Autonomous Vehicle Dispatching for Person Evacuation

Xin Liu, Yangming Zhao and Chunming Qiao
Department of Computer Science and Engineering, University at Buffalo

***Abstract*—The rapid development of Autonomous Vehicle (AV) technologies provides a new opportunity to evacuate vulnerable persons from their residences to shelters when some emergency event happens. One of the most important objectives is to minimize the evacuation time, which depends on the order to evacuate persons and which shelter each person is delivered to. We first formulate this AV dispatching problem as an Integer Linear Programming (ILP) model and prove this problem is NP-hard. Due to the problem hardness, an efficient algorithm based on Dynamical Programming (DP) is proposed. Through extensive simulations, we find that our algorithm can reduce the evacuation time by 58% compared with a greedy based algorithm, which is the common method to solve the Traveling Salesman Problem (TSP), a special case of our AV dispatching problem.**

I. Introduction

In the past decade, the Autonomous Vehicle (AV) technologies have seen rapid development [1]. Some of the AVs can drive itself by giving the source, destination and where to stop for a certain amount of time [2]. By leveraging these technologies, researchers have proposed to evacuate persons from their residences to shelters when some emergency event happens.

When dispatching AVs to evacuate persons, one objective is to minimize the time to complete the entire evacuation procedure, i.e. send all the persons who need to be transported to the shelters as soon as possible. The other objective is to evacuate all the persons with the minimum number of AVs that are required to evacuate all the persons before a given deadline. Though there are different objectives to pursue, the most important objective is the first one, i.e. to determine the order to pick up persons and which shelter each person should be sent to, such that the time for the entire evacuation procedure can be minimized. If the objective is to minimize the number of AVs, we can leverage the binary search method to find the minimum number of AVs that are required to evacuate all persons before the deadline. Accordingly, in this paper, we mainly focus on how to minimize the evacuation time with the given number of AVs, by determining the order to pick up persons and figuring out the optimal shelter for each person.

To minimize the evacuation time, we first formulate the problem as an Integer Linear Programming (ILP) model. However, the ILP model is intractable even in a small scale system, e.g. with dozens of persons and 3 AVs. After analyzing the problem, we find that the AV dispatching problem for person evacuation is a more general case of the Traveling Salesman Problem (TSP), which is a well known NP-hard problem. Accordingly, an efficient heuristic is required to solve the problem in a large scale system. To this end, we approach the problem based on divide and conquer. At first, we cluster all the persons into multiple small groups, and each group of persons can be carried by a single AV. Then, we figure out the order to pick up persons in each group, and the order to serve groups. Both ordering sub-problems can be determined with Dynamic Programming (DP). Through extensive simulations, it is shown that our algorithm can reduce the evacuation time by up to 58% compared with the greedy based algorithms.

This work is supported in part by NSF grant No.1626374 and US Department of Transportation grant DTRT13-G-UTC48.

The main technique contributions of our work is summarized as follows:

- Formulate the problem to solve as an Integer Linear Programming model.
- An efficient algorithm to solve the AV dispatching problem.
- Extensive simulations to show the effectiveness of our algorithm.

The rest of the paper is organized as follows. Section II briefly reviews the related works. Section III formulates the problem to solve as an Integer Linear Programming model and analyzes the complexity of our problem. In Section IV, we propose DP based algorithm to solve the AV dispatching problem. We evaluate the performance of the proposed algorithm by comparing it with three baseline algorithms in Section V, followed by a conclusion in Section VI.

II. Related Work

There are a number of related works on vehicle dispatching and emergency evacuation planning scheme. We review the most related ones.

In [3], a bi-level optimization model is used to determine the waiting locations and corresponding shelters. The main contribution is to propose a transit-based emergency evacuation plan and dispatch rescue buses toward the combinatorial locations. [4] design the optimal path for buses to pick up evacuees to leave a disaster scene while assume buses only pick up persons at one bus stop before deliver to a shelter which is a many to one approaches, [5] design path for weak users to evacuate in two approaches: one origin to one destination as shortest path problem and one origin to many destinations as vehicle routing problem, [6] are able to optimize paths and routes for emergency vehicles while taking into account time-dependence and optimizing the wait time at nodes, and specified the factors that affect one to one and many to one approaches. However, when solving the AV-to-person assignment sub-problem, we design many-to-many mapping approaches, which is more realistic and complex. [7] and [8] propose algorithms and frameworks on taxi carpooling dispatch problem, these works make contribution on making taxi dispatch more efficient in terms of maximize successful carpooling ratio while assuming at most one transfer is allowed. However, when solving dispatching problem in emergency evacuation, large capacities and multiple transfers could reduce total evacuation time. [9] design a comprehensive transit planning framework for hurricane evacuation, while the authors assume every one is assigned to the closest shelter prior to the route generation. However, this approach may not get the optimal solution in consideration of total evacuation time. To the best of our knowledge, there have been no existing theoretical works on emergency evacuation plans by using AV, while assume having multiple vehicles to rescue multiple persons to multiple shelters.

III. Formulation and Analysis

In this section, we first introduce the evacuation system to study, and then formulate the problem to solve as an Integer Linear Programming (ILP) model for formal definition. We will find that this ILP model cannot be solved even in a very small system. To study why this happens, we prove that the problem is NP-hard.

A. System Model and Notations

In our evacuation system, we assume there are a set of Autonomous Vehicles (AVs) $\boldsymbol{A}$, a set of persons who are needing evacuation $\boldsymbol{P}$, and a set of shelters to accommodate the evacuated persons $\boldsymbol{S}$. At the beginning of the evacuation, we know the position of each AV, person and shelter. Accordingly, we can calculate the AV cruising time between arbitrary two positions. Say t_{ij} is the AV cruising time from position i to position j, where $i, j \in \boldsymbol{A} \cup \boldsymbol{P} \cup \boldsymbol{S}$. For example, if $i \in \boldsymbol{P}$ and $j \in \boldsymbol{S}$, t_{ij} is the AV cruising time from the position of person i to shelter j.

As the first step to study the AV dispatching problem for person evacuation, we consider two types of constraints: 1) AV compatibility constraint, i.e. some persons can only be picked up by given set of AVs due to the equipment issue. We use $\boldsymbol{A}(p)$ to denote the set of AVs that are equipped to carry person p; 2) AV capacity constraint, i.e. each AV can carry at most c persons at the same time. We leave the shelter compatibility and capacity constraints to the further works.

B. Integer Linear Programming Formulation

Throughout the formulation, we use x_{ij}^k to indicate if AV i is at position j for its k^{th} stop, and y_{uv}^i to denote if AV i goes from position u to position v. The inherent relationship between these two variables can be formulated as

$$\begin{gathered}\frac{x_{iu}^k + x_{iv}^{k+1} - 1}{2} \leq y_{uv}^{ik} \leq \frac{x_{iu}^k + x_{iv}^{k+1}}{2} \\ y_{uv}^{ik} \leq y_{uv}^i\end{gathered} \tag{1}$$

for all u, v, i, k, where y_{uv}^{ik} is an auxiliary variable to denote if position u is AV i's k^{th} stop, while position v is its $k+1^{th}$ stop. To form a valid cruising trail, y_{uv}^{ik} should satisfy

$$\sum_v y_{vu}^{ik} - \sum_v y_{uv}^{i,k+1} = \begin{cases} -1 & \text{if } k = 0 \\ 1 & \text{if } k = K \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where K is the maximum number of positions each AV can stop by. This parameter can be set as $2|\boldsymbol{P}|$, because the longest possible trail is achieved when the capacity is 1, in which case the AV should stop by two positions (one for person and one for shelter) to pick up each person. As the initial condition, we have

$$x_{iu_0}^0 = 1 \tag{3}$$

if u_0 is the initial position of AV i. Since we should evacuate all the persons, we have

$$\sum_i \sum_k x_{ij}^k = 1 \tag{4}$$

for all $j \in \boldsymbol{P}$. To formulate the AV capacity constraint, we introduce a variable Δ_{ik} to denote the change of the person number on AV i at the k^{th} stop. Then, it satisfies

$$\Delta_{ik} = \begin{cases} \sum_j x_{ij}^k & \text{if } u \in \boldsymbol{P} \\ -\sum_{j, l<k} x_{ij}^l & \text{if } u \in \boldsymbol{S} \end{cases} \tag{5}$$

In every step, the number of persons on AV i should not exceed c. That is

$$\sum_{l \leq k} \Delta_{ik} \leq c \tag{6}$$

for all $k \leq K$. To ensure all the persons are dropped off at a shelter, there must be

$$\sum_{l \leq K} \Delta_{ik} = 0 \tag{7}$$

To formulate the compatibility constraint, we should fix

$$y_{uv}^i = 0 \quad \text{if } v \in \boldsymbol{P} \text{ and } i \notin \boldsymbol{A}(v) \tag{8}$$

Leveraging y_{uv}^i, the cruising time of AV i, T_i, can be calculated by

$$T_i \geq \sum_{u,v} t_{uv} y_{uv}^i \tag{9}$$

The evacuation process completes if and only if the latest AV completes its evacuation, and hence, we should minimize $\max_i\{T_i\}$. In summary, the ILP model to minimize the evacuation process can be formulated as following AV Dispatching Problem (AVDP):

$$\boxed{\begin{array}{ll} \text{minimize} & \max_i\{T_i\} \\ \text{Subject to:} & (1) \text{ — } (9) \end{array}}$$

C. Complexity Analysis

Though we have formulated the AVDP as a ILP model, it is difficult to find the optimal solution in a timely manner. In fact, we cannot solve AVDP with 15 persons, 3 AVs and 2 shelters on a desktop carrying Intel i7-2600 CPU and 8 GB memory. Following theorem shows that AVDP is NP-hard since Traveling Salesman Problem (TSP) is only a special case of AVDP.

Theorem 1. *Traveling Salesman Problem (TSP) is a special case of AVDP, and hence AVDP is NP-hard.*

Proof: To construct a special case of AVDP, we assume there is only one AV, and the shelter is very far from all the persons that need to be picked up. In addition, the capacity of the AV is enough to take all the persons. In this case, to minimize the evacuation time, we should find the shortest path to traverse all the persons, which is exactly the TSP. ■

IV. Algorithm Design

Since AVDP is NP-hard, we need an efficient algorithm to find a near optimal solution in a timely manner. We design such algorithm following the thought of divide and conquer. In Section IV-A, we first introduce the algorithm at a high level and show the relationship among all the sub-problems. Then, we solve these sub-problems one by one based on dynamic programming in Section IV-B – IV-E. At the end, we present some discussions on this algorithm and extend it to solve the problem that minimize the number of AVs with evacuation deadline constraint.

A. Algorithm Overview

Since we can solve a small scale NP-hard problem in a timely manner, we first cluster all the persons into several groups. Each group contains at most c, i.e. AV capacity, persons. Then, we first determine the order to pick up the person in each group, and figure out the order to serve each *group*.

The order to pick up persons in a group is obviously a TSP. When the size of each group is not too large, we can solve it through dynamic programming. For the order to serve each group, following corollary shows that this is NP-hard since TSP is only a special case.

Corollary 1. *TSP is a special case of determining the order to serve each group of persons in AVDP.*

Proof: When there is only one AV and the number of shelters is the same as the number of groups, each shelter is allocated at the exit of each group. Then this is a TSP to minimize the evacuation time. ■

If there are multiple AVs in the system, the problem would be harder. Accordingly, we first assume there is a single AV and find a path to minimize the evacuation time. Then, we split the single path into multiple paths, one for each AV, to further finalize the evacuation process.

In the following subsections, we will discuss algorithms to solve these sub-problems in detail. For clear presentation, we summarize the relationship among the proposed algorithms in Fig. 1. We first leverage Algorithm 1 to cluster all the persons into multiple groups. The trail to pick up persons in each group is derived by Algorithm 2. For the inter-group trail, we first leverage Algorithm 3 to find one trail under the assumption that there is only one AV in the system, and then Algorithm 4 is used to split the single trail into multiple ones each of which is for one AV.

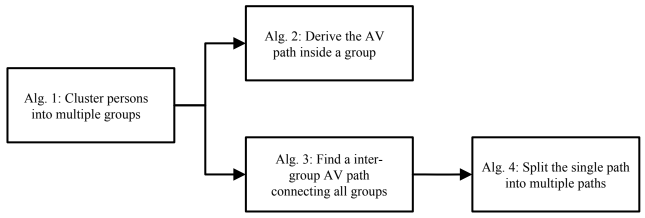
Alg. 1: Cluster persons into multiple groups
Alg. 2: Derive the AV path inside a group
Alg. 3: Find a inter-group AV path connecting all groups
Alg. 4: Split the single path into multiple paths

Fig. 1. Relationship among proposed algorithms.

B. Person Clustering

To cluster persons into multiple groups, we can design a k-means like algorithm. Suppose we are to divide persons into $G \geq \lceil \frac{|\boldsymbol{P}|}{c} \rceil$ groups, first, we select G centers for each group, and then assign each person to the group whose center is closest to him/her. After that, we update the center for each group to the position that has the minimum cruising time sum to all other persons in the same group. Formally, for a group C_g, its center should be updated to

$$i = \arg\min_i \sum_{j \in C_g} t_{ij} \tag{10}$$

After the cluster center updating, all the persons are reassigned to different groups. This iteration will end when the cluster center does not change any more.

Different from conventional k-means algorithm, in each group, there should be at most c persons in order to satisfy the AV capacity constraint. Accordingly, when the size of a group achieves c, no person can be assigned to it, and corresponding person should be assigned to the group whose center is the second nearest one.

Based on above discussions, we summarize the algorithm to cluster persons into multiple groups in Algorithm 1. In this algorithm, Lines 5–8 are assigning persons to different group, while Lines 9–11 update the center for each group.

C. Trail inside a Cluster

When all the persons are divided into multiple groups, we should find a way to cruise in each group such that the time to pick up all the persons in a group can be minimized. As we discussed earlier, this can be treated as a TSP. When there are only tens of persons in each group, we can solve it with Dynamical Programming (DP) in a timely manner.

Say $\boldsymbol{R}$ is the set of persons that have not been picked up, and $T[\boldsymbol{R}][i]$ is the minimal time to pick up all the persons in $\boldsymbol{R}$ from the start position i. Then, we have

$$T[\boldsymbol{R}][i] = \min_{k \in \boldsymbol{R}} \{t_{ik} + T[\boldsymbol{R} - k][k]\} \tag{11}$$

Algorithm 1: Person Clustering

```
Input: The AV cruising time between any two persons $t_{ij}$, number of groups $G$ and the maximum capacity of each group $c$
Output: Persons in each group $\{\boldsymbol{C}_g\}$
1: Initialize $C_g \leftarrow \Phi$ for all $g = 0, 1, \ldots, G-1$
2: Randomly select $G$ centers $\{a_g\}$
3: while true do
4:     clear $\{C_g\}$
5:     for each person $i$ do
6:         $g = \{\arg\min_g t_{i,a_g} | |C_g| < c\}$
7:         $C_g \leftarrow C_g \cup i$
8:     end for
9:     for each group $C_g$ do
10:        $a_g^* \leftarrow \arg\min_i \sum_{j \in C_g} t_{ij}$
11:    end for
12:    if $a_g^* = a_g$ for all $g$ then
13:        break
14:    else
15:        $a_g^* \leftarrow a_g$ for all $g$
16:    end if
17: end while
18: return $\{\boldsymbol{C}_g\}$
```

As the initial condition of this DP problem, the time to pick up all the persons in an empty set should be 0, i.e.

$$T[\boldsymbol{R}][k] = 0 \tag{12}$$

if $\boldsymbol{R} - k = \Phi$. Such DP based algorithm is shown in Algorithm 2. In this algorithm, the map $\boldsymbol{M}$ is used to record the minimal time to pick up different sets of persons from different start point, while $\boldsymbol{N}$ stores the next person to pick up given different sets of unpicked persons and different AV locations.

There is one remaining issue in Algorithm 2 that how to determine the start position. If each AV starts at different positions, the cruising time should be different. A naive method is to try all the positions one by one and pick up the one that derives minimal cruising time. This methods should invoke Algorithm 2 many times and is time consuming. To solve this problem, we introduce a dummy start position and set the cruising time between the dummy start position and arbitrary person in the given group to be 0. In this case, we start from the dummy start position, and can derive the optimal solution by invoking Algorithm 2 only once.

D. Trail for a Single AV

The next question is to find out how to pick up each group of persons and drop off at which shelter. For convenience, since persons are clustered into multiple groups, from now on, t_{ij} is the AV cruising time from a group to shelter or a shelter to a group, and $\boldsymbol{R}$ is the set of groups that have not been picked up.

Algorithm 2: Find trail to pick up persons in a group

```
Input: The AV cruising time between any two persons $t_{ij}$, the set of persons that need to be picked up in a group $\boldsymbol{R}$, start position $s$
Output: The order to pick up persons, $\boldsymbol{Q}$, such that the cruising time can be minimized, and the cruising time $T$
1: Initialize two maps $\boldsymbol{M}$ and $\boldsymbol{N}$, whose key consists of the start position and the set of persons unpicked, denoted as $(\boldsymbol{R}, i)$, $\boldsymbol{M}[(\Phi, k)] \leftarrow 0$, $\boldsymbol{N}[(\Phi, k)] \leftarrow k$, $\boldsymbol{Q} \leftarrow \Phi$
2: for all $k \in \boldsymbol{R}$ do
3:     if $\boldsymbol{M}$ does not have key $(\boldsymbol{R} - k, k)$ then
4:         Recursively call Algorithm 2 with input set of unpicked person $\boldsymbol{R} - k$ and start position $k$, say the minimal cruising time is $T'$
5:         $M[(\boldsymbol{R} - k, k)] \leftarrow T'$
6:     end if
7: end for
8: $T \leftarrow \min_{k \in \boldsymbol{R}} \{t_{ik} + T[\boldsymbol{R} - k][k]\}$
9: $n \leftarrow \arg\min_{k \in \boldsymbol{R}} \{t_{ik} + T[\boldsymbol{R} - k][k]\}$, $\boldsymbol{N}[(\boldsymbol{R}, k)] \leftarrow n$
10: while $\boldsymbol{R} \neq \Phi$ do
11:     $n \leftarrow N[(\boldsymbol{R}, s)]$, $\boldsymbol{Q} \leftarrow \boldsymbol{Q} \cup n$, $\boldsymbol{R} \leftarrow \boldsymbol{R} - n$
12: end while
13: return $\boldsymbol{Q}$, $T$
```

Again, we use $T[\boldsymbol{R}][i]$ to denote the minimal time to traverse all the groups from start position i, the DP model for only one available AV can be formulated as

$$T[\boldsymbol{R}][i] = \min_{k \in \boldsymbol{R}, j \in \boldsymbol{S}} \{t_{ik} + t_{k,j} + T[\boldsymbol{R} - k][j]\} \tag{13}$$

and the initial condition of this DP model should be

$$T[\boldsymbol{R}][i] = t_{ik} + \min_{j \in \boldsymbol{S}} \{t_{kj}\} \tag{14}$$

if k is the only element in $\boldsymbol{R}$. In this formulation, the start position i can be the position of arbitrary AV. The algorithm to calculate the inter-group trail is summarized in Algorithm 3. It should be noted that the cruising time returned by Algorithm 3 only includes the time to cruise between groups and shelters, but not the time to pick up persons in each group.

E. Trail Splitting for Multiple AVs

To address the multiple AVs issue, we should split the single trail obtained in last subsection into multiple ones, one for each AV. In this way, the evacuation process can be parallelized, and it can be speed up. To solve this problem, again, we leverage the DP. Given the traverse order derived in last subsection $Q = \{q_1, q_2, \ldots, q_n\}$, a_{ij} is the time to traverse from q_i to q_j including both the time for inter-group cruising and intra-group cruising. $T[i][m]$ is the minimal evacuation time to complete the cruising with m AVs starting at q_i. Then, the DP model can be formulated as

$$T[i][m] = \min_{j > i, q_j \in \boldsymbol{S}} a_{ij} + T[j][m-1] \tag{15}$$

Algorithm 3: Find single inter-group trail

```
Input: The AV cruising time between groups and shelters $t_{ij}$, the set of groups $\boldsymbol{R}$, start position $s$
Output: The order to traverse groups, $\boldsymbol{Q}$, such that the cruising time can be minimized, and the cruising time $T$
1: for every $k \in \boldsymbol{R}$ and $j \in \boldsymbol{S}$ do
2:     if $T[\boldsymbol{R} - k][j]$ has not been calculated then
3:         Recursively calculate $T[\boldsymbol{R} - k][j]$
4:     end if
5: end for
6: $T \leftarrow \min_{k \in \boldsymbol{R}, j \in \boldsymbol{S}} \{t_{sk} + t_{k,j} + T[\boldsymbol{R} - k][j]\}$
7: Form the traverse position order $\boldsymbol{Q}$
8: return $\boldsymbol{Q}$, $T$
```

Algorithm 4: Find single inter-group trail

```
Input: The combined single trail $Q$ derived by Algorithm 3, the set of AVs $\boldsymbol{A}$
Output: Path of each AV during the evacuation process
1: for every $j > 1, q_j \in \boldsymbol{S}$ do
2:     if $T[j][|\boldsymbol{A}| - 1]$ has not been calculated then
3:         Recursively calculate $T[j][|\boldsymbol{A}| - 1]$
4:     end if
5: end for
6: $T \leftarrow \min_{j > i, q_j \in \boldsymbol{S}} a_{ij} + T[j][m - 1]$
7: Form $|\boldsymbol{A}|$ subpath based on the solution of DP model
8: Assign each AV to the subpath starting at the group nearest to it
9: return The path of each AV
```

with the initial condition

$$T[i][1] = a_{in} \tag{16}$$

When we obtain the multiple trails based on above DP model, the start point should be a shelter. We should substitute the first hop to be the path from an AV to the group. In this step, we can directly assign each AV to the trail start at the group nearest to it. Accordingly, the algorithm to form the final evacuation trail for all the AVs can be summarized as Algorithm 4.

F. Discussions

In this section, we discuss some implementation and application issues.

Minimize the number of AVs Sometimes, there is a deadline for the evacuation procedure. One way to meet the deadline is to design a new algorithm. However, our algorithms can also be applied in this situation. We can first leverage all the AVs to calculate the minimal evacuation time. If the minimal evacuation time is less than the deadline requirement, we can use a binary search fashion to get the least number of AVs that satisfy the deadline. If the minimal evacuation time is larger than the deadline requirement, we cannot meet the deadline in any case.

Algorithm scalability In our algorithms, we leverage DP to solve several TSP like problems. However, DP based algorithm can solve TSP problem with about 25 points on current commodity desktop [10]. Accordingly, our current algorithm can solve a problem with about 500 persons, e.g. clustering all the person into 20 groups and each group has 25 persons. For most of the cases, this scale is enough for the emergency evacuation. If there is a much larger use case, we can further cluster all the groups into multiple pods, and determine the AV intra-pod route first. With this method, we can extend the problem scale that our algorithm can support to about 10,000 persons.

V. Performance Evaluation

In this section, we evaluate our algorithm with synthetic data. In the simulations, we randomly allocate all persons in a 10 km $\times$ 10 km area and assume the distances between them are approximately proportional to their Euler distance, AV's speed is 60 km/h. Then, we calculate the time to evacuate all the persons by implementing the algorithms proposed above. For comparison purpose, we also text following algorithms.

Greedy Algorithm Say we have m AVs and n persons, first we divide all the persons into m groups, and each AV should pick up all the persons in the same group. For each AV, it always selects the nearest person to pick up, until the capacity constraint achieves. At that time, this AV takes all the persons on it to the nearest shelter and continues the evacuation.

Parallel Greedy Algorithm Different from previous Greedy Algorithm, in Parallel Greedy Algorithm, we always assign the person closest to the feasible AV and consolidate this assignment. Whenever the capacity constraint of an AV is achieved, it should go to the nearest shelter. This procedure is repeated till all the persons are evacuated.

Parallel Greedy Algorithm with 2-Opt local search To further improve the performance of Parallel Greedy Algorithm, we introduce the 2-opt local search. In this scheme, based on the results derived by Parallel Greedy Algorithm, we randomly select two persons and swap the order to pick them up. If the evacuation time can be reduced, keep this swapping, otherwise, keep the original order. This local search should be executed multiple times to dig larger optimization space.

Experiments and Evaluation We conduct five sets of tests to evaluate the performance of our algorithm. In these tests, the default setting is 400 persons, 20 AVs with capacity 20 and 4 shelters. By changing the parameters, we derive the simulation results shown in Fig.2 – 5.

Fig.2 shows how the performance of our algorithms changes with the number of persons. From this figure, we can observe that the evacuation time under all four algorithms is linearly growing with the increase of number of persons, while our algorithm outperforms all other algorithms. The greedy algorithm avoids the situation that the AV go to pick up a further person and deliver to a further shelter, yield to a decent result. However, it does not consider the problem as a

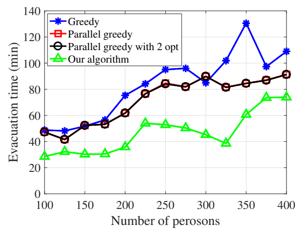

Fig. 2. Performance over persons.

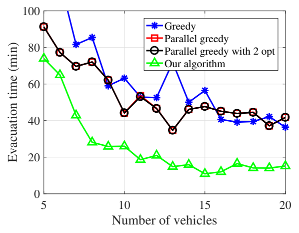

Fig. 3. Performance over AVs.

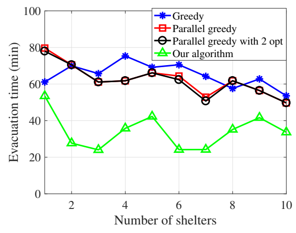

Fig. 4. Performance over shelters.

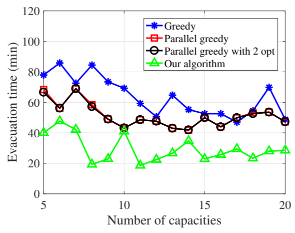

Fig. 5. Performance over capacities.

whole, sometimes pick up a further person might result better in whole evacuation process. The parallel greedy performs better than greedy because it find the next available person and shelter on the fly instead of group up and assign before the process. While the 2opt local search does not improve parallel greedy much. The main reason is that switch only 2 point based on parallel greedy is hard to get improvement, switch multiple points may potentially get better results, but also may increase the running time for k-opt. Overall, our algorithm outperforms greedy and parallel greedy algorithm with and without 2-opt by 35% and 44%, respectively.

Fig.3 shows how the number of AVs impacts the performance of different algorithms. The result shows that all four algorithms can get all persons evacuated in a relatively short time, while our algorithm outperformed by 53% and 42% compare to greedy and parallel greedy with or without 2-opt.

In Fig.4, the number of shelters is changed to evaluate the algorithm performance. It is worth noting that in this set of evaluation, there are only 200 persons, rather than 400 as in previous two sets of evaluations, since in a commodity desktop, we are only able to solve TSP for 25 groups with DP, so when we have 10 shelters, we can only have 15 groups for the persons, which is 300 persons. The result shows that all four algorithms can get all persons evacuated in a relatively short time, while our algorithm outperformed by 47% and 45% compare to greedy and parallel greedy with or without 2-opt. The reason that all four solution are stable over the increase of shelters is that the difference is only to choose over the shelters, so even if random pick a further shelter, the difference is not large and only count once. So the main reason that our algorithm performs better is in the intra-group shortest path part of the solution.

In Fig.5, we show how the evacuation time changes with the AV capacity. Due to the same reason discussed above, we only test the case that there are 100 persons in the area. When the AV capacity is 5, there are 20 groups for persons which can support 100 in total. The result shows that all four algorithms can get all persons evacuated in a relatively short time, while our algorithm outperformed by 58% and 52% compare to greedy and parallel greedy with or without 2-opt.

For the last group of evaluation, we set up a time constraint for the evacuation process and compare the least AVs requirement among all different algorithms, we consider 400 hundreds of persons to be evacuated with AV bus which has 20 capacities to 4 shelters. From the Table I, we can see that with 60 minutes constraint, all four algorithms performs well with only 7 to 11 AVs. When the time steps is limited to 20 minutes, our algorithm can still solve the problem with only 13 AVs, while Greedy needs 50 AVs.

TABLE I
AVs REQUIREMENT WITH TIME CONSTRAINT

Time (Min)	Our Alg.	Greedy	Parallel Greedy.	Parallel Greedy-2opt
20	13	50	42	42
40	8	17	14	14
60	7	11	10	10
80	5	9	6	6
100	5	7	5	5

VI. Conclusions

As autonomous vehicles (AVs) have developed significantly over last decade, we believe that using AVs to evacuate vulnerable people in an emergency event will be very common and efficient in the near future. In this paper, we have formalized the problem as an ILP model and proved the complexity. In addition, we have divided the problem into multiple sub-problems and proposed efficient algorithms based on dynamical programming to solve them. Furthermore, we have validated our design through extensive simulations and showed that our algorithm can achieve a significant improvement compare to greedy algorithm and parallel greedy algorithm with or without 2-opt local search.

References

[1] U.S. Department of Transportation, "Usdot automated vehicles activities," *https://www.transportation.gov/AV*.

[2] Olli Bus, "Meet olli," *https://localmotors.com/meet-olli/*.

[3] C.-C. Chen and C.-S. Chou, "Modeling and performance assessment of a transit-based evacuation plan within a contraflow simulation environment," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2091, pp. 40–50, 2009.

[4] H. Zheng, "Optimization of bus routing strategies for evacuation," *Journal of Advanced Transportation*, vol. 48, 2014.

[5] A. Q. . A. P. A. Vitetta, "Safety of users in road evacuation: algorithms for path design of emergency vehicles," *WIT Transactions on The Built Environment*, 2008.

[6] A. Polimeni and A. Vitetta, "Dynamic vehicle routing in road evacuation: A model for route design," *WIT Transactions on the Built Environment*, vol. 116, pp. 627–638, 2011.

[7] Y. Hou, X. Li, Y. Zhao, X. Jia, A. W. Sadek, K. Hulme, and C. Qiao, "Towards efficient vacant taxis cruising guidance," *2013 IEEE Global Communications Conference (GLOBECOM)*, 2013.

[8] Y. Hou, X. Li, and C. Qiao, "Tictac: From transfer-incapable carpooling to transfer-allowed carpooling," *2012 IEEE Global Communications Conference (GLOBECOM)*, 2012.

[9] R. Swamy, J. E. Kang, R. Batta, and Y. Chung, "Hurricane evacuation planning using public transportation," *Socio-Economic Planning Sciences*, vol. 59, pp. 43 – 55, 2017.

[10] C. Chauhan, R. Gupta, and K. Pathak, "Survey of methods of solving tsp along with its implementation using dynamic programming approach," *International Journal of Computer Applications*, 2012.