Universität Basel

Bachelorarbeit Präsentation

Optimization of Student Time Slot Allocation using Hungarian Method and Mixed-Integer Linear Programming

Artificial Intelligence Forschungsgruppe

Hamza Zarah - h.zarah@stud.unibas.ch

09. September 2024

Einleitung

- **Ziel** dieser Arbeit ist es, Methoden zur Optimierung der Zuordnung von Studenten zu Zeitfenstern zu untersuchen und ihre Effizienz und Effektivität zu vergleichen.
- Ein Ungarische Methode und zwei MILP-Ansätze wurden implementiert und evaluiert.

Problemstellung und Methoden

- Die Zuordnung von Studenten zu Zeitfenstern muss verschiedene Präferenzen und Gruppenwünsche berücksichtigen.
- **Herausforderung:** Optimierung unter Berücksichtigung von Präferenzen und Ressourcenbeschränkungen.

Gegeben:

Studenten $S = \{s_1, s_2, \dots, s_n\}$

Zeitslots $T = \{t_1, t_2, \dots, t_m\}$

Sprachen $L = \{l_1, \dots, l_k\}$

Zeitslot-Präferenzen $p_t : S \times T \to \{0,1,2\}$

Sprach-Präferenzen $p_l : S \times L \to \{0,1,2\}$

Gruppen-Präferenzen $p_g : S \times S \to \{0,1\}$

Einführung in Zuordnungsprobleme

1. **Definition:**

   Ein Zuordnungsproblem bezieht sich auf die Zuordnung von Ressourcen (Studenten) zu Aufgaben (Zeitslots).

2. **Ziel:**

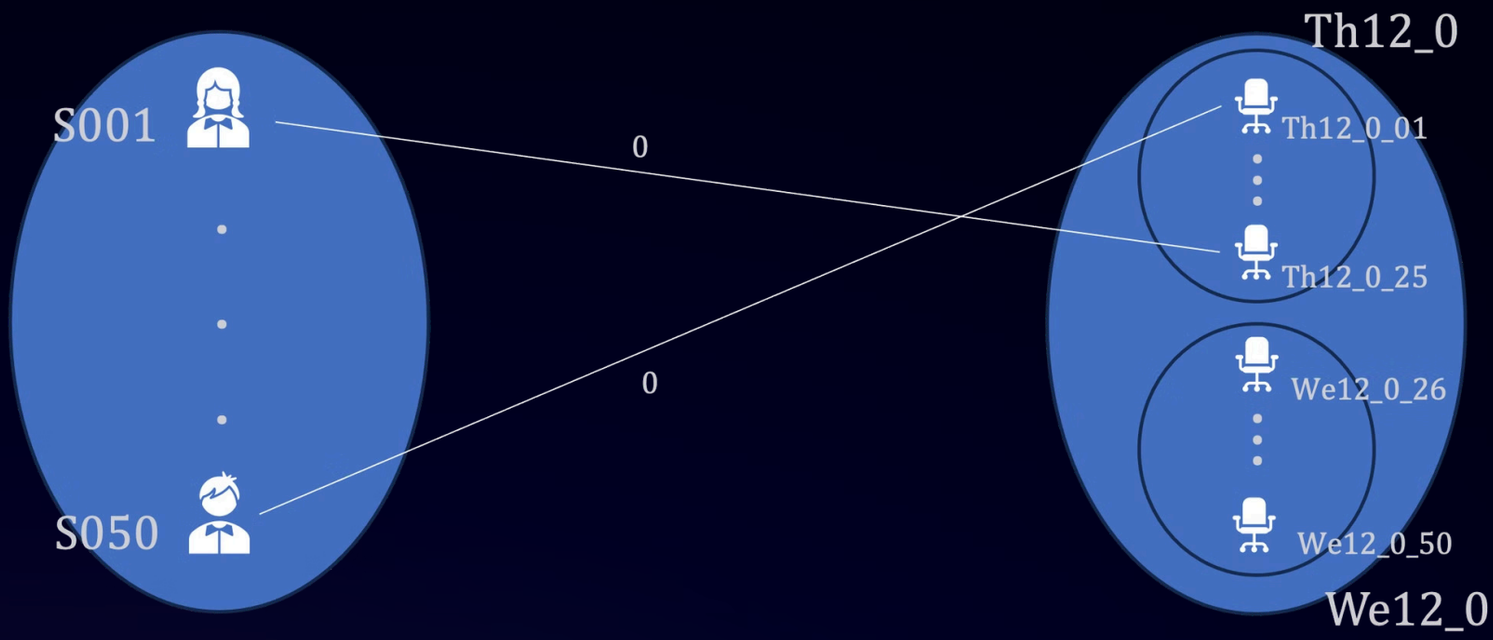
   Maximierung der Zufriedenheit der Studenten durch Berücksichtigung ihrer Präferenzen.

3. **Herausforderung:**

   Komplexität durch Präferenzen und begrenzte Ressourcen.

Einführung in die Ungarische Methode

- **Bipartiter Graph:**
  - Knoten: Studenten und Zeitslots.
  - Kanten: Verbindung basierend auf Präferenzen.
  - Gewichtung: Präferenzkosten auf den Kanten.
- **Perfect Matching:**
  - Erfordert gleiche Anzahl an Knoten in beiden Mengen (Studenten und Zeitslots).
    - **Lösung:** Einführung von "Subslots"
  - Minimierung der Gesamtkosten durch Perfect Matching im bipartiten Graphen.



Einführung in Gemischt-ganzzahliges lineares Programmieren (MILP)

- **Variablen:** Diskrete und kontinuierliche Variablen für Entscheidungen.
- **Nebenbedingungen:** Bedingungen, die die Zuordnungen einschränken (z.B. jeder Student nur ein Zeitslot).
- **Zielfunktion:** Gesamtkosten minimieren, alle Nebenbedingungen berücksichtigen.

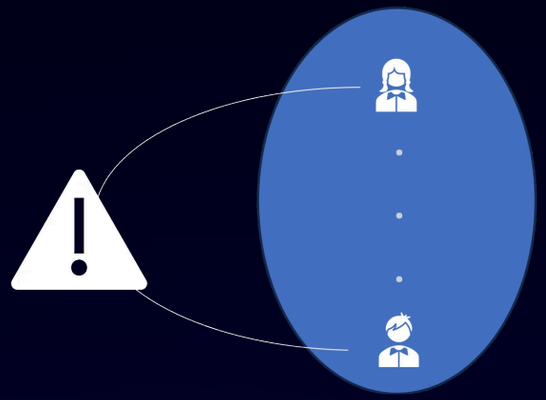
Beispiel Nebenbedingung: Ein Student muss genau einem Zeitslot zugeordnet werden.

$$\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall i \in \{1, \ldots, n\}$$

Vergleich der Methoden

Ungarische Methode:

- Klassische Methode zur Lösung von Zuweisungsproblemen.
- Polynomielle Laufzeit.
- Optimiert Zuordnungen ohne Berücksichtigung von Gruppenpräferenzen.



Gemischt-ganzzahliges lineares Programmieren (MILP):

- Flexibler Ansatz, der komplexere Neben-bedingungen wie Gruppenpräferenzen berücksichtigt.
- "SmartAlloc" Ohne Gruppenpräferenz: Nutzt MILP zur Optimierung.
- "SmartAlloc" Mit Gruppenpräferenz: Berücksichtigt zusätzliche Gruppenwünsche, komplexer.

Umgang mit Sprachkombinationen

- **Ungarische Methode:** Erzeugt eine Kostenmatrix für jede Kombination.
- "**SmartAlloc**" **(MILP):** Formuliert die Kombinationen als Variablen und Nebenbedingungen.
- Beide Ansätze gehen über alle Kombinationen, um optimale Lösungen zu finden.

Evaluationsmethodik

Evaluierung anhand von 510 Probleminstanzen, generiert auf Basis von zwei Parametern: Anzahl der Studenten (n) und Präferenztyp (p).

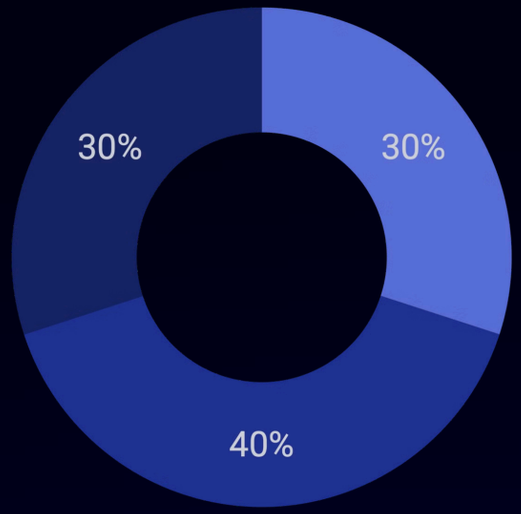
Problemgrößen: 50 bis 2000 Studenten.

Experimentelle Umgebung:

- sciCORE Cluster
- Zeitlimit: 30 Minuten
- Speicherlimit: 4 GB.

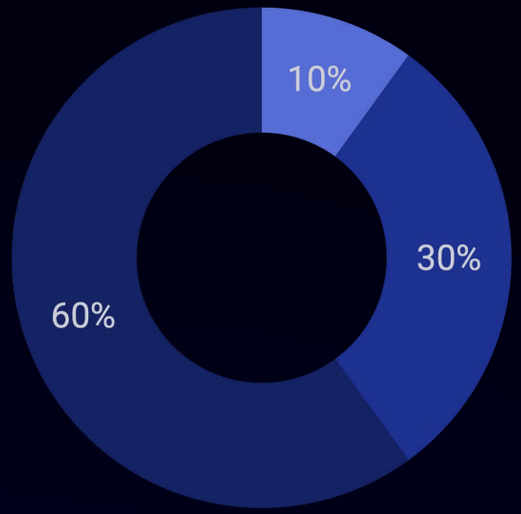
Scatterplot: Diagramm zur Veranschaulichung der Laufzeitunterschiede.

Präferenztypen der Problemstellung



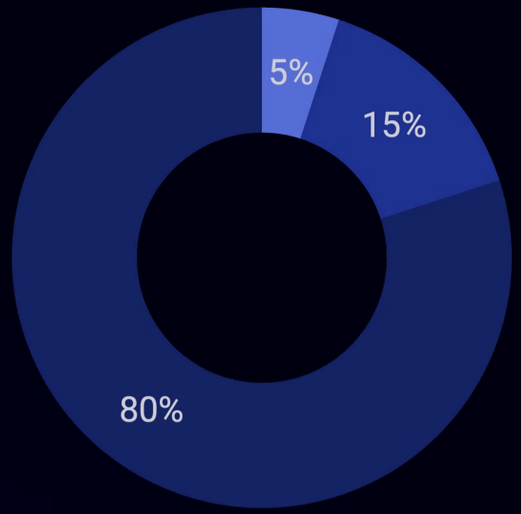
p0

Gleichmäßige Präferenzverteilung.



p1

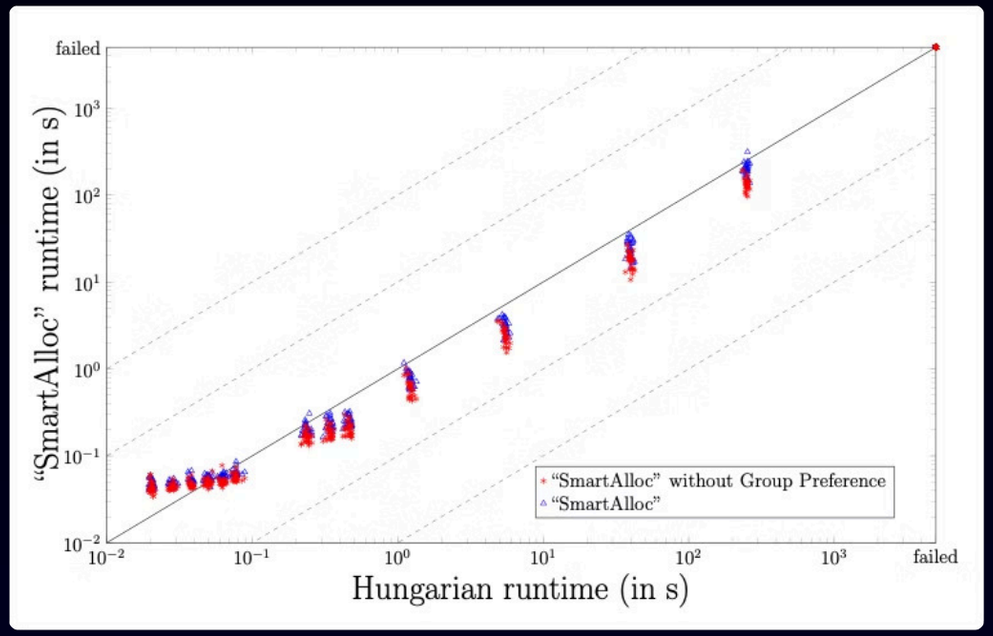
Ein bevorzugter und ein weniger bevorzugter Zeitslot.



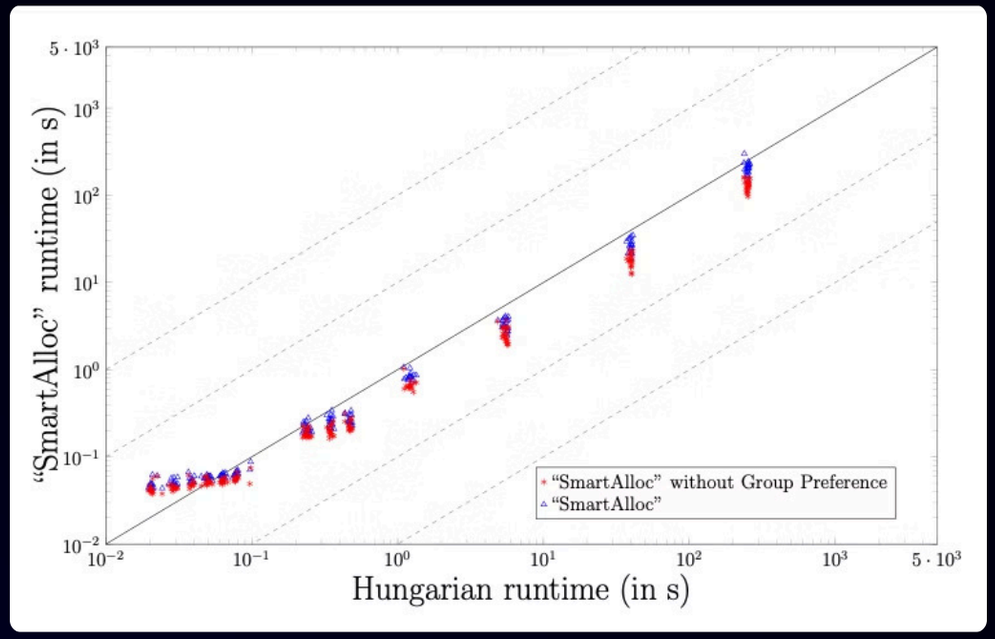
p2

Ein stark bevorzugter und ein stark abgelehnter Zeitslot.

Laufzeitanalyse der Methoden



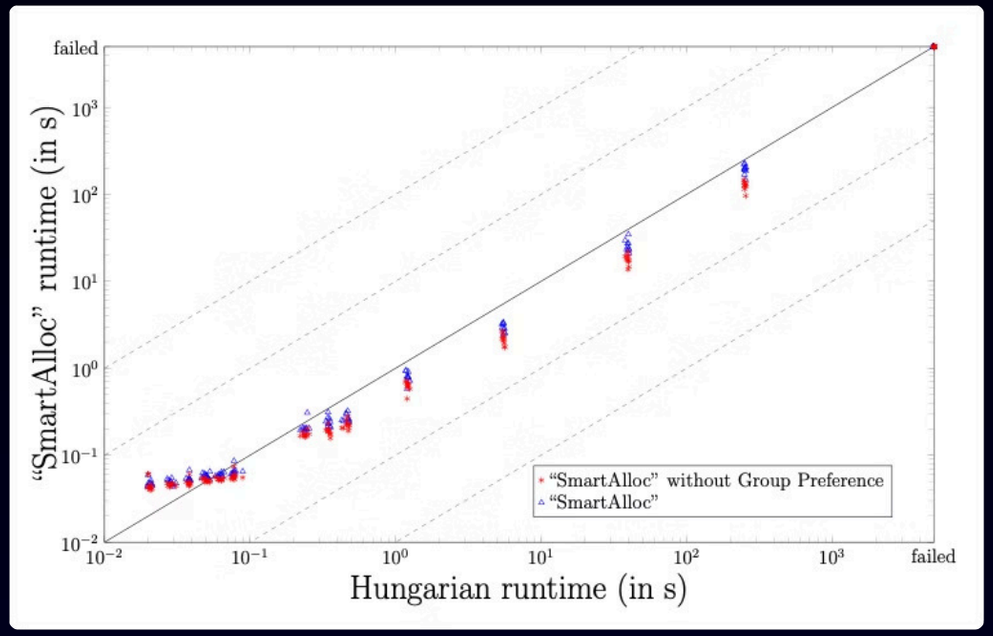
- Alle Probleme



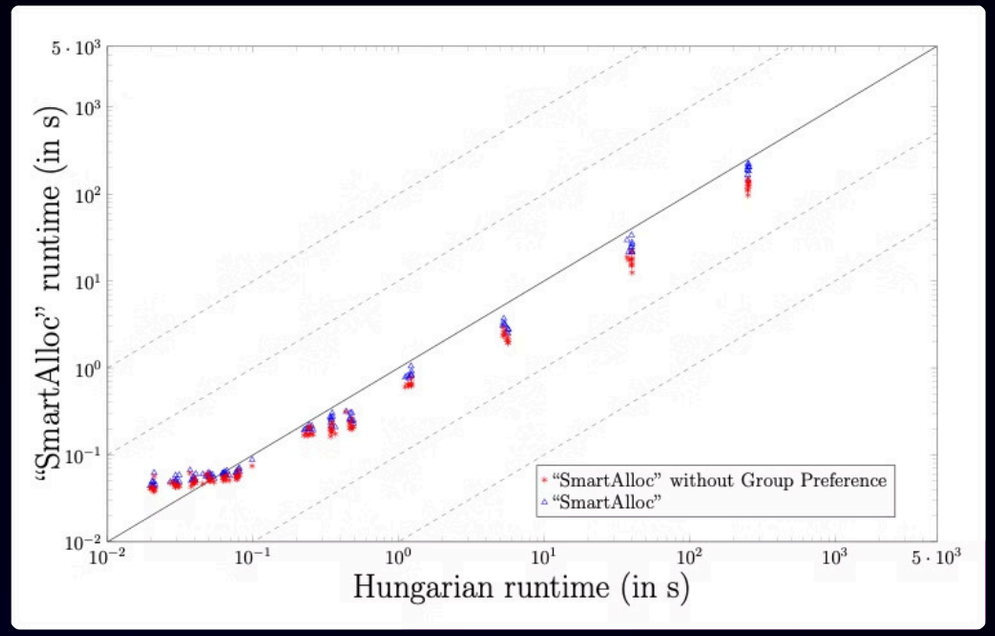
- Lösbare Probleme

- "SmartAlloc" zeigt generell schnellere Laufzeiten, besonders bei größeren Problemgrößen.

Vergleich der Laufzeit nach Präferenztyp P0



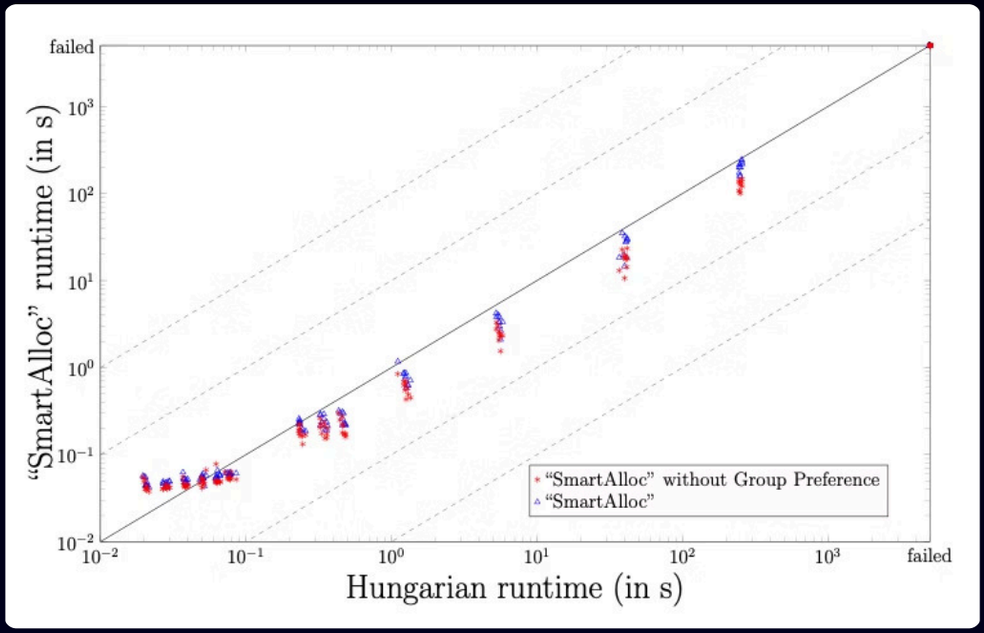
- Alle Probleme



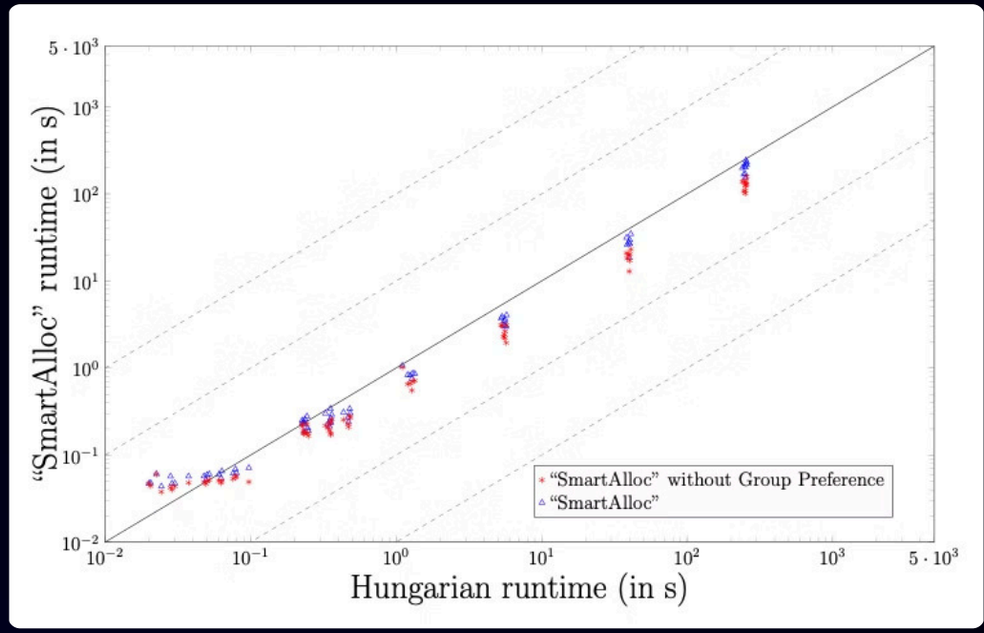
- Lösbare Probleme

Die Leistung variiert je nach Präferenztyp; "SmartAlloc" ist meist schneller, besonders bei den komplexeren Präferenztypen (p1, p2).

Vergleich der Laufzeit nach Präferenztyp P1



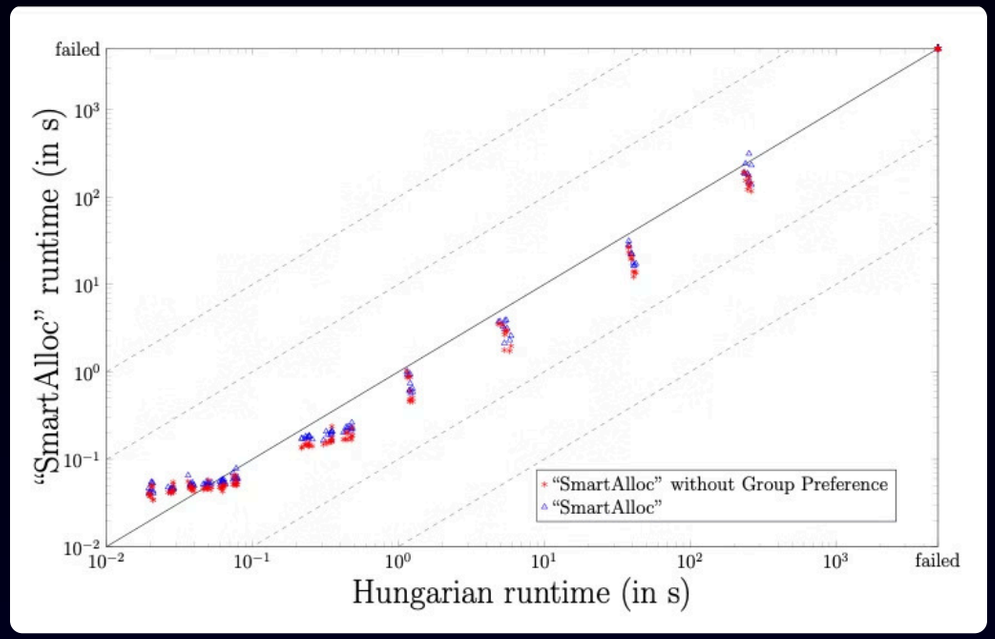
- Alle Probleme



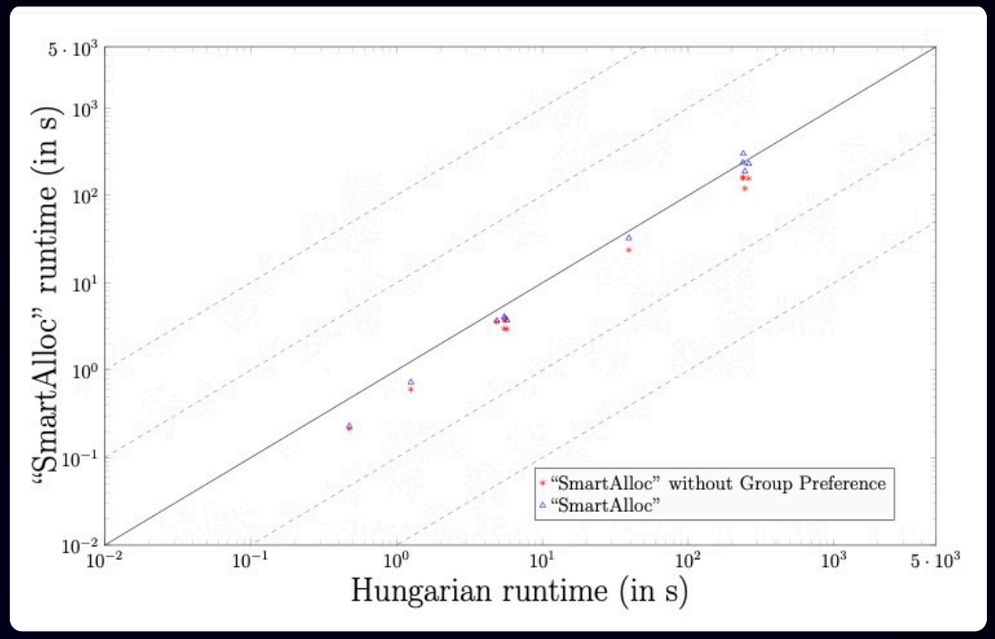
- Lösbare Probleme

Die Leistung variiert je nach Präferenztyp; "SmartAlloc" ist meist schneller, besonders bei den komplexeren Präferenztypen (p1, p2).

Vergleich der Laufzeiten nach Präferenztyp P2



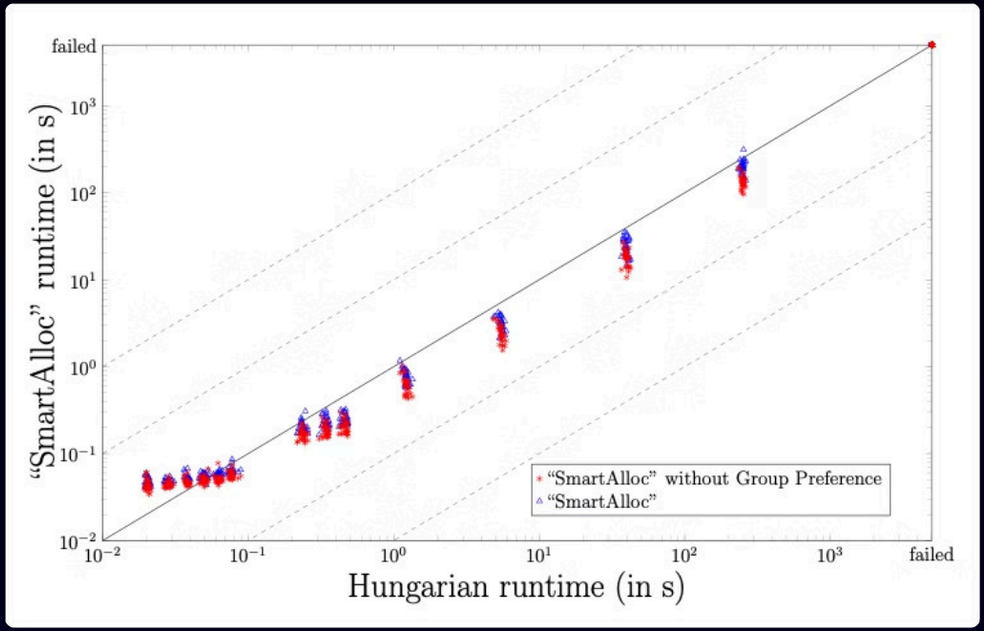
- Alle Probleme



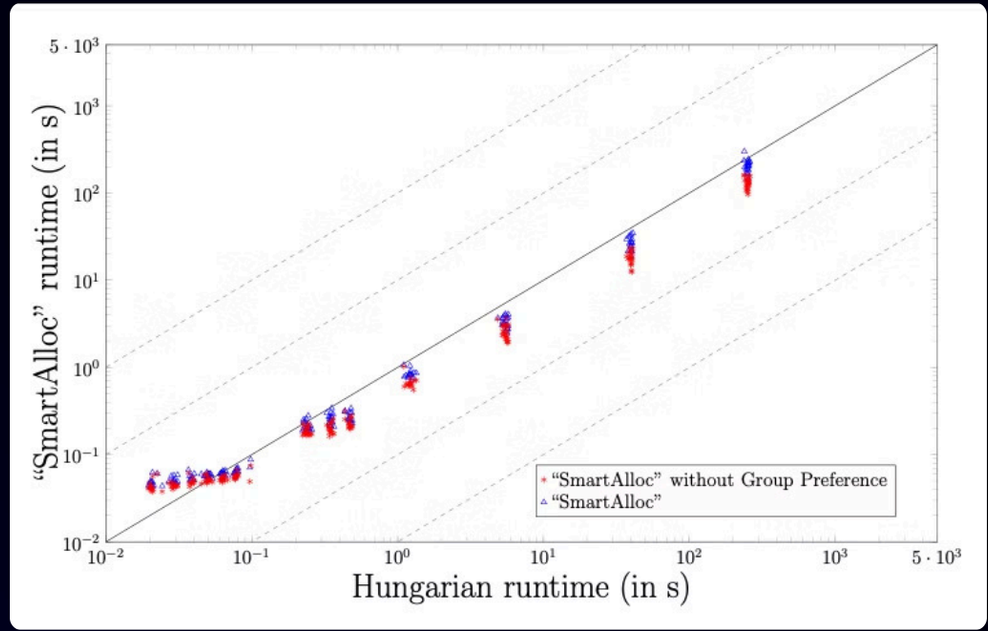
- Lösbare Probleme

Die Leistung variiert je nach Präferenztyp; "SmartAlloc" ist meist schneller, besonders bei den komplexeren Präferenztypen (p1, p2).

Vergleich der Laufzeiten: Gruppenpräferenz



- Alle Probleme



- Lösbare Probleme

- Trotz NP-Schwierigkeit der Gruppenpräferenz zeigt "SmartAlloc" keine signifikante Laufzeitverlängerung im Vergleich zur Ungarischen Methode.

Qualität der Lösung und Kostenanalyse

Algorithmus	Gelöst (Gültig)	Unlösbar gefunden	Nicht gelöst
Ungarische Methode	212	178	120
"SmartAlloc"	212	178	120
"SmartAlloc" ohne Gruppenpräferenz	212	178	120

- Die Ungarische Methode produziert Lösungen mit höheren Kosten, wenn harte Einschränkungen verletzt werden.
- "SmartAlloc" signalisiert ungültige Lösungen, indem keine Lösung ausgegeben wird.
- Die Berücksichtigung von Gruppenpräferenzen in "SmartAlloc" hatte keinen Einfluss auf die Gesamtkosten der Lösung, was darauf hindeutet, dass diese Präferenzen die Zufriedenheit nicht verschlechtert haben.

Herausforderungen und Grenzen

Exponentialwachstum: Das exponentielle Wachstum der Sprachkombinationen erhöht die Komplexität.

Rechenaufwand: MILP benötigt mehr Ressourcen, ist aber besser skalierbar.

Skalierungsprobleme: Beide Methoden haben Schwierigkeiten mit größeren Problemgrößen (750 +).

Schlussfolgerungen und Ausblick

Vergleichende Leistung:

Die Ungarische Methode eignet sich für einfache Probleme; "SmartAlloc" bietet mehr Flexibilität bei komplexen Szenarien.

Zukünftige Arbeit:

- Integration von Sprachkombinationen in das MILP-Modell könnte die Effizienz weiter steigern.
- Inkrementelle Generierung von Sprachkombinationen.

Takeaway:

"SmartAlloc" ist besser geeignet für komplexe Optimierungsszenarien.

Fragen?