Lösungsansätze für das Timetabling Problems mit LLMs

HAW HAMBURG
MULTI AGENT RESEARCH & SIMULATION
www.mars-group.org

Bjarne Martensen
bjarne.martensen@haw-hamburg.de

[1]

Motivation & Problemstellung

In die Erstellung von Stundenplänen an Schulen und Hochschulen fließt auch heute noch viel manuelle Arbeit. Bei Stundenplänen handelt es sich, wie bei Dienstplänen, Schichtplänen und Spielplänen, um ein Timetabling Problem. Dies sind Problemstellungen, bei welchen eine begrenzte Anzahl von Ressourcen zu bestimmten Zeitslots zugeteilt werden soll, unter der Voraussetzung von fachlichen Bedingungen und dass das Ergebnis valide und effektiv ist [2].
Aus Perspektive der Informatik ist dieses Problem NP-Vollständig. Das heißt, mit steigender Anzahl an Abhängigkeiten steigt die Komplexität eines Problems exponentiell. Aus diesem Grund ist es relevant, Lösungen zu optimieren, um in angemessener Zeit Ergebnisse zu erzielen [2].

Related work

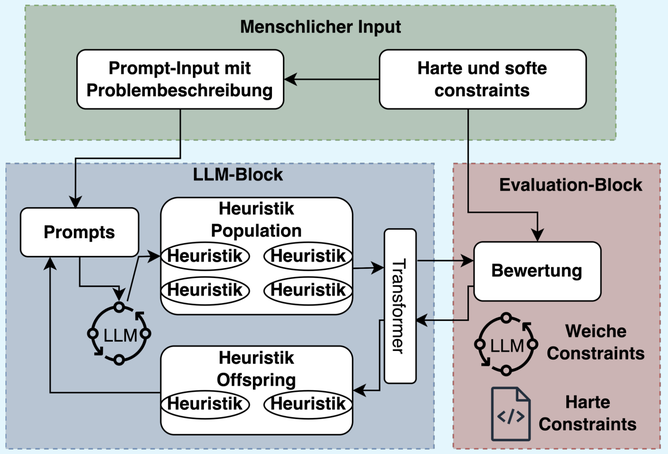
In der Literatur gibt es verschiedene Ansätze zum Lösen des Timetabling Problems. Folgend sind zwei Lösungsoptionen genannt, auf denen weiter aufgebaut wird:

- Nutzen von LLMs zur Generierung von heuristischen Lösungsverfahren mittels evolutionärer Algorithmen nach Lui et al. [3, 4] und Wu et al. [5]
- Anwenden von Automatic-Prompt-Engineering mittels des Gradientenabstiegs zur Generierung von Stundenplänen mit LLMs nach Chen et al. [6]

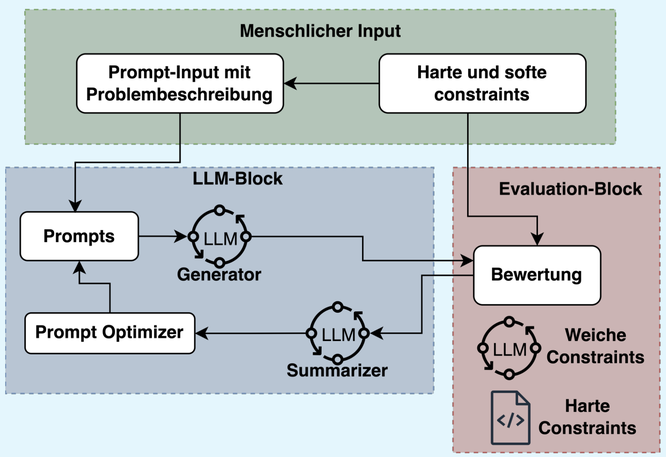
Problemaufbau

Zur Entwicklung von Lösungsansätzen wird ein Fokus auf Architekturen gesetzt, welche mittels LLMs vielversprechend sind, um ein heterogenes Feld an Anwendungsbeispielen zu optimieren.
Die Lösungsansätze aus der Literatur sind in den „LLM-Blöcken“ (blau) in den beiden Grafiken aufgezeigt.
Damit sinnvolle Timetabling Problemstellungen angegangen werden können, existiert neben den Modellen eine Form des menschlichen Inputs (grün). Dieser umfasst die Problembeschreibung und die dafür existieren zwingenden und optionalen Anforderungen.
Da Lösungen nicht immanent eine Bewertung haben, existiert noch ein „Evaluations-Block“ (rot), welcher die zwingenden Anforderungen hart codiert, konfigurierbar bewertet und die weichen Anforderungen über ein LLM evaluiert.

Evolutionärer Algorithmus & LLM



Automatic-Prompt-Engineering



Was ist daran neu?

1. Bisher behandelte Probleme typische Experimentszenarien mit einem überschaubaren Funktionsraum, weshalb neben den blauen Blöcken kein Umfeld nötig war
2. Die aufgezeigten Architekturen verfügen über einen neuen Evaluations-Block, welcher die erstellten Lösungen bewertet
3. Das Ergebnis ist schlussendlich keine Heuristik oder ein Text, sondern ein strukturiertes Datenkonstrukt

Anwendungsfall

Der Hauptanwendungsfall wird das Szenario der Stundenplanerstellung für Schulen sein. Dies nicht nur die Stundenpläne für Klassen, sondern auch Pläne für Lehrer und Räumlichkeiten. Dafür wird mit der GGS St. Jürgen Schule aus Lübeck zusammengearbeitet, von welcher Testdaten und fachlicher Input bereitgestellt werden.
Darüber hinaus werden zur Überprüfung der generalisierten Lösungsfindung zwei weitere Testszenarien aus der Literatur erstellt:

1. Einsatzplan Schichtplanung in Produktion
2. Einsatzplan von Busfahrern

Nächste Schritte

Die nächsten Schritte, um die dargestellte Architektur verproben zu können, sind folgende:

1. Erstellung der allgemeingültigen „grünen“ und „roten“ Bereiche zur Erstellung einer Test-Umgebung für Implementierungen
2. Implementieren des Evolutionären Algorithmus mit Anbindung eines LLMs (aktueller Stand: GPT-4o)
3. Implementieren des Automatic-Prompt-Engineerings unter Verwendung des gleichen LLMs zur Wahrung der Vergleichbarkeit
4. Vergleichen der Ergebnisse und Experimentieren mit den verschiedenen Anwendungsszenarien

References

[1] Icon made by Vector Markets from www.flaticon.com
[2] G. N. Beligiannis, C. Moschopoulos, and S. D. Likothanassis, “A genetic algorithm approach to school timetabling,” Journal of the Operational Research Society, vol. 60, no. 1, pp. 23–42, Jan. 1, 2009, ISSN: 0160-5682. DOI: 10 . 1057 / palgrave . jors . 2602525. [Online]. Available: https://doi.org/10.1057/palgrave.jors.2602525
[3] F. Liu, X. Tong, M. Yuan, and Q. Zhang. “Algorithm Evolution Using Large Language Model.” arXiv: 2311. 15249 [cs]. (Nov. 26, 2023), [Online]. Available: http: //arxiv.org/abs/2311.15249 (visited on 08/03/2024), pre-published
[4] F. Liu, X. Tong, M. Yuan, et al. “Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model.” arXiv: 2401 . 02051 [cs]. (Jun. 1, 2024), [Online]. Available: http://arxiv.org/abs/2401.02051 (visited on 08/02/2024), pre-published
[5] X. Wu, S.-h. Wu, J. Wu, L. Feng, and K. C. Tan. “Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap.” arXiv: 2401.10034 [cs]. (May 29, 2024), [Online]. Available: http://arxiv.org/abs/2401.10034 (visited on 08/03/2024), pre-published
[6] W. Chen, S. Koenig, and B. Dilkina. “RePrompt: Planning by Automatic Prompt Engineering for Large Language Models Agents.” arXiv: 2406 . 11132 [cs]. (Jun. 16, 2024), [Online]. Available: http :/ / arxiv. org /abs/2406.11132 (visited on 08/03/2024), pre-published