Cloud Kitchen: Using Planning-based Composite AI to Optimize Food Delivery Processes

Slavomír Švancár, Lukáš Chrpa, Filip Dvořák and Tomáš Balyo

Filuta AI, Inc., 1606 Headway Cir STE 9145, Austin, TX 78754, United States

Abstract. The global food delivery market provides many opportunities for AI-based services that can improve the efficiency of feeding the world. This paper presents the Cloud Kitchen platform as a decision-making tool for restaurants with food delivery and a simulator to evaluate the impact of the decisions. The platform contains a Technology-Specific Bridge (TSB) that provides an interface for communicating with restaurants or the simulator. TSB uses a planning domain model to represent decisions embedded in the Unified Planning Framework (UPF). Decision-making, which concerns allocating customers' orders to vehicles and deciding in which order the customers will be served (for each vehicle), is done via a Vehicle Routing Problem with Time Windows (VRPTW), an efficient tool for this problem. We show that decisions made by our platform can improve customer satisfaction by reducing the number of delayed deliveries using a real-world historical dataset.

1 Introduction

The global food delivery market accounted for $190 billion in 2022 and is forecasted to grow over $500 billion in 2032[1]. Such a market increase, accelerated by the recent COVID-19 pandemic, establishes (online) food delivery as a new norm in society [2, 11]. This opens a number of opportunities for AI-based services that can optimise the food delivery process by reducing costs and improving customer satisfaction. Apps such as UberEats™ involve courier services that can collect and deliver orders from multiple restaurants [12] and hence the problem they deal with can be modelled as a (dynamic) Pickup and Delivery problem [10, 1].

In contrast to courier services (e.g. UberEats), we consider restaurants having their own food delivery. Benefits of having their own food delivery include not paying provisions for third-party order and delivery services and not relying only on apps or web services (e.g. being able to accept orders by phone). Hence, the problem we deal with in this paper concerns assigning food orders to vehicles and providing routing for the vehicles, i.e., deciding in which order the deliveries are made. Such a problem can be modelled as a domain-independent planning problem [5], in PDDL (Planning Domain Definition Language) [3] which is currently the most established language for specifying (domain-independent) planning tasks. Therefore, a wide range of different planning engines as well as tools for plan verification and plan execution monitoring evolved around PDDL. A recent effort to unify these planning engines and tools motivated the development of the Unified Planning Framework (UPF) [9]. Although PDDL and UPF provide us a machinery for plan verification, explanation, and monitoring, the performance of existing domain-independent planners is rather poor for the problem. It is more effective to model our problem as a Vehicle Routing Problem with Time Windows (VRPTW) [7], which is specifically designed for it. Embedding VRPTW into the UPF [9] can keep both the efficiency of solving (by VRPTW) and the explainability of solutions [4]. It can be seen as a straightforward example of Composite AI [8], which aims at an effective combination of different AI techniques for addressing a given problem, or, in the planning context, as an example of Planning Modulo Theories [6]. Although we provide a rather trivial example of a Composite AI approach, it can be seen that the use of PDDL (a domain-independent language) and UPF (a domain-independent tool) can provide a useful machinery for for plan monitoring and explanation while embedding VRPTW (a specific solver) into it provides an efficient way to solve our problem.

This paper introduces *Cloud Kitchen*, a platform that provides a decision-making tool for restaurants (or kitchens) with food delivery. The platform is designed to run as a cloud service with a Technology-Specific Bridge (TSB) serving as its interface. TSB uses a PDDL model to represent decisions that are embedded into UPF. TSB passes information about new orders and available vehicles into a Decision-Making component, which formulates and solves VRPTW tasks. Solutions are then passed back to TSB, which interprets them as (PDDL) plans. Cloud Kitchen also includes a simulator that simulates decisions made on real historical datasets to demonstrate the usefulness of the platform for potential customers (i.e., restaurants). We empirically show on a historical dataset provided by our industrial partner that the Cloud Kitchen platform can improve customer satisfaction by considerably reducing the number of deliveries delayed by more than 10 minutes.

2 Vehicle Routing Problem with Time Windows

A *Vehicle Routing Problem with Time Windows (VRPTW)* $T = (V, C, \mathcal{G})$ consists of a set of vehicles V, a set of customers C, and a graph $\mathcal{G} = (N, E)$ such that $|N| = |C| + 2$. By convention, we denote the starting depot as n_0 and the returning depot as n_{k+1} and the location of a customer c_i as n_i (assuming that $|C| = k$). For each edge $(i, j) \in E$, we denote by $dist(i, j)$, resp. $time(i, j)$, the distance between locations i and j, resp. the driving time between i and j (including service time in j). Each vehicle $v \in V$ has its capacity q_v and each customer $c_i \in C$ has its demand d_i (representing the "size" of the delivery) and its *time window* $[a_i, b_i]$ representing that the delivery has to be made at time a_i at the earliest (the delivering vehicle might need to wait if it arrives earlier) and at time b_i at the

[1] https://www.precedenceresearch.com/online-food-delivery-market

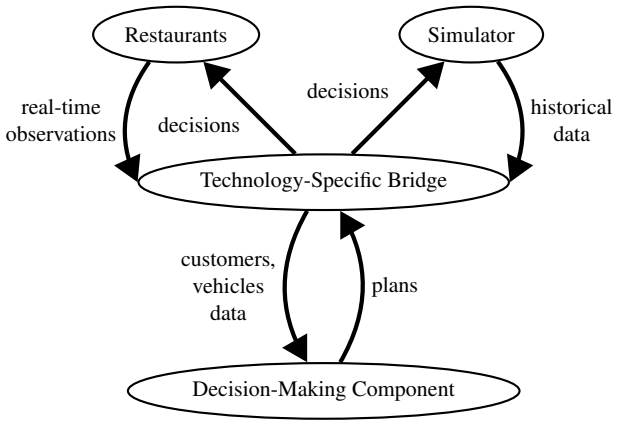

Figure 1. Architecture of the Cloud Kitchen platform

Algorithm 1 The high-level routine of the DM component

Require: Deadline extension step δ

while True **do**
  $C_{new} \leftarrow$ GetNewOrders()
  $V_{disp} \leftarrow$ GetDispatchedVehicles()
  $V_{new} \leftarrow$ GetReturnedVehicles()
  $V \leftarrow (V \setminus V_{disp}) \cup V_{new}$
  $C \leftarrow (C \setminus \bigcup_{v \in V_{disp}} C_v) \cup C_{new}$
  $\mathcal{G} \leftarrow$ GraphUpdate($\mathcal{G}, C$)
  $delay \leftarrow 0$
  repeat
    $T \leftarrow (V, C, \mathcal{G})$
    $\tau \leftarrow$ SolveVRPTW(T, $delay$)
    if τ is invalid **then**
      $delay \leftarrow delay + \delta$
    end if
  until τ is a valid solution of T
  SendSolution(τ)
end while

latest. We can also specify a *scheduling horizon* $[a_0, b_{k+1}]$ representing when a vehicle can leave the depot (a_0) and the latest time the vehicle returns to the depot (b_{k+1}).

A *solution* τ of a VRPTW T is a set of triples (C_v, π_v, t_v) for each $v \in V$ such that $C = \bigcup_{v \in V} C_v$, $C_v \cap C_{v'} = \emptyset$ (for all $v, v' \in V$ such that $v \neq v'$), $q_v \geq \sum_{c_i \in C_v} d_i$, π_v is a path $\langle n_{v_0}, n_{v_1}, \ldots, n_{v_j}, n_{v_{j+1}} \rangle$ ($C_v = \{c_{v_1}, \ldots, c_{v_j}\}$, $n_{v_0} = n_0$, $n_{v_{j+1}} = n_{k+1}$), and t_v, representing time of delivery, is defined on the set of nodes from π_v such that $t_v(n_{v_0}) \geq a_0$, $\max(a_{v_i}, t_v(n_{v_{i-1}}) + time(v_{i-1}, v_i)) \leq t_v(n_{v_i}) \leq b_{v_i}$ ($1 \leq i \leq k+1$). Solutions can be optimised, e.g., for the total traveled distance, or the total travel time, i.e., by minimising $\sum_{v \in V} \sum_{i=1}^{j+1} dist(v_{i-1}, v_i)$, or $\sum_{v \in V} t_v(n_{v_{j+1}})$, respectively.

For more details about VRPTW, the interested reader is referred to the literature [7].

3 Cloud Kitchen Platform

In a nutshell, *Cloud Kitchen* is a platform that aims at improving the efficiency of food delivery services in restaurants. Cloud Kitchen also includes a simulator that can simulate the process of food ordering and delivery based on historical data, which can be used to demonstrate the impact of the platform on the delivery business of restaurants. The architecture of the Cloud Kitchen framework is depicted in Figure 1. The Technology-Specific Bridge provides an interface to restaurants (the framework can serve multiple restaurants, each restaurant individually) or a simulator from which it receives real-time data (or historical data in the case of the simulator) about vehicles and customers. It provides decisions, on a real-time basis, that consist of information about which vehicle delivers food to what customer and in which order. TSB then forwards the information about new customer orders and about available and dispatched vehicles to the Decision-Making (DM) component that computes the routing plans.

3.1 Decision-Making Component

As mentioned above, the DM component deals with the problem of assigning orders to the vehicles as well as planning the delivery paths for them. Algorithm 1 summarizes the process. It initially monitors for customer orders that are almost ready (in the next five minutes), for orders that have been dispatched for delivery (i.e., vehicles with these orders have left the restaurant), and for vehicles that are about to return to the restaurant (in the next five minutes). Then, the set of customers C and the set of vehicles V are updated. According to the updated data about the customers and their locations, we update the graph $\mathcal{G}$, where $time$ and $dist$ functions (on the graph's edges) can be determined, for instance, by leveraging OSMNX, a library of OpenStreetMaps™ (OSM). Note that we do not explicitly consider vehicles' capacity since the amount of orders that are delivered never exceeds or even gets close to the capacity of a given vehicle. Then, we construct a VRPTW task $T = (V, C, \mathcal{G})$ while setting the time windows $[0, deadline(c)]$ for each $c \in C$, where $deadline(c)$ represents the deadline for the customer c. Deadlines are set by the restaurant (our customer) according to its processes. We then proceed to solve T, initially trying to meet all the deadlines and, if the task is not solved within a given time limit, we keep increasing the deadline for all customers by δ until the VRPTW task is solved. Such an approach was designed to guarantee that the delay is kept within reasonable bounds for each customer (in contrast to optimising average delay, which might impose an intolerably large delay for some customers).

The rationale behind considering orders that are not yet ready is twofold. Firstly, it is useful to adjust for aspects such as the time needed to physically group the orders and load them into a delivery vehicle or the later arrival of drivers. Secondly, locations of delivery for some later orders might be close to the delivery locations of earlier orders which might allow more efficient routing. The five-minute threshold is given by the restaurant (our customer) as it aligns with its internal processes. Decisions about when the vehicle is to be dispatched with the orders are made by the respective restaurant. Of course, by then all the orders have to be ready and the vehicle has to be at the restaurant.

3.2 Technology-Specific Bridge

TSB, as mentioned before, can be understood as an interface between the restaurants (or the simulator) and the DM component. To interpret (and verify) the solutions of the VRPTW tasks (returned by the DM component), we translate these solutions into sequential plans that contain actions, specified in PDDL, that are then forwarded to restaurants (or the simulator).

In particular, the PDDL model consists of six actions. Assigning customers' orders to the vehicles is done by actions assign-order, which assigns an order to a specific delivery, and assign-delivery,

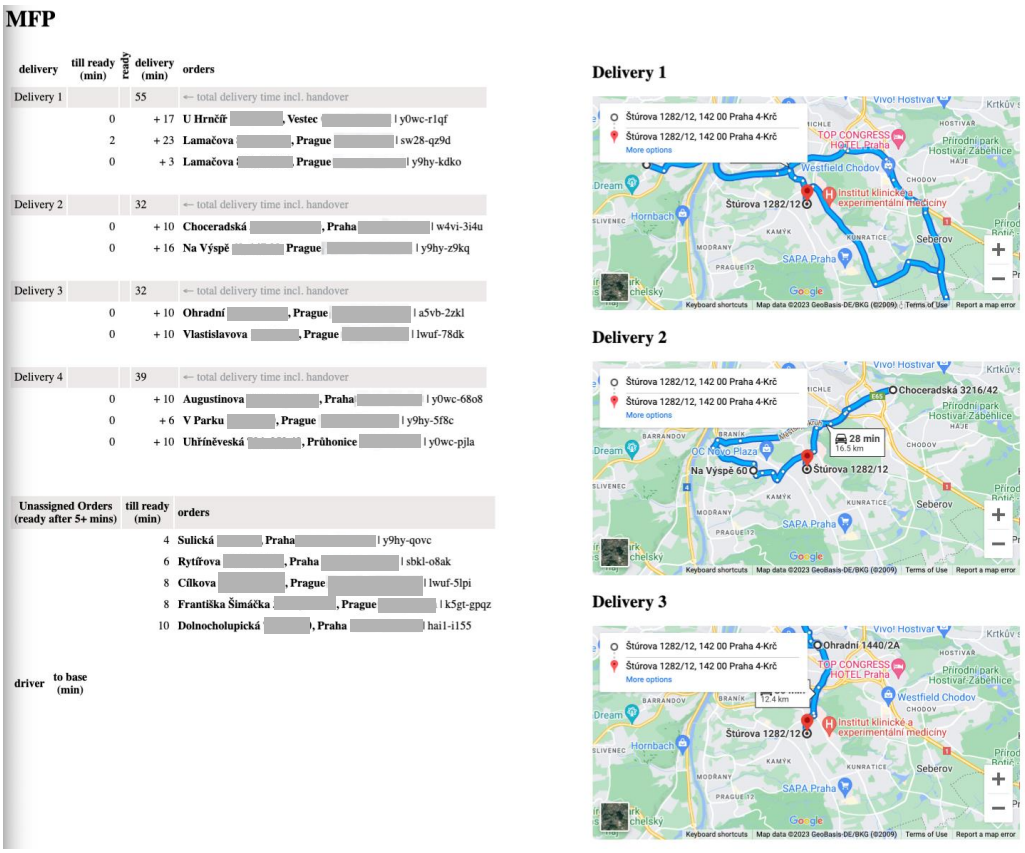

Figure 2. A screenshot of the recommender. On the left-hand side, the recommender shows the "grouped" deliveries (with not-yet-ready orders at the bottom). On the right-hand side, the recommender depicts the delivery routes in maps.

DT	DD	TD	PD	P10D
1.08	1.09	0.67	0.90	0.61

Table 1. Average improvement of Cloud Kitchen against historical data (measured by ratio to historical data). In particular, we consider total driven time (DT), total driven distance (DD), total delay (TD), and the number of orders delivered after the deadline (PD) or 10 minutes after the deadline (P10D).

which assigns a delivery to a specific vehicle. Note that these two actions correspond to a process in restaurants, where the finished orders are initially grouped before being loaded into a vehicle for delivery. The delivery process starts with the dispatch-delivery action, drive and deliver-order are for driving to the delivery location and handing the delivery to the customer, and the finish-delivery action represents that the driver returned to the restaurant. Again, these actions correspond to the processes that the restaurants implement for the delivery process.

These plans are then validated in UPF [9]. Then, the current plans are displayed in a *recommender*, which displays the groups of orders and visualises the delivery routes on a map (Google Maps™), so that the restaurant staff can make a decision about when the orders should be loaded to a vehicle and dispatched for delivery (a screenshot of the recommender is shown in Figure 2).

3.3 Simulator

The purpose of the simulator is to provide a realistic estimate of the impact of the decisions of the Cloud Kitchen platform made on historical data and visualize them to the potential customers (restaurants with food delivery). The simulator tracks the status of the orders, i.e., when the order is received by the restaurant, when it is cooked, then assigned for delivery, dispatched, en route, until it finally arrives to the customer who collects it. Similarly, we can track the state of each vehicle such as before delivering orders, the vehicle is ready (waiting at the restaurant), then the orders are assigned and loaded into it, and, after that, the vehicle goes to the customer to deliver the order, and after it delivers all the orders, it comes back to the restaurant.

Implementation-wise, the simulator is built on top of a process-based discrete event simulation framework named SimPy. The framework provides a linear, dimensionless timeline and allows for scheduling events to be executed at a specific time on this timeline. Events can then be processed, chained, or simply observed. One simulator tick represents one minute of real-time.

As mentioned above, driving distances and driving times are taken from OSM. To get a more realistic estimate of the actual driving time (in traffic conditions as they were in historical data), we have to determine a factor by which we multiply the estimated driving times from OSM. This is done by initially simulating the deliveries as they were arranged in the historical data and then by dividing driving times from the simulation by the driving time estimates from OSM.

Vehicle dispatching is automated such that if all orders in a batch (allocated to a single vehicle) are ready and there is some vehicle

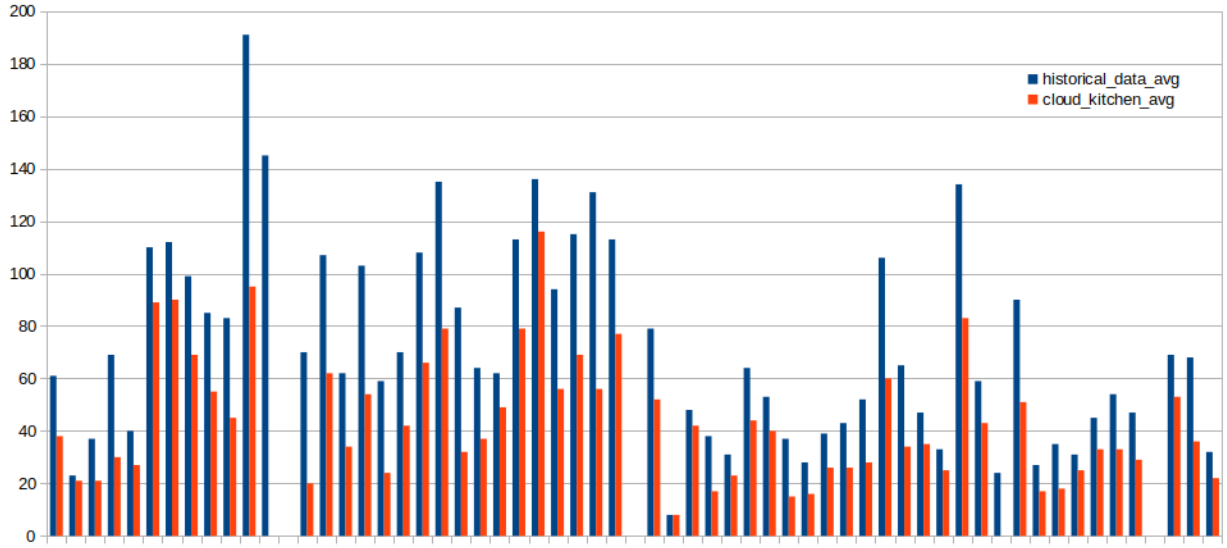

Figure 3. Comparison of the numbers of orders per day delivered later than 10 minutes (y-axis) in historical data and planned by the Cloud Kitchen platform. Particular days are on the x-axis.

waiting at the restaurant, then the vehicle is dispatched with that batch of orders.

The visual part contains an interactive map (based on Google Maps) consisting of pinpoints at the locations of customers that also indicate the time until the deadline and the current positions of the vehicles. On the right-hand side, there is a sidebar that shows the status of the orders.

4 Empirical Evaluation

To evaluate our Cloud Kitchen platform, we have simulated decisions on the platform on a real historical dataset consisting of one restaurant, nine vehicles, and more than 14000 customers (orders) spanning 61 days. As a solver for the VRPTW tasks we used Google OR Tools™, which we configured on the "path_cheapest_arc" option for the "first_solution_strategy". The reason for considering the "first_solution_strategy" is the need to operate near real-time. For that reason, we have also set the timeout for the solver to 50 milliseconds. Setting the "path_cheapest_arc" option was based on an empirical evaluation on a subset of our dataset. Also, we have found out that the VRPTW task has been solved within 30 milliseconds, or not at all (within 30 seconds), so, to give some leeway, we used the limit of 50 milliseconds per a VRPTW task. For the "deadline extension" loop (Lines 9–15 in Algorithm 1), we used a 1-second time limit (to keep near real-time reasoning).

As mentioned above driving distances and driving times are determined from OSM. To adjust for the conditions that were at the time of the dataset, we initially simulated the deliveries as they were arranged in the historical data and then we compared driving times (from the simulation) against the driving time estimates from OSM. From that comparison, we determined a factor (1.6666) by which we multiplied all driving time estimates from OSM to provide realistic estimates for our experiments. We would like to note that in real-world scenarios we use HERE maps to provide current driving time estimates for the recommender as HERE maps are more precise than OSM, however, the number of routing calls in the simulator is about three orders of magnitude higher than in the (real-world) recommender, which would make HERE maps usage in simulations computationally expensive.

The results, summarized in Table 1, show that while driving time and driving distance slightly increased (by 8% and 9%, respectively), the total delay has decreased by about one third, and the number of orders delivered later than 10 minutes after their deadline dropped by almost 40%. To give a better perspective, we provide, in Figure 3, day-by-day results comparing how many "later than 10 minutes after deadline" orders were in the historical dataset and were generated by the Cloud Kitchen platform, respectively. It can be seen that the numbers were not worse and often much better than in the historical dataset. There are two exceptions in days 12 and 50, where at least one planning episode failed (did not generate any routing plan within the time limit), and hence no results are shown in the simulator for that day as we could not validate the whole plan (for the day). In practice, if a planning episode fails, we can relax the deadlines to provide (at least) some recommendation for the restaurant.

From the results, we can see that the Cloud Kitchen platform has a strong potential to improve customer experience as it can considerably reduce the delay in order delivery. It has been confirmed by the restaurant (that provided us with the data) that orders delivered later than 10 minutes after their due time incur additional costs as the customers tend to ask for refunds and/or are less likely to order again. Despite the slight increase in costs (increased driving time and driving distance), minimising the delays beyond 10 minutes is hence much more beneficial for the restaurants.

5 Conclusion

In this paper, we have introduced the Cloud Kitchen platform that provides a decision-making tool for restaurants (or kitchens) with food delivery. In particular, the platform provides recommendations about which orders should be delivered together as well as providing routing for the vehicles (i.e., in which order the deliveries should be made). Such a problem can be modeled as VRPTW, which is tailored to address it. On the other hand, PDDL plans are better explainable to the restaurant staff who dispatch the orders as these plans are aligned

with the processes in the restaurant. Hence, we translate solutions of VRPTW tasks into PDDL plans and leverage UPF to validate the plans before they are displayed to the restaurant staff in the recommender. The Cloud Kitchen platform also contains a simulator that can simulate the decisions (made by the platform) on (real) historical data. The purpose of the simulator is to show the advantages of using the platform for potential customers (restaurants).

We have empirically evaluated our platform on a real historical dataset, where we have minimized the number of delayed deliveries and the total delay. Although, as we showed, such an optimization might increase driving time and distance and thus incur additional costs, the number of delayed orders, especially those delayed by more than 10 minutes, dropped considerably. Hence, the platform has the potential to improve customer satisfaction considerably, which has more value to the businesses despite the (slightly) increased costs of delivery (as confirmed by the restaurant).

In the future, we plan to extend our platform by considering different types of vehicles (e.g., scooters), distinguishing between different types of food (e.g., warm or cold food), and optimizing the process of preparing food in the kitchens.

Acknowledgements

The work is co-funded by the AIPlan4EU project, which is funded by the European Commission – H2020 research and innovation programme under grant agreement No 101016442

References

[1] J. Cai, Q. Zhu, Q. Lin, L. Ma, J. Li, and Z. Ming. A survey of dynamic pickup and delivery problems. *Neurocomputing*, 554:126631, 2023. doi: 10.1016/J.NEUCOM.2023.126631. URL https://doi.org/10.1016/j.neucom.2023.126631.

[2] L. T. Chai and D. N. C. Yat. Online food delivery services: Making food delivery the new normal. *Journal of Marketing advances and Practices*, 1(1):62–77, 2019.

[3] M. Fox and D. Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR)*, 20:61–124, 2003.

[4] M. Fox, D. Long, and D. Magazzeni. Explainable planning. *arXiv preprint arXiv:1709.10256*, 2017.

[5] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[6] P. Gregory, D. Long, M. Fox, and J. C. Beck. Planning modulo theories: Extending the planning paradigm. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*, 2012.

[7] B. Kallehauge, J. Larsen, O. B. Madsen, and M. M. Solomon. *Vehicle routing problem with time windows*. Springer, 2005.

[8] L. Martie, J. Rosenberg, V. Demers, G. Zhang, O. Bhardwaj, J. Henning, A. Prasad, M. Stallone, J. Y. Lee, L. Yip, et al. Rapid development of compositional ai. *arXiv preprint arXiv:2302.05941*, 2023.

[9] A. Rovetta, A. Trapasso, A. Valentini, A. Micheli, A. Bit-Monnot, E. Tosello, E. Scala, F. Chiariello, G. Röger, I. Serina, J. Rothe, L. Framba, L. Bonassi, R. Godet, S. H. S. S. Sadanandam, S. Goyal, and U. Köckemann. Unified planning framework. https://github.com/aiplan4eu/unified-planning, 2023. "Accessed: 2023-12-03".

[10] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transp. Sci.*, 29(1):17–29, 1995. doi: 10.1287/TRSC.29.1.17. URL https://doi.org/10.1287/trsc.29.1.17.

[11] A. Shankar, C. Jebarajakirthy, P. Nayal, H. I. Maseeh, A. Kumar, and A. Sivapalan. Online food delivery: A systematic synthesis of literature and a framework development. *International Journal of Hospitality Management*, 104:103240, 2022.

[12] Z. Steever, M. Karwan, and C. Murray. Dynamic courier routing for a food delivery service. *Computers & Operations Research*, 107: 173–188, 2019. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2019.03.008. URL https://www.sciencedirect.com/science/article/pii/S0305054819300681.