Towards A Natural Language Interface
for Flexible Multi-Agent Task Assignment

Jake Brawer[1][2], **Kayleigh Bishop**[1], **Bradley Hayes**[1], **Alessandro Roncone**[1]

[1] Department of Computer Science, University of Colorado Boulder, 1111 Engineering Drive, Boulder, CO USA
[2] US Army Research Laboratory, Aberdeen Proving Ground, MD 21005, USA
firstname.lastname@colorado.edu

Abstract

Task assignment and scheduling algorithms are powerful tools for autonomously coordinating large teams of robotic or AI agents. However, the decisions these system make often rely on components designed by domain experts, which can be difficult for non-technical end-users to understand or modify to their own ends. In this paper we propose a preliminary design for a flexible natural language interface for a task assignment system. The goal of our approach is both to grant users more control over a task assignment system's decision process, as well as render these decisions more transparent. Users can direct the task assignment system via natural language commands, which are applied as constraints to a mixed-integer linear program (MILP) using a large language model (LLM). Additionally, our proposed system can alert users to potential issues with their commands, and engage them in a corrective dialogue in order to find a viable solution. We conclude with a description of our planned user-evaluation in the simulated environment *Overcooked* and describe next steps towards developing a flexible and transparent task allocation system.

Introduction

In this paper we propose a flexible natural language interface for a task assignment and scheduling system. Task assignment and scheduling algorithms are powerful tools for coordinating large teams of robots, AIs, or humans while demanding minimal user oversight. As demonstrated by Gombolay et al. (2015), such systems are not only generally superior to users in delegating tasks but that users prefer to cede control to them. However, this finding has been challenged by work demonstrating a reversal in this preference when a user's workload is low (Karakikes and Nathanael 2023), such as when users are acting in a supervisory role. Indeed, when empowered to provide supervisory guidance to task assignment systems, users can sometimes quickly infer crucial parts of the optimal solution that would otherwise require substantial time to compute (Petersen, Kleiner, and Von Stryk 2013). A further complication is that the task assignment problem is often posed as a constrained mixed-integer linear program (MILP; Gombolay, Wilcox, and Shah 2013; Omar and Payeur 2019; Wang and Gom-

bolay 2020), which can be difficult to modify, let alone understand, without technical expertise or the use of inflexible, task-dependent GUIs. We aim to combine the power of algorithmic approaches with the intuition and experience of human supervisors by giving users the ability to easily control and modify task scheduling formulations.

Our work draws upon research interrogating the use of natural language interfaces for task assignment and scheduling, robot policy learning and motion planning, as well as recent innovations in transformer-based large language models (LLMS; Vaswani et al. 2017). Work in the former has demonstrated that user-provided symbolic constraints can be used to effectively guide a wide range of decision making and learning processes, including policy shaping (Brawer et al. 2023) and multi-agent task assignment (Petersen, Kleiner, and Von Stryk 2013). In terms of the latter, recent work has shown that LLMs can both transform unrestricted natural language prompts into actionable representations, such as robot policy code (Liang et al. 2023; Huang et al. 2023), as well as produce concise and accurate text summaries of extant source code (e.g. Ahmed and Devanbu 2022). Our approach combines both strands of research by allowing users to influence the task allocation process through natural language directives and using LLMs to enforce these directives as valid constraints on an underlying MILP. Crucially, our system can alert users when added constraints conflict with previously specified constraints, or prevent a satisficing solution from being found, engaging users in a corrective dialogue to amend or remove the faulty constraint. Additionally, users can interrogate the system as to why certain assignments were made, improving the transparency of the entire process.

In the remainder of this paper, we provide an overview of the proposed approach, as well as outline a planned user-study for evaluating the efficacy and ease-of-use of the approach in a collaborative gridworld environment based on the game *Overcooked* (Carroll et al. 2019).

Method

In this section we detail our proposed task-assignment dialogue interface. We first define the task-assignment problem for multiple robots moving and working in the same physical space as a mixed-integer linear program (MILP). Subsequently, we outline proposed architectural components that
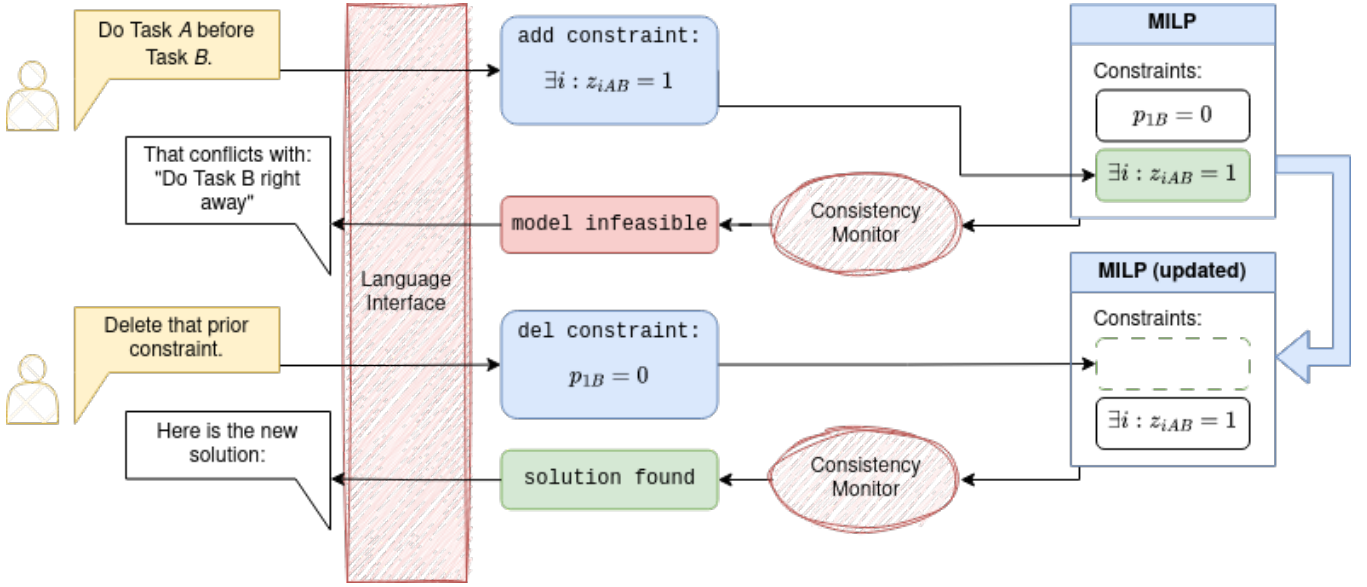
Figure 1: An example interaction with our proposed system. Our approach will enable users to modify and inspect a multi-agent task assignment system via the interface of spoken language. This will be accomplished by converting user commands into mathematical constraints on an associated MILP using an LLM. The consistency monitor ensures that user-provided constraints do not prevent task assignments from being found, and engages the user in a corrective dialogue if problematic constraints are identified.

enable a naive user-in-the-loop to modify and query the task assignment system. Refer to Fig. 1 for a high-level diagram of our approach.

Our approached builds upon and extends the method outlined by Petersen, Kleiner, and Von Stryk (2013). Here the authors propose a system enabling users to add constraints to a task-assignment and scheduling MILP designed to coordinate teams of robots. Our work builds on this in two important ways: 1) constraints will be specified entirely via natural language, rather than a bespoke, task-dependent GUI interface; 2) our system is specifically designed to be robust to the challenges and ambiguities inherent to natural language interactions as well as the complexities of real-time system adaptation made possible by such interactions. To do this we adopt the same basic formalism developed by Petersen, Kleiner, and Von Stryk (2013):

$$\text{minimize} \quad \sum_{i \in R} \sum_{j_1 \in T \cup \{0\}} \sum_{j_2 \in T} z_{ij_1j_2} \cdot k_{ij_1j_2} \tag{1}$$
$$- \sum_{i \in R} \sum_{j \in J} x_{ij} \cdot \rho_j + \sum_{j \in T} (p_{1j} \cdot \alpha_1 + p_{2j} \cdot \alpha_2)$$

subject to
$$C_{pre}(A, S, P)$$
$$C_{user}(A, S, P|U)$$

Here $x_{ij}, z_{ij_1j_2} \in A$ for agents $i \in R$ and tasks $j, j_1, j_2 \in T$ are binary decision variables indicating the assignment of an agent i to task j and the order in which tasks are assigned, respectively. $k_{ij_1j_2}, \rho_j \in P$ are scalar parameters

encoding the cost incurred by agent i for attempting task j_2 after j_1, and the reward for completing task j, respectively. $p_{1j}, p_{2j} \in S$ are continuous decision variables weighted by scalar parameters $\alpha_1, \alpha_2 \in P$ reflecting how early or late a task j has been scheduled, respectively. In sum, this objective seeks to find assignments and schedules that maximize the reward accrued while minimizing the costs incurred by these assignments and the deviations of task schedules from their pre-specified time windows.

C_{pre} and C_{user} are sets of constraints applied to the MILP. C_{pre} is a minimal set of pre-specified constraints comprised of the decision variables and parameters in A, S and P. These constraints ensure that a baseline, useful solution can be found for a wide range of multi-agent coordination problems. For example, the constraint $c_n \in C_{pre}$ where $c_n = \sum_{i \in R} x_{ij} \leq 1, \forall j \in T$ ensures that only one agent $i \in R$ is assigned to one task $j \in T$ at a time.

C_{user} is a set of user-defined constraints also comprised of the symbols in A, S and P, but conditional on user utterances U issued to the language interface (see following section). These constraints encode a user's high-level preferences or the assignment and scheduling process. For example, if a user identifies that agents a and b collaborate well on a task j, a user might tell the language interface "assign agent a to task i if agent b has already been assigned to it," which would be translated to the corresponding conditional hard constraint $x_{aj} \geq x_{bj}$.

Language Interface

We intend the language interface to act as a bidirectional interface between the task assignment process and the user's

preferences for the assignment process; it will allow users to both "program" the system by issuing verbal commands that will be converted into constraints applied to the MILP, as well as "debug" it, by engaging the user in corrective dialogue when issues arise. Ultimately this requires of our language interface two key abilities: i) the ability to transform language into a corresponding mathematical constraint in the form of a syntactically correct MILP optimization API call (Gurobi in this instance; Pedroso 2011), and ii) the ability to transform these same API calls back into comprehensible text explanations. To do this we will adapt recent methods for eliciting accurate task-oriented dialogue from LLMs (Hudeček and Dušek 2023; Yu et al. 2023). The main idea is to dynamically construct prompts, containing few-shot examples, based both on the system and task state as well as the user utterance itself. This is achieved in part by having the LLM perform multiple passes over a user utterance. In the initial, pre-processing pass, the LLM first attempts to discern the action the user wishes the system to perform (e.g. to add or remove a constraint, to produce an explanation, etc.). This determination is then used to select a more targeted prompt and set of examples for eliciting an accurate and relevant response from the LLM given the same user utterance.

As a consequence, the language interface can enable a user to debug "compile-time" and "run-time" errors raised by the task assignment system, to extend the programming metaphor further. Compile-time errors refer here to a failure of the system to find a valid solution to the MILP. Such a failure might be due to a user issuing commands that produce overly restrictive constraints, or constraints that are in conflict. As outlined in the following section, in order to correct this type of error, the language interface would engage the user in a corrective dialogue in order to remove or amend the offending constraints.

Run-time errors in the assignment process can occur if tasks are assigned, but do not line up with the user's communicated preferences. In this case, the user would have to take a more proactive role in the debugging process. The language interface will accommodate this by allowing users to issue interrogative, "why?" queries, such as, "why wasn't agent i assigned to task j?" Such a query would initiate a *counterfactual assignment* wherein the consistency monitor would attempt to make this assignment, and observe which constraints, if any, are violated. The violated constraints are then sent to the language interface, and a suitable answer to the user's query is generated.

Consistency Monitor

As we are designing our system with the intention that users will be regularly adding to and modifying its constraints, it's imperative that it be able to quickly check the consistency of these constraints. That is, it is conceivable that a user may inadvertently "break" the task assignment, either by suggesting two constraints that are in direct conflict (e.g. the constraints "task a should be completed after task b" followed by the constraint "task b should be completed after task a"), or a constraint that is overly restrictive.

In order to mitigate these potential challenges, our system will utilize a three-phased consistency check. In the first
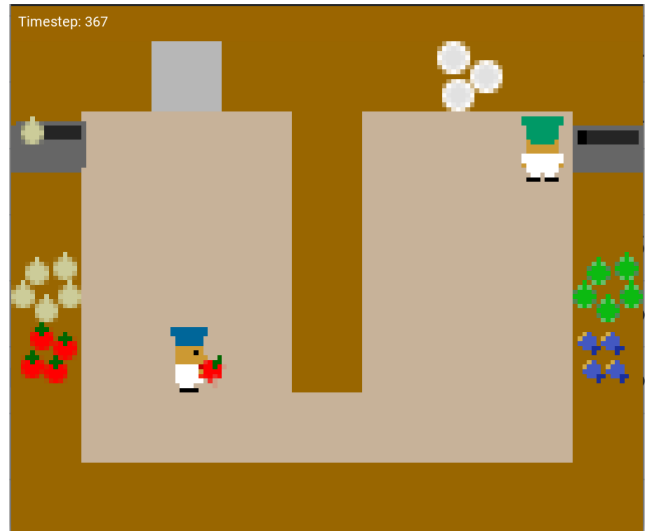

Figure 2: An example of the Overcooked environment. For an evaluation of our system, we would extend layouts used in previous papers to include more tasks and (potentially) more than two agents.

phase, our system will perform a *semantic check*, wherein we leverage an LLM's implicit common-sense and linguistic knowledge to check if any of the issued natural language commands are in conflict. After a user has issued a command via the language interface, the consistency monitor prompts the language interface's LLM to check this command against commands that have been previously issued. If a potential conflict is identified, the language interface will communicate this to the user and engage them in a corrective dialogue. During this dialogue the language interface will prompt the user to either amend or remove one of the candidate constraints, or ignore the warning and attempt to perform the task assignment without changing any of the constraints.

If the semantic check passes, the consistency monitor enters phase two. In this phase, our system will attempt to find a solution to the MILP using the constraints provided. If a solution cannot be found, the consistency monitor will perform a *relaxation check*, wherein our system will attempt to solve the MILP by relaxing constraints in C_{user} using, for example, a branch and bound relaxation approach (Clausen 1999). While this may produce a solution, the solution may violate the user's stated expectations for the task assignment. Thus, the consistency monitor sends the relaxed constraints to the language interface, wherein it will attempt to produce natural language explanations of the new constraints using its LLM. Much like in the first phase, the user will be prompted to accept, reject, or modify the proposed constraints.

The third and final phase is initiated when a relaxed solution cannot be found. In this phase, the consistency monitor will perform an *ablation check*, wherein it will attempt to solve this MILP by iteratively removing constraints from C_{user}. Removed constraints are then sent to language inter-

face, wherein a natural language explanation is generated, prompting the user to try and reformulate their commands, if they desire.

Evaluation

We plan to run a user-study in order to evaluate our approach. In this section we describe our planned experimental setup and outline questions we wish to be able to answer.

Environment and Setup

Our experiments will involve naive human users interacting with and overseeing a task-assignment system during a simulated multi-agent task. A good candidate for this task is the Overcooked environment (Carroll et al. 2019; Fig. 2), which has been gainfully employed across machine learning and human-AI teaming research (e.g. Hu et al. 2021; Aroca-Ouellette et al. 2023; Pearce et al. 2023). The Overcooked environment is a gridworld based on the popular video game *Overcooked*, in which agents must "cook" various dishes by performing actions like gathering ingredients and plating dishes throughout the shared environment. The shared reward for the agents is based on the number of dishes they are collectively able to prepare within the time limit. This domain is ideal for our purposes as it rewards teamwork and coordination among agents, though is simple enough for new users to quickly understand and follow. At the same time, effectively coordinating agents can be a challenge depending on the arrangement of the environment.

Experiments

With our experiments we wish to evaluate our approach along subjective and objective criteria. Our research aims can be summarized by the following three questions:

1. How accurately can our LLM-based language interface transform natural speech into valid symbolic constraints, and vice-versa?

2. How effectively can users interact with our system to generate productive and viable assignments?

3. How well do users' expectations for the system's behavior align with its actual behavior, and how does this impact user perceptions of the system overall?

In order to answer these questions, we employ a between-subjects user study. In our two experimental conditions, participants will be tasked with overseeing a game of Overcooked by providing supervisory constraints to teams of AI-controlled agents, with the goal of maximizing the agents' score across all rounds. In the first condition, users will provide these constraints using the language interface described above. In the second condition, users will be interacting with a system inspired by (Petersen, Kleiner, and Von Stryk 2013), where desired constrained are selected and applied using a drop-down GUI interface. Our control condition will have the task assignment and scheduling system running autonomously, without user oversight.

Answering question 1. will require us collect natural language commands from participants and their ground-truth

constraint counterpart, and comparing them against the output to our model. Empirically quantifying the accuracy of the model in this regard is critical, as the ability to effectively translate between user speech and symbolic constraints is central to the utility of our approach. Our primary empirical result will come from an exploration of question 2. We intend to compare participants across our experimental and control conditions in terms how well the agents perform in the Overcooked environment. This will allow us to quantify the performance impact a human-in-the-loop has on the assignment process as a function of interaction paradigm. Question 3 will be explored via post-experiment questionnaire. This questionnaire will enable us to interrogate the potential trade-offs between performance and interactivity.

Conclusion and Future Work

In this paper we outline ongoing work towards a flexible natural language interface for a multi-agent task-assignment system. Our proposed system enables users to modify and inspect the constraints on a MILP via spoken dialogue, granting them control and insight over the task assignment process. We provide a high level overview of the system architecture, and outline a planned user study for evaluating our approach.

As this work is in its preliminary stages, there are still a number of open questions and challenges that need to be resolved. For instance, the complete set of symbols that comprise the MILP and its constraints, and thus the scope of constraints the users can apply, has yet to be fully determined. Similarly maximizing the accuracy our language interface for producing constraints, while minimizing LLM "hallucinations" (Dinan et al. 2018) remains an ongoing challenge. Nevertheless, when our system is complete, we believe it will bridge the gap between non-technical users and task assignment systems, and foster more effective human-AI partnerships.

Acknowledgements

References

Ahmed, T.; and Devanbu, P. 2022. Few-shot training LLMs for project-specific code-summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 1–5.

Aroca-Ouellette, S.; Aroca-Ouellette, M.; Biswas, U.; Kann, K.; and Roncone, A. 2023. Hierarchical Reinforcement Learning for Ad Hoc Teaming. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multi-agent Systems*, 2337–2339.

Brawer, J.; Ghose, D.; Candon, K.; Qin, M.; Roncone, A.; Vázquez, M.; and Scassellati, B. 2023. Interactive Policy Shaping for Human-Robot Collaboration with Transparent Matrix Overlays. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, 525–533.

Carroll, M.; Shah, R.; Ho, M. K.; Griffiths, T.; Seshia, S.; Abbeel, P.; and Dragan, A. 2019. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32.

Clausen, J. 1999. Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen*, 1–30.

Dinan, E.; Roller, S.; Shuster, K.; Fan, A.; Auli, M.; and Weston, J. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.

Gombolay, M.; Wilcox, R.; and Shah, J. 2013. Fast scheduling of multi-robot teams with temporospatial constraints.

Gombolay, M. C.; Gutierrez, R. A.; Clarke, S. G.; Sturla, G. F.; and Shah, J. A. 2015. Decision-making authority, team efficiency and human worker satisfaction in mixed human–robot teams. *Autonomous Robots*, 39: 293–312.

Hu, H.; Lerer, A.; Cui, B.; Pineda, L.; Brown, N.; and Foerster, J. 2021. Off-belief learning. In *International Conference on Machine Learning*, 4369–4379. PMLR.

Huang, S.; Jiang, Z.; Dong, H.; Qiao, Y.; Gao, P.; and Li, H. 2023. Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model. *arXiv preprint arXiv:2305.11176*.

Hudeček, V.; and Dušek, O. 2023. Are LLMs All You Need for Task-Oriented Dialogue? *arXiv preprint arXiv:2304.06556*.

Karakikes, M.; and Nathanael, D. 2023. The effect of cognitive workload on decision authority assignment in human–robot collaboration. *Cognition, Technology & Work*, 25(1): 31–43.

Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; and Zeng, A. 2023. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 9493–9500. IEEE.

Omar, A.-B.; and Payeur, P. 2019. Probabilistic task assignment for specialized multi-agent robotic systems. In *2019 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, 1–7. IEEE.

Pearce, T.; Rashid, T.; Kanervisto, A.; Bignell, D.; Sun, M.; Georgescu, R.; Macua, S. V.; Tan, S. Z.; Momennejad, I.; Hofmann, K.; et al. 2023. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*.

Pedroso, J. P. 2011. Optimization with gurobi and python. *INESC Porto and Universidade do Porto,, Porto, Portugal*, 1.

Petersen, K.; Kleiner, A.; and Von Stryk, O. 2013. Fast task-sequence allocation for heterogeneous robot teams with a human in the loop. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1648–1655. IEEE.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, Z.; and Gombolay, M. 2020. Learning scheduling policies for multi-robot coordination with graph attention networks. *IEEE Robotics and Automation Letters*, 5(3): 4509–4516.

Yu, W.; Gileadi, N.; Fu, C.; Kirmani, S.; Lee, K.-H.; Arenas, M. G.; Chiang, H.-T. L.; Erez, T.; Hasenclever, L.; Humplik, J.; et al. 2023. Language to Rewards for Robotic Skill Synthesis. *arXiv preprint arXiv:2306.08647*.