制約付き情報推薦のための LLM による数理最適化モデルの自動生成

藤田侑星 [1]　佐々木稔 [1]
[1] 茨城大学 工学部 情報工学科
22t4065s@vc.ibaraki.ac.jp　minoru.sasaki.01@vc.ibaraki.ac.jp

概要

本研究では, 個人の関心と組織的制約を両立する制約付き推薦システム実現に向け,LLM による数理最適化モデルの自動生成手法を提案する. 従来の推薦は複雑な制約の遵守が困難で, 定式化に専門知識を要した. そこで自然言語から最適化モデルを直接生成する手法を検証した結果, 最新モデルは複雑な制約を正確に定式化でき, 構築時間を 52.7%短縮した. 特定のモデルにおいて, 出力形式の不備に起因するソルバー実行エラーが確認されたものの, 本手法により, 自然言語による曖昧な要求を厳密な数理モデルへ構造化する LLM の推論能力の有効性を示すとともに, 専門知識を持たない利用者でも高度な制約付き推薦を享受できる可能性を提示した.

1　序論

現代社会において, 膨大な選択肢の中から個人の関心や状況に合った情報を提示する推薦システムは, 日々の意思決定を支える身近な技術となっている. 従来の推薦システムの多くは, 過去の履歴に基づいた予測スコアの算出に重点を置き, 利用者の関心が高いと思われるアイテムを優先的に提示することで, 自分に合った情報を効率的に見つける手助けをしてきた.

しかし, 今後の推薦システムにおいては, 単に興味関心の高い順に提示するだけでなく, 実社会の複雑な条件を考慮した制約付き推薦システムとしての役割がより重要になると考えられる [1]. 例えば, 教育現場における履修計画や企業の研修プログラムの選定では, 個人の関心に加え, 履修単位の充足, 前提科目の修得, あるいは学習時間の上限といった物理的・論理的な制約条件を厳密に満たす必要がある. このように, 予測された興味度と多様な制約を同時に考慮し, 実行可能な解を提示する技術への転換が求められている.

この制約付き推薦を実現する有効な手法として数理最適化の導入が挙げられるが, 実用上の大きな障壁となっているのが, 個別のシナリオに応じた最適化モデルの構築プロセスである. 現実の問題を数理モデルとして記述するためには, 具体的な条件から, 目的関数や制約条件などの厳密な数式へと落とし込まなければならない. この工程には数学的な素養と定式化のスキルが必要であり, 非専門家である利用者が自らの状況に合わせて動的にモデルを構築・変更することは容易ではない.

本研究では, 自然言語による要求から厳密な数理モデルへの変換を言語処理における重要なタスクと捉え, 大規模言語モデル (Large Language Models, 以下 LLM) を用いて, 制約付き推薦のための最適化モデルを自動生成する手法を提案する [2, 3, 4]. 最適化モデルの構成要素のうち, 目的関数は推薦スコアの最大化・最小化として定型化しやすいが, ユーザーごとに内容が異なり, かつ記述の難易度が高いのは制約条件の構築である. そこで本稿では, 特に自然言語からの制約式の自動生成・支援に焦点を当てる. 具体的には, 大学の履修計画等のスケジューリング問題を対象ドメインとし [5],LLM が自然言語の記述から適切な最適化モデルを構築し, 実行可能なコードを生成するプロセスを検証する. 本アプローチを通じて, 自然言語による曖昧な要求を厳密な数理モデルへと構造化する LLM の推論能力の有効性を明らかにするとともに, 専門知識を持たない利用者であっても, 自身の個別の制約に基づいた最適な推薦結果を即座に得られる環境の実現を目指す.

2　実験方法

本実験では,LLM が自然言語で記述された複雑な要求から, 制約付き推薦のための最適化モデルをいかに正確に構築できるかを検証する. さらに本手法

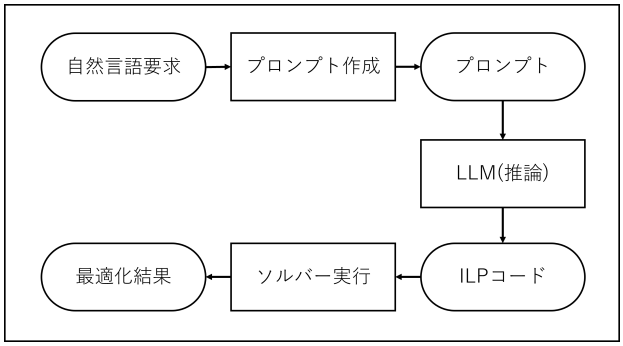

図 1　システム構成

を用いることで, 従来のエンジニアによる手作業と比較して, モデル構築の効率 (時間的コスト) をどの程度改善できるかを明らかにする. なお実験の対象とするドメインは, スケジューリング問題とする.

実験には,OpenAI 社の ChatGPT(GPT-5.2),Google 社の Gemini 3.0, および Anthropic 社の Claude 4.5 Sonnet の 3 種類のモデルを使用した. 本手法における, 入力からソルバー実行までのシステム構成を図 1 に示す. 各モデルに対して, 後述するプロンプトを単一のコンテキストとして入力を行い, ソルバーが直接読み取り可能な整数線形計画法 (Integer Linear Programming, 以下 ILP) 形式の出力を生成させた.

2.1　プロンプトの構造

本実験で使用するプロンプトは,LLM が段階的に問題を理解し, 推薦要件を定式化できるよう, 問題の定義, 目的関数・制約条件, 出力形式といった 3 つのコンポーネントで構成されている. そのうち目的関数・制約条件と出力形式に関する指示構造を以下に示す.

目的関数・制約条件の構造

以上で最適化モデルを定義しています。下の条件を基に最適化モデルを作成してください。なお目的関数のスコアは適当でよいものとし、「制約について」ではコースの種類で指定しているため通し番号に変換する事に留意する事。

コースについて
コースの種類 40、通し番号で 1〜40 とする
決定変数はコースの通し番号に合わせて x1,x2,...,x40 とする

制約について
必修制約
7 は取らなくてはならない

重複排除制約
1、2 のうち最大一つ
⋮
27、28 のうち最大一つ

容量制約
cap_t は 50
$credit_i$ はそれぞれ
1： 7
⋮
40： 21

前提条件制約
3 を取るには 1、2 のどれかを取らなくてはならない
⋮
40 を取るには 39 を取らなくてはならない

本セクションは, 定義された枠組みに基づいて具体的な課題データを適用する部分である. スケジューリング問題の定義に合わせた必修・重複排除・容量・前提条件といった具体的な制約条件を自然言語で記述している. また目的関数の決定変数の重みづけ (各コースの予測スコア) に関しては, 本実験の主眼が制約の正当な定式化にあるため, ランダムな値を指定している. 本実験では, 株式会社日立アカデミー [6] が提供する基盤製品 (データベース関連) および JP1 の実際の研修プログラムを参考に, それぞれ 40 種類・58 種類のコースから成るプロンプトを 2 つ作成した. それぞれのコースには固有の ID が存在しているが, ここでは分かりやすくするために通し番号で表し, また決定変数についても同様の理由で通し番号に合わせている.

出力形式の構造

また作成するにあたって以下のような ILP 形式のものをテキストベースで全て出力してください。なお OBJ には目的関数を記述し、subject to には C0、C1... と各制約条件を通し

番号で羅列し、binary には全ての変数を書いてください。さらにコメント文などは書かなくてよいです。

```
minimize
OBJ:+74 x1 +18 x2 +63 x3 +42 x4 +64 x5 +95 x6
+25 x7

subject to
C0: +1 x1 +1 x2 = 1
C1: +1 x3 +1 x4 +1 x5 <= 1
C2: +1 x1 +1 x2 +2 x3 +3 x4 +1 x5 +2 x6 +3 x7
<= 3
C3: +1 x1 +1 x2 -1 x6 >= 0

binary
x1 x2 x3 x4 x5 x6 x7
```

ここでは, 生成 AI の出力の揺らぎを抑制し, 評価を容易にするために, 特定の ILP 形式によるテキスト出力を指示している. また可読性を上げるためにコメント文は書かないように指示を行っている.

なお問題の定義では, 対象とするスケジューリング問題における集合 (研修コース I, 期間 T), 決定変数 $x_{i,t}$, およびパラメータ (単位時間 $credit_i$, 容量 cap_t, 予測スコア $score_i$) の定義を行っている. これによって,LLM に対してこれから構築すべきルールを数学的な枠組みとして明示している. なお定式化については,Parameswaran らの研究 [7] を参考にして定義している.

2.2 構築効率の評価手法

本研究では,LLM を用いた手法の有効性を検証するために, 手作業による最適化モデルの構築とプロンプト作成による構築の比較実験を行う. 具体的には, 株式会社日立アカデミー [6] が提供する JP1 の研修プログラムを対象として, それぞれの工程における所要時間を計測・比較する. なお情報工学科の学生 1 名が, 同一の仕様書に基づき手作業での定式化を行った。

3 実験結果

実験の結果, いずれの LLM においても, 入力された複雑な制約条件を正確に解釈し, 制約付き推薦に必要な最適化モデルを ILP 形式で出力することが確認された. 以下に,Claude から得られた出力結果 (40 コース) を示す.

実行結果

```
minimize
OBJ: +10 x1 +15 x2 +20 x3 +25 x4 +30 x5 +35
x6 +5 x7 +40 x8 +45 x9 +50 x10 +55 x11 +60 x12
+65 x13 +70 x14 +75 x15 +80 x16 +85 x17 +90
x18 +95 x19 +100 x20 +105 x21 +110 x22 +115
x23 +120 x24 +125 x25 +130 x26 +135 x27 +140
x28 +145 x29 +150 x30 +155 x31 +160 x32 +165
x33 +170 x34 +175 x35 +180 x36 +185 x37 +190
x38 +195 x39 +200 x40

subject to
C0: +1 x7 = 1
C1: +1 x1 +1 x2 <= 1
  ⋮
C7: +7 x1 +8 x2 ··· + 21x40 <= 50
C8 : +1x1 + 1x2 − 1x3 >= 0
  ⋮
C42 : +1x39 − 1x40 >= 0

binary
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14
x15 x16 x17 x18 x19 x20 x21 x22 x23 x24 x25
x26 x27 x28 x29 x30 x31 x32 x33 x34 x35 x36
x37 x38 x39 x40
```

各制約条件の妥当性についてだが, 生成されたモデルを精査したところ, 各制約式はプロンプトで与えた要求事項と完全に一致していた.

- C0(必修制約)：特性コース (アイテム) の選択が必須であることを正しく定義されている.
- C1〜C6(重複排除制約)：複数の選択肢から一つを選択する排他的な制約が適切に記述されている.
- C7(容量制約)：総受講時間 (リソース上限) と各アイテムのコストの関係が正確な不等式として生成されている.
- C8〜C42(前提条件制約)：コース間の依存関係 (前提条件) が, 変数間の線形制約式として論理的に正しく表現されている.

表 1　モデル構築手法による所要時間の比較

構築手法	所要時間（hh:mm:ss）	短縮率
手作業（従来手法）	01:02:10	—
プロンプト作成（提案手法）	00:29:26	52.7%

ChatGPT および Gemini においても,Claude と同様に論理的に正しいモデルがどちらのプロンプトでも生成された. モデル間で制約式の不等号の向き (例：$x_A - x_B \geq 0$ と $x_B - x_A \leq 0$) などの表記上の細かな差異や,Gemini においてのみ複数の制約式が 1 行に連結されて出力される傾向が確認された. 表記の差異については, 最適化問題としての数式の解釈は同一であり, 実運用上の支障はない. しかし,Gemini による 1 行での出力は, ソルバー実行時に構文エラーを引き起こす要因となる. 一方, 他モデルで生成された最適化モデルは, 修正なしでソルバーでの実行可能であることを確認した.

次に, 手作業による最適化モデルの構築とプロンプト作成による LLM を用いた構築に要した時間を測定した結果を表 1 に示す.

表 1 が示す通り, 手作業ではモデル構築に約 62 分を要したのに対し,LLM を用いた手法では約 29 分で完了した. これにより,LLM を活用することでモデル構築にかかる時間を 50%以上短縮できることが確認された.

4　考察

本実験の結果から,LLM を用いた最適化モデルの構築には, 従来のエンジニアによる手作業と比較して以下の 2 点の優位性が認められた.

- 手作業では, 変数番号の指定ミスや符号の誤りを防ぐために慎重に作業を進めるため, 多くの時間を費やしてしまうが, 提案手法では LLM が一括変換を行う. これにより, 人間はどのような制約で推薦を行うかという要件定義の作業に注力できるため, 構築時間を 50%以上削減するという大幅な効率化を達成したと考えられる.
- 本実験で使用した 3 種類のモデル (GPT-5.2,Gemini3.0, Claude 4.5 Sonnet) がいずれも論理的に正しい制約式を生成した事実は, 自然言語から数理構造を抽出する能力において,LLM が一定の汎用性を有していることを示している.

一方で, 実運用における課題も明らかになった.Gemini で見られた制約の 1 行出力は, ソルバー実行時にエラーの要因となる. これはプロンプトでコメント文は書かなくてよいと指示したことが,LLM に過度な圧縮 (改行の省略) を促した可能性がある [8]. したがって,LLM を実務に組み込む際には, モデルごとの出力傾向を考慮した事後処理や, プロンプトによる出力形式の厳密な指定が不可欠であると言える.

5　結論

本研究では, 推薦システムにおいて個人の興味関心と組織的な制約を両立させる意思決定支援を実現するため,LLM を用いた数理最適化モデルの自動生成手法を提案し, その有効性を検証した. スケジューリング問題を対象とした実験の結果, 最新の LLM(GPT-5.2,Gemini 3.0,Claude 4.5 Sonnet) は, 自然言語で記述された複雑な制約条件を正確に理解し,ILP 形式の数式へと正しく変換できることが示された.

本手法の導入により, 従来の手作業と比較して, モデル構築時間を 52.7%短縮することに成功した. これは, より本質的な要件定義に注力できることを意味しており, 専門知識を持たない利用者であっても高度な最適化技術を享受できる可能性を提示した. ただし, モデルによってはソルバー実行時にエラーを誘発する形式で出力されるケースも確認されたため, 実用化に向けてはこれらの出力形式の揺らぎを制御する機構の実装が求められる.

今後は, 問題の分割手法や大規模な変数を効率的に扱うためのプロンプトエンジニアリングの改良を行い, より広範かつ複雑な実業務への適用可能性を追求していく.

参考文献

[1] Sinan Seymen, Himan Abdollahpouri, and Edward C. Malthouse. A constrained optimization approach for calibrated recommendations. In **Proceedings of the 15th ACM Conference on Recommender Systems (RecSys '21)**, pp. 607–612, 2021. https://doi.org/10.1145/3460231.3478857

[2] Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. LLMOPT: Learning to define and solve general optimization problems from scratch. In **Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS 2024)**, 2024. https://arxiv.org/abs/2410.13213

[3] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In **Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)**, 2024. https://arxiv.org/abs/2309.03409

[4] Ganesh Prasath, and Shirish Karande. Synthesis of mathematical programs from natural language specifications. In **Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023)**, pp. 832–841, 2023. https://arxiv.org/abs/2304.03287

[5] Werayuth Charoenruengkit. Course planning optimization with conditional constraints using integer linear programming. In **Proceedings of the 2nd International Conference on Digital Technology in Education (ICDTE 2018)**, pp. 89–93, 2018. https://doi.org/10.1145/3284497.3284506

[6] 株式会社日立アカデミー.「研修コース体系図（製品別）2025 年度上期」. https://www.hitachi-ac.co.jp/service/opcourse/flow/2025s-flow/, (2026-01-04 閲覧).

[7] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. Recommendation systems with complex constraints: A course recommendation perspective. In **Proceedings of the 5th ACM Conference on Recommender Systems (RecSys '11)**, pp. 13–20, 2011. https://doi.org/10.1145/2037661.2037665

[8] Man-Fai Wong, Shangxin Guo, Ching-Nam Hang, Siu-Wai Ho, and Chee-wei Tan. Natural language generation and understanding of big code for AI-assisted programming: A review. **Entropy**, vol. 25, no. 6, p. 888, 2023. https://doi.org/10.3390/e25060888