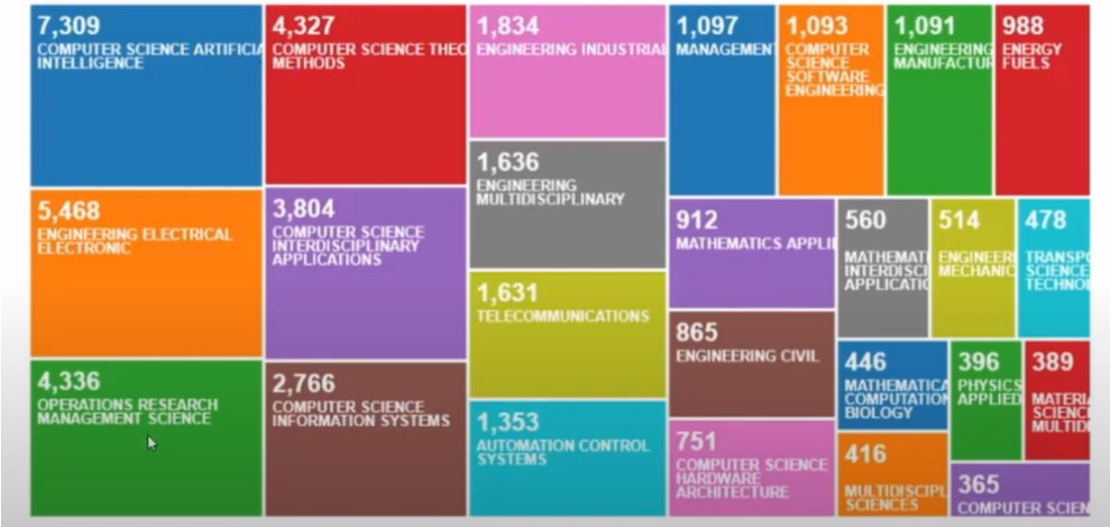
Metaheuristiken

und in sehr vielen Bereichen angewendet.[11]



Metaheuristiken

Keine einheitliche Definition, was eine Metaheuristik ist

> *A metaheuristic is formally defined as an **iterative generation** process which **guides a subordinate heuristic** by combining intelligently **different concepts for exploring and exploiting the search space**, learning strategies are used to structure information in order to **find efficiently near-optimal solutions**.*[13]

> *A metaheuristic is an **iterative master process** that **guides and modifies the operations of subordinate heuristics** to **efficiently produce high-quality solutions**. It may manipulate a complete (or incomplete) **single solution or a collection of solutions** at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method.*[14]

Eigenschaften von Metaheuristiken

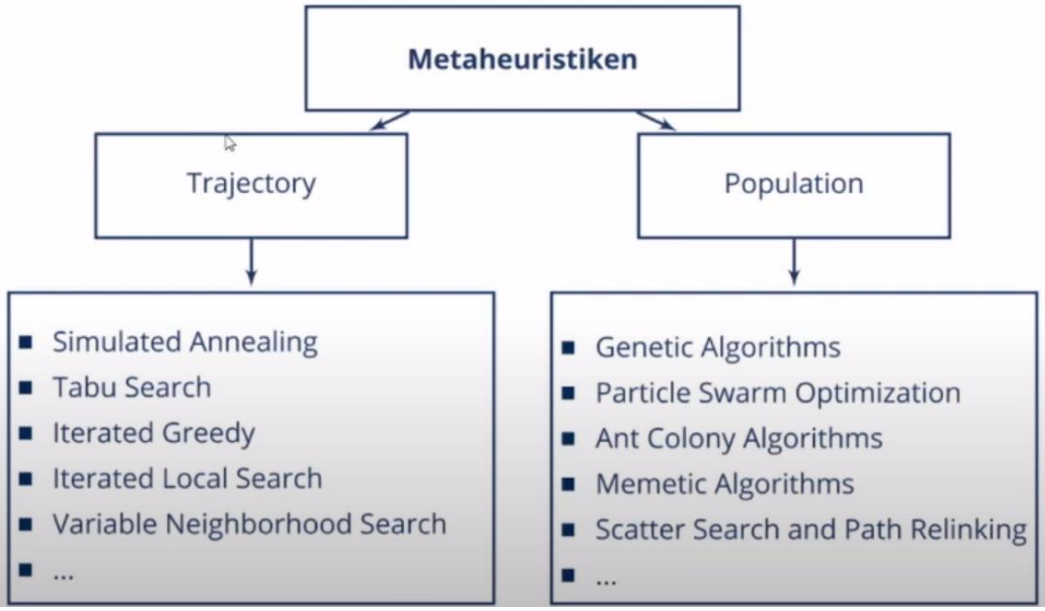
Ziel

Lösungsraum effizient nach sehr guten Lösungen absuchen

Metaheuristiken[15]

- sind eine übergeordnete, iterative Strategie, die den Suchprozess steuert.
- sind grundsätzlich problemunabhängig und daher abstrakt beschreibbar.
- nutzen problemspezifisches Wissen in Form von gesteuerten Heuristiken.
- sind approximativ und oftmals nicht-deterministisch.
- beinhalten Mechanismen, um lokalen Optima zu entkommen und den Lösungsraum großflächig zu untersuchen.

Überblick



→我们只用左边的

Grundprinzipien

Metaheuristiken bestehen aus verschiedenen Komponenten, die zwei grundsätzliche Wirkprinzipien umsetzen[16]

Intensivierung

- Intensive Suche nach sehr guten Lösungen in vielversprechenden Bereichen des Lösungsraums
- Hauptsächlich durch Veränderung der Zielfunktion getrieben

Diversifizierung

- Suche nach noch nicht erforschten Bereichen des Lösungsraum
- Einsatz von Zufall und anderen Steuerungsmechanismen als der Zielfunktion

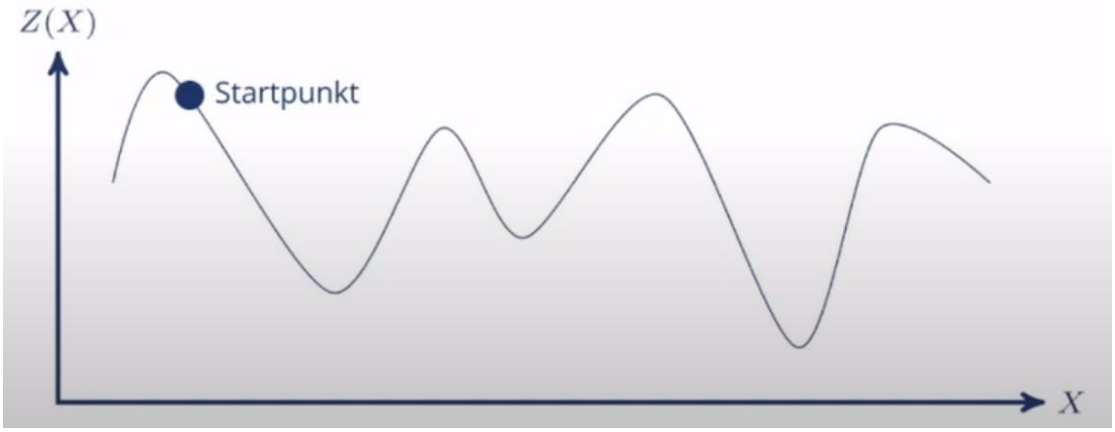
Gute Metaheuristiken steuern die Interaktion der Prinzipien **ausgewogen und dynamisch**, um schnell vielversprechende Bereiche zu finden und nicht zu viel Zeit in schlechten Bereichen zu verschwenden. Die Komponenten von Metaheuristiken besitzen dabei häufig Charakteristika beider Prinzipien, aber mit unterschiedlichen Ausprägungen.

尽可能尽快得出解决方案

Intensivierung und Diversifizierung bei ILS

Iterative Abfolge von

- einem Verfahren der **Lokalen Suche** (LS) ⇒ hauptsächlich Intensivierung
- **Perturbation** der Lösung (P) ⇒ hauptsächlich Diversifizierung

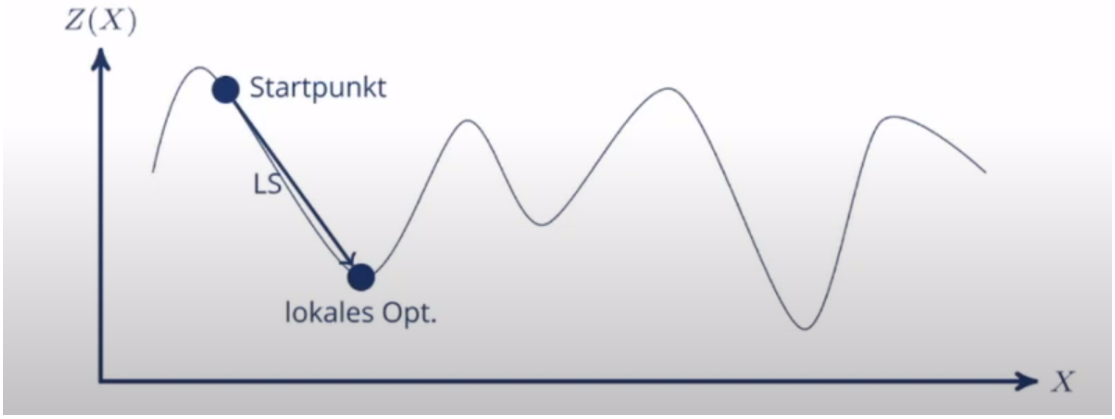


→ iterited local search，也是 Meta

Intensivierung und Diversifizierung bei ILS

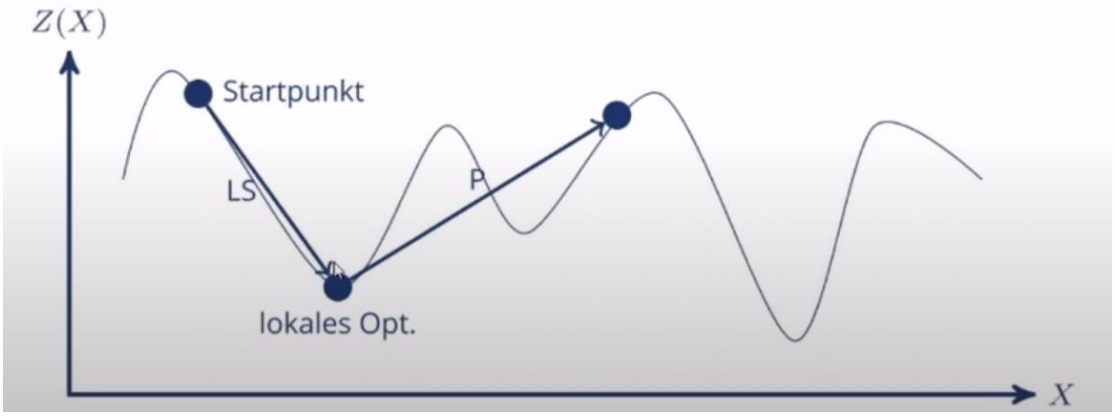
Iterative Abfolge von

- einem Verfahren der **Lokalen Suche** (LS) ⇒ hauptsächlich Intensivierung
- **Perturbation** der Lösung (P) ⇒ hauptsächlich Diversifizierung

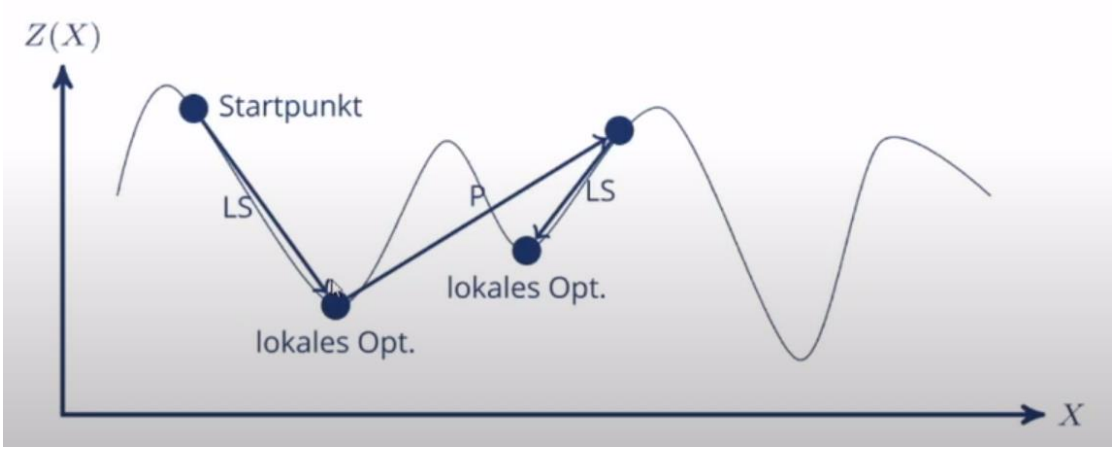


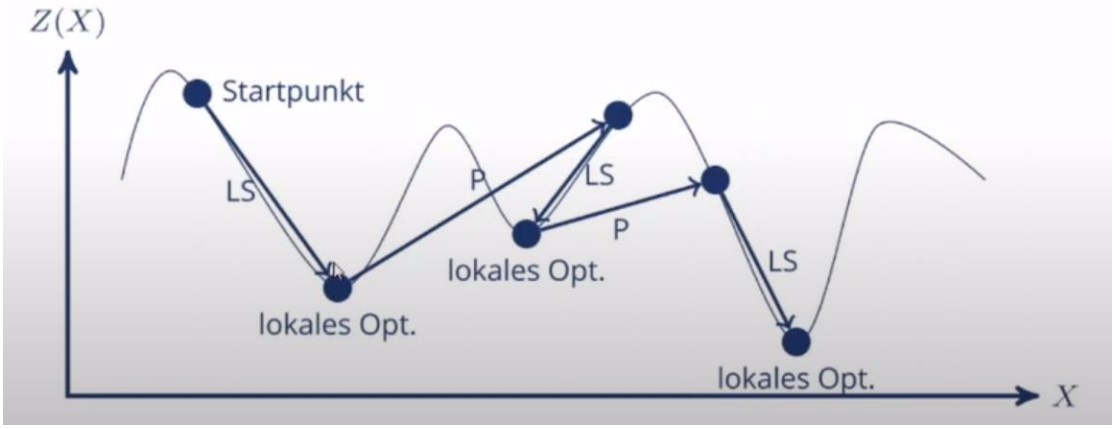
Startloesung: mit einer konstruktiver Verfahren → local serach → local opt
纵轴是目标函数

- **Perturbation** der Lösung (P) ⇒ hauptsächlich Diversifizierung



Pertubation 摄动，扰动 → springen in einen anderen Bereich des Loesungsraums → neue Lsg erzeugen → neue lsg als Startpunkt von LS





Komponenten von Trajectory Methoden (Auswahl)

Simulated Annealing:

- Akzeptanzkriterium
- Annealing Schedule

Tabu Search:

- Tabu-Liste
- Aspirationskriterium

Variable Neighborhood Search:

- Lokaler Suchalgorithmus
- Nachbarschaftsauswahl
- Shaking
- Akzeptanzkriterium

Iterated Greedy

Viele sehr gute (Lösungsqualität und Geschwindigkeit) Metaheuristiken sind

- sehr komplex,
- abhängig von vielen Parametern,
- extrem auf Problem anpasst und damit
- aufwendig zu implementieren.

Im Gegensatz dazu besteht Iterated Greedy aus zwei einfachen, **iterativ** ausgeführten Phasen

Destruktion	Konstruktion
Teilweise Zerstörung der aktuellen, vollständigen Lösung	Wiederherstellung einer vollständigen Lösung mittels konstruktiver Heuristik

Trotz der Einfachheit liefert Iterated Greedy sehr gute Ergebnisse für PFSP!

Iterated Greedy für PFSP

Vorgehensweise:[17]

1. Erzeuge Startlösung π mit NEH-Heuristik
2. Lokale Suche mit π *(optional)*
3. Bis Stoppkriterium erreicht, führe durch
   a) **Destruktion:** Erzeuge unvollständige Lösung π' durch Entfernen von d zufälligen Aufträgen aus aktueller Lösung π und füge entfernte Aufträge zu π_R hinzu
   b) **Konstruktion:** Erzeuge mit NEH-Heuristik neue vollständige Lösung π'' durch Hinzufügen der Aufträge aus π_R zu π'
   c) Lokale Suche mit π'' *(optional)*
   d) Wenn Akzeptanzkriterium erfüllt, setze Lösung π'' als aktuelle Lösung π
   e) Speicher π'' als beste Lösung, wenn π'' besser als bisher beste gefundene Lösung π_B
4. Gib beste gefunden Lösung π_B zurück

Permutation Flow Shop Problem

Komponenten des Iterated Greedy

Destruktion und Konstruktion

Optionale Lokale Suche

- Hauptsächlich Intensivierung
- Verschiedene Lokale Suchverfahren möglich
- Ruiz, Stützle (2007): Insertion Nachbarschaft mit First Improvement

Akzeptanzkriterium

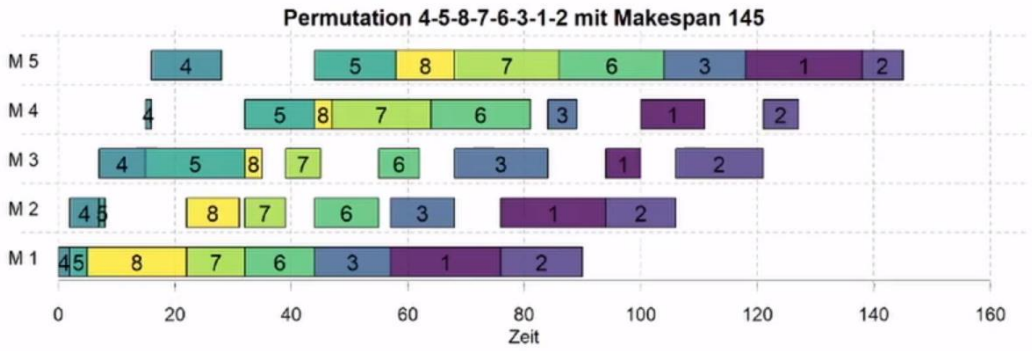
- Diversifizierung und Intensivierung
- Bessere Lösungen werden immer akzeptiert
- Schlechtere Lösungen werden mit gewisser Wahrscheinlichkeit akzeptiert
- Ruiz, Stützle (2007): Vereinfachung des Kriteriums von Simulated Annealing

$$e^{-\frac{C_{\max}(\pi'')-C_{\max}(\pi)}{Temperature}} \text{ mit } Temperature = T \cdot \frac{\sum Bearbeitungszeiten}{n \cdot m \cdot 10}$$

Gedächtnis für beste Lösungen

Iterated Greedy - Beispiel I

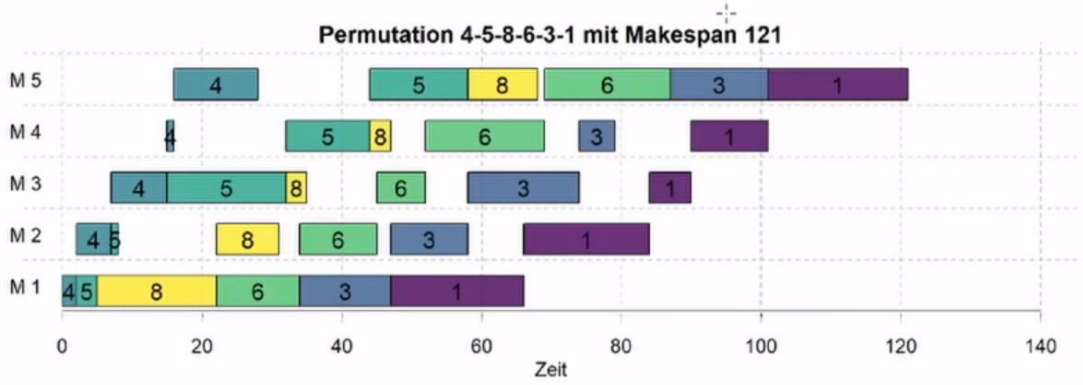
1. Startlösung: Erzeuge $\pi = [4, 5, 8, 7, 6, 3, 1, 2]$ mit NEH-Heuristik



Iterated Greedy - Beispiel II

3.a) Destruktion: Erzeuge unvollständige Lösung π' durch Entfernen von d zufälligen Aufträgen aus aktueller Lösung π und füge entfernte Aufträge zu π_R hinzu

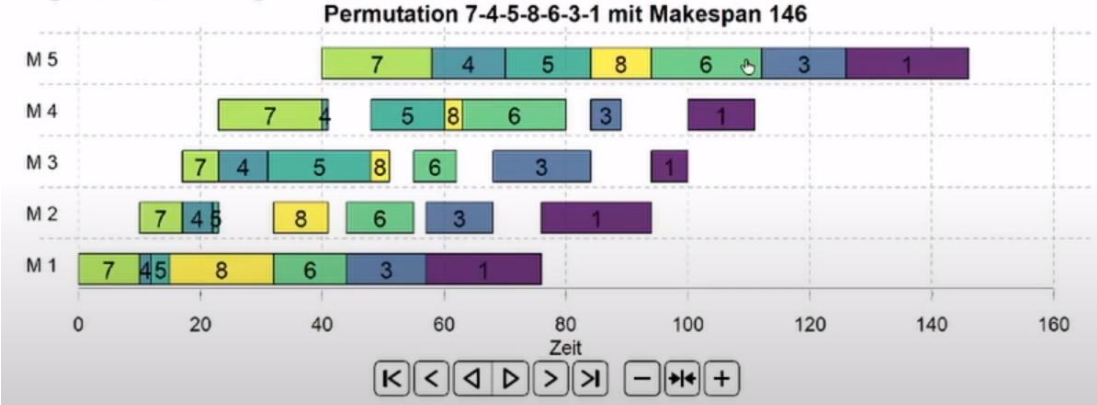
Mit $d = 2$ werden zufällig Aufträge 7 und 2 aus π entfernt
$\Rightarrow \pi_R = [7, 2]\,,\ \pi' = [4, 5, 8, 6, 3, 1]$



Iterated Greedy - Beispiel III

3.b) Konstruktion: Erzeuge mit NEH-Heuristik neue vollständige Lösung π'' durch Hinzufügen der Aufträge aus π_R zu π'

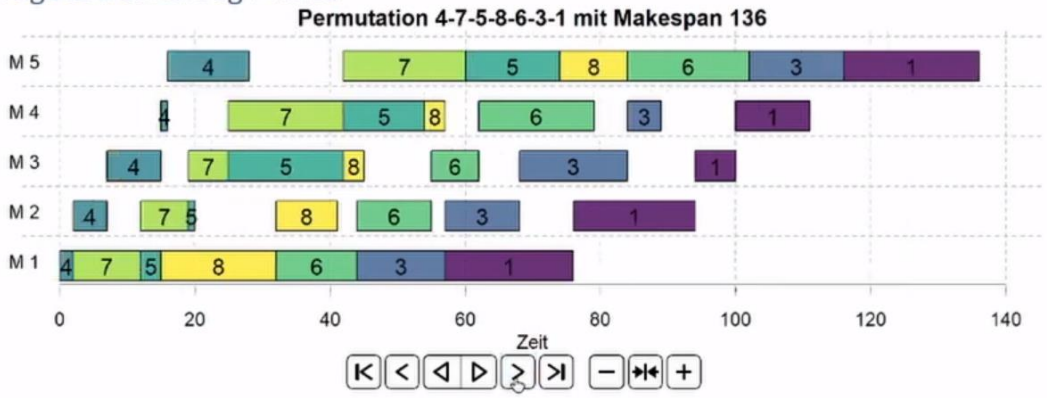
Füge zuerst Auftrag 7 hinzu



Iterated Greedy - Beispiel III

3.b) Konstruktion: Erzeuge mit NEH-Heuristik neue vollständige Lösung π'' durch Hinzufügen der Aufträge aus π_R zu π'

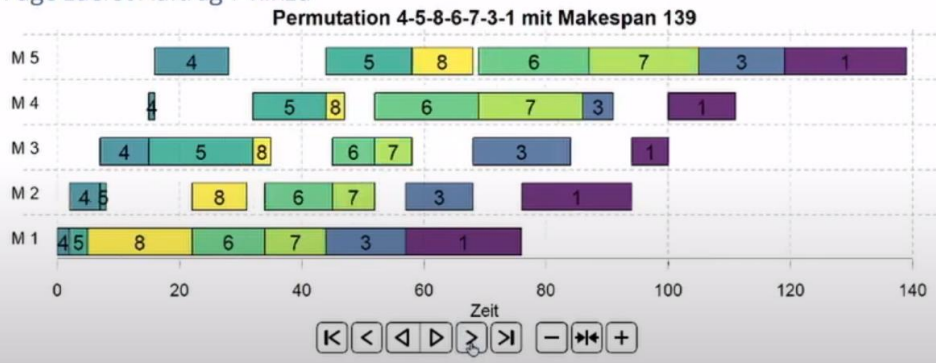
Füge zuerst Auftrag 7 hinzu



Iterated Greedy - Beispiel III

3.b) Konstruktion: Erzeuge mit NEH-Heuristik neue vollständige Lösung π'' durch Hinzufügen der Aufträge aus π_R zu π'

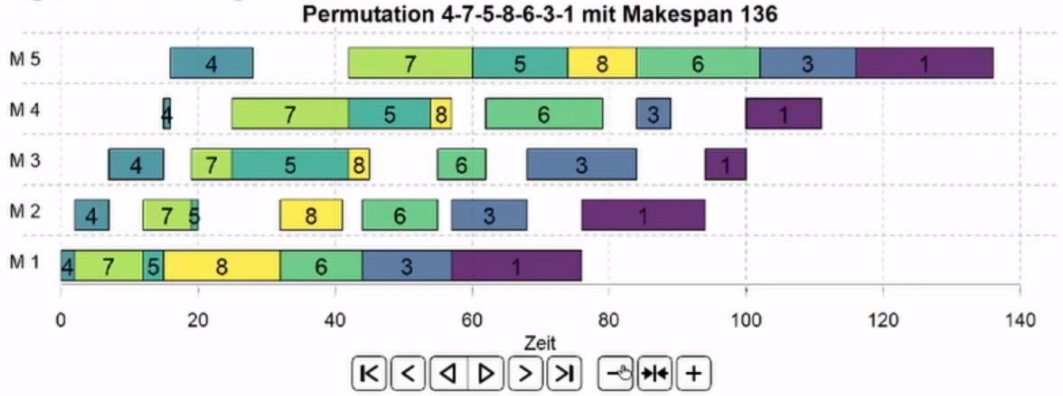
Füge zuerst Auftrag 7 hinzu



Iterated Greedy - Beispiel III

3.b) Konstruktion: Erzeuge mit NEH-Heuristik neue vollständige Lösung π'' durch Hinzufügen der Aufträge aus π_R zu π'

Füge zuerst Auftrag 7 hinzu

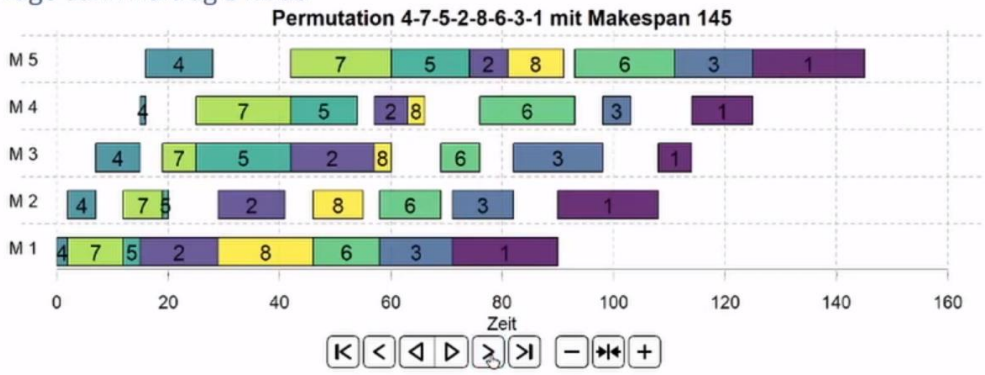


$\Rightarrow \pi' = [4, 7, 5, 8, 6, 3, 1]$

7 的位置确定了

3.b) Konstruktion: Erzeuge mit NEH-Heuristik neue vollständige Lösung π'' durch Hinzufügen der Aufträge aus π_R zu π'

Füge dann Auftrag 2 hinzu



现在来尝试 2

Iterated Greedy - Beispiel IV

3.d) Akzeptanz: Wenn Akzeptanzkriterium erfüllt, setze Lösung π'' als aktuelle Lösung π

Makespan von π'' (143) < Makespan von π (145) $\Rightarrow \pi = \pi''$

3.e) Gedächtnis: Speicher π'' als beste Lösung, wenn π'' besser als bisher beste gefundene Lösung π_B

Makespan von π'' (143) < Makespan von π_B (145) $\Rightarrow \pi_B = \pi''$

Exkurs: DEAP

Distributed Evolutionary Algorithms in Python (DEAP)[18]

- ist eine Framework für (u.a.) populationsbasierte Metaheuristiken
  - Genetic Algorithms
  - Particle Swarm Optimization
- unterstützt auch multikriterielle Optimierung
- ist besonders geeignet für schnelles Prototyping

Installation

- Anaconda Prompt öffnen[19]
- `conda install -c conda-forge deap` ausführen[20]

Mehr in Live Session