Multi-Vehicle, Multi-Warehouse Package Delivery Using Ant Colony Optimization

Adrian Machado
Department of Computer Engineering
University of Waterloo
Waterloo ON, Canada
amachado@edu.uwaterloo.ca

Irfan Sharif
Department of Computer Engineering
University of Waterloo
Waterloo ON, Canada
i4sharif@edu.uwaterloo.ca

YuChen Tang
Department of Computer Engineering
University of Waterloo
Waterloo ON, Canada
eytang@edu.uwaterloo.ca

Tu Trung Nguyen
Department of Computer Engineering
University of Waterloo
Waterloo ON, Canada
tt2nguye@edu.uwaterloo.ca

Abstract— **Package delivery is a rapidly growing industry, with the routing of delivery vehicles becoming a bottleneck on optimized parcel delivery networks. This 'last-mile delivery' is modeled to be the Multi Warehouse/Depot Vehicle Routing Problem with Pickup and Delivery lending to optimization strategies reliant on heuristics-based algorithms. In this paper we comparatively survey several algorithms such as Tabu Search, Genetic Algorithm and newly introduce Ant Colony Optimization (ACO) as it pertains to the last-mile delivery problem. We demonstrate the relative final cost efficiency of ACO with our novel cost computation leveraging warehouse distribution and accounting for vehicle capacity for multiple vehicles. We also demonstrate simulation efficiency of ACO by leveraging its internal parallelizable structure and conclude ACO to be the strongest overall solution in contrast to previous literature focusing on Genetic Algorithms.**

Keywords—Multi-vehicle routing, ant colony optimization

I. Introduction

With the rise of e-commerce, more retailers are now required to be deliver goods directly to consumers. The global parcel delivery market has experienced a massive growth in the volume of packages delivered in recent years, growing from 7.8 to 8.8 billion parcels distributed from 2015 to 2016[1]. A large part of this delivery process is the 'last mile delivery system', shipping the packages from local warehouses to the individual customers or picking up packages and bringing them back to the local warehouses.

The last mile delivery system usually requires one to many warehouses with multiple vehicles depending on the region that takes place in. For example, there are over twenty Fedex shipping centers in Manhattan alone with probably as many as a hundred delivery trucks active at any given time. Good coordination between fleet vehicles is essential to ensure that packages are delivered in a timely manner whilst minimizing cost undertaken in finishing deliveries. A lot of recent research has focused on this problem, notably the work from Serna M., Uran C., et. al. in *'Vehicle routing to multiple warehouses using a memetic algorithm'*[2] where the 'Multi Warehouse/Depot Vehicle Routing Problem with Pickup and Delivery' (MDVRPPD) is first introduced.

In this report we describe using a variant of the Ant Colony Optimization (ACO) algorithm in optimizing for the Multi Warehouse/Depot Vehicle Routing Problem with Pickup and Delivery problem, specifically the total cost of delivering/picking up all required packages as measured in the distance travelled by all shipment vehicles. We compare our ACO implementation with earlier revisions using Tabu Search techniques and Genetic Algorithms. Section 2 provide a cursory review of existing literature on the subject, various techniques already adapted and explored. In Section 3 we formulate the problem precisely and demonstrate how we modelled it in each of our algorithm implementations. The modeling lent to comparable analysis between our various approaches in solving the problem. In Section 4 we elaborate on our proposed solution using a variant of ACO. We elaborate on the underlying mechanism that demonstrate convergence properties of the procedure to near optimal solutions. In Section 5 we present comparative quantitative performance evaluations between ACO, GA and Tabu Search techniques as applied to the MDVRPPD problem. We conclude in Section 6 summarizing our central contributions in this paper, and recommendations to others looking to solve the same problem.

II. Literature review

In *'Multiple Ant Colony System for Vehicle Routing Problems with Time Windows'*[3], the authors demonstrate that using ACO can create solutions competitive in terms of results and runtime performance. The attempt was to optimize the number of vehicles required for deliveries within a region in addition to the cumulative traveling distance across all vehicles. In this paper, the optimization technique uses two different ant colonies, one for the number of vehicles, and another for the cumulative length of routes across all vehicles. The information from each colony was shared with the other in a form of mutual reinforcement optimization. This paper also assumes the presence of only one warehouse and that delivery trucks do not revisit the warehouse multiple times to pickup/dropoff more packages. It's our understanding that most high density regions today with delivery fleets are comprised of multiple warehouses and multiple corresponding delivery fleets. We also assume delivery vehicles, with finite package capacities, need to make multiple round trips to and from central warehouses to make room for more packages to pickup/dropoff. Due the evolved nature of delivery routing today, the effectiveness and utility of MACS-VRPTW is diminished.

'Vehicle routing to multiple warehouses using a memetic algorithm'[2] portrays the problem that is more reflective of the modern last mile delivery problem. It utilizes Genetic Algorithms solving for configurations with multiple warehouses and multiple vehicles in the delivery fleet, optimizing for the total route time as measured by total distance traversed by all vehicles. The result shows that in less than ten generations, the generated solution saturate. The authors also note that the total runtime of their JAVA implementation on a Athlon X2 Dual Core processor took around two minutes. The authors however fail to compare their approach to other algorithms, lending no indication to comparative performance. In their conclusion, the authors state that though sub-optimal, GA has desirable runtime performance characteristics. The authors also state that the fast compute time lent to quick saturation of the result, meaning not enough of the search space had been explored.

III. Problem Formulation & Modelling

The Multiple Depot/Warehouse Vehicle Routing Problem with Pickup and Delivery is quite complex and is composed of several constraints and parameters that are reflected by real-life counterparts seen in last-mile delivery. The problem is rooted in TSP and is therefore modelled by a set of nodes located in the search space. These nodes are labelled as either a pickup node (ex. A household that has a package to be picked up and delivered to the warehouse), a dropoff node (ex. A household that is expecting a package to be delivered to them), or a warehouse (which contains packages in transit). In last mile delivery, these households are not equidistant, and as a result have travel times between each other. This is modelled as distance values between the different nodes. In order to simulate different search spaces with uneven distribution of pickups and dropoffs, nodes are randomly selected to be either a pickup, dropoff or warehouse. A limit is set on how many warehouses can exist to avoid unrealistic scenarios (ex. More warehouses than dropoff/pickup points). The travelling salesman in MDVRPPD is not a single agent, rather a collection of vehicles that are initially stationed at warehouses. These vehicles are initially loaded with the packages they intend to deliver (based on the generated solution), and can pick up packages from pickup nodes so long as the package won't put the vehicle over capacity. This problem is considered solved if a delivery/pickup route is created such that the overall distance travelled by the vehicles is minimized.

In essence, the problem can be broken down in the following summary:

1) *State:*
   a) The set of nodes yet to be visited
   b) The current position of the vehicles
   c) The number of packages that have yet to be delivered
   d) The total capacity of each vehicle
   e) The number of delivery packages each vehicle is carrying
   f) The number of packages each vehicle has picked up on its current trip
2) *Goal:*
   a) Deliver all 'drop-off' packages in the warehouses to their respective dropoff nodes
   b) Pick up all 'pick-up' packages from the pickup nodes and unload them at a warehouse
3) *Initial State*:
   a) All vehicles are located at a randomly selected warehouse and contain no packages
4) *Action:*
   a) Actions take place within a simulation round in which each vehicle takes one actions
   b) A vehicle will choose a node to travel to that has not already been visited
   c) A vehicle can travel to a warehouse to drop off packages it picked up, and load packages that have yet to be delivered
5) *Cost:*
   a) Travelling from one node to another has a cost based on the distance between the two nodes
   b) Travelling to a pickup node when the vehicle is at capacity has a cost of infinity
   c) Travelling to a dropoff node when the vehicle has no packages to deliver has a cost of infinity

IV. Ant Colony Optimization

Ant Colony Optimization (ACO) is one of the more popular population based metaheuristic methods. It draws from colony behaviours of ants where individual ants navigate an unknown terrain looking for food and excrete pheromones along the path taken. Crucially, pheromone deposits are proportional to how close the food source is taking a given path (owing to the natural evaporation process diluting pheromone deposits over longer path traversals). Subsequent ant explorations consider previous pheromone deposits and distance to the immediate next node when navigating, favoring paths with higher concentration of deposits and consequently a cheaper path to the food source. The very initial ant traversals are random traversals of the grid.

ACO is a simulation of this process where at a given starting point M ants are placed and multiple exploration rounds are simulated where in each round each ant traverses the terrain by a path determined probabilistically by earlier pheromone deposits. Following the traversal, each ant deposits pheromones along the path inversely proportional to the cost of the path.

The motion of the ants, though non-deterministic, is stochastic in nature as it converges on a global path near optimal in nature. An important characteristic is that a few ants continuously explore non-preferred paths attempting to find even better paths.

Applying ACO to the MDVRPPD problem had a few interesting characteristics, and is not an approach that has been studied previously. Firstly, given each delivery vehicle had a fixed, finite capacity regarding the number of packages it could hold at any given time, it did not suffice to simply traverse through the pickup/dropoff points while minimizing cost. We needed to ensure that appropriate trips back to central depots/warehouses were interleaved within the traversal so as to make room for subsequent package pickups or dropoffs. Secondly, the solution generated was one intended for traversals of multiple vehicles. Classical formulations applying the ACO techniques to the Traveling Salesman Problem were optimizing a tour for a single individual. In MDVRPPD it's the sumtotal cost of paths across all vehicles in the feel that's being optimized for.

Given these two constraints, we adapt the classical ACO algorithm to generate multiple sub-tours through the graph where its only across all generated sub-tours that we guarantee the visitation of all nodes in the graph. Each sub-tour corresponds to the generated traversal of each vehicle, and within each sub sub-tour we also guarantee the vehicle capacity is not violated. The classical ACO algorithm is shown in Figure 1.

```
Initialize pheromone trails

While (termination condition not met) Do:
  Construct ant solutions
  Apply local search
  Update pheromone trails
End
```

Figure 1. ACO algorithm summarized

In typical adaptations of ACO to the TSP problem, updates to the pheromone trails are contingent on the costs of the constructed ant solutions which in turn represent the cost of a single agent tour through the entirety of the grid. In our adaptation of the algorithm, we compute this cost differently. For a given grid with labeled warehouses, pickup, and dropoff points, we generate a sub-grid not containing any warehouses. Each ant within each simulation round maintains an internal tabu list so as to not revisit already visited nodes, thus producing a single tour through all pickup and dropoff points. At this point none of the generated solutions pertain to multi-vehicle tours nor are valid with respect to the capacity of any delivery vehicle. We first assume the generated path enlists the targeted points for a single vehicle and annotate it with interleaved warehouses as appropriate whilst minimizing cost. This is done so by iterating over the generated path and keeping track of capacity needed as the vehicle traverses through pickup or dropoff points. If at any point the vehicle is over-capacity, we visit the closest warehouse available and dropoff all packages previously picked up. If additionally, the vehicle finds itself at a dropoff point but with not packages left to dropoff, we similarly visit the nearest warehouse to pick up the packages to be dropped off. The motivating assumption here is vehicle capacities are sufficiently large such that individual visits to warehouse mid-traversal doesn't affect cost significantly. After generating a single valid path with capacity in consideration, we repurpose the tour and partition it such that there is roughly equal distribution of work among all the vehicles. We do this by identifying the various interleaved warehouses in the tour, and given we know that the path between consecutive warehouse visits satisfies vehicle capacity, we take all *n* partitions of the tour (partition being every sub-path between consecutive warehouses) and assign it evenly across the provided *m* vehicles. The total distance traversed cumulatively across all vehicles is then used as the cost per solution in the classical ACO algorithm, with the pheromone trails updated accordingly

V. Performance Evaluations

The datasets used for the problem are standard TSP datasets that we retrofit to model a region with multiple pickup/dropoff points. The datasets include *uk12* (12 nodes), *ha30* (30 nodes), *kn57* (57 nodes), *sgb128* (128 nodes) and *usca312* (312 nodes) and were retrieved from Florida State University's Department of Scientific Computing[4]. Nodes in the dataset were labeled to be pickup points, dropoff points, or warehouses and the distances between nodes were taken to be node traversal costs.

Table 1. Summary of results derived from collected data for ACO

Dataset	**UK12**	**HA30**	**KN37**	**SGB128**	**USCA312**
Average search time (seconds)	0.23	0.294	0.756	374.1	900+
Average final path length	3351.4	789.4	23103.4	42952	NA

In Table 1 we summarize our results for various ACO simulations as applied to the MDVRPPD problem for differently size datasets. From the average search time trend, we observe a non-linear relationship between runtime and input size. It's safe to conclude that for much larger input sizes, ACO's runtime will become very expensive. However, for reasonably small input sizes, such as vehicle delivery within a small region, ACO can perform very well: achieving low cost paths with a fast runtime. Figure 2 demonstrates that convergence to near optimal paths happen quite rapidly within a few simulation rounds, where most later rounds only reduce total cost slightly so. This fact can be leveraged in path computation where operators can simply simulate the algorithm on large datasets for as little while needed.

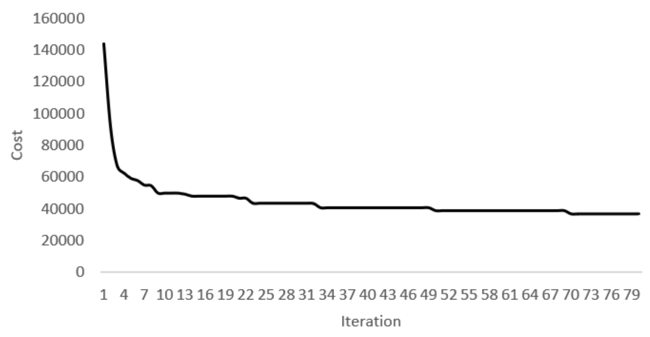

Figure 2. Path cost after each iteration for 128-nodes

ACO, by default, has a large set of parameters to tune. In order to minimize costs and simulation runtimes, we simulated an ACO parameter sweep varying the pheromone evaporation rate, α (influence of pheromone in next step decision making), β (influence of distance to next node in next step decision making) and M (number of ants) in isolation. The resulting plots, plotting between the parameter value and path cost, are presented in Figure 3, Figure 4, Figure 5 and Figure 6, respectively, and were all executed over the *sgb128* dataset.

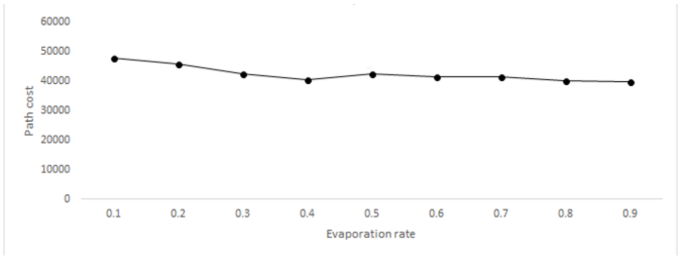

Figure 3. Path cost vs. Evaporation rate

We tuned the evaporation rate, sweeping from 0.1 to 0.9 in order to determine the effect on the solution cost, the results of which are displayed in Figure 3. As the evaporation rate was increased the path cost decreases near linearly. The higher evaporation rates lent to a higher incidence of ant exploration, and thus more optimal solutions.

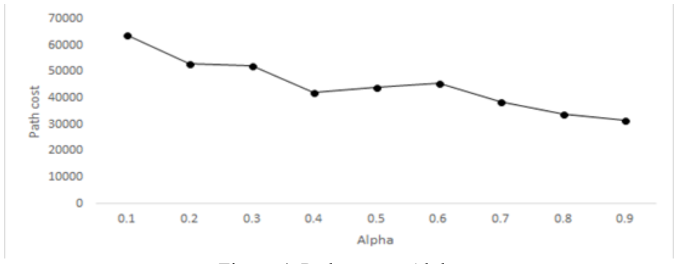

Figure 4. Path cost vs Alpha

Similarly tuned was α, the relative importance of the pheromone concentration in next step decision making, with the results displayed in Figure 4. Increasing α resulted in reduction in path cost due to higher reliance on the overall colony intelligence. Higher α's also lent to fast convergence of optimal solutions, at a cost to more exploration, but we observed a good balance in using a sufficiently high evaporation rate in addition to high α.

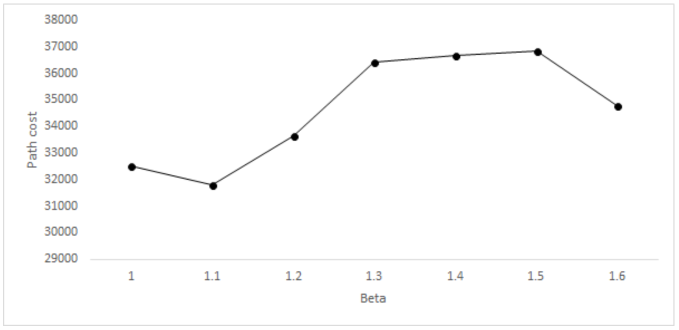

Figure 5. Path cost vs Beta

The parameter sweep for β, the relative importance of visibility heuristic, is shown in Figure 5. It's observed that the best β is approximately 1.1, with deviations from this value resulting in increased path costs. We additionally tuned M, the number of ants made available per round of simulation and is shown in Figure 6. We observe the higher the ant count, the lower the path cost. This is expected as higher ant counts leads to more exploration, albeit at a cost of longer simulation times.

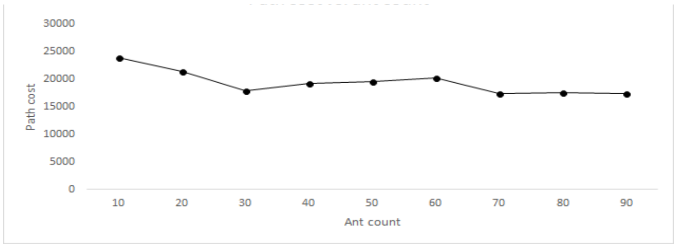

Figure 6. Path cost vs Ant count

The approach of varying one parameter while fixing other parameters is a classic method to show a variable's contribution to the final output of a system. We made the assumption that the parameters are completely independent. However, this is also a downside to our approach: we did not consider the possible correlation between the different parameters. This means it is possible that, for example, Alpha's values and its impact on the overall optimality may change with the variations of other variables.

In this report, we did not include results for Simulated Annealing (SA) because Tabu Search, also a trajectory-based search algorithm, performed better than SA in terms of runtime and comparably in terms of optimality.

A. Comparison With Tabu Search

Table 2. Summary of results derived from collected data for tabu search

Dataset	UK12	HA30	KN37	SGB128	USCA312
Runtime (s)	0.0303	0.65	0.86	565.8	900+
Average final path length	2223.2	971.6	21163.4	41396.4	NA
Cost decrease (%)	30.5	58.2	56	70.1	NA

In terms of runtime, ACO is comparable to Tabu Search for small number of nodes (12, 30 and 57). For larger datasets, Tabu Search runs faster than ACO due to ACO's highly non-linear nature. In terms of optimality, Tabu slightly beats out ACO in our simulations, but the gains by Tabu is not large enough to claim conclusive result. Therefore, it is recommended that for very large dataset, one is better off to use Tabu over ACO.

B. Comparison With GA

Table 3. Summary of results derived from collected data for GA

Dataset	UK12	HA30	KN37	SGB128
Average search time (seconds)	70.68	146.02	289.32	615.2
Average final path length	2243.8	771	22430.8	59566
Average cost decrease (%)	26.19	48.21	58.05	61.2%

GA's runtime is quite long compared to ACO, even for small-sized search spaces, as seen in Table 3. This is most likely due to the fact that ACO's internal structure lends to easy parallelizability, which our implementation leveraged. This arises in the exploration stage of the M ants following which global updates to the pheromone distribution across the map are made, where the traversal of each ant can be run independently.

ACO also lends to far greater exploration compared to GA in larger data sets, where despite the high iteration count traits from the very initial solution pair can be seen in the final results. ACO on the other hand isn't as directly related to early solutions in comparison. In our dataset the starting point of the path matters significantly less so given how closely the vehicle routing problem relates to TSP, as any starting point would have to be part of any optimal path, and ACO is able to compute near optimal paths from any point.

For small to medium sized datasets, GA takes much longer, but typically provides a more optimal solution on average compared to ACO, likely due to more iterations resulting in the creation of a more substantial portion of every possible path, essentially exploring more of the search space (ex. exploring 12! possible permutations vs 30! with the same number of iterations and population size). By nearly 'brute-forcing' the search space the trait of sticking too close to the initial solution has its impact softened. In smaller data sets, the initial solution is also more likely to be closer to the optimal solution as well, as there are fewer permutation, so retaining traits from the initial solution may be beneficial in some cases.

VI. Conclusion and Recommendations

The problem of multi-vehicle, multi-warehouse routing for performing pickup and delivery is a complex problem to optimize and one with many real-world applications. By modelling the problem as a TSP variant with multiple depots, multiple salesman, and some additional constraints, algorithms typically used for solving TSP can be utilized to create efficient routes for the vehicles. These algorithms include Tabu Search, Genetic Algorithm and Ant Colony Optimization, which were all tested and evaluated. Given the main two criteria of solution optimality and program runtime, we demonstrate that ACO maintains a strong balance between speed and optimality. ACO provides similar levels of optimality as Genetic Algorithm in a fraction of the time for most data sets. Similarly, ACO provides more optimal solutions than Tabu Search in nearly as short of a time. Our ACO solution can be further improved with the tuning of the evaporation rate, α, β, and the ant count, as well as more adaptive techniques varying parameters as per cost progression, lowering the total path cost while maintaining a quick runtime.

Although ACO is a strong overall solution, that does not imply it is the best solution for all problem configurations, especially in real-world scenarios. In some neighborhoods, there are very few pickups and dropoffs per day, so the algorithm would only be operating on a small data set, in which using GA is more suitable given the runtime is sufficiently low and often generates more optimal paths. For very dense neighborhoods with many drop-offs/pickups, or for near real-time routing, a sub-optimal solution that is generated faster than ACO, namely using Tabu Search, might be preferable.

References

[1] Statista. (2018). Number of parcels distributed worldwide from 2014 to 2016. [online] Available at: https://www.statista.com/statistics/737418/parcel-traffic-worldwide-by-sector/, Dec 1, 2017 [Accessed 17 May 2018].

[2] Serna M., Uran C., Cortes J., Benitez A., "Vehicle routing to multiple warehouses using a memetic algorithm". XI Congreso de Ingeniería del Transporte, Columbia, 2014.

[3] Gambardella L., Taillard E., Agazzi G., "MACS-VRPTW: Multiple Ant Colony System for Vehicle Routing Problems with Time Windows". Technical Report Idisa, Logano Switzerland, 1999.

[4] Burkardt J., "CITIES - Cities Distance Dataset," [Online]. Available: https://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html. [Accessed 17 May 2018].