2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Decentralized Dynamic Multi-Vehicle Routing via Fast Marching Method

Jesper Karlsson and Jana Tumova

Abstract— **While centralized approaches to multi-vehicle routing problems typically provide provably optimal solutions, they do not scale well. In this paper, an algorithm for decentralized multi-vehicle routing is introduced that is often associated with significantly lower computational demands, but does not sacrifice the optimality of the found solution. In particular, we consider a fleet of autonomous vehicles traversing a road network that need to service a potentially infinite set of gradually appearing travel requests specified by their *pick-up* and *drop-off* points. The proposed algorithm synthesizes optimal assignment of the travel requests to the vehicles as well as optimal routes by utilizing Fast Marching Method (FMM) that restricts the search for the optimal assignment to a local subnetwork as opposed to the global road network. Several illustrative case studies are presented to demonstrate the effectiveness and efficiency of the approach.**

I. Introduction

This work is motivated by a mobility-on-demand scenario, where a fleet of autonomous vehicles coordinate in order to service a demand given by customers at different points in time and space. An essential problem to address is the optimal assignment of vehicles to satisfy these demands. For example, consider a road network with K locations, N vehicles, and M demands, such as "Pick me up at location A and drive me to location B." Each demand is associated with a deadline and a priority depending on their relative importance in the system given, for example, by the customer's subscription status. Clearly, some vehicles might be better suited to service certain tasks than others, depending on their relative position in the road network. The goal is to determine which vehicles should service which demands so that their respective tasks are fulfilled as fast as possible, and to find the optimal routes. Finding the optimal assignment of demands to vehicles is not trivial. Simply evaluating all the potential assignments and routes for all N vehicles in a centralized fashion is not tractable due to state space explosion. We propose a decentralized approach that promotes scalability while at the same time maintains the guarantees on optimality, a combination not present in other decentralized approaches for the same problem.

We model the road network as a Weighted Transition System (WTS), with the states representing locations of interest and the transition describing the road segments connecting these locations. A fleet of vehicles operate in the network. Demands arrive continuously as time progresses, and each demand is defined with a *pick-up* location and a *drop-off* location together with its time of arrival, its deadline, and priority. Our goal is to assign demands to vehicles and to determine the paths in the WTS that yield the minimization of the cumulative delay of servicing demands weighted by their respective priorities. The main contribution of this paper is a decentralized approach based on Fast Marching Method (FMM) that grows a local region around a pick-up location of a given demand. We identify stopping criteria for the region growth and prove that the optimal assignment is to a vehicle within this region. Thus, we limit the search space and potentially decrease necessary computations when determining the optimal routes.

The problem we consider is a variant of Dynamic Vehicle Routing Problem (DVRP) [1], [2], which is usually solved using a centralized approach [3], [4], [5], [6]. In these works, the problem is formulated in such a way that tasks are unknown beforehand and appear according to some stochastic behaviour. These works solve the problem by defining an objective function, most often depending on travel time and some form of customer inconvenience, which is then minimized using a meta-heuristic, genetic algorithm or some sort of combinatorial optimization method (e.g., Mixed Integer Linear Programming problem, MILP). The downside to these approaches is that they scale poorly to large networks, which makes them unrealistic to real world applications. To the best of our knowledge works that handle decentralized approaches [7], [8], [9], [10], [11] generally partitions the environment and allocate one vehicle in each segment. The solution is found by employing a local optimization strategy. However, these segmentation methods, such as Voronoi partitioning, can not provide global guarantees on optimality. At best, the solutions found will be at some provable distance from the optimal. In this work we combine the scalability of the decentralized approaches with the provable guarantees of the centralized approaches. In general, our approach is inspired by research in formal methods for optimal path planning and can be viewed in the context of work such as [12], [13], in which unsatisfiable specifications are accomplished in the maximally satisfying manner. In our previous work [14], we addressed the problem of single-agent least-violating routing from scLTL specification. In contrast, our focus here is mainly on multi-agent task assignment.

The remainder of the paper is organized as follows. In Sec. II, the necessary preliminaries are introduced. In Sec. III, we describe the system model and formalize the optimal multi-vehicle routing problem. Sec. IV introduces the key methods used in the solution and describes the proposed algorithm in detail. Sec. V provides simulation descriptions and results. Sec. VI concludes and discusses the work, and outlines future research directions.

The authors are with the Department of Robotics, Perception, and Learning (RPL), KTH Royal Institute of Technology. Email: {jeskarl, tumova}@kth.se. This work has been supported by the Swedish Research Council (VR).

II. Preliminaries

Let $\mathbb{R}_+$ denote the set of positive real numbers. Given a set S, let 2^S denote the set of all subsets of S.

Definition 1 (Weighted transition system (WTS))**.** *A weighted deterministic transition system (WTS) is a tuple $\mathcal{T} = (S, R, W, \Pi, L)$, where S is a finite set of states; $R \subseteq S \times S$ is a transition relation; $W : R \to \mathbb{R}_+$ is a weight function; Π is a set of atomic propositions that are evaluated either as true or false; and $L : S \to 2^\Pi$ is a labelling function that assigns to each state the subset of atomic propositions that are true therein.*

Given that a system is in a state s at time t, it can reach state s' at time $t' = t + W(s, s')$ if $(s, s') \in R$. A *trace* is a finite or infinite sequence of states $\tau = s_1 s_2 \dots$ such that $(s_i, s_{i+1}) \in R$ for all $i \geq 1$. The length of a finite trace $\tau = s_1 s_2 \dots s_n$ is $\mathit{length}(\tau) = \sum_{i=1}^{n-1} W(s_i, s_{i+1})$. The fragment of a trace $\tau = s_1 s_2 \dots$ starting at time t_1 and ending at time t_2 is denoted by $\tau^{t_1,t_2} = s_i \dots s_j$, where $\mathit{length}(s_1 \dots s_{i-1}) < t_1$, $\mathit{length}(s_1 \dots s_i) \geq t_1$, $\mathit{length}(s_1 \dots s_{j-1}) < t_2$, and $\mathit{length}(s_1 \dots s_j) \geq t_2$. $\tau^{0,t}$ is a prefix of τ up to time t, and $\tau^{t,\infty}$ is its suffix, which we denote also τ^t.

A weighted transition system can be viewed as a graph and the standard terminology from graph theory thus applies. A *subgraph* of $\mathcal{T}$ is given by a subset of states S and transitions R of $\mathcal{T}$, the subset of states S must include all end states of the transition subset. A *radius* of a subgraph measured from a state s is the maximal length of the shortest path to any other state in the subgraph.

III. Problem Formulation and Approach

In this section, we state the problem of optimal routing of a fleet of vehicles in a road network, whose goal is to collectively service customers as efficiently as possible. The customers' transport requests are each characterized by their pick-up and drop-off locations, the time of the request appearance, the deadline of the drop-off, and the customer's priority, given, e.g., by the type of membership the customer holds in the mobility-on-demand system. Later on in Sec. VI, we discuss that we can easily replace simple pick-up and drop-off requests by more complex requests, given as scLTL formulas. However, for the simplicity of the presentation, we first restrict our attention to pick-up and drop-off requests. For convenience, Table I provides the notations used throughout this work.

TABLE I: Table of Notations.

D_j	Demand
t_j	Arrival time of demand D_j
T_j	Deadline of demand D_j
p_j	Priority of demand D_j
$A(t)(j)$	Assignment of demand D_j at time t
$\tau_i(t)$	Trace of vehicle i at time t
τ^{t_1,t_2}	Trace from time t_1 to t_2
d_j	Duration to service demand D_j
$\Delta_j = d_j - T_j$	Delay of servicing demand D_j
$dist(\textit{pick-up}, \textit{drop-off})$	Shortest path between *pick-up* and *drop-off*
r	Radius of *relevant region*
$\mathcal{S}^\star(t)$	Optimal solution at time t
$\mathcal{S}^r(t_j)$	Optimal solution assuming an assignment of D_j to a vehicle within radius r from *pick-up*$_j$

A. System Model and Specification

A *road network* is modeled as a weighted transition system $\mathcal{T} = (S, R, W, \Pi, L)$, where S corresponds to a set of locations, such as intersections, or points of interest, and a transition $(s, s') \in R$ models the existence of a road segment connecting locations s and s'. The weight $W(r)$ gives an estimate on the travel time associated with traversing the road segment r. In this work, we assume that $W(r)$ does not change over time, and considering a dynamically changing weight function is subject to our future work. Π is a set of properties associated with locations, such as "location A", "shopping mall", "grocery store", or similar. The labeling function L then labels each location with a subset of properties. We assume that each location $s \in S$ has a unique identifier, i.e. that there exists $\pi \in L(s)$, such that $\pi \notin L(s')$, for all $s' \neq s$.

A *fleet of vehicles* $\mathcal{N} = \{1, \dots, N\}$ traverse the road network $\mathcal{T}$. We denote by $\mathfrak{s}_i(t) \in S$ the current position of vehicle $i \in \mathcal{N}$ in the road network at time t. We assume that each vehicle can stay in its current location for an arbitrary amount of time, which can be modeled by adding transitions (s, s) for all $s \in S$ to the transition relation R. The system is initiated at time $t = 0$ and the initial location $\mathfrak{s}_i(0)$ of each vehicle $i \in \mathcal{N}$ is known. As each vehicle $i \in \mathcal{N}$ moves, it produces a trace τ_i of $\mathcal{T}$, whose first state is $\mathfrak{s}_i(0)$. Transport requests may appear at any time and prior to that, they are unknown. The j-th transport request is modelled as a *demand* $D_j = (\textit{pick-up}, \textit{drop-off}, t_j, T_j, p_j)$, where

- *pick-up*$_j \in S$ is the pick-up location of the demand;
- *drop-off*$_j \in \Pi$ is the proposition that has to hold in the drop-off location. With a slight abuse of notation, we will use *drop-off* to denote any state $s \in S$, such that *drop-off* $\in L(s)$;
- t_j is the time of the demand arrival;
- $T_j \in \mathcal{R}_0$ is the demand completion deadline; and
- $p_j \in \mathcal{N}$ is the priority.

The set of all demands is denoted by $\mathcal{D}$.

B. Routing

A demand $D_j \in \mathcal{D}$ *has been serviced* by vehicle $i \in \mathcal{N}$ if $\tau_i^{t_j, t_j + d_j} = \mathfrak{s}_i(t_j) \dots \textit{pick-up}_j \dots \textit{drop-off}_j$. If a drop-off state of D_j only occurs once on $\tau_i^{t_j, t_j + d_j}$, d_j denotes the *service duration*. The *delay* of demand D_j is $\Delta_j = d_j - T_j$. If the demand is serviced before the deadline, then $\Delta_j < 0$. If it is serviced right on time, then $\Delta_j = 0$. If the demand is not serviced at all, we define $\Delta_j = \infty$.

Definition 2. *A solution $\mathcal{S}$ to the routing problem is a set of traces τ_i, $i \in \mathcal{N}$, and a mapping $A : \mathcal{D} \to \mathcal{N}$, called* assignment*, such that each demand $D_j \in \mathcal{D}$ has been serviced by the vehicle $A(j)$.*

We associate each solution with a cost that represents the global efficiency of the system. In particular, we aim to achieve the "global social optimum" with respect to delays on servicing the demands weighted by the priorities:

$$\lambda(\tau_1, \dots, \tau_N, A) = \sum_{j \in \mathcal{D}} p_j \Delta_j \tag{1}$$

C. Optimal Routing Problem

The most natural goal would be to find the solution to the routing problem minimizing the cost in (1). However, since demands are initially not known, we instead aim to periodically re-route as the demands appear, in order to find the optimal future routing at each time instant t. To formalize the problem we introduce several intermediate definitions.

Instead of a single routing problem solution, we work with time-dependent routing problem solutions, i.e., a set of traces $\tau_i(t), i \in \mathcal{N}$, and an assignment $A(t)$, $t \geq 0$. Loosely speaking, a valid time-dependent routing problem solution cannot change what happened in the past. It does not change the prefix of the trace executed up to time t, and it does not change the assignment of a demand that is already in process of being serviced, i.e. a demand of a customer that was already picked up. Formally, we have the following.

Definition 3. *$\mathcal{S}(t) = (\tau_1(t), \dots, \tau_N(t), A(t))$ is a valid routing solution at time t if*

- *$\tau_i(t)^{0,t'} = \tau_i(t')^{0,t'}$ for all $t' \leq t, i \in \mathcal{N}$; and*
- *$A(t')(j) = A(t)(j)$ for $D_j \in \mathcal{D}$ and $t' \leq t$ with the property that $t' \geq t_j$, and pick-up$_j \wedge$drop-off $\in \tau^{t',t}_{A(t')(j)}$.*

Definition 4. *A demand is* active *if it has appeared, but has not been serviced, yet. In other words, D_j is active at time t if $t_j \leq t$, but $d_j > t - t_j$. We denote the set of active demands at time t by $\mathcal{D}(t)$.*

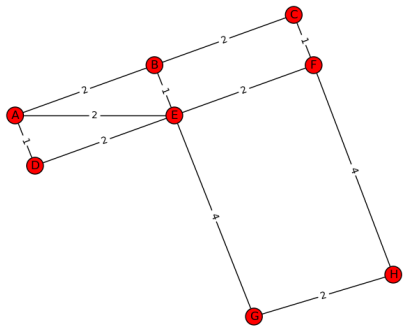
Problem 1. *Consider a road network in the form of a* WTS*, a fleet of vehicles $\mathcal{N} = \{1, \dots, N\}$, and a (possibly infinite) set of gradually appearing demands $\mathcal{D}$. At each time instant t, find a valid routing problem solution $\mathcal{S}(t)^\star = (\tau_1(t)^\star, \dots \tau_N(t)^\star, A(t)^\star)$ that minimizes*

$$J(\tau_1(t)^\star, \dots, \tau_N(t)^\star, A(t)^\star) = \sum_{j \in \mathcal{D}(t)} p_j \Delta_j \tag{2}$$

An example of an instance of Problem 1 is given in Fig. 1.

IV. Solution

One way to solve Problem 1 is to determine the optimal assignment and traces centrally by taking into account the whole network. However, such an approach scales poorly for large networks. Another approach is to segment the network into subregions, one for each vehicle, and assign the demands within a subregion to the corresponding vehicle. However, this type of solutions does not provide global optimality guarantees. We propose to find a *subgraph* of $\mathcal{T}$, C_j s.t. $A(t)^\star \in C_j$. C_j is centered around the *pick-up* location of D_j, and we refer to this subgraph as the *relevant region*.



	pick-up	*drop-off*	Arrival	Deadline	Priority
D_1	F	E	0	10	1
D_2	D	E	0	5	10

Fig. 1: A simple road network $\mathcal{T} = (S, R, W, \Pi, L)$, where $S = \{A, \dots, H\}$, $\Pi = \{A, \dots, H\}$, $L(s) = s$ for all $s \in S$. The figure together with the table illustrate an example input to Problem 1 inspired by example in [14], i.e. a road network modeled as a WTS, and a set of demands, respectively. The states are illustrated as nodes, the transitions as edges, the weights as the labels of the edges and the atomic propositions that hold in states as the labels of the nodes. Suppose that there are two vehicles, vehicle 1 and 2 in the system, initially located at A and H, respectively. Two demands are considered; the first one to travel from the pick-up location F to the drop-off location E, while the second one to travel from the pick-up location D to E. The problem is to determine which allocation will minimize J. In this case D_1 arrives at $t = 0$ and D_2 at $t = 0$. The optimal allocation in this case will be that D_1 is assigned to the vehicle in H and that D_2 is assigned to the vehicle in A. The reason being that either vehicle will perform equally for D_1, but the vehicle in A will service D_2 better than the vehicle in H.

A. Efficient Assignment

Suppose that given an assignment $A(t)(j)\ \forall\ D_j \in \mathcal{D}(t)$, we can compute valid optimal traces $\tau_1(t)^\star, \dots, \tau_N(t)^\star$ respecting the assignment. In this work we use the method described in [14], but it can also be done using classical approaches, such as through MILP. In this section, we introduce our main contribution, which is an algorithm that efficiently computes the optimal assignment $A(t)^\star$ at time t.

Upon appearance of a new demand D_j at time t_j, we utilize the Fast Marching Method (FMM) to grow the *relevant region* around its pick-up location. Intuitively, the best assignment for a demand is to the closest vehicle, provided that this vehicle would service this demand only. However, it is not necessarily the best assignment in general due to the fact that the same vehicle might have been assigned to earlier demands, which now compete for the vehicle's travel time with the new demand D_j. The new demand D_j thus can cause delays in servicing the earlier demands, or vice versa. Hence, it might be advantageous to continue growing the relevant region to find a better assignment, or to re-assign an earlier demand. The assignment procedure is in fact recursive. A candidate assignment that newly assigns demand D_j to vehicle i at time t_j leads to a growth of a relevant region around every demand in $A(t)^{-1}(i)$ that is active, but not in the process of being serviced, i.e. whose pick-up location has not been visited, yet. Each candidate assignment is evaluated by computing the valid optimal traces and the corresponding value of the J function. To consider only promising candidate assignments, we specify stopping criteria for the growth of the relevant region by FMM. The first criterion is an existence of a vehicle that has currently no other demand assigned. The second criterion

is that the relevant region reaches the size of a calculated threshold. Both the criteria are discussed below, after we briefly overview FMM technique.

1) Fast Marching Method (FMM): FMM [15] is an algorithm for modelling the motion of a physical wave front. Consider the scenario where a stone is dropped into a pond. As rings spread from the point where the stone broke the surface of the water, the FMM determines the time it takes for the rings to reach every point in the search space. At every point, the wave front is described with the Eikonal equation, namely: $1 = F(x)|\nabla T(x)|$; where $F(x)$ is the expansion speed of the wave at point x, and $T(x)$ is the time it takes for the wave to reach it. At each point it is possible to determine $T(x)$, by solving the Eikonal equation.

FMM can be used for path planning [16] as follows: The starting location $s \in S$ (in our case the pick-up location) is the centre of the growing circle. A path to the desired target $s' \in S$ (in our case a location of a vehicle that a demand can be assigned to) is determined by expanding a wave. It is assumed that $F(s')$ is constant. $T(s')$ is then in fact the length of the shortest path between the starting location and the target (in our case the radius of the growing relevant region). In that sense, FMM is closely related to Dijkstra's shortest path algorithm [17]. In this work, however, we could not use Dijkstra's algorithm directly, since we are not interested in simply assigning a newly arrived demand to the closest vehicle.

2) Growth Stopping Criteria: Suppose that r is the current radius of the relevant region, centred around the *pick-up* location of demand D_j, with the arrival time t_j. Let $\mathcal{S}(t_j)^r$ denote the solution that minimizes J among all possible solutions, where D_j is restricted to be assigned to a vehicle inside the relevant region. Furthermore, let $\mathcal{S}(t_j^-)^\star$ denote the optimal solution right before the arrival of demand D_j, i.e. the solution that has been executed up till the arrival of D_j. This section discusses how the threshold of r is derived, and proves that $\mathcal{S}(t_j)^r = \mathcal{S}(t_j)^\star$.

Definition 5. *Vehicle $i \in \mathcal{N}$ is called* free *at time t if $A(t)^{-1}(i) \cap \mathcal{D}(t) = \emptyset$. Initially, all vehicles are free.*

The following Lemma provides the first sufficient condition for stopping the growth of the relevant region around the pick-up location of demand D_j at time t_j.

Lemma 1. *If a free vehicle i is found at distance r from the pick-up location of D_j, then any vehicle i' found at distance $r' > r$ will yield*

$$J(\tau_1(t_j)^\star, \ldots, \tau_N(t_j)^\star, A(t_j)) < J(\tau_1(t_j)'^\star, \ldots, \tau_N(t_j)'^\star, A(t_j)')$$

for the assignments $A(t_j), A(t_j)'$, where $A(t_j)(j) = i$, $A(t_j)'(j) = i'$, and $A(t_j)(\ell) = A(t_j)'(\ell)$ for all $\ell \in \mathcal{D}(t)\backslash\{D_j\}$, and $\tau_1(t_j)^\star \ldots \tau_N(t_j)^\star$ and $\tau_1(t_j)'^\star \ldots \tau_N(t_j)'^\star$ being the optimal traces respecting the assignments $A(t_j)$ and $A(t_j)'$, respectively.

Proof. Suppose that i' is free. The lemma follows from the fact that the task duration associated with servicing D_j is larger for the latter vehicle than for the former one by at least $r' - r$. On the other hand, suppose that i' is not free. Similar argument applies, and furthermore, additional delays to earlier arrived demands might be introduced. □

The consequence of Lemma 1 is that once we find a free vehicle i at time t in the growing relevant region around the pick-up location of demand D_j, we can safely stop the growth, set $A(t)^\star(j) = i$, and compute $\tau(t)^\star(i)$ without any changes to $\tau(t)^\star(k)$ for $k \in \mathcal{N} \setminus \{i\}$. A more complex scenario is when there keep appearing only non-free vehicles within the growing relevant region. At some point, vehicles in the relevant region become so far from the pick-up location that the assignment of the new demand to them cannot yield the global optimum even if these vehicles were free. In the following, we introduce the threshold radius of the relevant region. Initially, this threshold radius is set to size of the whole network, however, as the relevant region grows and better and better candidate assignments are revealed, this threshold radius shrinks till it becomes smaller than the radius of the current relevant region. At that point, the growth can stop. We illustrate the idea in Fig. 2.

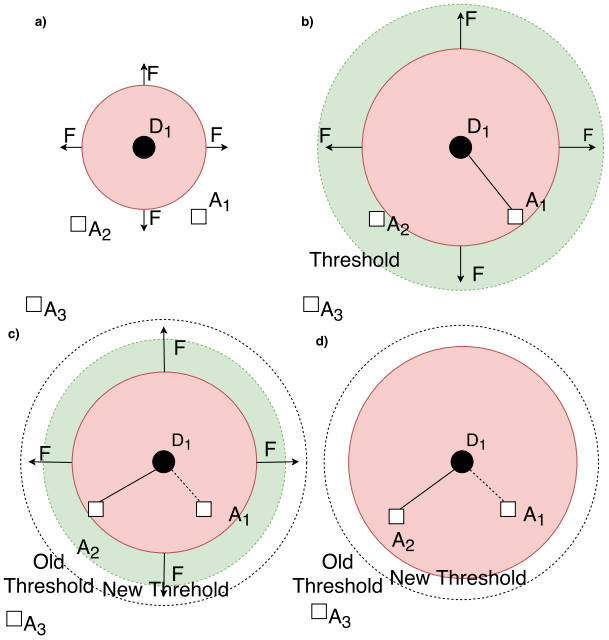


Fig. 2: Three vehicles A_1, A_2 and A_3, and a new demand D_1 that arrives at time t_1 are considered. Assume that neither of the vehicles are free. Top left illustrates the wave form (red circle) that is initiated at the pick-up location of the demand D_1. The relevant region is grown until vehicle A_1 is found, at which point D_1 is assigned to it in top right. Note that, had A_1 been *free*, i.e. had it no other assigned demands, it would have provided the optimal assignment. Instead a threshold to the radius of the relevant region is determined, describing the region where a better solution *might* exist. This first assignment provides us with an initial threshold radius, as is illustrated in top right in green. The wavefront is grown further, until vehicle A_2 is discovered (bottom left in red). At this point the costs associated with the two assignments are compared in order to determine whether or not the value of J is improved. In this case, it is, and a new threshold radius is provided (bottom left in green). The process is terminated when this threshold is reached (bottom right). Note that at this point it is unnecessary to consider A_3 as it is guaranteed to perform worse than any of the vehicles considered.

Formally, we set the threshold radius as follows. Suppose that the relevant region is grown around a pick-up location

of a demand D_j at time t_j. Initially, the relevant region is grown and we obtain the first candidate solution $\mathcal{S}(t_j)^{r_1}$. To determine whether or not a better solution $\mathcal{S}(t_j)^\star$ will be found if the relevant region is grown more, we compare the difference between the cost of the current candidate solution $J(S^r(t_j))$ and the cost of the optimal solution that assumes that a virtual free vehicle is placed at the boundary of the relevant region and that D_j is assigned to it, which is $p_j\Delta_j + J(S(t_j^-)^\star)$. Lemma 1 ensures that, if we find a free vehicle we can stop growing the relevant region. Therefore, we can stop growing, if the virtual free vehicle results in a higher cost. In other words, any allocations found outside this threshold will be suboptimal. With this in mind we formulate the following Lemma:

Lemma 2. *If the size of the relevant region radius is greater or equal than*

$$r = T_j - dist(\textit{pick-up}_j, \textit{drop-off}_j) + \frac{J(\mathcal{S}^r(t_j)) - J(\mathcal{S}^\star(t_j^-))}{p_j} \tag{3}$$

the optimal assignment $A(t)^\star(D_j)$ is guaranteed to be within this relevant region.

Proof. The proof follows directly from the discussion above. Any free vehicle would perform worse when $J(\mathcal{S}^r(t_j)) = p_j\Delta_j + J(\mathcal{S}(t_j^-)^\star)$. The cost for the virtual free vehicle is $p_j\Delta_j = p_j(r + dist(\textit{pick-up}_j, \textit{drop-off}_j) - T_j)$. Together, we have $J(\mathcal{S}^r(t_j)) = p_j(r + dist(\textit{pick-up}_j, \textit{drop-off}_j) - T_j) + J(\mathcal{S}(t_j^-)^\star)$, which immediately gives us the threshold in Eq. 3. □

In summary, the growth of the relevant region can be stopped if (i) A free vehicle is found; or (ii) The radius of the region has reached the stopping threshold (3).

3) Algorithm Overview: With the growth stopping criteria in mind, Algorithm 1 gives a solution to Problem 1 at time t_j when a new demand D_j has arrived. The threshold is initially set to infinity. However, if known it can be initialized to the size of the network. The initial value of the radius of the *relevant region*, *r* is 0, defining the centroid of the wavefront at the pick-up location. The set *fixed* signifies the demands that have already been considered for re-assignment and is reset to $\emptyset$ at the arrival of each new demand. We also consider customers already picked up as *fixed*. The *relevant regions* are continuously grown as long as the wavefront does not pass beyond the threshold (line $2-4$), while new vehicles are constantly being discovered and considered in searching for a candidate solution to Problem 1. When a vehicle is found in the growing relevant region, the following three cases are treated:

1) The vehicle is free; The demand D_j is assigned to it and the corresponding route is returned (lines $5-9$). At this point Lemma 1 ensures that any vehicle outside this point will not perform better and the algorithm terminates.

2) The vehicle has already some demands assigned to it, and re-assignment of these demands results into considering previously unconsidered candidate solutions; Therefore, it is necessary to recursively call the algorithm on these demands

Algorithm 1: Planning

Input: Input to Problem 1 including (WTS $\mathcal{T}$, Demand D, vehicle locations $\{N\}$)

Output: An assignment and a set of feasible state trajectories

Initialisation : $Threshold \leftarrow \infty; r \leftarrow 0; fixed \leftarrow \emptyset$

1: Initialize FMM with origin in D's *pick-up* location
2: **while** $r < Threshold$ **do**
3: Grow relevant region, r
4: **for** vehicles i in relevant region **do**
5: **if** vehicle i is Free **then**
6: Assign D to vehicle i
7: $\tau_i \leftarrow Routing(D, i)$
8: **return**
9: **end if**
10: **if** demands assigned to i and these are not $fixed$ **then**
11: Add demand D to $fixed$
12: **for** $unfixed$ demands, D_j **do**
13: Algorithm 1 for D_j
14: **if** allocation unchanged **then**
15: Add D_j to $fixed$
16: **end if**
17: **end for**
18: **end if**
19: Assign D to vehicle i
20: $\tau_i \leftarrow Routing(D, i)$
21: Calculate $Threshold$
22: **end for**
23: **end while**

(lines $10-18$). If the assignment is unchanged, i.e. the current assignment is still optimal, the demand is marked as *fixed*.

3) This vehicle has already some demands assigned to it, but all of these demands have been previously marked as *fixed*; In this case re-assignment of these demands would not result into an improvement. This is because they have already been considered for reassignment but no better assignment was found. (lines $18-20$).

Note that, it is only in the second and the third case that we need to calculate the threshold (lines $21-25$). Once a free vehicle is found within the current threshold we can stop growing and terminate the algorithm, as the threshold signifies the point where a free vehicle would perform worse than current consideration.

B. Solution to Problem 1

With the outline of the algorithm and underlying routing procedure described, it is now possible to formulate the following theorem regarding the correctness of the algorithm.

Theorem 1. *At each time t, Algorithm 1 returns the solution to Problem 1.*

Proof. The proof follows from the fact that Lemmas 1 and 2 guarantee that the subnetworks found include the optimal assignment for each demand. Since all possible solutions in these subnetworks are considered we can guarantee that the proposed solution does in fact minimize J from (2). □

V. Simulations

The proposed algorithm was implemented in Python and tested on several cases, ranging from simple networks of

TABLE II: Description of the simple cases in the case study

Case	Demands	pick-up	drop-off	Arrival/ Deadline/ Priority
1a	D_1	D	E	0/9/1
	D_2	F	E	0/4/1
	D_3	H	G	0/4/5
1b	D_1	D	E	0/10/1
	D_2	F	E	0/5/2

TABLE III: Resulting assignments and thresholds in cases 1a and 1b.

Case	Demands	Assigned Vehicle's Initial Position:	Threshold
1a	$D_1/D_2/D_3$	E/H/H	2/4/0
1b	D_1/D_2	E/H	2/4

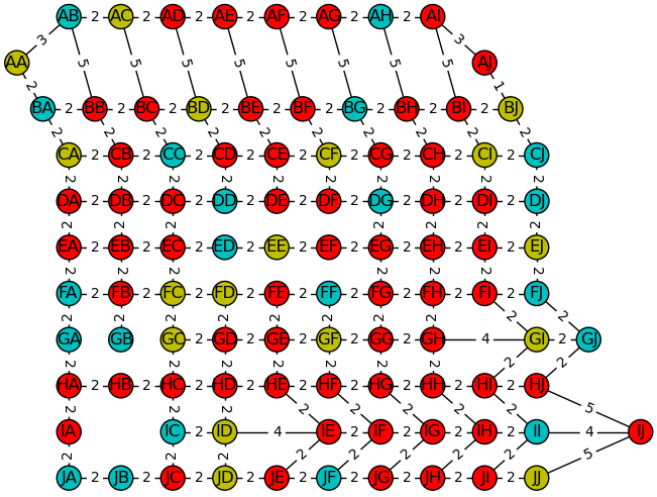


Fig. 3: Case 2, with 99 nodes, 17 vehicles and 21 demands. The vehicles' initial locations in yellow and the demands' pick-up location in cyan.

8 nodes and 2-3 demands and vehicles to larger networks of 100 nodes, both with a sparse and dense distribution of demands and vehicles.

The first case is illustrated in Fig. 1. This case considered a network of 8 nodes, and two set of demands listed in Table II as cases 1a and 1b, and two vehicles V_1, V_2 initialized at E, H, respectively. The resulting assignments for cases 1a, and 1b and are provided in Table III.

A larger and more densely distributed network is illustrated in Fig. 3. In this case it consists of 99 nodes, 17 vehicles and 21 demands. Finally, in Fig. 4 we consider a model of a realistic traffic network. This model consist of 94 nodes, 7 demands and 5 vehicles.

Run time properties of the algorithm are listed in Table IV. The benefit of the threshold can be best seen in Fig. 3 and 4. In Fig. 3 an exhaustive search for every demand is infeasible. Table IV illustrates that we can limit the search to a subnetwork that is only a small part of the entire network. There are still some demands where the relevant region spans large portions of the network, depending on the priority, or how many demands are currently being serviced. However, these are in a minority and a solution is readily found, which is not the case had the threshold not been implemented. Fig. 4 contains a complex network, in this case too, the threshold

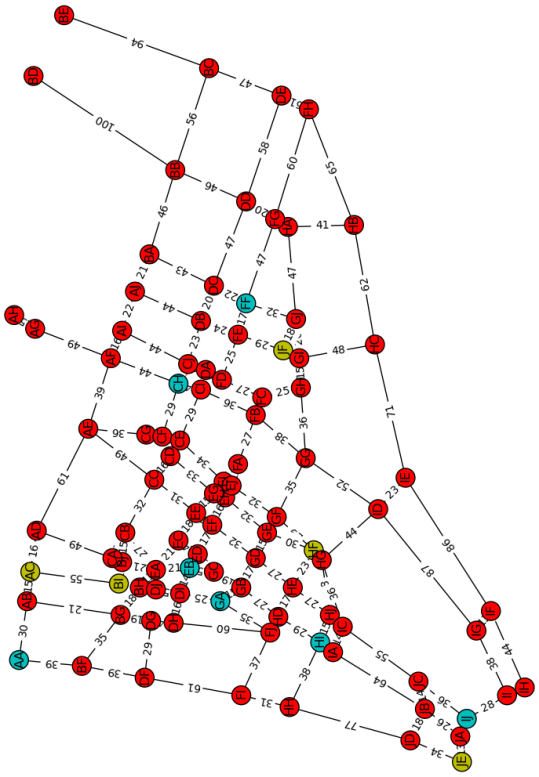


Fig. 4: Case 3, network modelled after Manhattan containing 7 demands and 5 vehicles. The vehicles' initial locations in yellow and the demands' pick-up location in cyan.

TABLE IV: Description of the run time properties of the algorithm in the cases in the case study.

Case	Smallest/Largest Threshold	min/max Depth of Recursion	Run Time [s]
1a	0/4	0/1	0.23
1b	2/4	0/1	0.13
2	2/12	0/4	7.70
3	23/167	0/2	2.96

limits the search to a very small part of the network. These results show that by utilizing the threshold as described in Sec. IV, we can reduce the complexity of the problems by a wide margin.

VI. Conclusion and Discussion

In this work, we proposed a decentralized scheme for task assignment and routing for a fleet of autonomous vehicles in a mobility-on-demand scenario. The road network was modelled as a Weighted Transition System and gradually appearing travel requests were described through pick-up and delivery locations, priorities, and deadlines. It was shown that it is possible to limit the search for the desired assignment of demands to vehicles to a particular subnetwork around the pick-up locations. We utilized the Fast Marching Method and derived growth stopping criteria that ensured the optimality of the solution.

Complexity and efficiency: The intended application for this work is a light-load autonomous mobility-on-demand service. A typical problem instance involves a large network with K locations, N vehicles, and M demands, where

$K \gg N > M$, while the length of a typical trip is $\ell \ll K$. The considered problem is generally NP-hard. In contrast to other approaches that also guarantee optimal solution, but are centralized, we split the problem defined on the overall network into several suproblems defined on subnetworks. In practice, for typical problem instances, these subnetworks are much smaller than the overall network, which is the key improvement offered by our solution. However, in the very worst case, our algorithm still meets the theoretical complexity of the centralized solution.

Related decentralized methods formulate routing subproblems by partitioning the environment (utilizing e.g., Voronoi diagrams). However, such methods are not quite suitable for typical dynamic multi-vehicle routing scenarios on large networks with a large number of vehicles and a non-uniform distribution of demands. Furthermore, as opposed to this type of decentralized, our approach can ensure the global optimum in case the routing algorithm applied to a particular subproblem guarantees optimum. Note that different existing heuristics solving a multi-vehicle VRP can be also applied on the subproblems, resulting in even more increased efficiency of the approach, while sacrificing the optimality.

Extension to complex demands: This work can be easily extended to handle scLTL (syntactically co-safe Linear Temporal Logic) formulas that can rigorously capture demands, such as "Pick me up at location A, take me home, and on the way first stop by at a shopping mall and then at my child's kindergarden." Loosely speaking, instead of *pick-up* and *drop-off*, a demand would be characterized by an scLTL formula over Π and the routing algorithm from [14] would be used for finding traces of individual vehicles respecting a given assignment.

Extension to limited capacity of vehicles: For the simplicity of presentation, we have assumed that the vehicles have unlimited capacity. This assumption can be easily relaxed by considering the respective constraint representing the limit on the vehicle's capacity in the selected routing algorithm. For instance, the algorithm from [14] builds on a MILP problem formulation that can be straightforwardly enhanced in the desired way.

Future work: Our future work will focus on dynamically changing weights of the WTS associated with changing traffic in road networks. We believe that we will be able to address this aspect through FMM, that can be seen as an extension of Dijkstra's to the continuous domain. With the use of anisotropic velocities, we will combine the weights in the graph, signifying the traversal time of the roads as a function of their lengths, with the movement speed of the waveform as a function of e.g., traffic density at a given time and point. This will allow us to effectively handle dynamic weights in the graph, without having to alter the graph in real-time. Another direction for future research is to exploit possible statistical knowledge regarding pop-up rates of demands in different locations of the road network.

Acknowledgements: We would like to thank Cristian Ioan Vasile for fruitful discussions on the topic.

REFERENCES

[1] S. N. Kumar and R. Panneerselvam, "A Survey on the Vehicle Routing Problem and Its Variants," *Intelligent Information Management*, vol. 04, no. 03, pp. 66–74, 2012. [Online]. Available: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/iim.2012.43010

[2] H. N. Psaraftis, "Dynamic Vehicle Routing Problems," *Vehicle Routing: Methods and Studies*, pp. 223–248, 1988.

[3] S. F. Ghannadpour, S. Noori, R. Tavakkoli-Moghaddam, and K. Ghoseiri, "A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application," *Applied Soft Computing Journal*, vol. 14, no. PART C, pp. 504–527, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2013.08.015

[4] 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. [Online]. Available: http://dx.doi.org/10.1016/j.ejor.2009.04.024

[5] D. Sáez, C. E. Cortés, and A. Núñez, "Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering," *Computers and Operations Research*, vol. 35, no. 11, pp. 3412–3438, 2008.

[6] E. Hyytiä, A. Penttinen, and R. Sulonen, "Non-myopic vehicle and route selection in dynamic DARP with travel time and workload objectives," *Computers and Operations Research*, vol. 39, no. 12, pp. 3021–3030, 2012.

[7] M. Pavone, E. Frazzoli, and F. Bullo, "Distributed and Adaptive Algorithms for Vehicle Routing in a Stochastic and Dynamic Environment," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1259–1274, 2011. [Online]. Available: http://arxiv.org/abs/0903.3624

[8] A. Arsie, K. Savla, and E. Frazzoli, "Efficient routing algorithms for multiple vehicles with no explicit communications," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2302–2317, 2009.

[9] R. Claes, T. Holvoet, and D. Weyns, "A decentralized approach for anticipatory vehicle routing using delegate multiagent systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 364–373, 2011.

[10] E. Frazzoli and F. Bullo, "Decentralized algorithms for vehicle routing in a stochastic time-varying environment," *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 4, pp. 3357–3363, 2004.

[11] M. Pavone, E. Frazzoli, and F. Bullo, "Decentralized algorithms for stochastic and dynamic vehicle routing with general demand distribution," *Proceedings of the IEEE Conference on Decision and Control*, pp. 4869–4874, 2007.

[12] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Proceedings of the 16th international conference on Hybrid systems: computation and control - HSCC '13*. New York, New York, USA: ACM Press, 2013, p. 1. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2461328.2461330

[13] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal motion planning for hybrid systems in partially unknown environments," *Hscc*, p. 353, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2461328.2461380

[14] J. Tumova, S. Karaman, C. Belta, and D. Rus, "Least-violating planning in road networks from temporal logic specifications," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–9.

[15] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations," *Journal of computational physics*, vol. 79, no. 1, pp. 12–49, 1988.

[16] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path planning for mobile robot navigation using voronoi diagram and fast marching," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 2376–2381.

[17] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec 1959. [Online]. Available: https://doi.org/10.1007/BF01386390