

2-opt

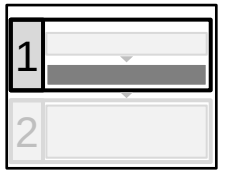
2 Kanten der derzeitigen Lösung werden aufgelöst und die **zwischenliegende Jobreihenfolge wird invertiert**

Auftragsfolge

X = [1, **5, 4, 3, 2**, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



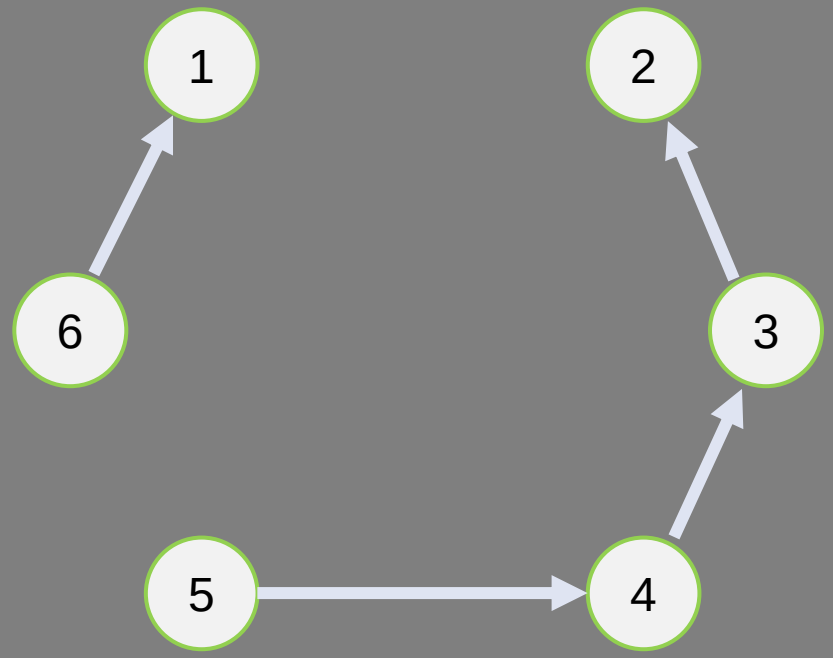
Vorgehen:

2-opt

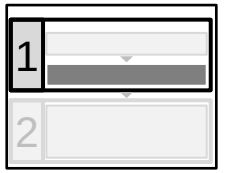
2 Kanten der derzeitigen Lösung werden aufgelöst und die **zwischenliegende Jobreihenfolge wird invertiert**

Auftragsfolge

X = [1, **2, 3, 4, 5**, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



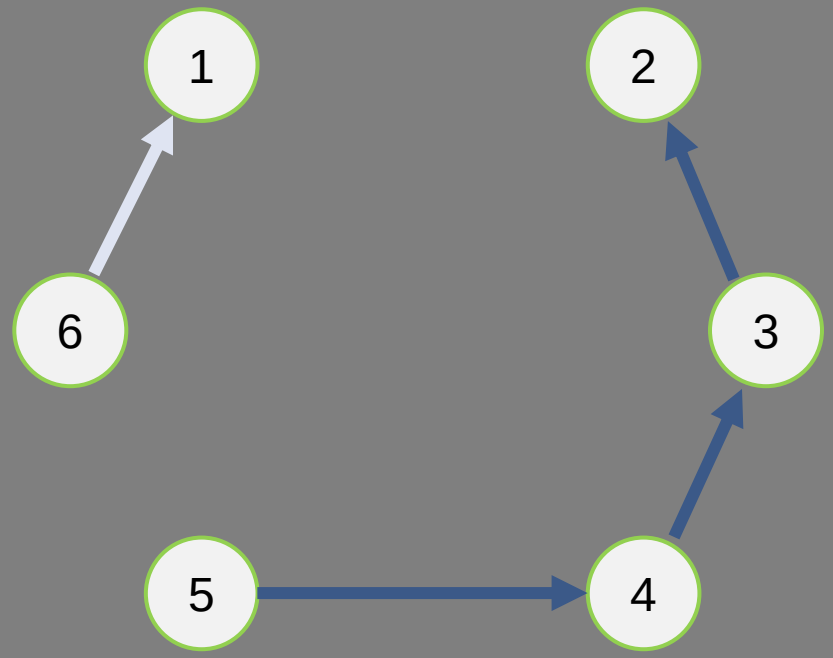
Vorgehen:

2-opt

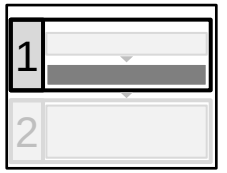
2 Kanten der derzeitigen Lösung werden aufgelöst und die **zwischenliegende Jobreihenfolge wird invertiert**

Auftragsfolge

X = [1, **2, 3, 4, 5**, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



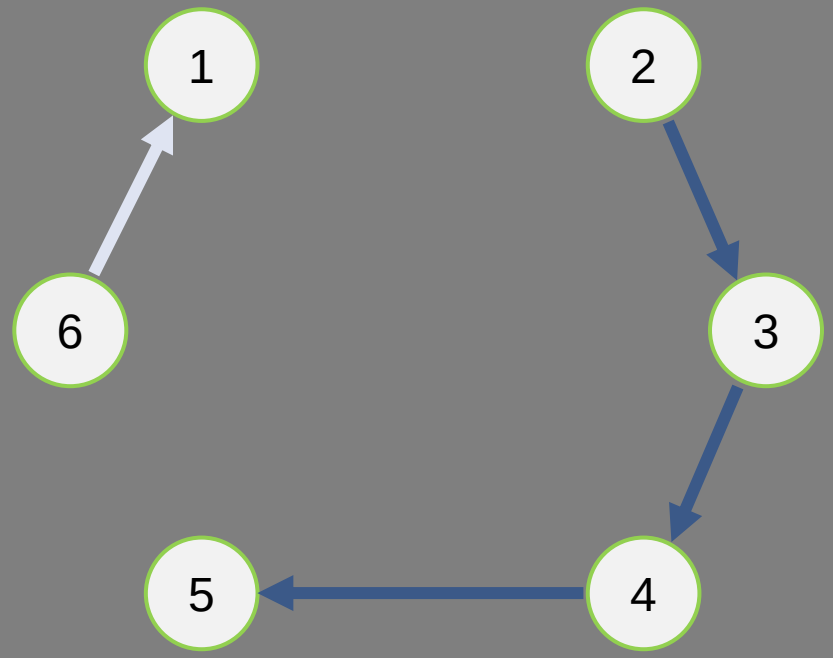
Vorgehen:

2-opt

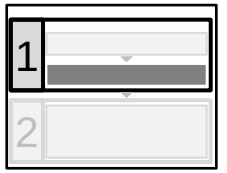
2 Kanten der derzeitigen Lösung werden aufgelöst und die **zwischenliegende Jobreihenfolge wird invertiert**

Auftragsfolge

X = [1, **2, 3, 4, 5**, 6]



Schritt 1: Verbesserung einer zulässigen Lösung durch Variable Neighborhood Descent (VND)



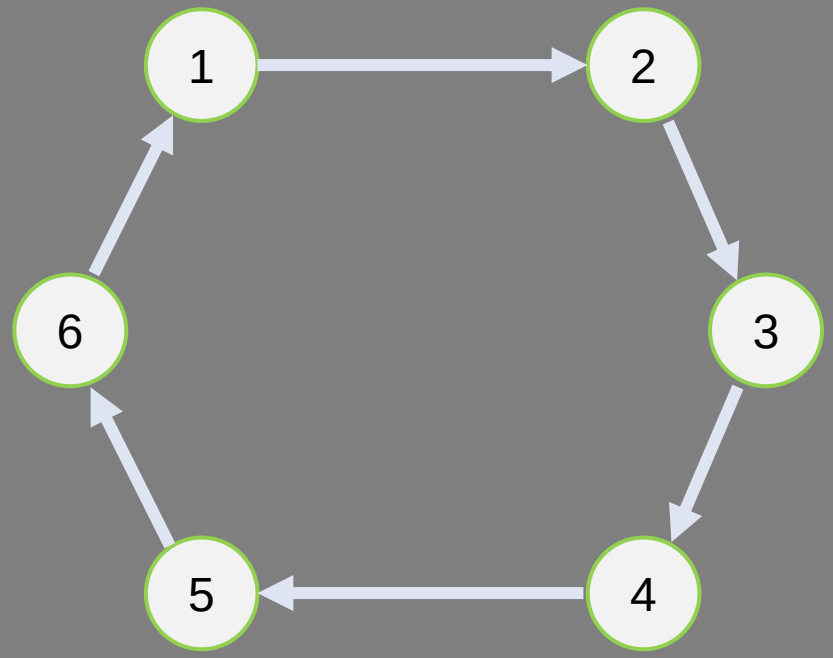
Vorgehen:

2-opt

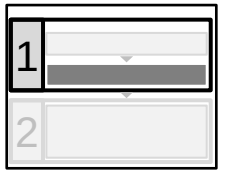
2 Kanten der derzeitigen Lösung werden aufgelöst und die zwischenliegende Jobreihenfolge wird invertiert

Auftragsfolge

X = [1, 2, 3, 4, 5, 6]



Schritt 1: Weitere Verbesserung durch zusätzliche Iterationen und Perturbation um lokalen Minima zu entkommen



Algorithmus

Verfahren der iterative Nachbarschaftssuche mit Perturbation

Algorithm 5: GVNS nach Mladenovic

```
1  X = build feasible solution by VNS;
2  repeat
3      k = 1;
4      while k <= kmax do
5          X' = Shake(x, k);
6          X" = SeqVND(X');
7          k = k +1;
8          if X" is better than X then
9              X = X", k = 1;
10         end
11     end
12 until t <= tmax;
```

Quelle: Mladenovic et al. 2013

Weiteres Vorgehen und Abbruchkriterium

- Sukzessive Erhöhung von *k* zur Perturbation der Lösung, wenn Nachbarschaftssuche keine Verbesserung bringt bis k_{max} erreicht
- wird eine verbesserte Lösung durch VND-Suche gefunden, zurücksetzen von *k* auf 1 und update der aktuell besten Lösung
- iteratives Vorgehen bis definiertes time-out t_{max} erreicht ist

Schritt 2: Die Lösung des TSPTW dient als Input für folgendes MILP

Ziel

Minimierung der Energiekosten auf Basis des zuvor erstellen Ablaufplans.

Restriktionen

Energiebedarf für Job i basierend auf zurückgelegter Strecke des Krans.

Ein Job kann über mehrere Perioden mit unterschiedlichen Strompreisen andauern, daher Linearisierung der Gesamtverbrauchsgleichung:

$$RE_i = EC_{ip} \cdot z_{ip}$$

Job j kann frühestens nach der Startzeit von Job i zzgl. der benötigten Abfertigungszeit für i beginnen. Big-M-Methode, um nicht aufeinanderfolgende Jobs auszuschließen.

Zeitfensterbeschränkungen $[a_i, d_i]$ergeben sich durch Fahrplan der Züge.

Zielfunktion

Strompreis in Periode p

$$\min \sum_{p \in P} \sum_{i \in R} c_p EC_{ip}$$

Energiebedarf für Job i in Periode p

Nebenbedingungen

$$RE_i = \sum_{j \in J} e_{ij} x^*_{ij} \qquad \forall i, j \in R$$

$$EC_{ip} \leq M z_{ip} \qquad \forall i \in R, \forall p \in P$$
$$EC_{ip} \leq RE_i \qquad \forall i \in R, \forall p \in P$$
$$EC_{ip} \geq RE_i - (1 - z_{ip})\, M \qquad \forall i \in R, \forall p \in P$$
$$EC_{ip} \geq 0 \qquad \forall i \in R, \forall p \in P$$

$$ST_j \geq ST_i + x^*_{ij} t_{ij} - \left(1 - x^*_{ij}\right) M \qquad \forall i, j \in R$$
$$a_i \leq ST_i \leq d_i \qquad \forall i \in R$$
$$\sum z_{ip} = 1 \qquad \forall i \in R$$
$$RE_i \geq 0 \qquad \forall i \in R$$
$$ST_i \geq 0 \qquad \forall i \in R$$
$$z_{ip} \in \{0, 1\} \qquad \forall i \in R, \forall p \in P$$

Quelle: Pohl et al. 2019

Analyse des Kostensenkungspotenzials durch Energy Demand Side Management in elektrisch betriebenen Containerterminals

1 Betrachtungsgegenstand Containerhafen

2 Problemformulierung

3 Vorstellung Lösungsverfahren

4 **Ergebnisse**

5 Fazit und Diskussion

Ergebnisse – Ermittlung der Kostenreduktionspotentiale

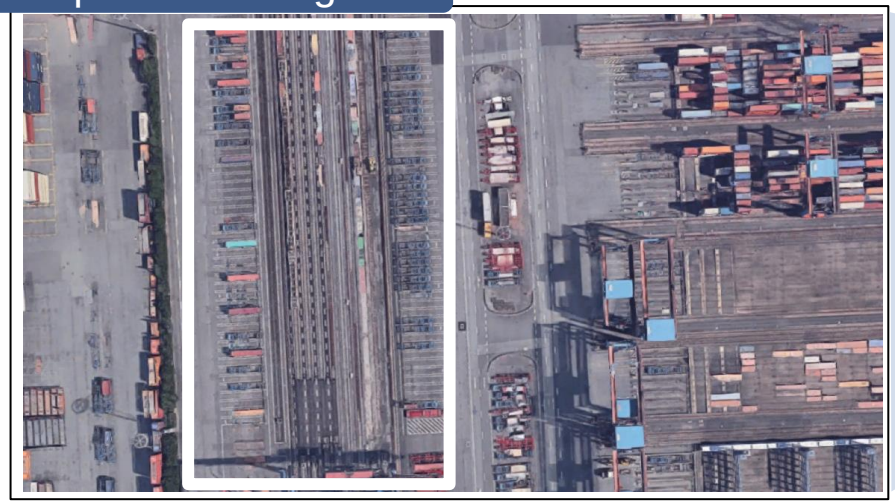
Hinweise und Annahmen

- Implementierung der Heuristiken und des MILP in Python 3
- Preisdaten: Großhandelspreise DE für das Jahr 2019 (Bundesnetzagentur | Smard.de)
- Anwendung des Lösungsverfahren auf Probleminstanzen mit…
  - … unterschiedlicher Anzahl an zu betrachtenden Zügen…
  - … und unterschiedlicher Anzahl an zu betrachtenden Containern pro Zug.
- Berechnung auf täglicher Basis für jeden Tag des Jahres

Für die verwendeten Probleminstanzen ergeben sich die folgenden Kostenreduktionspotentiale

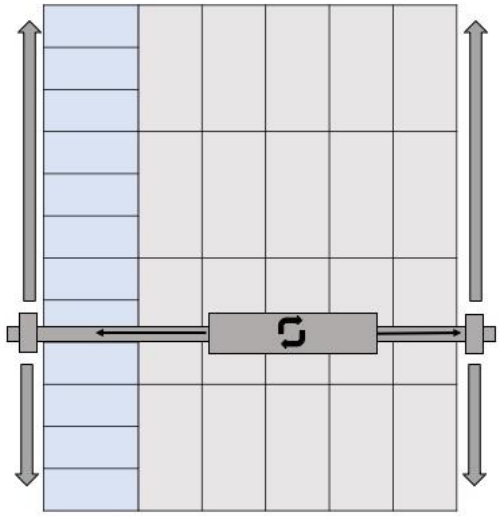
Erläuterung der verwendeten Probleminstanzen – Beispiel t4c10

Fahrpläne der Züge



TrainID	ETA	ETD	Track
t1	5100	29400	Track3
t2	6000	30300	Track2
t3	20700	45000	Track1
t4	26100	50400	Track4

Informationen zu den Containern



t=1

In	Size	StartX	StartY	TargetX	TargetY	Out	Size	StartX	StartY	TargetX	TargetY
c1	40	8,25	37	12,25	20	**c6**	20	8,125	20	4,125	37
c2	20	20,625	37	28,625	20	**c7**	20	20,375	20	12,375	37
c3	40	33	37	29	20	**c8**	20	24,625	20	20,625	37
c4	40	49,5	37	53,5	20	**c9**	20	28,875	20	28,875	37
c5	20	61,875	37	69,875	20	**c10**	20	29,125	20	37,125	37

⋮ ⋮

t=4

In	Size	StartX	StartY	TargetX	TargetY	Out	Size	StartX	StartY	TargetX	TargetY
c1	40	8,25	30	12,25	20	**c6**	40	12,25	20	8,25	30
c2	20	20,625	30	24,625	20	**c7**	20	20,625	20	20,625	30
c3	40	33	30	41	20	**c8**	20	40,875	20	28,875	30
c4	40	49,5	30	61,5	20	**c9**	20	45,125	20	37,125	30
c5	20	61,875	30	53,875	20	**c10**	20	45,375	20	45,375	30

Ergebnisse für Probleminstanzen mit bis zu 4 Zügen und 40 Containern je Zug

Instanz	Gesamte Energiekosten ohne DR in 2019 [€]	Gesamte Energiekosten mit DR in 2019 [€]	Relatives Kostenpotential durch DR
t1c5	51,9	37,5	27,7%
t1c10	117,2	85,0	27,5%
t1c20	286,6	207,7	27,5%
t1c40	566,4	410,4	27,5%
t2c5	106,0	76,8	27,5%
t2c10	239,1	173,3	27,5%
t2c20	542,3	392,5	27,6%
t2c40	1.269,6	1.004,7	20,9%
t3c5	141,5	105,9	25,2%
t3c10	332,1	247,6	25,4%
t3c20	762,9	569,7	25,3%
t3c40	1.792,6	1.519,1	15,3%
t4c5	206,5	173,7	15,9%
t4c10	499,0	422,2	15,4%
t4c20	1.132,7	970,0	14,4%
t4c40	2.346,8	1.988,4	15,3%

Energieverbrauch Kran

0,024 kWh pro Meter bzw.
0,034 kWh pro Sekunde

Instanzbezeichnung

t: Anzahl der betrachteten Züge

c: Anzahl an betrachteten Containern je Zug

Ergebnisse für Probleminstanzen mit bis zu 4 Zügen und 40 Containern je Zug

Instanz	Gesamte Energiekosten ohne DR in 2019 [€]	Gesamte Energiekosten mit DR in 2019 [€]	Relatives Kostenpotential durch DR
t1c5	51,9	37,5	27,7%
t1c10	117,2	85,0	27,5%
t1c20	286,6	207,7	27,5%
t1c40	566,4	410,4	27,5%
t2c5	106,0	76,8	27,5%
t2c10	239,1	173,3	27,5%
t2c20	542,3	392,5	27,6%
t2c40	1.269,6	1.004,7	20,9%
t3c5	141,5	105,9	25,2%
t3c10	332,1	247,6	25,4%
t3c20	762,9	569,7	25,3%
t3c40	1.792,6	1.519,1	15,3%
t4c5	206,5	173,7	15,9%
t4c10	499,0	422,2	15,4%
t4c20	1.132,7	970,0	14,4%
t4c40	2.346,8	1.988,4	15,3%

Energieverbrauch Kran

0,024 kWh pro Meter bzw.
0,034 kWh pro Sekunde

Globale Betrachtung

- spezifischer Verbrauch an elektrischem Strom für einen Kran: 90 kWh/Bh
- Insgesamt für alle am Terminal eingesetzten Bahnkräne ca. 685.000 kWh/a
- Relative Kostenersparnis von 15% bis 25% bei den Stromkosten entspricht bei Industriestrompreis i.H.v. 18,44 ct/kWh (BDEW 2019) somit ca. 25.000 € pro Jahr.

Analyse des Kostensenkungspotenzials durch Energy Demand Side Management in elektrisch betriebenen Containerterminals

1 Betrachtungsgegenstand Containerhafen

2 Problemformulierung

3 Vorstellung Lösungsverfahren

4 Ergebnisse

5 Fazit und Diskussion

Fazit und Diskussion

- Implementierung einer Heuristik zur Bestimmung von zulässigen und möglichst energiesparenden Ablaufplänen für praxisnahe Problemgrößen mit bis zu 4 Zügen und 40 Containern je Zug.
- Vorstellung eines MILP zur Ausnutzung von eventuell auftretenden Pufferzeiten, um weitere Kostenreduktionspotentiale durch Teilnahme an Demand Response Programm zu realisieren.
- Für den betrachteten Containerterminal ergibt sich auf Basis der Lösung über Dekomposition ein jährliches Kostenreduktionspotential von ca. 25.000 € durch Teilnahme an Demand Response und Anpassung des Ablaufplans.

Literaturverzeichnis

- Albadi, M. H., & El-Saadany, E. F. (2007). Demand response in electricity markets: An overview. In *2007 IEEE power engineering society general meeting* (pp. 1-5). IEEE.
- Bundesnetzagentur (2019): Großhandelspreise 2019. Online verfügbar unter smard.de, zuletzt geprüft am 12.03.2020.
- Bundesverband der Energie- und Wasserwirtschaft e.V. (2019): Strompreis für die Industrie. Online verfügbar unter https://www.bdew.de/service/daten-und-grafiken/strompreis-fuer-die-industrie/, zuletzt geprüft am 2.10.2020
- Da Silva, R. F., & Urrutia, S. (2010). A General VNS heuristic for the traveling salesman problem with time windows. Discrete Optimization, 7(4), 203-211.
- Mladenović, N., Todosijević, R., & Urošević, D. (2013). An efficient general variable neighborhood search for large travelling salesman problem with time windows. *Yugoslav Journal of Operations Research*, *23*(1), 19-30.
- Pohl, E., Lauven, L. P., & Geldermann, J. (2019). An Exact Method for Cost-Minimal Deployment of a Rail Crane Using Demand Response. In *Operations Research Proceedings 2018* (pp. 183-189). Springer, Cham.
- Schmidt, J., Lauven, L. P., Ihle, N., & Kolbe, L. M. (2015). Demand side integration for electric transport vehicles. *International Journal of Energy Sector Management*.
- Steenken, D., Voß, S., & Stahlbock, R. (2004). Container terminal operation and operations research-a classification and literature review. *OR spectrum*, *26*(1), 3-49.
- Strbac, G. (2008). Demand side management: Benefits and challenges. *Energy policy*, *36*(12), 4419-4426.36(12):4419–4426.