On-Demand Grocery Delivery from Multiple Local Stores with Autonomous Robots

Kronmueller, Maximilian; Fielbaum, Andres; Alonso-Mora, Javier

DOI
10.1109/MRS50823.2021.9620599

Publication date
2021

Document Version
Final published version

Published in
Proceedings of the International Symposium on Multi-Robot and Multi-Agent Systems, MRS 2021

2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)
November 4-5, University of Cambridge, United Kingdom

On-demand Grocery Delivery From Multiple Local Stores With Autonomous Robots

Maximilian Kronmueller[1], Andres Fielbaum[1] and Javier Alonso-Mora[1]

Abstract— **The advances in the area of autonomous delivery robots combined with customers' desire for fast delivery, bare potential for same-day delivery operations, specifically with small time windows between ordering and delivery. Most same-day deliveries are operated using a single depot and with vehicles' routes planned and fixed when leaving the depot. In this paper, we relax these two assumptions and focus on on-demand grocery delivery using a fleet of autonomous vehicles or robots. The problem features the opportunity to pick up goods at multiple local stores or depots, for example, supermarkets within the city, and allows robots to perform depot returns prior to being empty, if beneficial. This allows for more agile planning and on average shorter distance to the next depot. We propose a novel dynamic method for the same-day delivery problem, where we aim to deliver orders as fast as possible, minimally within the same day. In each time step (every few seconds or minutes) the following is executed: For each order potential pick-up locations are identified and feasible trips, i.e., sequences to pick up goods and deliver orders, are calculated. To assign trips to robots an integer-linear program is solved. We simulate one day of service in a city under different conditions with up to 30 autonomous robots, 30 depots and 10,500 orders. Results underpin the advantages of the proposed method and show its versatility with respect to different situations.**

I. INTRODUCTION

The development of autonomous delivery vehicles and robots has made big steps forward during the last years. Many different robots are under development, tested and deployed by different companies and universities, like Nuro, Amazon, Starship Technologies or Postmates. The possibility to order and have one's parcel delivered within the next hours, minimally within the same day, is appreciated by many customers. During the last years many young companies started to rise, offering grocery delivery in minutes, such as Gorillas, Flink, Getir or GoPuff. As soon as fleets of autonomous robots are scaled to full operations, the planning and routing algorithm in charge of computing robots' plans is of major importance. Combining these two developments bares huge potential.

This leads to planning for same-day delivery (SDD) operations. Most of them are operated using a single depot and with vehicles' routes planned and fixed when leaving the depot. In this paper, we relax these two assumptions and focus on on-demand grocery delivery using a fleet of autonomous vehicles or robots. SDD can generally be described as follows: Orders are placed continuously throughout the day and need to be delivered before the end of the day. To achieve the latter, parcels need to be picked up and delivered to customers' locations leveraging a fleet of robots. Each robot has a maximum capacity. For each robot, a route needs to be found such that a given objective function is minimized or maximized, for example maximizing the number of delivered packages or minimizing the sum of delays. In this paper, we extend on a SDD problem and aim to deliver orders as fast as possible, minimally within the same day. Further, we increase options to pick-up orders from a single depot to multiple.Moreover, most SDD operations assume that their robots deliver all loaded orders, afterward returning to the depot, and then become available to load and service new customers. We relax this assumption and allow for depot returns prior to being empty (pre-empty depot returns). This allows to flexibly incorporate new occurring orders within existing plans of robots. We show that those additions can lead to more efficient routes and cheaper operations.

We approach the above-described problem in a rolling fashion, i.e., we divide the full-day problem into multiple subsequent sub-problems. The problem evolves with time as new requests arrive throughout the day. Further, the operation is planned (solving sub-problems) and executed (following the obtained routes) simultaneously, changing the load and position of robots constantly. To solve a single sub-problem at time t, we first identify potential pick-up locations for each order. Second, potential feasible trips, i.e., sequences to pick up goods and deliver orders, are calculated. To assign these trips to robots an integer-linear program is solved. As a result, each robot has a plan to follow, i.e., which orders to pick-up and where, as well as in which succession to deliver them.

The remainder of this paper is structured as follows: First, the related literature and the contribution of this work (Section II) are presented. Afterwards, the problem (Section III) and our proposed method (Section IV) to solve it are explained in detail. Section V presents results obtained by running an extensive computational analysis simulating one day of service. The paper is concluded with Section VI.

II. RELATED WORK

The same-day delivery problem can be categorized as a dynamic and stochastic pick-up and delivery problem with incomplete information. Thus falling into the family of dynamic vehicle routing problems (DVRP), a comprehensive overview can be found in [1] and [2]. [3] provides an overview focusing specifically on dynamic pick-up and

[1] Department of Cognitive Robotics at the Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Mekelweg 5, 2628 CD Delft, The Netherlands

2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS) | 978-1-6654-2926-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/MRS50823.2021.9620599

delivery problems. Additional related problems are the meal delivery routing problem [4]–[6], vehicle routing to transport people (dial-a-ride problem) [7], [8] and the multi-robot task assignment problem [9].

The SDD literature splits in two: First, work focusing on robot dispatching or order acceptance followed by a separate routing step e.g. [10]–[15] and second, work focusing on SDD routing directly. [16] tackles the SDD problem using a multi-scenario sampling approach leveraging waiting strategies and apply it to scenarios with up to 800 requests and up to 13 vehicles. In contrast, our approach can handle larger problem sizes and allows to pick up orders at multiple depots, but works myopic. Similar to our work, [17] allows for preemptive depot returns, i.e. depot returns before finishing the currently planned tour based on expectations of future events. The authors proposed a method building on approximate dynamic programming combined with an insertion routing heuristic. Both methods allow vehicles to return to depots before finishing their current routes, but our approach differs on the point that pre-empty depot returns do not use anticipation of the unknown future but only use currently available information. The method proposed in [17] is able to plan for a single vehicle, whereas our approach scales up to large fleet sizes.

The multi-depot vehicle routing problem (MDVRP) is a VRP featuring multiple depots and thereby introducing a choice where goods are picked up. The combination of DVRP and MDVRP is called the dynamic multi-depot vehicle routing problem (DMDVRP). [18] and [19] decompose the DMDVRP into multiple DVRPs by assigning each order to one fixed depot. In contrast, we include the decision of which depot should be used within the routing decision itself.

The routing method proposed in this work is based on methods for ride-sharing. An overview of these methods can be found in [20]–[22]. More specifically, our proposed method builds upon a routing method for transporting people in metropolitan areas [7]. Our problem mainly differs in two aspects. The pick-up locations of orders are not pre-defined through themselves (nor optimized in a close area around them, as in [23]) but limited to a smaller set in the whole service area, the depots. Second, the urgency in serving a request right away is lower for delivering goods than for transporting people.

In this paper, we propose an optimization method to route a fleet of autonomous robots performing on-demand grocery delivery (same-day delivery operations), whose main virtues are threefold. First, we consider multiple depots at which orders can be picked up, and it is the method that decides which is the optimal one for each order. Second, we admit robots to visit a depot to receive more parcels before distributing their loads, when this increases efficiency. Finally, our method is able to scale up to scenarios with thousands of requests and to find good quality solutions online.

III. PROBLEM FORMULATION

A. Definitions

Environment: Let $G = (N, A)$ be a weighted directed graph where N defines a set of nodes and A defines a set of weighted arcs. The arcs' weights represent the traveling times between two connected nodes. The travel time between two locations $x_1, x_2 \in N$ is given by the function $travel(x_1, x_2)$. The travel time equals the sum of the weights of all arcs that need to be traversed following the shortest path between the two locations. A depot $d \in \mathcal{D}$ is a specific node where goods can be picked up. All $\mathcal{H}$ depots are summarized in the set of depots $\mathcal{D} \subset N$. We assume that every depot has all goods on stock.

Robot Fleet: Robots can drive along the graph's arcs to load and deliver goods to customers. The amount of orders they can load is restricted through a maximum capacity C. The autonomous robot fleet $\mathcal{R}$ consists of $\mathcal{M}$ identical robots. At each time t, a single robot $r \in \mathcal{R}$ is fully described by its current location $l_{r,t}$ and the orders it has loaded (picked-up and not yet dropped-off), summarized in the set $\mathcal{LO}_{r,t}$.

Demand: Each order $o = (t_o, g_o) \in \mathcal{O}$, where $\mathcal{O}$ is the demand set, is revealed at time $t_o \in [0, T_{end} - \delta_T]$, with T_{end} the end of the day and δ_T a constant time span, in which no more orders are placed, and requires that its goods are delivered to its specified destination $g_o \in N$. A total of $\mathcal{N}$ orders are placed. For simplicity, we do not look at individual goods contained in the order but assume that all orders are of the same size, set to one. This assumption can easily be extended to variable order sizes. Note that an order itself does not define a pick-up location $p_o \in N$. The demand set $\mathcal{O}$, can be split into subsets depending on the status of each order $o \in \mathcal{O}$. The status of an order depends on the current time t and thus also the subsets, which we define as follows.

- $\mathcal{UO}_t$ is the set of all unknown orders, consisting of all orders $o \in \mathcal{O}$ with $t < t_o$.
- The set $\mathcal{PO}_t$ consist of all orders $o \in \mathcal{O}$ that are known $t \geq t_o$ but are not yet picked-up by any robot $r \in \mathcal{R}$.
- The set $\mathcal{LO}_t$ are all orders $o \in \mathcal{O}$ that are currently loaded to a robot $r \in \mathcal{R}$, i.e., $\mathcal{LO}_t = \cup_{r \in \mathcal{R}} \mathcal{LO}_{r,t}$.
- The set $\mathcal{DO}_t$ are all orders $o \in \mathcal{O}$ that already got delivered to their destinations g_o.
- The set $\mathcal{IO}_t$ summarizes all orders that can not be delivered within given constraints anymore.

The subsets are defined such that each order only belongs to one subset at time t, thus they are disjoint, $\mathcal{O} = \mathcal{UO}_t \cup \mathcal{PO}_t \cup \mathcal{LO}_t \cup \mathcal{DO}_t \cup \mathcal{IO}_t$. At $t = 0$, the beginning of the day, all orders are unknown $\mathcal{UO}_0 = \mathcal{O}$. At $t = T_{end}$, the end of the day, all orders are either delivered or ignored $\mathcal{DO}_{T_{end}} \cup \mathcal{IO}_{T_{end}} = \mathcal{O}$ and $\mathcal{UO}_{T_{end}} = \mathcal{PO}_{T_{end}} = \mathcal{LO}_{T_{end}} = \emptyset$.

Times: We assume that a robot needs some constant time to load or deliver a single order, δ_{load} and $\delta_{service}$ respectively. The time at which an order $o \in \mathcal{O}$ is picked up by a robot $r \in \mathcal{R}$ is $t_{pick,o}$ and the time it is delivered to its destination g_o is $t_{drop,o}$. The earliest time an order can be delivered is described by $t_{ideal,o}$. To do so, an

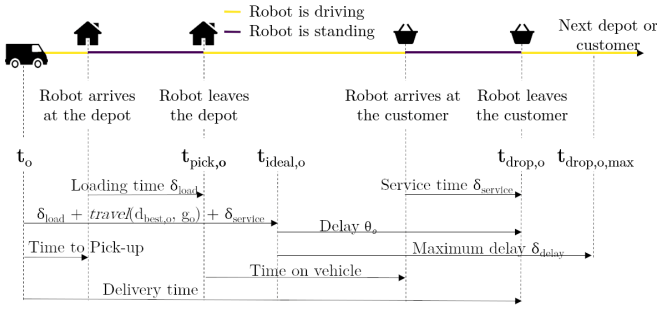

Fig. 1: Visualization of the different times and time spans for one order.

idle robot needs to be located at the closest depot to the order's destination, $d_{\text{best},o}$, and start serving the customer immediately without any detours, resulting in $t_{ideal,o} = t_o + \delta_{load} + travel(d_{\text{best},o}, g_o) + \delta_{service}$. Thus, the lower bound of the drop off time is $t_{ideal,o}$, i.e., $t_{drop,o} \geq t_{ideal,o}$. We define the difference between the optimal and actual delivery time as delay $\theta_o = t_{drop,o} - t_{ideal,o} \geq 0$. Further, we assign each order a maximal drop off time $t_{drop,o,max} = t_{ideal,o} + \delta_{delay}$, with δ_{delay} the maximally allowed delay per order. δ_{delay} is predefined by the operator to ensure a desired service level. A summary of all involved points in time for one order is illustrated in Figure 1.

B. Problem

Consider a directed graph $G = (N, A)$, a set of depots $\mathcal{D}$, where orders can be picked up, and a fleet of autonomous robots $\mathcal{R}$ in an initial state (locations of each robot $l_{r,0}$ and robot's load $\mathcal{LO}_{r,0}$). The robot fleet $\mathcal{R}$ needs to serve a demand $\mathcal{O}$. The operation starts at $t = 0$ and ends at $t = T_{end}$, in this time span each orders $o \in \mathcal{O}$ is revealed at $t = t_o \in [0, T_{end} - \delta_T]$. Find an assignment Ω of orders $o \in \mathcal{O}$ to robots $r \in \mathcal{R}$, including the choice of a depot $d \in \mathcal{D}$ to pick-up the order and the routes the robots take, to minimize a given cost function $\mathcal{J}$ subject to a set of given constraints.

Ω represents the routes of all robots, which are defined through the orders to serve, where to pick them up, and in which sequence the individual orders are carried out. The set of constraints we consider are:

- Each robot has a maximum capacity of C
- Each order is allowed to have a maximum delay of δ_{delay} and needs to be delivered before the end of the operation $t = T_{end}$

In our case, we assume that at the beginning of the day ($t = 0$) all robots $r \in \mathcal{R}$ are distributed over all depots $d \in \mathcal{D}$ and are empty $\mathcal{LO}_{r,0} = \emptyset \ \forall r \in \mathcal{R}$. Situations in which it might be unfeasible to serve all orders without violating any constraint can occur. Orders that can not be delivered within the given constraints will not be served. We define the cost function $\mathcal{J}$ as follows, taking two opposing interests into account: Operators' cost and customers' cost, the latter measuring the service quality. The customer's cost of an order $o \in \mathcal{O}$ is defined as its delay θ_o, thus the faster an order is delivered the better. The operator's costs are defined as the sum of the traveling time of each robot tt_r to serve all orders assigned to it, expressed mathematically as $\sum_{r \in \mathcal{R}} tt_r$. The two costs are combined in a convex fashion, using a weight β. If an order is not served a penalty α is charged. If α is set considerably larger than the sum of the other two cost terms the system aims at maximizing the number of orders that are served, followed by minimizing the combination of operators' cost and customers' cost. In this work, we set α as a sufficiently large constant, as we consider the urgency to deliver orders in SDD less than delivering as many orders as possible.

$$\mathcal{J}_{T_{end}} = \left[(1-\beta) \cdot \sum_{o \in \mathcal{DO}_{T_{end}}} \theta_o + \beta \cdot \sum_{r \in \mathcal{R}} tt_r + \sum_{o \in \mathcal{IO}_{T_{end}}} \alpha \right] \tag{1}$$

Equation 1 represents the overall objective function at $t = T_{end}$. The penalty α can be interpreted as the cost the operator has to cover if a third party, to deliver the respective order, is hired. Generally, the cost function and constraints are universal and could be changed to fit other requirements. As a result our problem combines several NP-hard problems, including the capacitated vehicle routing problem [24], the multi-depot vehicle routing problem [25] and dynamic optimization, while looking at large fleet sizes.

IV. METHOD

A. Overview

We tackle the introduced problem in a rolling fashion, meaning that a state of the problem is solved, then time is propagated forward (robots follow their plans and new orders emerge) until a next state to solve occurs. This happens repeatedly until the end of the day. We propagate time in fixed time steps of duration Δt. To simplify notation, we enumerate all decisions consecutively by k. The time at assignment k is $t_k = k \cdot \Delta t$. This results in $\mathcal{K} = T_{end}/\Delta t$ decisions from start to end of the operation. We need to describe the state of the problem at any time t. The state $\mathcal{S}_t$ at time t, is fully characterized by the time itself, the robots fleet state $\mathcal{R}_t$, characterized by all the robots' states at t ($(l_{r,t}, \mathcal{LO}_{r,t}) \forall r \in \mathcal{R}$), and the set of known but not yet loaded orders $\mathcal{PO}_t$ at t. This results in the state definition as

$$\mathcal{S}_t = (t, \mathcal{R}_t, \mathcal{PO}_t).$$

A single decision is the assignment of currently open orders to robots, considering the corresponding state $\mathcal{S}$, and optimizing the assignment according to the objective function. Solving this assignment $\mathcal{K}$ times consecutively gives a solution for the overall problem formulated in Section III. Our approach is myopic, i.e., not taking future states into account. We regard this assumption as reasonable, as the time passed between updating the problem and robots' routes is rather short. To solve one specific state of the problem, we propose a method divided into four steps: First, potential pick-up locations for each order are found. Second, orders with associated pick-up locations are grouped into potential

trips, taking the current location of each robot into account. A trip is an ordered sequence of locations to pick up and deliver orders executed by a robot. Here all possible trips for each robot are calculated if enough computational time is given. Third, these potential trips are assigned to specific robots. Last, robots execute part of their plans as time is propagated forward. An overview of the approach is depicted in Figure 2.

B. Pick-up Locations

Each individual order $o \in \mathcal{O}$ needs to be assigned to a specific pick-up location $p_o \in \mathcal{D}$. Each depot $d \in \mathcal{D}$ which is close enough to the order's destination $g_o \in N$ such that a robot can go to the depot and deliver before the maximum delay has passed, is a feasible option. One of these options needs to be selected. We introduce and define the term candidate c_o of an order $o \in \mathcal{O}$ as follows.

Definition: A candidate c_o belonging to order $o \in \mathcal{O}$ is a tuple of the order $o \in \mathcal{O}$ itself and an associated pick-up location $p_{c_o} \in \mathcal{D}$. Thus a candidate is described as $c_o = (o, p_{c_o})$.

A candidate c_o is unique, but one order $o \in \mathcal{O}$ can have multiple candidates associated with it, each having a different pick-up location $p_{c_o} \in \mathcal{D}$. $\mathcal{I}_o^{\mathcal{C}}$ denotes the set of candidates that belong to order $o \in \mathcal{O}$. The set of all candidates is denoted by $\mathcal{C}$. $\mathcal{C}_t$ is the set of candidates at time t corresponding to all placed orders $o \in \mathcal{PO}_t$. We introduce a heuristic to select a subset of pick-up locations. We consider the x depots closest to the destination measured in travel time. The parameter x can be tuned. This results in maximally x candidates per placed order. If $x = \mathcal{H}$ all depots in reach (not violating any constraint) are considered and if $x = 1$ the closest depot is used for each order. We do so to control the number of candidates per order and thus the number of potential trips for each vehicle, which is directly correlated to the required computational effort.

C. Trip Generation

A trip T is an ordered sequence of locations to pick up and deliver orders executed by a robot. To calculate potential feasible trips for a robot, we look at sets of candidates separately for each robot. A trip's size l, measured as the number of considered candidates, is thereby step-wise increased starting at a size of one until a maximum size η is reached. The operator sets η, additionally huge trips are prevented by having a latest drop-off time for each order $t_{drop,o,max}$. The result of this step is a set of potential trips for each robot. The trip definition contains a set of candidates $c_o \in T$, which are delivered in the trips route, and thus also a set of orders $o \in T$. The algorithm to calculate all trips at time t, the set of all feasible trips $\mathcal{T}_t$, is shown in Algorithm 1. In Algorithm 1 we use three functions: $CandidateRobot()$, $TwoCandidates()$ and $bestRoute()$. $CandidateRobot()$ is valid if a specific robot can serve a specific candidate in the current state of the problem without violating any constraint. $TwoCandidates()$ checks if two candidates are combinable, i.e., whether they can be served by a hypothetical robot located at a depot satisfying all the constraints. As multiple candidates per order exist, we add an additional constraint to the existing time and capacity constraints. For two candidates to be combinable into one trip we require them to share their pick-up location. This implies that two candidates do not belong to the same order. The step of finding the sequence to visit all locations for a given set of candidates (of one trip), is covered by the function $bestRoute()$. Algorithm 1 checks whether $bestRoute()$ is valid, which is the case if a single trip for robot r delivers all candidates and doesn't violate any constraint. If $bestRoute()$ is valid, it returns a trip T_ℓ. Most likely, there will be more than a single feasible route to visit all locations of the trip. The function $bestRoute()$, chooses the route that minimizes the given cost function for the trip. The cost for visiting a sequence of locations (trip $T \in \mathcal{T}_t$) executed by the associated robot $r \in \mathcal{R}$ is given by $\gamma_{T,r}$ and is derived from Equation 1;

$$\gamma_{T,r} := (1-\beta) \cdot \sum_{o \in T} \theta_o + \beta \cdot travel(T) \tag{2}$$

Algorithm 1: Trip Generation for decision k at t_k

```
input : S_{t_k}, C_{t_k}, η
output: All feasible trips T_{t_k}
begin
    T_{t_k} = ∅ ;
    foreach r ∈ R do
        T_ℓ = ∅   ∀ℓ ∈ {1, ..., η} (Set of all trips of size ℓ);
        [add trips of size 1]
        foreach c ∈ C_{t_k} do
            if CandidateRobot(r, c) valid then
                T_1 ← T_1 ∪ (c)(Add trip to set of trips)
            end
        end
        [add trips of size 2]
        foreach (c_i), (c_j) ∈ T_1 do
            if TwoCandidates(c_i, c_j) valid and
              bestRoute(r, c_i, c_j) valid then
                T_2 ← T_2 ∪ (c_i, c_j)
            end
        end
        [add trips of size ℓ]
        for ℓ ∈ {3, ..., η} do
            foreach T_i, T_j ∈ T_{ℓ-1} with |T_i ∪ T_j| = ℓ
              (Two trips with ℓ candidates combined) do
                if ∀h ∈ {1, ..., ℓ}, {c_1, ..., c_ℓ} \ c_h ∈
                  T_{ℓ-1} then
                    if bestRoute(r, T_i ∪ T_j) valid then
                        T_ℓ ← T_ℓ ∪ (T_i ∪ T_j);
                    end
                end
            end
        end
    end
    return T_{t_k} ← ∪_{ℓ∈{1,...,η}} T_ℓ
end
```

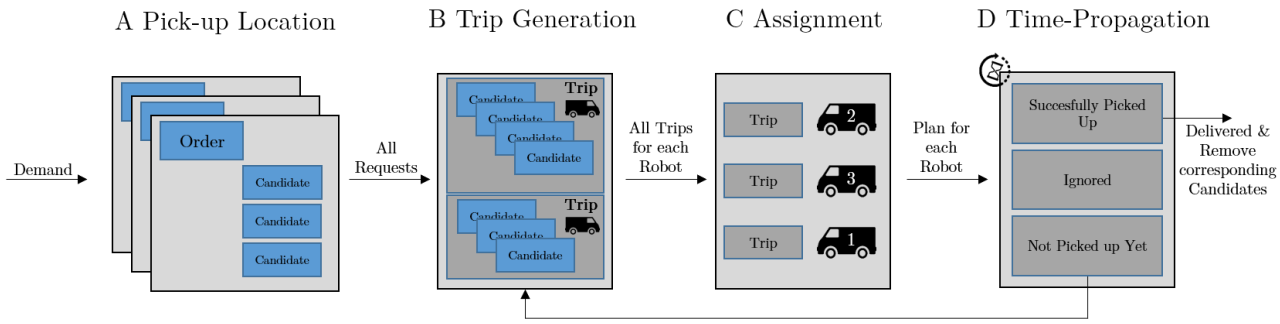

Fig. 2: Schematic overview of our solution approach. Step A assigns a number of potential pick-up locations to each order. During step B individual candidates (combinations of orders to specific pick-up locations) are combined to feasible trips. Step C performs an assignment of trips and individual robots. Within step D we propagate time and robots follow their assigned plans.

where $travel(T)$ represents the total driven distance to complete trip T. To come up with the different evaluated sequences we perform an exhaustive search. For large group sizes a routing heuristic, for example an insertion heuristic, could be used. For robots that already contain prior load, trips include those loaded orders. The sequence according to which the combination of prior loaded orders and new ones are served is thereby unconstrained. Herein the possibility of pre-empty depot returns occurs. We only keep the route (sequence of visited locations) that minimizes the costs (2) of the trip for a specific robot and a set of candidates. Taking the minimal cost route is included in the subsequent notation of a trip T. Further, calculations for one robot are stopped if a predefined time, ρ_{max}, has passed. In this case, the trips generated up to this point are considered.

D. Assignment

After calculating potential feasible trips, summarized in $\mathcal{T}_t$, we need to decide which of them should be carried out and by which robot to minimize the overall objective function. We call this step the assignment. This assignment is formulated as an integer linear program (ILP), which is solved incrementally. First, it is initialized by a greedy solution. The greedy solution selects trips decreasingly by their number of served candidates l and in case they serve the same amount, increasingly by their associated costs. The ILP is presented in Algorithm 2. Thereby, $\epsilon_{\mathcal{TR}}$ denotes the set of all feasible trip robot combinations, and $\epsilon_{T,r}$ is the corresponding binary variable, taking the value 1, if the combination is feasible. Further, we create the following sets: $\mathcal{I}_r^T$, the set of trips that can be serviced by a robot $r \in \mathcal{R}$; $\mathcal{I}_{c_o}^T$, the set of trips that contain candidate c_o; $\mathcal{I}_T^{\mathcal{R}}$, the set of robots that can service trip T; $\mathcal{I}_o^{\mathcal{C}}$, the set of candidates that belong to order o. Further, χ_{c_o} is a binary variable, taking the value of one if the corresponding candidate is ignored, introduced for each candidate $c_o \in \mathcal{C}$ and $\mathcal{X}$ is a set of all variables $\mathcal{X} = \{\epsilon_{T,r}, \chi_{c_o}; \forall \epsilon_{\mathcal{TR}} \text{ and } \forall c_o \in \mathcal{C}\}$.
Equation 3 describes the objective function for a single state. Note that the considered costs are relative. From the costs of a robot's route $\gamma_{T,r}$ (see Equation 2), the costs for the considered robot to serve its already loaded parcels are subtracted, $\gamma_{loaded,r}$. Thus, we only account for changes in the robot's plan. If a robot's plan is not changed, by not assigning any new orders, no costs are posed in the assignment. Note that this does not affect the optimization according to the global objective function. Equation 4 ensures that each robot is at most assigned to one trip. Equation 5 ensures that each order is assigned to a single robot or is rejected and the penalty α is charged. Furthermore, it ensures that only one candidate belonging to the same order is chosen. Equation 6 ensures that χ_{c_o} is binary. χ_{c_o} takes the value one if its associated order $o \in \mathcal{O}$ can not be served by any robot or is ignored. Equation 7 defines $\epsilon_{T,r}$ as binary. As a result, each robot is assigned to a trip. If a robot receives no new orders, it will follow its current plan of delivering the currently loaded parcels or it will be considered idle if it has none.

Algorithm 2: Assignment

input : Greedy assignment of trips to robots Ω_{greedy}
output: Assignment of trips to robots Ω_{optim}
begin
Initalize with Ω_{greedy} ;
Solve;

$$\Omega_{\text{optim}} = \text{argmin}_{\chi} \sum_{T,r \in \epsilon_{\mathcal{TR}}} (\gamma_{T,r} - \gamma_{loaded,r})\epsilon_{T,r} + \sum_{c_o \in \{1,\dots,|\mathcal{C}_t|\}} \alpha \chi_{c_o} \tag{3}$$

$$\sum_{T \in \mathcal{I}_r^T} \epsilon_{T,r} \leq 1 \qquad \forall r \in \mathcal{R} \tag{4}$$

$$\sum_{c_o \in \mathcal{I}_o^{\mathcal{C}}} \sum_{T \in \mathcal{I}_{c_o}^T} \sum_{r \in \mathcal{I}_T^{\mathcal{R}}} \epsilon_{T,r} + \chi_{c_o} = 1 \quad \forall o \in \mathcal{O} \tag{5}$$

$$\chi_{c_o} \in \{0,1\} \tag{6}$$

$$\epsilon_{T,r} \in \{0,1\} \tag{7}$$

return Ω_{optim}
end

E. Return of Idle Robots

If some robots are considered idle after an assignment has been done, we instruct them to move towards the closest depot from their current location. Nevertheless, they might still be assigned otherwise, before reaching the depot.

F. Time-Propagation

In this step, we propagate the time and all elements affected by it, until the next decision $k+1$ is triggered, $t_{k+1} = t_k + \Delta t$. During this time each robot follows its plan determined in the previous steps (the trip assigned to it). Each order can take one of the following five states:

- An order can be **picked up** by a robot at a depot - As soon as an order is picked up it is fixed to this robot, it can not be unloaded anymore, meaning that it can not be reassigned to any other robot. Additionally, as multiple candidates belonging to one order exist, but only one of them gets served, the other candidates belonging to this order are removed.
- An order can be **delivered** to its destination.
- An order can be assigned to a trip, but the planned pick-up time is later than t_{k+1}, the time of the next decision, thus we consider the order as **not picked up**, yet. All not picked up candidates are reinserted into the trip generation step for the next decision, thus allowing for reassignment. Note that such reassignments are not possible if robots cannot adapt their routes prior to being empty.
- An order can be not assigned to any trip. This order is reinserted into the trip generation step for the next decision.
- An order can be **ignored**, meaning that it can not be picked up such that all constraints can be satisfied within the current plan. This is the case if an order $o \in \mathcal{O}$ can't be delivered until the latest drop-off time $t_{drop,o,max} = t_{ideal,o} + \delta_{delay}$.

$t_{drop,o,max}$ is mainly influenced by the value of δ_{delay}. The smaller δ_{delay} is set, the harder it is to combine multiple candidates to be served by one robot. On the other hand, if δ_{delay} is set too large, the number of possible combinations becomes vast, which can hinder solving the problem in the first place. A good balance has to be found by the operator. We distinguish between $\delta_{\text{delay,real}}$, defined by the service level and $\delta_{\text{delay,heuristic}}$, the maximum delay at which the method performs well. In case of $\delta_{\text{delay,heuristic}} < \delta_{\text{delay,real}}$ we allow a candidate to be reinserted into the problem after it has been ignored. The candidate gets reinserted with a new release time of t, the current time. Each candidate can be ignored up to a limit of ζ times,

$$\zeta = (\delta_{\text{delay,real}} - (\delta_{\text{delay,real}} \mod \delta_{\text{delay,heuristic}}))/\delta_{\text{delay,heuristic}}.$$

If a candidate got ignored ζ times it is removed from the problem. It is noteworthy that if an order gets not assigned to a robot it is not necessarily ignored.

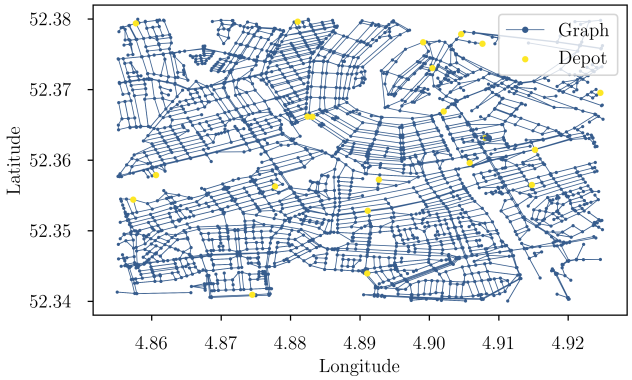

Fig. 3: A visual representation of the underlying graph $G = (N, A)$ is shown. The set $\mathcal{D}$ of 20 depots is highlighted in yellow.

V. RESULTS

In this section we first analyze one simulation run, representing a day of on-demand grocery delivery in Amsterdam, in detail (Section V-A). To investigate the benefits of the proposed method, we evaluate the approach by comparing it with a scenario that uses a single depot and a scenario that does not allow for pre-empty depot returns, (Section V-B). In Section V-C we present the results of a sensitivity analysis of the main parameters.

A. Base Scenario: One Scenario in Detail

To analyze the proposed algorithm we simulate a potential day in a city center. We represent the city as a directed graph containing 2706 nodes and 5632 edges, shown in Figure 3. The travel times between nodes are calculated as their distance divided by a constant speed of $36\frac{\text{km}}{\text{h}}$. The simulated demand pattern features 10,000 orders, homogeneously distributed in space and covers a time span from 8 a.m. to 9 p.m., including a noon and a stronger evening peak. The last 10 minutes, prior the end of the day T_{end}, no more orders are placed, $\delta_T = 10$. 20 depots to pick up orders are distributed over the whole service area following a greedy k-center algorithm. 30 autonomous robots with a maximum capacity of ten are used. The maximum trip size η equals the capacity of the robots. A maximum delay, $\delta_{\text{delay,real}}$, of 24 minutes is allowed and $\delta_{\text{delay,heuristic}}$ is chosen as eight minutes, resulting in a ζ of three. Per order, the five closest depots to the final destination, ($x = 5$), are considered. To load and service an order, we assume $\delta_{load} = 15\,\text{sec}$, implying that all orders are prepared and need to be loaded only, and $\delta_{service} = 30\,\text{sec}$, assuming that all customers are ready to grab their groceries at the front door. The algorithm runs in time spans Δt of 100 seconds. The penalty to ignore an order is set to equal 10^4 seconds. We weighted the two different objectives equally with $\beta = 0.5$. These values have been chosen to reach a service rate close to 100%, while limiting resources such that the system is forced to work as efficient as possible. To solve the ILP, described in Algorithm 2, we leverage Mosek (7.1) [26]. We run the solver to optimality (around 75% of the times), with a maximum time budget of 50 seconds. To

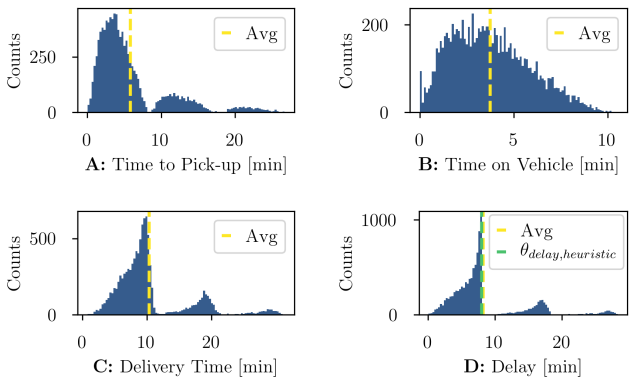

Fig. 4: Distributions of time to pick-up, time of parcels spend loaded to a robot, the total delivery time and delay of the base scenario.

evaluate the capabilities of the proposed method, we look at four different performance measures.

First, we evaluate the service rate, which is defined by the percentage of placed orders that got served. For this scenario, a service rate of 99.02% was achieved, which equals an absolute number of 98 ignored orders. Most ignored orders happen during peak times.

Second, we analyze different time spans (time KPIs) involved in the delivery process. Figure 4 shows the distributions of the time until pick-up (Avg: 5 min 51 s), the time a parcel is loaded onto a robot (Avg: 3 min 45 s), the delivery time (Avg: 10 min 21 s) and the associated delay (Avg: 8 min 14 s). For comparison, the average distance of all nodes to their closest depot is 1 min 22 s.

Third, we analyze how each robot is utilized by the proposed method. We summarize by calculating the mean number of loaded parcels of all robots over time to 1.78 loaded parcels per robot.

Fourth, we analyze the total traveled distance. In the base scenario a distance of 8,930.86 km is traveled by all 30 robots together. Results of the base scenario are illustrated in Figure 5 in purple.

B. Comparison

To investigate the benefits of our proposed method, we compare the run analyzed in detail in the previous section with two similar scenarios. First, we claim that the use of multiple depots is beneficial, so we compare to a single depot scenario. The location of the depot was chosen as the center of the graph. The center of the graph minimizes the total sum of the distances to all other nodes of the graph. Worse results in all performance measure were obtained, as shown in Figure 5 (yellow). The service rate drops from 99.02% to 66.08%. All time KPIs nearly double in time, for example, the overall delivery time increases from 10 min 21 s to 19 min 03 s. Even having fewer parcels delivered overall the total driven distances increases compared with the base scenario. Those improvements prove the value of using multiple depots, as assumed a priori, and puts numbers onto their effects. To operate depots is costly, thus posing the question "How many depots should be used?".

Second, we explicitly allow for pre-empty depot returns arguing to improve obtained results, thus we prohibit those depot returns, meaning only empty robots can load new orders. Results are also depicted in Figure 5 in green. Results show the worth of allowing for pre-empty depot returns as all performance measures stay similar or are slightly worse compared with allowing pre-empty depot returns. For example, delivering fewer parcels overall more total driven distance was required. Further experiments show that allowing pre-empty depot returns has larger effects in their absolute magnitude the fewer depots are present. Note that these improvements are fully achieved by modifying the routes without the need of additional infrastructure.

C. Sensitivity Analysis

1) Number of considered depots per order, x: In Section IV-B we introduced a heuristic that only the x closest depots to an order's destination are considered as potential pick-up locations. The more depots are considered the higher the number of candidates will be, yet, the higher the computational burden.

Table I lists results if one, three, five or seven depots per order are considered. For five and seven considered depots the available computational time was increased to avoid a bottleneck at this point and allow for a fair comparison. The service rate increases from one to five depots and drops for seven. Despite that more parcels have been delivered the total driven distance decreases for more considered depots, again excluding the seven depot case. Time KPIs show a similar behavior. The number of mean loaded parcels increases the more depots are considered.

On one hand, we see that considering multiple depots is beneficial, shown in the improvements compared with considering the closest depot only. On the other hand, too many depots can be considered, leading to worse obtained results. This can be explained by the myopic nature of our approach, which can lead the system into unfavorable states to serve future demand. For a single state using more depots is better, but due to chaining multiple states dynamically with each other, the overall problem's solution can worsen. Chaining better solutions for a dynamic problem is not necessarily better in the end. Thus, those results confirm the benefits of introducing a heuristic. We conclude:

- Considering only one depot as [18] and [19], which in this paper would be the closest one, can be inferior to considering multiple ones.
- To consider as many depots as possible can be inefficient for dynamic problems having imperfect anticipation.

2) Total number of depots $\mathcal{H}$: We varied the number of placed depots within the service area $\mathcal{H}$. We simulated scenarios featuring 1, 10, and 30 depots. Service rate improves at decreasing rates the more depots are available. Time KPIs also improve with the number of available depots. The number of mean loaded parcels (except for a single depot) and total driven distance both decrease the more depots are

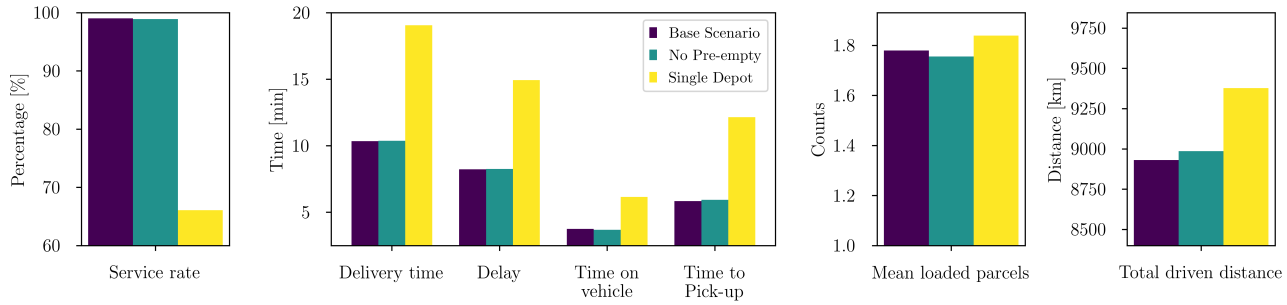

Fig. 5: Service rate, time KPIs, mean loaded parcels and total driven distance of the base scenario (purple), the same scenario with prohibit pre-empty depot returns (green) and a case using a single depot and our method (yellow) are illustrated.

	Service rate [%]	**Delay [min:s]**	**Delivery time [min:s]**	**Time on robot [min:s]**	**Time to Pick-up [min:s]**	**Mean loaded parcels**	**Total distance [km]**
Base scenario	99.0	08:14	10:21	03:45	05:51	1.78	8930.9
Single depot	66.1	14:56	19:03	06:07	12:09	1.84	9377.3
No pre-empty	98.9	08:15	10:22	03:41	05:56	1.76	8986.3
Considered depots per order							
1 de. per order	98.3	07:58	10:04	03:23	05:56	1.62	9036.6
3 de. per order	99.0	07:58	10:05	03:41	05:39	1.76	8951.8
5 de. per order	99.1	07:52	09:59	03:44	05:29	1.78	8946.7
7 de. per order	99.0	08:07	10:14	03:48	05:41	1.80	8955.1
Total depots							
10 total depots	93.4	11:30	14:16	04:44	08:47	2.07	9300.2
30 total depots	99.6	06:48	08:41	03:15	04:41	1.58	8799.1
Demand patterns							
9,500 orders	99.9	06:58	09:06	03:34	04:47	1.63	9114.9
10,500 orders	98.6	08:39	10:46	03:52	06:10	1.91	8943.0

TABLE I: Precise results of all performance indices of all executed runs are listed. In the first column results of the base scenario are shown (Section V-A). In the second and third column the results corresponding to the comparison section are presented (Section V-B). The reminder of the table lists the different runs corresponding to the different parameters of the sensitivity study (Section V-C).

available. The magnitude of changes in performance varies while the number of depots is increased linearly in steps of 10. Table I lists the corresponding results.

3) Number of orders $\mathcal{N}$: We created two alternative demand scenarios, featuring different numbers of customer orders $\mathcal{N}$ (9,500 and 10,500) over the course of the full day. Both scenarios resemble the distribution of the 10,000 order case in time and space. Table I lists the corresponding results. The more orders are placed the lower the service rate, because available resources were kept constant. Nevertheless, the absolute number of delivered parcels increases (9,487, 9,902 and 10,351). Despite that more parcels have been delivered this is not shown in the total driven distance. Compared with the 9,500 order case, driven distance decreases for both 10,000 orders and 10,500 orders. In general, if there are more orders to serve, it will be easier to find customer destinations close to each other, so the average distance between them decreases. However, those improvements do not hold for the time KPIs as all of them worsen with more placed and served orders. The number of mean loaded parcels increases in the same manner.

VI. CONCLUSION

In this paper, we looked at a fleet of autonomous grocery delivery robots operating a SDD operation featuring the opportunity to pick up goods at multiple depots and allowing robots to perform pre-empty depot returns, if beneficial. This allows for a reduced average distance to customers' homes and more agile planning. We propose an iteratively working method, meaning one state of the problem is solved before time propagates and the next state to solve is reached. To solve one given state, orders are assigned to potential pick-up locations, followed by checking how they could be combined into trips. For each robot, as many trips as possible are generated, bounded by predefined constraints. Robots are assigned to the generated trips via solving an integer-linear program.

The proposed method is able to handle large problem sizes. Extensive computational experiments simulating one day of service have been carried out. Looking at one scenario in detail, in which 10,000 orders are placed and 30 robots are available to serve those, a service rate of 99.0% was achieved. The average delay accounts for 8 min 14 s and 8,930.9 km needed to be driven. Further, simulations showed the value of using multiple depots, exemplary, the service rate dropped about 33%, if only one depot was used, and the value of performing pre-empty depot returns, shown in a decrease in total driven distance. A sensitivity study showed the varying influence of individual parameters on the obtained solution.

Future research could extend the proposed method to look ahead, such that the risk that the system gets into unfavorable states is reduced. Further, the possibility to plan for heterogeneous fleets of robots could be added. Currently, the time to pick an order is constant and rather short. Additionally, we assume that all products are always available. To relax these assumptions, namely to consider variable picking times of orders and to include product availability in depots, are further research questions.

ACKNOWLEDGMENT

This research was supported by Ahold Delhaize. All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[2] H. N. Psaraftis, M. Wen, and C. A. Kontovas, "Dynamic vehicle routing problems: Three decades and counting," *Networks*, vol. 67, no. 1, pp. 3–31, 2016.

[3] G. Berbeglia, J.-F. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, no. 1, pp. 8–15, 2010.

[4] D. Reyes, A. Erera, M. Savelsbergh, S. Sahasrabudhe, and R. O'Neil, "The meal delivery routing problem," *Optimization Online*, 2018.

[5] B. Yildiz and M. Savelsbergh, "Provably high-quality solutions for the meal delivery routing problem," *Transportation Science*, vol. 53, no. 5, pp. 1372–1388, 2019.

[6] M. W. Ulmer, B. W. Thomas, A. M. Campbell, and N. Woyak, "The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times," *Transportation Science*, vol. 55, no. 1, pp. 75–100, 2021.

[7] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, 2017.

[8] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem: models and algorithms," *Annals of Operations Research*, vol. 153, no. 1, pp. 29–46, 2007.

[9] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative Robots and Sensor Networks 2015*, pp. 31–51, 2015.

[10] G. Ghiani, E. Manni, A. Quaranta, and C. Triki, "Anticipatory algorithms for same-day courier dispatching," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, no. 1, pp. 96–106, 2009.

[11] N. Azi, M. Gendreau, and J.-Y. Potvin, "A dynamic vehicle routing problem with multiple delivery routes," *Annals of Operations Research*, vol. 199, no. 1, pp. 103–112, 2012.

[12] M. A. Klapp, A. L. Erera, and A. Toriello, "The one-dimensional dynamic dispatch waves problem," *Transportation Science*, vol. 52, no. 2, pp. 402–415, 2016.

[13] ——, "The dynamic dispatch waves problem for same-day delivery," *European Journal of Operational Research*, vol. 271, no. 2, pp. 519–534, 2018.

[14] ——, "Request acceptance in same-day delivery," *Transportation Research Part E: Logistics and Transportation Review*, vol. 143, p. 102083, 2020.

[15] M. W. Ulmer and S. Streng, "Same-day delivery with pickup stations and autonomous vehicles," *Computers Operations Research*, vol. 108, pp. 1–19, 2019.

[16] S. A. Voccia, A. M. Campbell, and B. W. Thomas, "The same-day delivery problem for online purchases," *Transportation Science*, vol. 53, no. 1, pp. 167–184, 2017.

[17] M. W. Ulmer, B. W. Thomas, and D. C. Mattfeld, "Preemptive depot returns for dynamic same-day delivery," *EURO Journal on Transportation and Logistics*, vol. 8, p. 327–361, 2019.

[18] B. Yu, N. Ma, W. Cai, T. Li, X. Yuan, and B. Yao, "Improved ant colony optimisation for the dynamic multi-depot vehicle routing problem," *International Journal of Logistics Research and Applications*, vol. 16, no. 2, pp. 144–157, 2013.

[19] H. Xu, P. Pu, and F. Duan, "A hybrid ant colony optimization for dynamic multidepot vehicle routing problem," *Discrete Dynamics in Nature and Society*, 2018.

[20] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.

[21] A. Mourad, J. Puchinger, and C. Chu, "A survey of models and algorithms for optimizing shared mobility," *Transportation Research Part B: Methodological*, vol. 123, pp. 323–346, 2019.

[22] S. Narayanan, E. Chaniotakis, and C. Antoniou, "Shared autonomous vehicle services: A comprehensive review," *Transportation Research Part C: Emerging Technologies*, vol. 111, pp. 255–293, 2020.

[23] A. Fielbaum, X. Bai, and J. Alonso-Mora, "On-demand ridesharing with optimized pick-up and drop-off walking locations," *Transportation Research Part C: Emerging Technologies*, vol. 126, p. 103061, 2021.

[24] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. T. Jr., "On the capacitated vehicle routing problem," *Math. Program.*, vol. 94, no. 2-3, pp. 343–359, 2003.

[25] J. R. Montoya-Torres, J. López Franco, S. Nieto Isaza, H. Felizzola Jiménez, and N. Herazo-Padilla, "A literature review on the vehicle routing problem with multiple depots," *Computers Industrial Engineering*, vol. 79, pp. 115–129, 2015.

[26] M. ApS, *MOSEK for c++. Version 7.1.*