



Prescriptive Analytics

Belegarbeiten

I. Multiple Knapsack Problem

- Erweitertes Knapsack Problem mit Beachtung von nicht genutzten Ressourcen
- Zielfunktion: Gewinn (Maximierung)

II. Quadratic Assignment Problem

- Standortplanung von Anlagen
- Zielfunktion: Gesamttransportleistung (Minimierung)

III. Capacitated Profitable Tour Problem

- Profitable Tour Problem mit begrenzter Kapazität
- Zielfunktion: Profit der besuchten Knoten (Maximierung)

IV. Traveling Salesman Problem with precedence constraints

- TSP mit festgelegter Reihenfolgebeziehung
- Zielfunktion: Gefahrene Distanz zum Besuchen aller Knoten (Minimierung)

Zielstellung und Bearbeitungsmodus

Zielstellung

- Die KursteilnehmerInnen erhalten je eines der dargestellten Planungsprobleme.
- Ziel ist die Lösung des Planungsproblems mit der beschriebenen Heuristik innerhalb einer definierten Rechenzeit. Diese kann innerhalb der Arbeitsgruppen abgestimmt werden.
- Die Bearbeitungsdauer beträgt sechs Wochen.
Ausgabetermin: 23.05.2022
Abgabetermin: 04.07.2022
- Am Ende erfolgt eine Kurzpräsentation (10 Minuten) der Ergebnisse. (Präsentationstermine: 11./12.07.2022)

Bearbeitung

- Die Bearbeitung erfolgt in Visual Studio Code unter Verwendung der Programmiersprache Python.
- Zur Bearbeitung werden zu jedem Problem Datensets bereitgestellt. Nach Abgabe erfolgt einer Überprüfung der Lösungsqualität mit einem Validation Data Set.
- Zur Lösung der Aufgabenstellung sind Klassenstruktur, Lösungsalgorithmus und die notwendige Solverarchitektur zu entwickeln.
- Neben der Programmierung ist der Lösungsweg in einem Jupyter Notebook nachvollziehbar zu dokumentieren.

Dokumentation

- Während die eigentliche Entwicklungsarbeit in normalen Python-Programmen/ Dateien stattfindet, sollte die Dokumentation über ein Jupyter Notebook erfolgen.
- Im Programm-Code sollten kurze Hinweise und Kommentare stehen.
- Zur Veranschaulichung können im Jupyter Notebook Code-Zellen verwendet werden, welche auf entwickelten Programmcode zurückgehen/verweisen.
- Die Dokumentation sollte die folgenden Punkte enthalten:
 - Beschreibung des Planungsproblems & Inputdaten
 - Klassenstruktur und Erläuterung zur Architektur des Programms
 - Beschreibungen zur Codierung und Bewertung von Lösungen
 - Erzeugung von Startlösungen und Vorgehensweise des Näherungsverfahrens
 - Hinweise und Erläuterungen zu verwendeten Parametereinstellungen
 - Auswertung der Ergebnisse & Einschätzung der Lösungsgüte
- Referenzieren Sie von Ihnen verwendete Literatur in Ihrem Jupyter Notebook

Worauf achten wir

- Ihr Solver beinhaltet alle notwendigen Elemente des geforderten Lösungsverfahrens: problemspezifische konstruktive Bausteine, sowie verfahrensspezifische Intensivierungs- und Diversifikationsmechanismen.
- Ihr Solver läuft fehlerfrei durch und erzeugt zulässige Ergebnisse für das Planungsproblem.
- Sie haben sinnvolle Klassenstrukturen und Solverarchitektur in Anlehnung an das Seminar erstellt.
- Ergebnisse sollten durch die Verwendung eines Zufallszahlengenerators reproduzierbar sein.

Beschreibung

- Iterative Lokale Suche mit Akzeptanzkriterium
 - Intensivierung:
Local Search
 - Diversifikation:
Akzeptanzkriterium und Perturbation: Stören einer Lösung, z.B. durch einen zufälligen (verschlechternden) Tausch
 - Unterschied zu Iterated Greedy: kein *Zerstören* der Lösung und darauffolgendes Zusammensetzen mit konstruktiven (problemspezifischen) Regeln
- Mögliche Variationen:
 - Mehrere Nachbarschaften für Lokale Suche → VND
 - Verschiedene Perturbationsmechanismen: mehrere oder problemspezifische Perturbationen
 - Periodischer Neustart von bester gefundener Lösung
 - ...

Ablauf

Grundlegender Ablauf von Iterated Local Search

```
1: Input: start solution  $s_0$ 
2: Initialize:  $s = s_0$ 
3: while abort criterion is not reached do
4:    $s' = \text{Perturb}(s)$  // not in first iteration
5:    $s'' = \text{LocalSearch}(s')$ 
6:   if acceptance criterion is satisfied then
7:      $s = s''$ 
8:   end if
9: end while
```

Vgl. Gendreau, M., & Potvin, J. Y. (Eds.). (2019). *Handbook of metaheuristics* (Vol. 3, p. 129). New York: Springer

Variable Neighborhood Search

Beschreibung

- Systematischer Wechsel zwischen verschiedenen Nachbarschaften
 - Intensivierung:
Local Search mit ausgewählter Nachbarschaft oder auch VND
 - Diversifikation:
Shaking: Zufälliger, auch verschlechternder Tausch in der aktuellen Nachbarschaft
- Mögliche Variationen:
 - Dynamische Auswahl der Nachbarschaft, z.B. Anhand von gesammelten Statistiken während des Lösungsverlaufs
 - Verschiedene Perturbationsmechanismen: mehrere oder problemspezifische Perturbationen
 - Periodischer Neustart von bester gefundener Lösung
 - ...

Ablauf

Grundlegender Ablauf von Variable Neighborhood Search

- 1: **Input:** default neighborhood k_0 , start solution s_0
- 2: **Initialize:** $k = k_0, s = s_0$
- 3: **while** abort criterion is not reached **do**
- 4: $s' = \text{Shaking}(s, k)$ // not in first iteration
- 5: $s = \text{LocalSearch}(s', k)$
- 6: $k = \text{NeighborhoodChange}(k)$
- 7: **end while**

Vgl. Gendreau, M., & Potvin, J. Y. (Eds.). (2019). *Handbook of metaheuristics* (Vol. 3, p. 57). New York: Springer

Simulated Annealing

Beschreibung

- Analogie zu physikalischen Prinzipien beim Ausglühen von kristallinen Substanzen:
 - Nach dem Erhitzen folgt die Abkühlung des Werkstoffs und die Ausbildung von Gitterstrukturen
 - Robustheit bzw. Stabilität hängt von der Ausgangstemperatur und der Abkühlungsrate ab
- Die Metaheuristik simuliert diesen Abkühlungsprozess, wobei eine Lösung des Planungsproblems den Zustand des Werkstoffes repräsentiert
- Der Zielfunktionswert stellt dabei den energetischen Zustand des Werkstoffes dar
- Cooling Schedule (Initialtemperatur, Gleichgewichtszustand, Abkühlungsfunktion, Stoppkriterium)

Ablauf

Grundlegender Ablauf des Simulated Annealing

```
1: Input: Cooling Schedule, start solution  $s_0$ 
2: Initialize:  $s = s_0, i = 0, T_0 = T_{max}$ 
3: while  $T_i > T_{min}$  do
4:   while Equilibrium State = False do
5:     Generate random neighbor  $s'$ 
6:      $\Delta E = ZF(s') - ZF(s)$ 
7:     if  $\Delta E \leq 0$  then
8:        $s = s'$ 
9:     else
10:      Set  $s = s'$  with probability  $P(\Delta E, T_i) = e^{\frac{-\Delta E}{T_i}}$ 
11:    end if
12:  end while
13:   $i = i + 1, T_i = g(T)$ 
14: end while
```

Vgl. Gendreau, M., & Potvin, J. Y. (Eds.). (2019). *Handbook of metaheuristics* (Vol. 3, p. 1). New York: Springer

Beschreibung

- Zulassen von verschlechternden Tauschen um lokalen Optima zu entkommen
- Verbieten von zuletzt besuchten Lösungen („Cycling“)
 - Kurzzeitgedächtnis Tabu-Liste
 - Länge Tabu-Liste steuert Diversifizierung und Intensivierung
- Aspirationskriterium um Tabu-Liste zu umgehen
- Weitere Gedächtnisformen
 - Mittelfristig: Speicherung schlechter/ guter Strukturen
 - Langzeit: Neustart von sehr guten Lösungen

Ablauf

Grundlegender Ablauf des Tabu Search

- 1: **Input:** start solution s_0
 - 2: **Initialize:** Tabuliste $T^L = \emptyset$, $s = s_0$
 - 3: **while** abort criterion is not reached **do**
 - 4: Suche beste Lösung s' in Nachbarschaft von s , die
 - (i) *nicht tabu* ist oder
 - (ii) ein *Aspirationskriterium* erfüllt
 - 5: $s = s'$
 - 5: Update T^L
 - 6: **end while**
-

Vgl. Gendreau, M., & Potvin, J. Y. (Eds.). (2019). *Handbook of metaheuristics* (Vol. 3, p. 37). New York: Springer

Feedback

Wo sehen Sie Verbesserungspotential?

Was hat Ihnen gefallen?

Würden Sie die Veranstaltung weiterempfehlen?



Themenzuweisung

| Nr. | Thema | Heuristik | Zuweisung |
|-----|--|-----------|-----------|
| 1 | QAP Minimale Transportleistung Dr. Janis Neufeld | VNS | 4694274 |
| 2 | | TS | 5012185 |
| 3 | | SA | 4873373 |
| 4 | | ILS | 4904196 |
| 5 | MKP Maximaler Gewinn Benedikt Zipfel | VNS | 4795742 |
| 6 | | TS | 4685025 |
| 7 | | SA | 4805234 |
| 8 | | ILS | 4802768 |
| 9 | CPTP Maximaler Profit Florian Linß | VNS | 4629203 |
| 10 | | TS | 4874889 |
| 11 | | SA | 4694185 |
| 12 | | ILS | 4997523 |
| 13 | TSP-PC Minimale Distanz Lisa Wesselink | VNS | 4712326 |
| 14 | | TS | 4886335 |
| 15 | | SA | 4803099 |
| 16 | | ILS | 4623033 |