A Simulation System for Autonomous Carts

Boumediene Belkhouche\textsuperscript{1}, Abderrahmane Lakas\textsuperscript{2}

\textsuperscript{1}Computer Science and Software Engineering Department
College of Information Technology, United Arab Emirates University, Al Ain, UAE
b.belkhouche@uaeu.ac.ae

\textsuperscript{2}Computer and Network Engineering Department
College of Information Technology, United Arab Emirates University, Al Ain, UAE
alakas@uaeu.ac.ae

Abstract: We elaborate a simulation model of a Cooperative Autonomous Reactive Taxi System (CARTS) to investigate a software architecture that supports decision decentralization and the impact of such an architecture on optimization issues associated with the pickup and delivery problem (PDP). Our PDP models cart transportation services, whose use is becoming highly popular within campuses. Given dynamic stochastic variables that include a set of carts, a set of customers, a set of stations (e.g., bus stop), and a set of transport missions, we devise autonomous intelligent carts capable of effectively carrying out these missions to satisfy the quality of service requirements and any associated constraints. To assess our approach, we develop a real-time stochastic system to simulate different aspects, such decentralized decision-making, request scheduling and allocation, route construction, and cooperative behavior. The simulation results allow us to carry a cost-efficiency analysis by determining the correlation between the number of carts required to service the daily requests, the carts seat capacity needed in each cart, and the arrival rates of the passengers in each stop.

Keywords: pickup and delivery problem; scheduling; route planning; decentralized control; cart transportation; software architecture.

I. INTRODUCTION

To promote sustainability and to encourage people to protect the environment, a growing number of communities have bicycle stations all over town. These bicycles are available to people for free use to move from one place to another. On many campuses, and various other places (e.g., airports), electric golf carts are used to transport passengers for short distances from one point to another (pickup and delivery of passengers). Because of the lack of automation and decision-making sophistication, their performance is suboptimal based on time, scheduling, passenger satisfaction, availability, and human resources. Thus, we would like to develop a system in which to endow these carts (that we call cooperative autonomous reactive taxi system, CARTS) with the spirit of cooperation among themselves and the computing capabilities to make optimal decisions on their own. Our idea is twofold: (1) to develop and integrate novel logistic algorithms (scheduling, allocation, route planning) based on cooperation, communication, navigation, and real-time decision-making, to optimize the quality of services rendered by these carts and (2) to demonstrate the feasibility of our solution by implementing it using robot carts.

As shown in Figure 1, we will illustrate our discussion using a familiar scenario to any campus community. In the map, we identify the following: (1) physical infrastructure (roads, buildings, passenger stations); (2) communication infrastructure (WiFi, antennas); (3) passengers waiting at stations; and (4) carts picking up and delivering passengers. Then, consider the current situation on any campus. We see carts roving all over taking passengers from one station to another. Typically, cart owners have at their disposal their carts; others may have to request a specific ride. A major issue is that some of the carts are idle at some station most of the time, whereas passengers are waiting for an undetermined amount of time at another station. At other times, the majority of carts may be all sitting idle at just one station. Real-time information about carts, drivers, and passenger requests is not available leading to ad-hoc and suboptimal services offered by these carts, impacting passengers and their productivity within the campus. Our approach is to automate the logistics and to give autonomy to the carts so that they can provide optimal services.

Fig. 1. CARTS Framework.
The question we address is: “How can we fulfill passengers transport requests by minimizing total service time (waiting plus transport) and optimizing their satisfaction with the service” [1]. We state our problem as follows:

Given a 2D space in which we have a set of N carts pick up and deliver M passengers from/to a set of K locations. A location can serve as a pickup and/or delivery. Each cart has a number of seats S and a battery life with finite energy E. A number D of dynamic transportation requests (pickup/delivery) is to be fulfilled by assigning carts to requests. The assignment process requires scheduling, allocating passengers to carts, and route planning. Since requests arrive dynamically and randomly, the assignment process is continuous and is carried accordingly. Carts evaluate received requests and select among themselves the best routes for assignment. Subsequently, each cart carries out autonomously its assigned task, while being responsive to new assignments. The variables N, M, K, S, D, and E are assumed to be stochastic, that is, having a random distribution that we assume to be a Poisson distribution.

Various solutions for the PDP have been, and are still being, proposed from an operations research perspective, that is, using traditional optimization techniques ([2], [3], [4]) and heuristics ([5], [6]). Because of the NP-hard complexity, proposed solutions tend to work only for small-size problems. More importantly, the available solutions are based on a single centralized server and the vehicles are mere traditional vehicles without any computing power. Our proposal integrates distributed computation by endowing our vehicles with cooperative computing capabilities to allow each one of them to devise its own optimal solution for a given situation.

II. PROBLEM OVERVIEW AND FORMULATION

Our problem formulation specializes the pickup and delivery problem (PDP) to on-campus cart transportation ([1], [7], [8]). Items (goods, people) are picked up at locations and delivered at other locations. The objective function is to optimize the quality of service. For example, in the case of passengers, the objective function is to minimize the operational cost and the total time (waiting and trip time) in order to satisfy customers. Given requests for transport, the basic idea is to design schedules and sequencing routes to satisfy the objective function [9]. Requests arrive dynamically and the sequencing of pickup and delivery is out carried in real-time, that is, rescheduling and re-planning routes are performed while vehicles are in motion. These functions are generally performed by a central dispatcher.

Our approach differs from existing PDP solutions in the following aspects: (1) it supports decentralization, autonomy and cooperation vs. centralized decision-making; (2) it supports autonomous dynamic path planning for carts vs. vehicle routing; and (3) it supports communication among the vehicles to allow cooperation. These differences, and mainly the cooperative/autonomous dynamic route planning, introduce novel issues whose solution cannot be adapted from the proposed exact or heuristic solutions.

Table I provides a cursory summary of the major features of a sample of references dealing with the PDP problem similar to our formulation.

III. PROPOSED FRAMEWORK

Our ultimate aim is to deploy in a real-life environment a system of cooperative autonomous carts that are capable of efficiently carrying out pickup and delivery missions in real-time. To achieve this aim we identified as our main objective the definition of what we term as the “Autonomous Pickup and Delivery Problem” (APDP) [10].

Fig. 2. CARTS Framework.

Our modeling is inspired by actual systems as depicted earlier in Figure 1. Figure 2 is an abstract representation of the real-life pickup/delivery system. The map is represented as a grid and the transport network as a graph consisting of stations (nodes represented as numbered circles) and paths between the stations (links represented as weighted lines connecting the circles wherein the weight represents the distance). The major components of the proposed model are:

1) Map: it represents the actual map where services are rendered. The underlying structure of the map is expressed as \( M = \{ (x, y) \in N \} \) endowed with a function \( f \) defined by:
   a) \( f(x, y) = \text{free if the cell } (x, y) \text{ is free (nothing is in this location)} \)
   b) \( f(x, y) = \text{oc if it is occupied by a cart} \)
   c) \( f(x, y) = \text{op if it is occupied by a passenger} \)
   d) \( f(x, y) = \text{oo if it is occupied by an obstacles (e.g., building)} \)
   e) \( f(x, y) = \text{or if it is occupied by a path} \)
   f) \( f(x, y) = \text{os if it is occupied by a station} \)
   g) Note that the size of the occupant (e.g., station, building) and the size of the cell determine how many cells an occupant takes up.
2) Carts: these are the vehicles (carts) that move passengers between stations. A cart has a capacity (# of passengers it can carry) and a battery (energy source). Each car is endowed with computing power to perform planning and navigation tasks and to cooperate with other components of the systems (carts and dispatcher).

3) Passengers: these are the people who show up at a given station and initiate a transportation request.

4) Passenger stations: these are the locations where vehicles pick up and drop off passengers.

5) Routes: These are the paths between the stations. Vehicles can only travel over these paths while avoiding obstacles.

6) Communication infrastructure: this consists of the WiFi to support communication among the components of the system.

7) Obstacles: these are two types: static obstacles, such as buildings and trees, and dynamic obstacles, such as pedestrians and other vehicles using the same routes as our carts. The cart navigation capabilities will avoid these, as they may constitute a danger.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Static/Dynamic</th>
<th>Deterministic/Stochastic</th>
<th>Number of Vehicles</th>
<th>Time Window</th>
<th>Model: Analytical/Simulation</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psaraftis 1980</td>
<td>Static</td>
<td>Deterministic</td>
<td>1</td>
<td>None</td>
<td>Analytical</td>
<td>Dynamic programming</td>
</tr>
<tr>
<td>Psaraftis 1983</td>
<td>Static</td>
<td>Deterministic</td>
<td>1</td>
<td>Yes</td>
<td>Analytical</td>
<td>Dynamic programming</td>
</tr>
<tr>
<td>Dumas 1991</td>
<td>Static</td>
<td>Deterministic</td>
<td>Many</td>
<td>Yes</td>
<td>Analytical</td>
<td>Approximation algorithm</td>
</tr>
<tr>
<td>Swihart 1999</td>
<td>Dynamic</td>
<td>Stochastic</td>
<td>1</td>
<td>None</td>
<td>Analytical Simulation</td>
<td>Probabilistic</td>
</tr>
<tr>
<td>Attanasio 2003</td>
<td>Dynamic</td>
<td>Deterministic</td>
<td>Many</td>
<td>None</td>
<td>Simulation</td>
<td>Heuristics</td>
</tr>
<tr>
<td>Ritzinger 2016b</td>
<td>Static</td>
<td>Deterministic</td>
<td>Many</td>
<td>Yes</td>
<td>Analytical Simulation</td>
<td>Dynamic programming +heuristics</td>
</tr>
</tbody>
</table>

We assume that customers arrive randomly at the designated stations with a rate that follows a Poisson distribution such that probability of arrival is given by: $P(k) = e^{-\lambda} k! / k!$, where $\lambda$ is the average number of customers arriving at various stops per interval and $k$ denotes the number of customers arriving per interval. $k$ takes values 0, 1, 2. In this model, we assume that there is a dispatch system where customers, using an App, login their requests and receive notifications. All carts are equipped with a communication system allowing them to communicate with the dispatching system as well as among themselves. When a customer logs in a request, the dispatching system broadcasts it to all carts using the available networking infrastructure. The main information contained in a request includes the following: Request = $(reqId, t, src, dst, Tw)$, where $reqId$ is a unique identifier associated with the requester, $t$ is the time at which the request is made, $src$ is the pickup location, and $dst$ is the delivery location. $Tw$ is the preferable time window within which a customer would like his/her request to be satisfied.

To best satisfy customer requests, carts communicate and exchange their whereabouts as well as their availability and travel status. Based on the exchanged information, the carts make the best possible decision by processing and comparing the cost that each will incur if selected to execute the request. The carts with the least cost is selected to carry on with the request. The cost incorporates aspects like the (a) travel distance, (b) the travel time, (c) the waiting time of the requester, as well as (d) the energy cost in terms of estimated battery usage. In addition, the calculation of the cost is done by considering many parameters such as the availability of the cart given its current load and its current battery level, and the time window specified by the requester. We assume that each cart has a maximum number of passengers to accommodate and its onboard battery allows it to travel to the required maximum distance.

![Fig. 3. Request Transition States](image)

The life cycle of a request is illustrated by Figure 3 and consists of the following states:

1) Created: This state refers to when a request is created. Once a request is created it is tagged with creation time $t_0$. It is in this state, that the dispatch system verifies the credentials of the customer and decides whether the customer’s request should be accepted, in which case the request state is moved to “waiting”, or rejected in which case, the request state is changed to canceled.

2) Waiting: In this state, the customer is waiting for CARTS to decide which cart will execute its request. The request
in this state is tagged with $tw$ indicating the time the customer waiting time started.

3) Pick-up: When the customer boards the cart, the request state changes to “pick-up” indicating the start of the trip. The request is then tagged with start of the trip time $tp$, and by the same token the customer’s waiting time (at state “waiting”) is calculated as $WT = tp - tw$. The algorithm operating the CARTS system will always try to satisfy the time window logged in by the customer, that is, $WT: Tw$.

4) Drop-off: This state refers to the end of the customer’s trip. The request is tagged with time $td$ and the trip time is calculated: $TT = td - tp$. The optimization algorithm used by CARTS will always try to minimize the total time $WT + TT$; that is the time spent between the states “Waiting” and “Drop-off”.

5) Completed: This state refers to the completion of the positive request. Note that a completion may implicitly happen if the customer decides to get off the cart at any stop before the final destination.

6) Canceled: This state refers to the cancellation of a request. A request can be canceled in various situations. A request is canceled when the requester failed to show credentials. A cancellation may be caused by the requester himself/herself if the preferable time window cannot be satisfied, or if during the “waiting” state, the customer decides not to wait any longer for the dispatched cart. Upon receiving a request, each cart executes an algorithm which calculates a cost, that is, minimize:

$$\text{cost} = \sum_{i=1}^{V} \sum_{j=1}^{W} \text{dist}(s_i, s_j) x_{i} + \sum_{i=1}^{V} t_{w} y_{i}$$

subject to

$$\text{load}(c_i) \leq L_{\text{max}}, \forall c_i \in C$$

$$\text{dist}(c_i) \leq D_{c_i} \text{max}, \forall c_i \in C$$

IV. SIMULATION MODEL

Finding a mathematical solution to a complex problem such as of a pick-up and delivery system using traditional trucks and drones is a hard task. Many research studies have provided formal proofs for the NP-hardness of this problem [11]. Therefore our research methodology includes simulation-based experiments that will assess the performance of our solution, and benchmark it against existing solutions. We advocate the use of distributed discrete event-based simulation techniques which embed mobility models, a communication and cooperation model, as well as the dispatching model. The simulation will focus on the evaluation of specific path planning and vehicle routing algorithms. The objective of this simulation is to evaluate the performance of the proposed solution with regards to the delivery time and efforts as well as the battery energy consumption. Various performance parameters will be considered including the payload weight, coverage area, number of carts, number of delivery requests, etc. The communication technology considered in the simulation includes IoT WiFi and associated standards. In Figure 4, a screenshot of our simulation software shows a top view of the various components that are modeled by the simulation. The snapshot consists of a map of a typical campus – here, UAE University campus, with a set of pathways taken by the golf carts (represented as dashed lines in the figure). The map is modeled as a graph where the circles represent the junctions connecting path segments, each with a specific distance. Some of the junctions, colored in solid red, represent the stops where the carts can mark a stop to pick up or drop passengers. Therefore, in this model, passengers can only request transportation from and to designated stops. The blue brackets on top of each designed stop indicate the ID of the requests emitted by passengers. For instance, in stop #22, there are 3 passengers waiting for transportation, and represented by requests #81, #93, and #94. The solid-green colored squares represent carts. In this scenario, 10 carts are operating and each cart is assigned a number of requests – the request IDs are shown between brackets, based on its maximum capacity. For instance, cart #4, at the time of the snapshot is carrying 2 passengers represented by requests #77 and #105. The requests arrive according to a Poisson distribution, and are assigned to available cart based on the time they arrive, the current availability of seats in the cart, the whereabouts of the cart, and the current battery level of the cart.

Algorithm 1 provides an abstract description of the pickup and delivery management system. It expresses the process that executes three major subprocesses concurrently. These are: (1) receive requests; (2) process requests; and (3) path planning. Algorithm 2 refines the first algorithm. One step that is noteworthy is the the fitness score computation carried out by each cart. After constructing the route segment for the new request, our heuristics divides the decision space into three cases: (1) the segment is embedded within the current route; (2) the segment overlaps the current route; and (3) the segment is outside the current route.

Algorithm 1 Pick-up and Delivery Management Algorithm

1. Initialize the system parameters
2. Fork (These tasks are performed concurrently)
   - Receive requests $R_i$
   - Process requests $R_i$
   - Plan path for cart $C_i$
3. end fork

V. SIMULATION RESULTS

A. Simulation Environment

To test CARTS, we developed a simulation environment which takes into account realistic parameters and a campus-
wide map. The simulator is implemented in Python running on MacBook Air with a 1.4 GHz Intel Core i5 processor and a 4 GB 1600 MHz DDR3 RAM. The simulation results obtained below are based on the $2km \times 1km$ map of the UAE University campus. Detailed parameters are shown in Table II. The numbers are averages of 3 simulation runs of each experiments.

Two type of tests are considered here: (1) tests where we study the effect of the the number of cars available in the system on the performance of CARTS, and (2) tests where we study the effect of the requests arrival rate on the performance of CARTS. In this study, and for each of the two types of tests, we principally observe three performance criteria:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campus size (UAE Univ.)</td>
<td>$2km \times 1km$</td>
</tr>
<tr>
<td>Nb. of junctions</td>
<td>94</td>
</tr>
<tr>
<td>Nb. of stops</td>
<td>24</td>
</tr>
<tr>
<td>Nb. of road segments</td>
<td>115</td>
</tr>
<tr>
<td>Max. cart speed</td>
<td>5 mph (24 km/h)</td>
</tr>
<tr>
<td>Request arrival rate (Poisson)</td>
<td>2 to 18 requests per mn</td>
</tr>
<tr>
<td>Nb. of cars</td>
<td>2 to 18</td>
</tr>
<tr>
<td>Nb. seats per car</td>
<td>4</td>
</tr>
</tbody>
</table>

Fig. 4. Graph Modeling the UAE University Campus
times: the average time of passengers at each station, from the point they issue a request until they are picked up.

(b) seat occupancy: the average number of seats occupied by passengers in each car in the system. Occupancy is measured as percentage out of the car’s capacity.

(c) waiting queue sizes: the average number of passengers waiting at each station for a pickup.

B. Simulation Results

Figure 5 illustrates the effect of increasing the number of cars in the system on the average waiting times of the passengers. In this experiment we used a request arrival rate of 1 request per minute. The graph, here, clearly shows that making at least 10 cars available in the campus can drop the waiting times from around half-an-hour (1800 seconds) to 3 minute (200 seconds).

![Fig. 5. Average Waiting Time](image)

The next result illustrated in Figure 6 shows the effect of increasing the number of cars on the length of request queues waiting to be served at the stations. Similarly to the previous experiment, results indicate a drop from around 2.2 requests at 2 cars in the system) to 0.5 request when 10 cars are available.

![Fig. 6. Average Queue Size](image)

The results in Figure 7 shows the effect of increasing the number of cars on the seat occupancy. The average number of occupied seats in the cars tends to increase with the increase of the number of cars in the system. However, we note that this increase is relatively slow.

Algorithm 2 Detailed Pickup and Delivery Management

1: Initialize the system parameters
   
   Map selection: a map is selected randomly.
   Cars: a random number N of cars is generated and N cars are positioned at random locations.
   Stations: Similarly, for stations, a random number M is generated and M stations are distributed at random locations.

2: Initiate requests
   Passengers arrive following a Poisson distribution (random).
   Scan in passenger
   Passenger sends a random request.
   Request $R_i$ is received by the dispatcher.

3: Dispatcher broadcasts $R_i$ to all carts $C_i$
   Carts are polled for their results

4: Process requests $R_i$ Cart $C_i$ receives request $R_i$
   Cart $C_i$ computes a fitness score $FS_i$
   Cart $C_i$ returns its $FS_i$

5: Dispatcher receives all the $FS_i$
   Dispatcher selects $C_i$ with the best $FS_i$
   Dispatcher assigns $R_i$ to optimal cart $C_i$

6: Plan path for $C_i$ from current location to source $S_i$
   Pick up passenger $P_i$: Execute path from $S_i$ to destination $D_i$

7: end Process requests

8: Receive requests
   Passenger $P_i$ sends pickup request $R_i$
   Add $R_i$ to queue $Q_i$ associated with $P_i$

9: end Receive requests

Next, we consider the effect of the requests arrival rate on the performance of CARTS. Figure 8 shows an increase in the average waiting time of the passengers with the increase in the arrival rate of the requests. That is, from a 2.5 minutes (150 seconds) at 4 cars in the system, the waiting time increases almost linearly to around 6.5 minutes (400 seconds) at 16 cars in the system.

For the requests queue sizes, Figure 9 shows an almost proportional increase in the number of requests in the waiting queue with the increase in the arrival rate.
VI. CONCLUSION

We developed a model and a software simulation to address the architecture and optimization issues stemming from the growing use of electric carts as a means of transportation within campuses. Unlike previous research, our model is based on a decentralized architecture that takes advantage of the IT infrastructure. Consequently, the components of the model consisting of stations, carts, and people, are endowed with computing and communication capabilities that allow the system to distribute the functionalities over these components.

Specifically, carts are made capable of evaluating locally the cost of requests and sharing their results globally in order to select the best choice. Our current simulation system can be used to fine-tune the parameters (e.g., number of carts, waiting time, queue length, arrival rate) in order to identify an optimal mix for a given campus, thus leading to cost savings.

REFERENCES


