Zusammenfassung - BWL: Prduktion, Logistik und Wirtschaftsinformatik

Julian Shen

1. April 2023

1 Einführung in die Logistik und SCM

Logistik:

- **Definition**:
  - Planung, Implementierung und Kontrolle
  - von effizienten, effektiven Vor- und Rückflüssen
  - sowie der Lagerung von Gütern, Dienstleistungen und Informationen
  - zwischen Ursprungs- und Verbrauchsort
  - mit dem Ziel, die Kundenanforderungen zu erfüllen
- **Aufgabe der Logistik** ist es,
  - den Kunden mit dem richtigen Produkt, am richtigen Ort, zur richtigen Zeit,
  - unter gleichzeitiger Optimierung eines vorgegebenen Leistungskriteriums (z. B. Minimierung der Gesamtkosten),
  - und unter Berücksichtigung gegebener Anforderungen (z. B. Servicegrad) und Beschränkungen (z. B. Budget) zu versorgen
- **7 R's der Logistik**:
  - Richtiges Produkt
  - Richtige Zeit
  - Richtiger Ort
  - Richtige Menge
  - Richtige Qualität
  - Richtige Kosten
  - Richtige Information
- **Auf was bezieht sich Logistik heute?**
  - Alle arbeitsteiligen Wirtschaftssysteme, in denen es auf zeit-, kosten- und mengenabhängige Verteilung von Gütern und Dienstleistungen ankommt

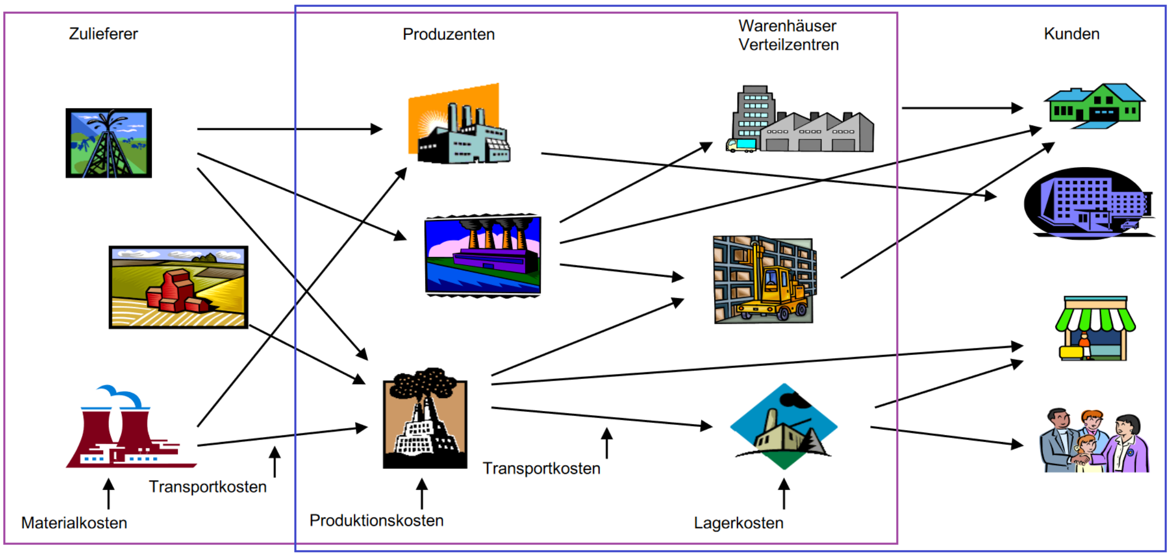
Supply Chain:

- Komplexes, unternehmensübergreifendes, interlogistisches System, das die Vorgänge und Funktionen der Beschaffung, Produktion, Verarbeitung, Lagerung und Distribution von Objekten umfasst
- Keine einfache Kette, sondern ein komplexes Netzwerk mit sich verzweigenden und zusammenführenden Informations- und Materialflüssen

Supply Chain Management (SCM):

- Koordination und Kollaboration von Stakeholdern entlang der gesamten Supply Chain, d.h. auch über die eigene Organisation hinaus, insbesondere mit Zulieferern, Zwischenhändlern, Service-Dienstleistern und Kunden
- Umfasst alle Aktivitäten des Logistik Management sowie Produktionsaktivitäten, Vertrieb, Produktdesign, Finanzen und IT

Supply Chain Network:

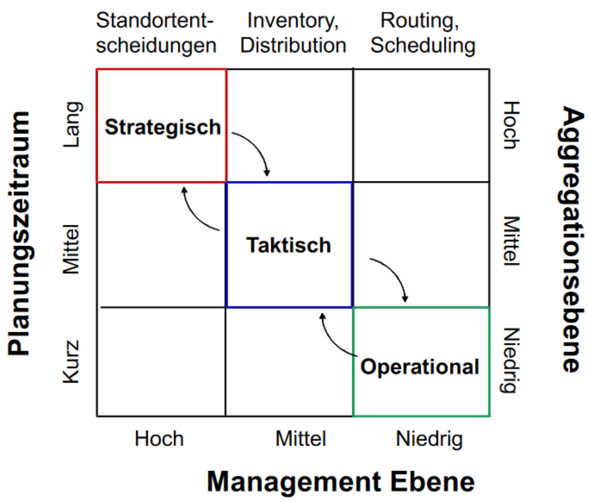


- Quellen, Lieferanten, Auslieferer stellen Objekte zur Verfügung, z.B. Rohstofflager, Produktionsanlagen, Fabriken, Vorratslager, Importlager, Logistikzentren
- Senken oder Anlieferstellen haben Nachfragen nach Objekten, z.B. Einzelhändler, Märkte, Filialen, Konsumenten, Müllverbrennungsanlagen
- Warenquellen können selbst Empfänger von Gütern aus anderen Quellen sein
- Handel und Konsumenten sind wiederum Quellen von Leergut, Restoffen und Verpackungsabfall, die entsorgt werden müssen → **Reverse Logistics**

Planungsebenen des Supply Chain Managements:

- **Strategisch – Supply Chain Configuration:**
  - Entscheidungen mit langfristigem Effekt und hohem Kapitalaufwand
  - Planungszeitraum: mehrere Jahre
  - Daten: aggregiert, basieren auf Vorhersagen, oft unvollständig oder ungenau
  - **Beispiele**: Anzahl, Standorte und Kapazitäten von Einrichtungen, Investitionen in Produktions- und Lageranlagen, Layout von Einrichtungen

- **Taktisch – Supply Chain Planning:**
  - Entscheidungen, die die effektive Allokation von Produktions- und Distributionsressourcen betreffen
  - Planungszeitraum: 3 Monate bis 1 Jahr
  - Daten: detailliert, basieren auf Vorhersagen
  - **Beispiele**: Beschaffungs- und Produktionsentscheidungen, Wahl von Transport- und Versandstrategien, Lagerbestandsplanung
- **Operativ – Supply Chain Execution:**
  - Erstellt zeit- und mengengenaue unmittelbar umsetzbare Vorgaben für die Ausführung der Prozesse
  - Planungszeitraum: täglich, wöchentlich
  - Daten: sehr konkret, detailliert, bis auf unvorhergesehene Störungen vollständig aus ERP System bekannt
  - **Beispiele**: Scheduling (Produktion), Zuweisung von Aufträgen zu Maschinen, Auftragsverarbeitung, Fahrzeug-Routing, LKW-Beladung



Aggregationsebene: Wie detailliert sind die Daten

Logistik vs. SCM:

- **Logistik**: Betrachtung der Material- und Erzeugnisflüsse unter Berücksichtigung von Informations- und Wertströmen innerhalb der eigenen Organisation
- **SCM**: Gesamtes logistisches Wertschöpfungsnetz mit Lieferanten, Produzenten, Händlern, Konsumenten

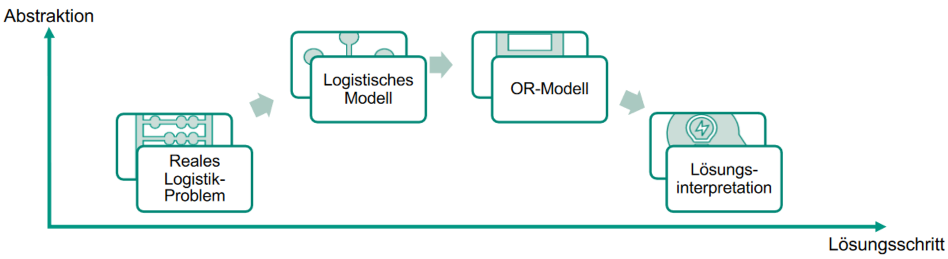
  Koordination und Kollaboration von Stakeholdern entlang der gesamten Supply Chain, auch über die eigene Organisation hinaus

Operations Research:

- Analysiert praxisnahe, komplexe Problemstellungen, um möglichst gute Entscheidungen zu treffen
- Probleme werden mithilfe mathematischer Modelle formuliert und mit mathematischen Lösungsmethoden gelöst
- Anwendbar auf verschiedenste Probleme in Logistik und SCM

Vorgehen beim Lösen von Problemen mit OR:

- Überführe realwirtschaftliches Logistikproblem in abstraktes, logistisches Modell
- Wandle logistisches Modell in OR-Modell (LP/MILP/MIP) um und löse mit bekannten Werkzeugen
- Interpretation der OR-Modell-Lösung und Schlussfolgerung für das reale Problem



- *Beispiel siehe Logistik VL 1, F27-34*
- *Rechenbeispiele siehe Logistik Tutblatt 1*

Wichtige Software für die Logistik:

- **Enterprise Resource Planning Systeme (ERP)** erfassen Daten aller wesentlichen Geschäftsfunktionen (z.B. Buchhaltung, Personalwesen) konsistent und up-to-date und machen diese unternehmensweit verfügbar (z.B. SAP, Oracle)
- Erweiterung zu **Advanced Planning Systems (APS)** helfen, komplexe Planungsaufgaben im SCM zu erfüllen und rationale Entscheidungen zu unterstützen
- APS nehmen die im ERP-System erhobenen Daten in Modelle entgegen und lösen die so entstandenen Probleme mittels OR-Algorithmen

2 Scheduling

Was ist Scheduling?

- Zuordnung von Aufträgen (**Jobs**) zu Arbeitsträgern, z.B. Maschinen, unter Beachtung von Nebenbedingungen zum Optimieren einer oder mehrerer Zielgrößen

Scheduling Notation:

- n **Jobs** müssen auf m **Maschinen** bearbeitet werden
- Job j hat auf Maschine i eine **Prozesszeit** p_{ij}
- Job j kann ein **Gewicht** w_j haben → Repräsentiert die Wichtigkeit des Jobs
- Job j kann einen **Liefertermin** d_j haben
- Notation eines **Scheduling-Problems**: $\alpha \mid \beta \mid \gamma$
  - α: Maschinenumgebung
  - β: Auftragscharakteristik und Beschränkungen
  - γ: Zielgröße

Performanz-Kenngrößen:

- **Fertigstellungszeitpunkt (Completion Time)** $\boldsymbol{C_j}$:
  - Zeitpunkt, zu welchem Job j fertiggestellt ist
  - Bei mehreren Maschinen C_{ij} (Fertigstellung von Job j auf Maschine i) gilt: $C_j = \max_{i \in I}\{C_{ij}\}$
- **Unpünktlichkeit (Lateness)** $L_j = C_j - d_j$ beschreibt die Abweichung vom Fertigstellungszeitpunkt zum Liefertermin. Negativ, wenn Produkt zu früh fertig
- **Verspätung (Tardiness)** $T_j = \max\{C_j - d_j, 0\}$ wie Lateness, aber erlaubt keine negativen Werte
- **Einheits-Strafe (Unit penalty)** $U_j = \begin{cases} 1\,, & \text{wenn } C_j > d_j \\ 0\,, & \text{sonst} \end{cases}$

  erhebt eine Einheitsstrafe, wenn Fertigstellungszeitpunkt zu spät

Maschinenumgebung ($\boldsymbol{\alpha}$):

- **Einzel Maschine (1)**
- **Parallele Maschinen ($\boldsymbol{Pm, Qm, Rm}$):**
  - Mehrere Maschinen, die gleichzeitig Jobs abarbeiten
  - Pm: m identische Maschinen (gleiche Geschwindigkeit)
  - Qm: m Maschinen mit unterschiedl., job-unspezifischen Geschwindigkeiten
  - Rm: m Maschinen mit unterschiedl., job-spezifischen Geschwindigkeiten
- **Flow-Shop ($\boldsymbol{Fm}$)**: m Maschinen in Serie, alle Jobs müssen diese durchlaufen (selbe Maschinen-Reihenfolge)

- **Job-Shop (Jm)**: m Maschinen, alle Jobs müssen diese durchlaufen, haben jedoch unterschiedliche Maschinen-Reihenfolge

Auftragscharakteristik (β):

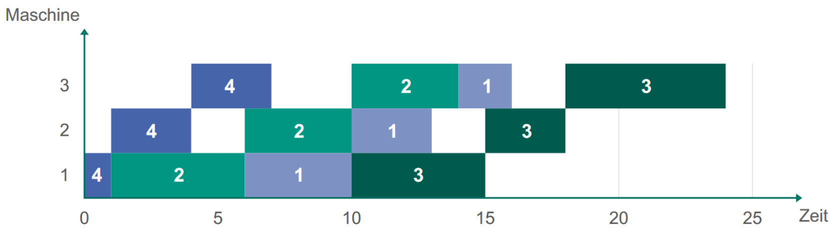
- **Freigabezeiten (Release dates) (r_j)**: Auftrag kann nicht vor diesem Zeitpunkt gestartet werden
- **Unterbrechungen (Preemptions) ($prmp$)**: Bearbeitung eines Auftrags kann unterbrochen und später fortgesetzt werden
- **Permutation ($prmu$)**: Job-Reihenfolge auf der ersten Maschine muss beibehalten werden
- **Rüstzeiten (Setup times) (s_{jk}, s^i_{jk})**:
  - Bevor mit Auftrag k begonnen werden kann, ist Maschine i durch Umrüstung blockiert
  - s_{jk}: Rüstzeit ist nur von den aufeinanderfolgenden Jobs j und k abhängig
  - s^i_{jk}: Rüstzeit ist zusätzlich von Maschine i abhängig

Zielfunktion (γ):

- **Makespan (C_{max})**: Entspricht Gesamtproduktionszeit, also der Zeit, wenn der letzte Job fertiggestellt ist: $C_{max} = \max_{j \in J}\{C_j\}$
- **Gesamtfertigstellungszeiten (Total completion time) ($\sum C_j$):** Summe der Fertigstellungszeiten der Jobs
- **Gewichtete Gesamtfertigstellungszeiten (Total weighted completion time) ($\sum w_j C_j$):** Summe der gewichteten Fertigstellungszeiten der Jobs
- **Gesamtverspätung (Total tardiness) ($\sum T_j$):** Summe der Verspätungszeiten
- **Anzahl verspäteter Jobs (Number of tardy Jobs) ($\sum U_j$):** Summe der Einheitsstrafen

Gantt-Charts:

- Visualisierungsmöglichkeit von Scheduling-Lösungen
- Block für die Bearbeitung von Job j auf Maschine i ist auf Höhe von i und Länge des Blocks entspricht Prozesszeit p_{ij}
- Innerhalb des Blocks steht die Job-Nummer oder Prozesszeit (problemabhängig)



2.1 Ein-Maschinen-Probleme

- **Problemstellung**: n Jobs sollen auf einer Maschine in Reihenfolge gebracht werden
- Jeder Schedule kann als Permutation der Jobs $1, \dots, n$ angesehen werden $\rightarrow n!$ verschiedene Schedules

Minimierung der Fertigstellungszeiten:

- Problem $1 \,||\, C_{max}$ ist trivial, da $C_{max} = \sum_{j=1}^{n} p_j$ für jeden Schedule
- Problem $1 \,||\, \sum_{j=1}^{n} C_j$ lässt sich mit **SPT-Regel** (Shortest Processing Time first) optimal lösen $\rightarrow$ Individuelle Fertigstellungszeitpunkte so gering wie möglich halten

Job	1	2	3	4
p_j	6	3	9	4

$\rightarrow$

Job	2	4	1	3
p_j	3	4	6	9
C_j	3	7	13	22

$\rightarrow \sum_{j=1}^{n} C_j = 45$

Minimierung gewichteter Fertigstellungszeiten:

- Problem $1 \,||\, \sum_{j=1}^{n} w_j C_j$ lässt sich mit **WSPT-Regel** (Weighted shortest processing time) optimal lösen

Job	1	2	3	4	5	6	7	8
p_j	8	6	5	9	4	5	4	7
w_j	2	3	1	3	0,5	5	2	1
p_j/w_j	4	2	5	3	8	1	2	7

- Ergebnis: $S = \{6, 2, 7, 4, 1, 3, 8, 5\}$ mit $\sum_{j=1}^{n} w_j C_j(S) = 329$

Minimierung der Anzahl verspäteter Jobs:

- Problem $1 \,||\, \sum_{j=1}^{n} U_j$ lässt sich mit **Moore's Algorithmus** optimal lösen

1. **Initialisierung**: Sortiere alle Jobs in aufsteigender Reihenfolge nach Lieferterminen $\rightarrow$ Schedule S und setze $J = \emptyset$
2. **Job-Auswahl**:
   - Wenn ein verspäteter Job in S existiert $\rightarrow$ Betrachte ersten verspäteten Job j' in S
   - Sonst: Gehe zu 4.
3. **Job-Entfernung**: Wähle Job u mit größter Prozesszeit, der vor j' kommt und setze $S := S \setminus \{u\}$ und $J := J \cup \{u\}$
4. **Terminierung**: Füge Jobs aus J in beliebiger Reihenfolge an S

Job	1	2	3	4	5	6	7	8
p_j	8	6	5	7	4	8	4	7
d_j	13	12	15	24	20	19	30	40

EDD-Regel

Job	2	1	3	6	5	4	7	8
p_j	6	8	5	8	4	7	4	7
d_j	12	13	15	19	20	24	30	40

Entfernen von Job 1, da $p_1 > p_2$

Entfernen von Job 6, da größtes p_j unter platzierten Jobs

Jod-ID	2	1	3	6	5	4	7	8	J
C_j in ZE	6	14							$\emptyset$
C_j in ZE	6	*							$\{1\}$
C_j in ZE	6	*	11	19	23				$\{1\}$
C_j in ZE	6	*	11	*	15				$\{1, 6\}$
C_j in ZE	6	*	11	*	15	22	26	33	$\{1, 6\}$

- Ergebnis: $S = \{2, 3, 5, 4, 7, 8, 1, 6\}$ oder $S = \{2, 3, 5, 4, 7, 8, 6, 1\}$ mit $\sum_{j=1}^{n} U_j(S) = 2$

2.2 Flow-Shop-Umgebung

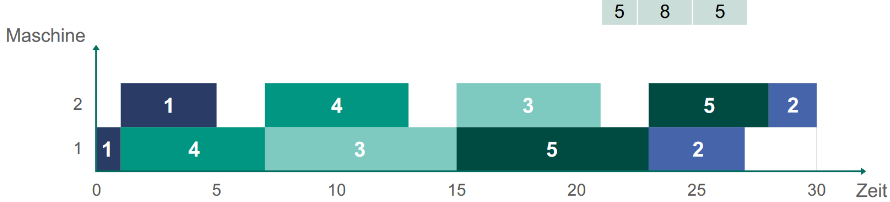
- **Problemstellung**: n Jobs durchlaufen selbe Maschinensequenz mit m Maschinen. Reihenfolge der Jobabarbeitung kann an jeder Maschine variieren
- Unterscheidung nach **Buffer-Typen**:
  - **Unbegrenzter Zwischenspeicher**: Keine Blockierung vorhergehender Maschinen möglich (*im Folgenden angenommen*)
  - **Begrenzter Zwischenspeicher**: Blockierung vorhergehender Maschinen möglich, d.h. wenn Produkt auf Maschine 1 fertig ist, dann kann es nicht direkt auf Maschine 2 geschoben werden und blockiert somit Maschine 1
- Schedule heißt **Permutationsschedule**, wenn Jobs in gleicher Reihenfolge auf allen Maschinen abgearbeitet werden

Minimierung des Makespan:

- Problem $Fm \mid\mid C_{max}$, wobei m die Anzahl der Maschinen ist
- **Satz**: Für $Fm \mid\mid C_{max}$ existiert für jede Probleminstanz ein optimaler Schedule, bei welchem die Jobsequenz für die ersten zwei Maschinen für die letzten zwei Maschinen gleich ist
- **Folgerung**: Für $F2 \mid\mid C_{max}$ und $F3 \mid\mid C_{max}$ existieren optimale Schedules die Permutationsschedules sind
- Für **Permutationsschedules** gilt:
  - $C_{i,j_1} = \sum_{l=1}^{i} p_{l,j_1}$ für $i = 1, \ldots, m$: Fertigstellungszeitpunkt von Job 1 auf Maschine i ist die Summe der Prozesszeiten von Job 1 auf allen vorherigen Maschinen
  - $C_{1,j_k} = \sum_{l=1}^{k} p_{1,j_l}$ für $k = 1, \ldots, n$: Fertigstellungszeitpunkt von Job k auf Maschine 1 ist die Summe der Prozesszeiten von allen vorherigen Jobs auf Maschine 1
  - $C_{i,j_k} = \max\{C_{i-1,j_k}, C_{i,j_{k-1}}\} + p_{i,j_k}$ für $i = 2, \ldots, m$ und $k = 2, \ldots, n$: Erst, wenn Job k auf vorheriger Maschine $i-1$ fertig ist und wenn Job $k-1$ auf Maschine i fertig ist, kann mit Job k auf Maschine i angefangen werden
- $F2 \mid\mid C_{max}$ lässt sich mit **Johnson's Algorithmus** optimal lösen:
  1. **Initialisierung**: Speichere Jobs mit $p_{1,j} \leq p_{2,j}$ in J_1 und Jobs mit $p_{1,j} > p_{2,j}$ in J_2
  2. **Job-Sortierung**: Sortiere Jobs in J_1 aufsteigend nach Prozesszeiten auf Maschine 1 und Jobs in J_2 absteigend nach Prozesszeiten auf Maschine 2
  3. **Terminierung**: Füge J_2 an J_1

- Beispiel
  - Initialisierung: $J_1 = [1,4]$, $J_2 = [2,3,5]$
  - Job-Sortierung: $J_1 = [1,4]$, $J_2 = [3,5,2]$
  - Terminierung: $S = \{1,4,3,5,2\}$, $C_{max} = 30$

j	$p_{1,j}$	$p_{2,j}$
1	1	4
2	4	2
3	8	6
4	6	6
5	8	5



Betrachtung der Komplexität:

- $F2 \mid\mid C_{max}$ lässt sich mit Johnson's Algorithmus in polynomialer Zeit lösen

- $Fm \mid\mid C_{max}$ für $m \geq 3$ ist NP-schwer $\rightarrow$ Lösung durch Heuristiken

Johnson's Algorithmus für 3 Maschinen:

- Für bestimmte Probleminstanzen von $F3 \mid\mid C_{max}$ findet eine modifizierte Form von Johnson's Algorithmus ebenfalls eine optimale Lösung
- **Voraussetzung**: $\max\limits_{i \in J}\{p_{2i}\} \leq \min\limits_{k \in J}\{p_{1k}\}$ oder $\max\limits_{i \in J}\{p_{2i}\} \leq \min\limits_{k \in J}\{p_{3k}\}$ mit $i \neq k$
- **Vorgehen**:
  1. Berechne $p^*_{1j} = p_{1j} + p_{2j}$ und $p^*_{2j} = p_{2j} + p_{3j}$ für alle $j \in J$
  2. Führe den normalen Johnson Algorithmus für p^*_{1j} und p^*_{2j} durch

Job-ID	1	2	3	4
p_{1j}	8	9	3	10
p_{2j}	6	3	4	2
p_{3j}	10	8	6	7

Job-ID	1	2	3	4
p^*_{1j}	14	12	7	12
p^*_{2j}	16	11	10	9

- Initialisierung: $J_1 = [1,3]$, $J_2 = [2,4]$
- Job-Sortierung: $J_1 = [3,1]$, $J_2 = [2,4]$
- Terminierung: $S = \{3,1,2,4\}$, $C_{max} = 42$
- Optimal? → Ja, denn $\max\limits_{j \in J}\{p_{2j}\} = 6 \leq \min\limits_{j \in J}\{p_{3j}\} = 6$

Job-ID	3	1	2	4
p_{1j}	3	8	9	10
p_{2j}	4	6	3	2
p_{3j}	6	10	8	7
C_{1j}	3	11	20	30
C_{2j}	7	17	23	32
C_{3j}	13	27	35	**42**

- Einträge C_{ij} wird nach den Rechenvorschriften für Permutationsschedules (*siehe vorherige Seite*) bestimmt

Heuristiken:

- Verfahren zur Bestimmung eines zulässigen Punktes eines Problems, dessen Wert möglichst nahe am Optimalwert liegen soll, mit akzeptablem Aufwand
- **Arten von Heuristiken**:
  - **Konstruktionsheuristiken**: Finden eines ersten zulässigen Punktes
  - **Verbesserungsheuristiken**: Ausgehend von einem zulässigen Punkt wird nach Verbesserungen gesucht
  - Heuristiken zur Bestimmung von Schranken

NEH-Heuristik:

1. Berechne für jeden Job j: $T_j = \sum\limits_{i=1}^{m} p_{ij}$
2. Sortiere Jobs in absteigender Reihenfolge ihrer T_j in einer Liste, wenn mehrere Jobs dieselben Werte für T_j haben, sortiere aufsteigend nach Job-IDs

3. Nehme die ersten beiden Jobs der Liste und finde die beste Sequenz aus diesen Jobs $\rightarrow$ Berechne Makespan für beide möglichen Sequenzen und wähle Sequenz mit niedrigerem Makespan. Setze $i \coloneqq 3$
4. Nehme Job an i-te Stelle aus der Liste von Schritt 2. Füge diesen an alle möglichen Positionen der bisher generierten Sequenz ein. Die generierte Sequenz, welche den minimalen Makespan aufweist, wird für den nächsten Schritt berücksichtigt.
5. Wenn $i = n \rightarrow$ Stop, sonst $i = i + 1$ und gehe zu Schritt 4

Beispiel siehe Übung, Folie 14-18

MILP für Flow-Shop Maschinenumgebung (allgemein):

- **Entscheidungsvariablen**:
  - x^i_{jk} : Binärvariable: Angabe, ob j vor k auf i produziert wird
  - C_{ij} : Fertigstellungszeitpunkt von j auf Maschine i
- **Zielfunktion**: $Z \rightarrow \min$ mit $Z = \max\limits_{j=1,\dots,n} \{C_{mj}\}$
- **Nebenbedingungen**:
  1. $C_{ij} \geq C_{(i-1)j} + p_{ij}, \forall i = 1, \dots, m; j = 1, \dots, n$ (Maschinenreihenfolgensicherstellung)
  2. $M \cdot x^i_{jk} + C_{ij} - C_{ik} \geq p_{ij}, \forall i = 1, \dots, m; j = 1, \dots, n-1; k = j+1, \dots, n$ (Jobreihenfolge, wenn k vor j)
  3. $M \cdot \left(1 - x^i_{jk}\right) + C_{ik} - C_{ij} \geq p_{ik}, \forall i = 1, \dots, m; j = 1, \dots, n-1; k = j+1, \dots, n$ (Jobreihenfolge, wenn j vor k)
  4. $Z \geq C_{mj}, \quad \forall j = 1, \dots, n$ (Definition Makespan Z)
  5. $C_{0j} = 0, \forall j = 1, \dots, n$ (NNB für virtuelle Maschine 0 $\rightarrow$ Job j kann erst ab Zeitpunkt 0 auf Maschine 1 produziert werden)
  6. $x^i_{jk} \in \{0; 1\} : x^i_{jk} = \begin{cases} 1, & \text{wenn Auftrag } j \text{ vor Auftrag } k \text{ auf Maschine } i \text{ produziert wird} \\ 0, & \text{sonst} \end{cases}$
- M ausreichend groß wählen $\rightarrow$ Sorgt für Erfüllung der Ungleichungen, wenn j vor k (2.) oder k vor j (3.) bearbeitet wird
- Bei anderer Zielfunktion $\rightarrow Z$ austauschen und 4. eventuell streichen

MILP für Permutation-Flow-Shop:

- **Entscheidungsvariablen**:
  - x_{jk} : Binärvariable: Angabe, ob j an Position k (auf allen Maschinen) produziert wird

- C_{ik} : Fertigstellungszeitpunkt des Jobs auf Position k auf Maschine i

- **Zielfunktion**: $Z \rightarrow \min$ mit $Z = \max\limits_{j=1,\ldots,n} \{C_{mj}\}$
- **Nebenbedingungen**:
  1. $\sum\limits_{j=1}^{n} x_{jk} = 1, \forall k = 1, \ldots, n$ (Jede Position hat einen Job)
  2. $\sum\limits_{k=1}^{n} x_{jk} = 1, \forall j = 1, \ldots, n$ (Jeder Job hat eine Position)
  3. $C_{(i-1)k} + \sum_{j=1}^{n} p_{ij} x_{jk} \leq C_{ik}, \forall i = 1, \ldots, m; k = 1, \ldots, n$ (Maschinenreihenfolgensicherstellung)
  4. $C_{i(k-1)} + \sum_{j=1}^{n} p_{ij} x_{jk} \leq C_{ik}, \forall i = 1, \ldots, m; k = 1, \ldots, n$ (Jobreihenfolgensicherstellung)
  5. $Z \geq C_{mj}, \quad \forall j = 1, \ldots, n$ (Definition Makespan Z)
  6. $C_{01} = 0$ (NNB für die virtuelle Maschine 0 $\rightarrow$ Job auf Position 1 kann erst ab Zeitpunkt 0 auf erster Maschine produziert werden)
  7. $x_{jk} \in \{0; 1\} : x_{jk} = \begin{cases} 1, & \text{wenn Auftrag } j \text{ an Position } k \text{ produziert wird} \\ 0, & \text{sonst} \end{cases}$
- Bei anderer Zielfunktion $\rightarrow$ Z austauschen und 5. eventuell streichen

Flow-Shop-Optimierung durch Solver:

- Obige MILPS können durch Solver (z.B. CPLEX) gelöst werden
- Große einfache Probleme $\rightarrow$ Nutze Heuristiken, da diese schneller
- Schwere Probleme $\rightarrow$ Nutze Solver