1 Introduction

Waste collection industry is one of the key sectors in urban communities around the world. Today, everyone has waste to dispose due to the current lifestyle of fast food, online shopping and shipping, and non-bulk packaging. Every day, about 35,000 tons of municipal solid waste are produced in the Philippines. In Baguio City, daily garbage increased by more than 20% from 2008 to 2013.[1] As of 2018, Baguio City has 14 functioning serviceable waste collection trucks with a pending purchase of 4 more vehicles on January, 2018.[2] These trucks are responsible for servicing the 128 barangays (villages) from 3 a.m. in the morning to 11 p.m. at night. On the other hand, two compactors are utilized for waste collection in the central business district area. This move of purchasing vehicles was said to boost efficiency and to keep-up with the growing tourist influx during the weekends, holidays and the incoming summer vacation. Waste Analysis and Characterization Study (WACS) investigated the output of garbage in the city and found that the residents produce 402 tons of mixed waste daily with 150 to 167 tons being residual waste while the rest are classified as biodegradable and recyclable. From 2017, the city has approved and is en route to establishing and developing several waste collection facilities in the next ten years, namely, centralized materials recovery facility, engineered sanitary land-fill, anaerobic digester, waste-to-energy plant, Environmental Recycling System machines, health and medical waste treatment plant, and special waste treatment plant.[3] Indeed, the city is doing it's part to reduce the carbon footprint and employ better waste management by employing the no plastic policy or the "Plastic and Styrofoam-Free Baguio Ordinance" of 2017. This city ordinance regulates the sale, distribution and use of plastic bags and styrofoam in the city. Instead of plastics and styrofoam containers, vendors are encouraged "to provide or make available to customers for free or for a cost, paper bags or reusable bags or containers made of paper or materials which are biodegradable, for the purpose of carrying out goods or other items from the point of sale.[4]

In line with these city policies and activities, we study the current efficiency of the routing and scheduling of waste collection vehicles. Specifically, we study the most efficient route and scheduling of vehicles as well as the minimum number of vehicles required to handle the job.

Vehicle routing problem is a combinatorial optimization programming problem aiming to provide service to a number of customers with a fleet of vehicles. This problem was proposed by Dantzig and Ramser[5] in 1959. In the recent decades, vehicle routing has been the focus of some researchers and application developers (such as WasteRoute[6]). From public transportation to product deliveries, vehicle routing solutions has helped not only businesses but also the common man. Optimization of the vehicle routing problem has given rise to efficient fuel consumption as well as time management. In this study, we will focus on routing waste collection vehicles in Baguio City given time windows for servicing consumers.

There are a number of ways to solve a vehicle routing problem. Researchers have used both exact algorithms and heuristic algorithms. Exact algorithms are able to obtain exact solutions but are computationally complex to perform. Computations increase exponentially as the problem size increases. Some methods under the exact algorithm are the Branch-and-bound method, Dynamic Programming[7], and Integer Programming. Heuristic algorithms on the other hand are able to obtain solutions to the problem but they may return sub-optimal ones. Some methods under the heuristic algorithms include Genetic Algorithms[8], Particle Swarm Optimizations[9, 10], Neighborhood Search Heuristic[11] and some hybrid algorithms[12, 13, 14]. In this study, the problem will be tackled using a hybrid meta-heuristic algorithm, Particle Swarm Optimization (PSO) coupled with Genetic Algorithm (GA).

2 Vehicle Routing Problem with Time Window

The vehicle routing problem (VRP) is an non-deterministic polynomial-time hard problem. This means that finding solutions to these kinds of problems will take exponentially longer to obtain as the problem size increases. Solutions to the VRP, although difficult to produce, are quickly verifiable due it being able to be computed in polynomial time for a deterministic Turing Machine to check. We may say that a solution is feasible but it may not be the best one (optimal) in the whole solution space. A solution may not even exist at all. The VRP is typically a combinatorial problem that deals with arranging multiple specified locations along a single path while satisfying known constraints. VRP is usually concerned with the minimization of the temporal and/or geographical aspects of traveling along paths while accommodating the most amount of customer demands along the way. Customers are usually distributed at different locations and each customer has a certain amount of demand assigned that utilizes the capacity of the vehicles servicing them. VRP models also include minimizing the number of vehicles needed to satisfy the total customer demand. In waste collection, a regular VRP can be used to model the residential sector since trucks only need to traverse along each location, collecting waste and moving them to disposal sites.

The vehicle routing problem with time windows (VRTPW) is an extension of the VRP problem. VRPTW is also an np-hard problem and given a fixed number of vehicles, it becomes an np-complete problem. Here, some customers identify a time interval when the service is needed or expected. This adds time dimension to the problem, hence, we now consider minimizing both spatial and temporal costs. Service time is also taken into consideration, this is the amount of time required for the loading and unloading processes at locations. VRPTW specifically covers the business sector in waste collection. Usually, businesses and waste collection services draft contracts for accommodation at a particular time and day.

Further extending the VRPTW, Buhrkal[11] add a lunch break, required by law, for the drivers. In addition to the previously mentioned constraints, drivers will have to take a certain amount of time to rest and eat in between any two locations at most once along the entire route.

Adding some specifications, waste collection also involves the holding capacity of a garbage truck as well as a disposal period wherein the vehicle needs to visit a disposal sight to refresh it's capacity after it is filled along the route and before the vehicle returns to the depot to retire for the day.

3 Review of Related Literature

In 1987, Solomon[15] proposed and implemented some insertion heuristics to the some problems sets he crafted (called "Solomon Benchmark Problems") for benchmarking VRPTW solution methods. The first is the savings heuristics which starts out with each customer having dedicated routes, then the algorithm tries to combine the best pair of routes with each other until the minimum amount of routes are produced. The best pair of routes is decided by some savings equation which gives the amount of cost saved if two routes are combined rather than separate. The next heuristic is a greedy approach, the time-oriented nearest-neighbor heuristic which starts by selecting the "closest" (in terms of travel distance and travel time) node from the depot and attaching it to the current route. We repeat the process for the last added node until the schedule is full. We then proceed to create the next route. The next heuristic introduced is the insertion heuristic which can construct routes sequentially. Each node is inserted in the best location, between two nodes in the route. The best location for insertion is determined by some function that shows how efficient the route becomes after insertion. There are three proposed ways of evaluating the efficiency of the routes each called the I1, I2, and I3 respectively. When the node is inserted, the nodes succeeding it are pushed forward, meaning that servicing these nodes are adjusted based on how much time is used to accommodate the inserted node. The last heuristic discussed is the time-oriented sweep heuristic which groups customers and assigns them to a vehicle and the schedule for each vehicle is constructed from there. The results show that among the proposed heuristics, the insertion algorithm (specifically the I1) proved to be the most effective is solving the benchmark test cases because if focuses more on correct sequencing customers rather than grouping and assigning customers to vehicles.

In 2000, Son[9] utilized a Chaotic Particle Swarm Optimization (CPSO) algorithm to generate routes and schedules of the different waste collection vehicles at Danang City Vietnam. The CPSO obtained data on the roads and waste collection facilities from a Geographic Information System (GIS) that simulates a continuous environment and models the road network and waste collection system of Danag City. There are three different kinds of vehicles available, namely, tricycles, hook-lifts and forklifts which take up different roles in the waste collection system. The objective in this case was to maximize the amount of garbage collected in the simulation.

In 2005, Nuortio et.al.[16] improved the inflexible and inefficient waste collection scheduling and routing in Eastern Finland by creating a conceptual model based on the available data and employing a heuristic based on the guided variable neighborhood thresholding meta heuristic. The results showed that the schedule produced by the heuristic significantly reduced travel costs of vehicles.

In 2007, Cordeau et.al.[17] compiled and defined general models of the vehicle routing problem and it's extensions. They also compiled and cited the approaches done by researchers over the years to tackle the VRP and it's extensions. They also give a brief summary of the process of how each algorithm solves the problems presented.

In 2012, Burhkal et.al.[11] set-up a model for waste collection VRPTW with lunch breaks after two test cases, namely the Waste Management Inc. which is responsible for waste collection in parts of Northern America and the Henrik Tofteng Company responsible for handling waste collection at Denmark. These two cases have different policies for lunch break hours, limits on the number of customers served per route, and total amount collected at each route. They provided both cases with solutions using an adaptive large neighborhood search heuristic.

In 2015, Akhtar, Hannan and Basri[10] proposed a method of solving Waste Collection Vehicle Routing Problem by distributing customers into bins and creating a traveling salesman problem for each bin then applying Particle Swarm Optimization to create routes.

Masrom et.al.[14] developed a hybrid PSO by incorporating the mutation mechanism of GA. They w

4 PSO-GA Approach

PSO-GA is a hybrid of PSO and GA. Harish Garg has proposed a PSO-GA[18] which supplements the particular disadvantages of both PSO and GA with the advantages of each. The algorithm attempts to balance the exploration and exploitation ability of both algorithms. Exploration happens in PSO when particle fly through the search space. It is less applicable to GA since the algorithm only utilizes what is current known in the population. It only occurs for GA through Cross-over and Mutation. Exploitation happens in PSO when a particle flies to or near an area containing a possible solution, every other particle in the population will tend to flock towards that area in order to find the solution. PSO's problem is that local optima may trap the whole population. Exploitation happens in GA during the Selection operator, wherein the members with the fittest values have a higher chance of being chosen for Cross-over and Mutation. Hence, more chances of exploring that particular gene pool.

In GA, if an individual is not selected, the information contained by that individual is lost but in PSO, the memory of the previous best position is always available to each individual. Without a selection operator, PSO may waste resources on poorly located individuals. PSO-GA by Garg[18] combines the ability of social thinking in PSO with the local search capability of GA.
PSO's velocity vector guides the population to a certain solution point while GA's selection and cross-over replaces infeasible solutions with feasible ones by creating an individual from the set of feasible solutions.

4.1 Parts of PSO-GA

The algorithm for PSO-GA is shown below

1. Set PSO and GA parameters
    - Set current PSO iteration, $PSO_{CurrIt} = 0$ and max iteration PSO_{MaxIt}
    - Set PSO population size PSO_{PopNum}, cognitive and social bias constants c_1 and c_2, maximum and minimum inertial weights w_{max} and w_{min}

- Set GA parameters, crossover probability GA_{cross}, mutation probability GA_{mut}
- Set GA parameters: rate of the number of PSO particles affected by GA γ and rate of increasing GA maximum iterations β, maximum and minimum number of individuals to be selected GA_{NumMax} and GA_{NumMin}, maximum and minimum GA population sizes $GA_{MaxPopSize}$ and $GA_{MinPopSize}$, maximum and minimum GA iteration numbers GA_{MinItr} and GA_{MaxItr}
- Set the PSO dependent GA parameters, number of individuals affected by GA GA_{Num}, GA population size $GA_{PopSize}$ and GA maximum iteration GA_{MaxItr} using the equations

$$GA_{Num} = GA_{NumMax} - (\frac{PSO_{CurrIt}}{PSO_{MaxIt}})^{\gamma} \times (GA_{NumMax} - GA_{NumMin}) \quad (1)$$

$$GA_{PopSize} = GA_{MinPopSize} + (\frac{PSO_{CurrIt}}{PSO_{MaxIt}})^{\gamma} \times (GA_{MaxPopSize} - GA_{MinPopSize}) \quad (2)$$

$$GA_{MaxItr} = GA_{MinItr} + (\frac{PSO_{CurrIt}}{PSO_{MaxIt}})^{\beta} \times (GA_{MaxItr} - GA_{MinItr}) \quad (3)$$

PSO Section

2. Generate a random population of particles of PSO_{PopNum} members in D dimensions, each with a corresponding random velocity v
3. Increment PSO_{CurrIt} by 1
4. Evaluate each particle's objective function value $F(PSOx)$
5. Update *gbest* and *pbest* positions and values of each $PSOx_i$ in the population ($i \in 1, 2, 3, ..., PSO_{PopNum}$)
6. Update each particle's velocity and position with the equations,

$$w = w_{max} - (w_{max} - w_{min}) \times (\frac{PSO_{CurrIt}}{PSO_{MaxIt}}) \quad (4)$$

$$v_i = v_i \times w + c_1 \times rand() \times (pbest_i - PSOx_i) + c_2 \times rand() \times (pbest_g - PSOx_i) \quad (5)$$

where $i \in 1, 2, 3, ..., PSO_{PopNum}$ and g is position/individual in the PSO population that is currently designated as global best (*gbest*) individual

$$PSOx_i = PSOx_i + v_i \quad (6)$$

GA Section

7. Set the number of currently selected individuals $GA_{CurrNum} = 0$
8. Increment $GA_{CurrNum}$ by 1
9. Choose a random position/individual $PSOx_s$ from the PSO population.

10. Generate a random population of $GA_{PopSize}$ individuals in the same D dimensions.
11. Set the first individual GAx_1 in the GA population to be a randomly selected individual $PSOx_s$ from the PSO particle population.
12. Set the current GA iteration $GA_{CurrItr} = 0$
13. Increment $GA_(CurrItr)$ by 1
14. Perform elitism
    - set the replacing individual GA_{rep} as the randomly selected PSO particle $PSOx_s$ if $GA_{CurrNum} = 0$
    - otherwise, check each individual in the current GA population, if $F(GAx_i)$ is less fit than $F(PSOx_s)$, then replace GAx_i with $PSOx_s$

$$GAx_i = \begin{cases} PSOx_s & \text{if } F(PSOx_s) < F(GAx_i) \\ GAx_i & \text{otherwise} \end{cases} i \in 1, 2, ..., GA_{PopSize}$$

15. Perform selection, crossover and mutation to generate the next GA population
16. Evaluate the penalizing objective fitness values $F(GAx_i)$ for each individual in the GA population
17. Check if maximum GA iterations is reached
    - If reached, proceed to step 18
    - otherwise, go back to step 13
18. Replace the selected PSO particle $PSOx_s$ with the best individual in the GA population
19. Check if the maximum number of replacements have occurred
    - If reached, proceed to step 20
    - otherwise, go back to step 9
20. Update the PSO dependent GA parameters using equations (1), (2) and (3)
21. Check if the maximum number of PSO iterations have been reached or if the population has converged
    - If reached, end
    - otherwise, go back to step 3

As you can see, the algorithm follows the both PSO and GA algorithms in succession. PSO is first done to the population to obtain points across the search space. GA is then applied to some of the best individuals. This is done to replace the worst individuals in the population with those closer to the better ones.

After forming the new population with PSO, some of the individuals in the population will get replaced. Some not all because if we have a huge population, it would take a long time to complete. This number is given by GA_{Num}. After selecting the best individuals from the population, the algorithm aims to create a new population by replacing points in the current population with better points via the genetic principles, selection, cross-over and mutation. After all selected individuals have been processed, we change the GA variables, $GA_{PopSize}$ and GA_{MaxItr} which are for the population size in GA and the maximum iterations done for GA respectively by the equations (1), (2) and (3).

Judging from the equations 1, 2 and 3, GA_{Num} will initially be GA_{NumMax} and slowly become GA_{NumMin} as the number of iterations increases. This is because the fraction PSO_{CurrIt}/PSO_{MaxIt} is raised to γ which is a positive whole number as given by Garg[18], hence, the whole term $(PSO_{CurrIt}/PSO_{MaxIt})^{\gamma}$ will initially be very small and eventually will be equal to 1 when $PSO_{CurrIt} = PSO_{MaxIt}$.

This is also the case for both $GA_{PopSize}$ and GA_{MaxItr}. $GA_{PopSize}$ will initially start equal to $GA_{MinPopSize}$ then slowly become $GA_{MaxPopSize}$. GA_{MaxItr} will initially start equal to GA_{MinItr} then slowly become GA_{MaxItr}. Since the factors will be in fractions, there is a need to get the floor values of GA_{Num}, $GA_{PopSize}$ and GA_{MaxItr}. This is because GA_{Num}, $GA_{PopSize}$ and GA_{MaxItr} must be positive integers because they dictate array sizes. However, in the case of Inertial Weight w, which changes according to the equation $w = w_{max} - (w_{max} - w_{min})(PSO_{CurrIt}/PSO_{MaxIt})$, it is most of the time a fraction. Garg[18] started $w = 0.9$ initially then becoming $w = 0.4$ as the number of iterations increases.

5 Problem Formulation

We start with defining some variables. Then we show how each particle or chromosome is decoded into tours(set of routes). From there we define the objective functions and the constraints that will be used in modeling the problem.

The objective in Waste Collection Vehicle Routing Problem with Time Windows (WCVRPTW) is find a set of routes for vehicles, minimizing travel time and distance, such that all consumers are visited exactly once within the time window while satisfying the constraints imposed upon by vehicle capacities.

We first set up some definitions and assumptions essential to creating our model. We represent our network of customers (also called "bins" or "collection sites"), depot(s), and disposal site(s) as a graph $G = (V, E)$ of V vertices and E edges. The set of nodes V represent and encapsulate the set of depot(s) (V^d), consumers (V^c), and disposal site(s) (V^p), that is $V = \{V^d \cup V^c \cup V^p\}$. The size of V (denoted as $|V|$) is therefore computed as the sum of the number of depot(s) ($|V^d| = n_d$), consumers ($|V^c| = n_c$), and disposal site(s) ($|V^p| = n_p$), that is $|V| = |V^d| + |V^c| + |V^p| = n_d + n_c + n_p$. In our case, we consider only one central depot, represented as $V^d = \{0\}$, n_c consumers $V_c = \{1, ..., n_c\}$ and n_p disposal sites $V^p = \{n_c + 1, ..., n_c + n_p\}$, respectively. Each node $\in \{1, 2, ..., n_c\}$ represents each of the consumers taken into consideration. Hence, $|V| = n_c + n_p + 1$. The set of edges $E = \{(i,j)|i,j \in V, i \neq j\}$, that is each $(i,j) \in E$ connects the pair of nodes i and j in V. We can visualize graph G as in figures 1 and 2.

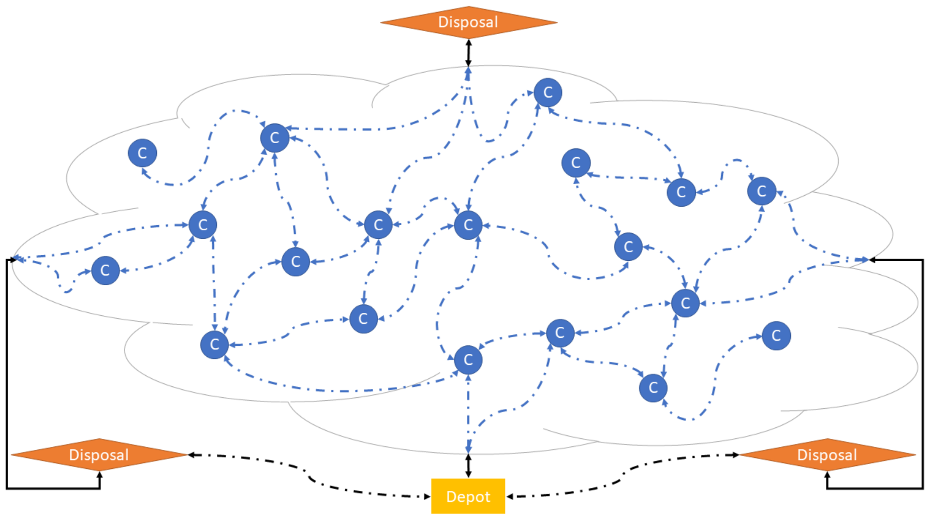

Figure 1: Graph Representative Visualization

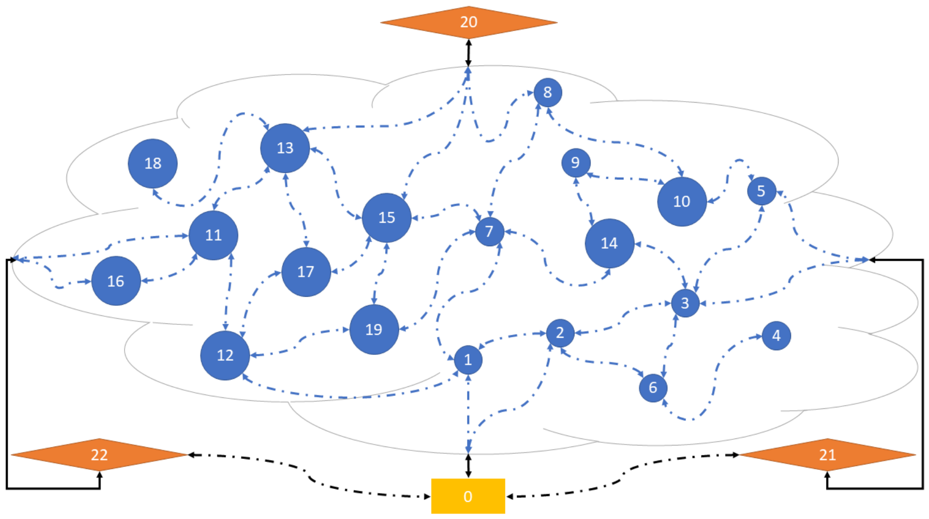

Figure 2: Graph Numerical Visualization

Let $K = \{1, ..., k\}$ be the set of waste collection vehicles and let $time_{(i,j)}$ and $dist_{(i,j)}$ be the travel time and distance associated with edge $(i,j) \in E$, respectively. This means, that if for any vehicle $l \in K$ that traverses edge (i,j), it travels from nodes i to j and covers a distance of $dist_{(i,j)}$ space units in $time_{(i,j)}$ time units. Each node $i \in V$ is associated with service time $serve_i$ and time window $[a_i, b_i]$. We define q_i as the amount of waste collected at a consumer $i \in V^c$. It is assumed that all vehicles can carry at most Q_{cap} amount of waste. As vehicle $l \in K$ traverses it's route, it cumulatively collects the amount of waste for every consumer $i \in V^c$ it services. When the cumulative amount reaches some $90 - 100\%$ of Q_{cap}, the vehicle then travels to a disposal site $i \in V^p$ to dispose of its load and replenish its full carrying capacity. Service time $serv_i$ is the amount of time needed to collect the waste at consumer $i \in V^c$ or the amount of time needed to unload the vehicle at disposal site $i \in V^p$. Note that no waste is collected at either the depot $\{0\}$ and at disposal sites $i \in V^p$ and that the service time at the depot is not necessary, hence $serv_0 = 0$. If any consumer $i \in V^c$ has a greater amount of waste than the maximum carrying capacity of the vehicle, that is $q_i > Q_{cap}$ then we create more instances of consumer i and divide the amount of waste accordingly.

We now identify our assumptions for the model. Some may be repetitions of the above definitions.

1. Each vehicle must start and end at the depot. Also, no garbage is collected at the depot.

2. All our vehicles $l \in K$ are homogeneous, that is, they all have the same model and carrying capacity Q_{cap}.

3. Graph G is connected. That is, all nodes $i \in V$ are reachable from any node $j \in V$. Mathematically stated as "there exists a simple path (made up of edges in E) that connects any pair of nodes $i, j \in V$". It follows that all vehicles $l \in K$ can travel on land to reach any location $u \in V$ using a sequence of nodes $(i, j) \in E$. We do not include locations that cannot be reached or traveled to using our vehicles.

4. Graph G is complete. That is, there exists a path, direct or indirect from any $i \in V$ to all other nodes $j \in V \setminus i$. Also that the path is the shortest possible path from i to j. We can satisfy this using the Floyd-Warshall algorithm or applying the Dijkstra's Shortest Path algorithm for all nodes $i \in V$. We consider this because not all locations in a map can have a direct roadway that connects them, but we can generate an indirect path using known roadways.

5. We consider a divide-and-conquer method wherein we only consider creating a route for each vehicle per day. We proceed with generating routes for each day of the seven days of the week.

6. All values for distance and time for any edge $(i, j) \in E$ is known and that graph G is symmetric.

We now proceed to model the problem. We discuss how we decode or 'make-sense' of the orderable list of consumers given by/and utilized in the PSO-GA algorithm. From the previous section, we defined that each particle or chromosome in the population is composed of some n_c dimensions representing bins or garbage collection sites around the city. Each dimension $dim_f, f \in [1, n]$ is assigned a real number in $(0, 1)$. These real numbers will be used to sort the n_c collection sites and place them into a list similar to the implementation of Masrom[14]. The visualization of these steps are shown on figures 3 and 4. An example of mapping the sorting is shown on figure 5.

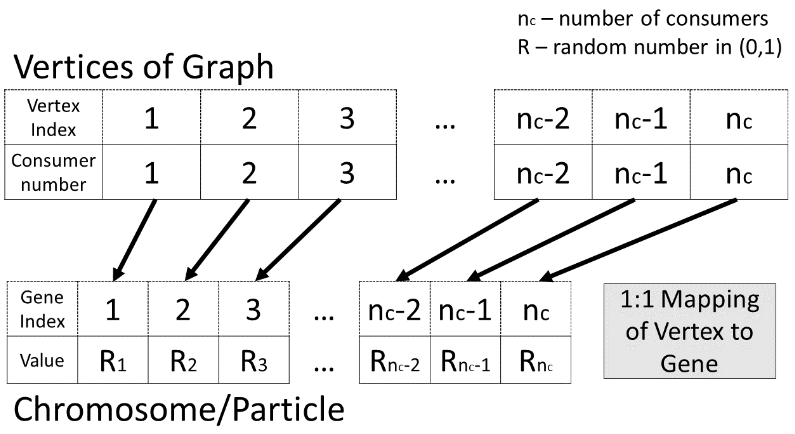

Figure 3: Vertex to Chromosome/Particle Gene

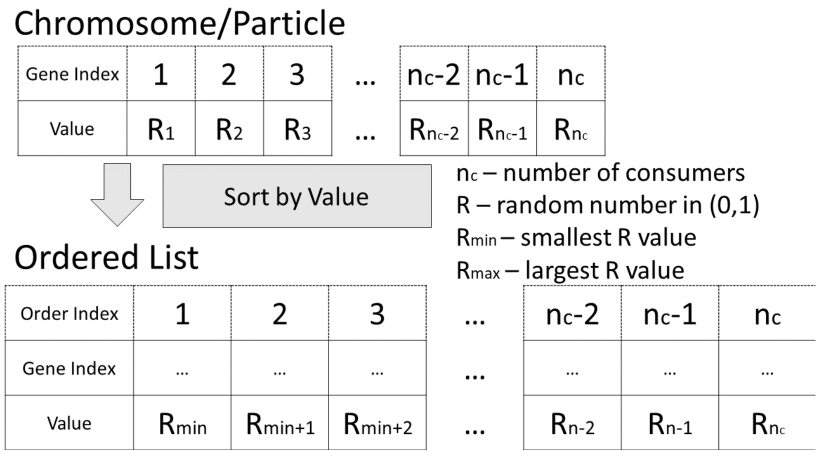

Figure 4: Chromosome/Particle to Ordered List

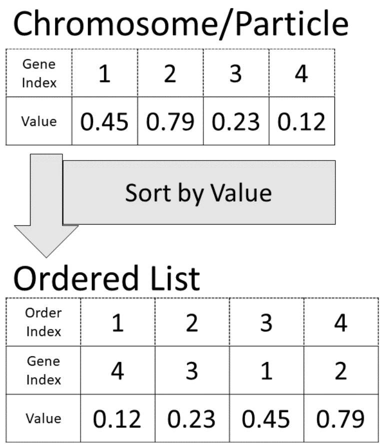

Figure 5: Chromosome/Particle to Ordered List Example

Vehicle routes are then made from the sorted list using some sort of insertion heuristic inspired from Solomon[15]. We define our decision variable $edgeT_{(i,j),l} \in \{0,1\}$ is 1 if and only if vehicle $l \in K$ utilizes edge $(i,j) \in E$, otherwise, it is 0. $accumW_{i,l}$ represents the accumulative waste collected at node $i \in V$ for vehicle $l \in K$. $begS_{i,l}$ represents the start time of service at node $i \in V$ for vehicle $l \in K$. $arrival_{i,l}$ represents the time of arrival of any vehicle $l \in K$ at any node $i \in V^c$.

Our insertion heuristic is given as follows:

Initialization

- Set $edgeT_{(i,j),l} = 0, \forall (i,j) \in E, l \in K$
- Set $accumW_{i,l} = 0, \forall i \in V, l \in K$
- Set $begS_{i,l} = 0, \forall i \in V, l \in K$
- Set $arrival_{i,l} = 0, \forall i \in V, l \in K$

Insertion

(a) Start with the route sequence $(0,0)$ with vehicle $l = 1$. Thus, the initial route length $routeLen = 2$. Also, set $edgeT_{0,0} = 1$

(b) For each node $u \in [1, n_c]$ in the ord list ord do the next item. Note that $ord_u \in V^c$, also, that u is the index of the elements in the ord list while the value of ord_u is associated to nodes in V^c (recall the mapping of nodes to gene indices to order indices).

(c) For every two consecutive nodes ($routeN_z$ and $routeN_{z+1}$, $z \in [0, routeLen - 1]$) in the route, try inserting ord_u. Note: revert any changes if ord_u cannot be inserted between $routeN_z$ and $routeN_{z+1}$.

- Flip the values of $edgeT_{(routeN_z, routeN_{z+1}),l}$, $edgeT_{(routeN_z, ord_u),l}$ and $edgeT_{(ord_u, routeN_{z+1}),l}$. This means that vehicle l will not traverse edge $(routeN_z, routeN_{z+1})$ and instead, it traverses edges $(routeN_z, ord_u)$ and $(ord_u, routeN_{z+1})$. Note that $edgeT_{(i,j),l}$ takes on binary values.
- Calculate the arrival time of vehicle l at node ord_u using:

$$arrival_{ord_u,l} = begS_{routeN_z,l} + serve_{routeN_z} + time_{(routeN_z, ord_u)} \tag{7}$$

- Calculate the time when vehicle l starts service at $orderd_u$ is using:

$$begS_{ord_u,l} = \begin{cases} a_{ord_u} & \text{if } arrival_{ord_u,l} < a_{ord_u} \\ arrival_{ord_u,l} & \text{if } arrival_{ord_u,l} <= b_{ord_u} \end{cases} \tag{8}$$

  The first case also constitutes that vehicle l has to wait at node ord_u before servicing said node.

- If vehicle l's starts servicing node ord_u beyond its allocated time window $[a_{ord_u}, b_{ord_u}]$, (expressed as $begS_{ord_u,l} > b_{ord_u}$) then ord_u cannot be inserted between $routeN_z$ and $routeN_{z+1}$.
- Calculate the value of the accumulated waste collected at ord_u using:

$$accumW_{ord_u,l} = accumW_{routeN_z} + q_{ord_u} \tag{9}$$

- If the accumulated waste collected at ord_u is at least $90 - 100\%$ of Q_{cap} and is not already succeeded by a disposal site, add the nearest disposal site $disp \in V^p$ to the list after ord_u.

  In adding the disposal site, we flip the values of $edgeT_{(ord_u,routeN_{z+1}),l}$, $edgeT_{(ord_u,disp),l}$ and $edgeT_{(disp,routeN_{z+1}),l}$.
- If on the other hand, the accumulated waste collected at ord_u is greater than the capacity Q_{cap} of vehicle l then ord_u cannot be inserted.
- **Push forward** the nodes that come after ord_u in the route list. It is done for each node $routeN_i$ that comes after ord_u. Push forward at $routeN_i$ is done as follows:
  i. If $routeN_i$ is a **consumer** ($routeN_i \in V^c$), do the following
     - Adjust the arrival time of vehicle l at $routeN_i$ using the equation

$$arrival_{routeN_i,l} = begS_{routeN_{i-1},l} + serve_{routeN_{i-1}} + time_{routeN_{i-1},routeN_i} \tag{10}$$

     - Adjust the start of service at $routeN_i$ using the equation

$$begS_{routeN_i,l} = \begin{cases} a_{routeN_i} & \text{if } arrival_{routeN_i,l} < a_{routeN_i} \\ arrival_{routeN_i,l} & \text{if } arrival_{routeN_i,l} <= b_{routeN_i} \end{cases} \tag{11}$$

       The first case also constitutes that vehicle l has to wait at node $routeN_i$ before servicing said node.
     - If start of service at $routeN_i$ is still within it's time window ($begS_{routeN_i,l} \in [a_{routeN_i}, b_{routeN_i}]$), proceed to the next item. Otherwise, declare that ord_u cannot be inserted between $routeN_z$ and $routeN_{z+1}$
     - Adjust the value of the accumulated waste collected at $routeN_i$ using equation

$$accumW_{routeN_i,l} = \begin{cases} 0 & \text{if } routeN_i \in V^p \text{ (a disposal site)} \\ accumW_{routeN_{i-1}} + q_{routeN_i} & \text{otherwise} \end{cases} \tag{12}$$

     - If the accumulated waste collected at $routeN_i$ is at least $90 - 100\%$ of Q_{cap} and is not already succeeded by a disposal site, add the nearest disposal site $disp \in V^p$ to the list after $routeN_i$.

       In adding the disposal site, we flip the values of $edgeT_{(routeN_i,routeN_{i+1}),l}$, $edgeT_{(routeN_i,disp),l}$ and $edgeT_{(disp,routeN_{i+1}),l}$.
     - If on the other hand, the accumulated waste collected at $routeN_i$ is greater than the capacity Q_{cap} of vehicle l then declare that ord_u cannot be inserted between $routeN_z$ and $routeN_{z+1}$
     - If the accumulated waste collected at a node is at least $90 - 100\%$ and is not already succeeded by a disposal site, add the closest disposal site to the list after this node and continue to the next node i.e. (increment i)

ii. If $routeN_i$ is a disposal site ($routeN_i \in V^p$), do the following
   - If the accumulated waste collected at the previous node $routeN_{i-1}$ is at least 90% the capacity of vehicle l then proceed to the next item. Otherwise, remove this node ($routeN_i$) from the route.
   - Adjust the arrival time of vehicle l using equation (10).
   - Adjust the start of service at $routeN_i$ using equation (11).
   - If start of service at $routeN_i$ is still within it's time window ($begS_{routeN_i,l} \in [a_{routeN_i}, b_{routeN_i}]$), proceed to the next item. Otherwise, declare that ord_u cannot be inserted between $routeN_z$ and $routeN_{z+1}$
   - Adjust the value of the accumulated waste collected at $routeN_i$ using equation (12).
   - Proceed to the next if it exists. (i.e. increment i)

iii. If $routeN_i$ is a depot ($routeN_i \in V^d$), do the following
   - Adjust the arrival time of vehicle l using equation (10).
   - Adjust the start of service at $routeN_i$ using equation (11).
   - If start of service at $routeN_i$ is still within it's time window ($begS_{routeN_i,l} \in [a_{routeN_i}, b_{routeN_i}]$), proceed to the next item. Otherwise, declare that ord_u cannot be inserted between $routeN_z$ and $routeN_{z+1}$
   - If $routeN_i$ is not preceded by a disposal site, then make a disposal trip before this node and go back one node (i.e. decrement i).

(d) If we managed to insert node ord_u, then remove it from the list. Otherwise, go to the next node in the list if it is not empty (i.e. increment u).

(e) If we have exhausted the list of nodes that can possibly be inserted into the route of vehicle l, and there are still nodes in the list then move on to the next vehicle, that is $l = l + 1$. Otherwise, the we have completely inserted every consumer node then we proceed to evaluating the objective function value of the particle/chromosome.

Our objective function is as follows:

$$\min F(X) = \lambda \left(\sum_{(i,j)\in E} \sum_{l \in K} dist_{(i,j)} \cdot edgeT_{(i,j),l} \right) + \delta \left(\sum_{(i)\in V^c} \sum_{l\in K} (\max\{a_i - arrival_{i,l}, 0\}) \right) + \omega \left(\sum_{l\in K} 1 \right) \tag{13}$$

where $\lambda + \delta + \omega = 1$

For all vehicles $l \in K$, we add the distances of all the edges that the vehicle utilized in servicing consumers in it's route. We know which edges are utilized by vehicle l when $edgeT_{(i,j),l} = 1$. If the edge is not in the route of vehicle l, then no distance is added to the total sum because $edgeT_{(i,j),l} = 0$. We also add the amounts of time that all vehicles in K wait at any node $i \in V^c$. Waiting time is the time expended when a vehicle $l \in K$ arrives at consumer/node i before the alloted time slot of consumer i. (Expressed as $arrival_{i,l} < a_i$) Lastly, we add the number of vehicles or routes produced in the entire route. (Note that each vehicle is designated a route to follow for the whole day.) The values λ, δ, and ω dictate how much each term (distance, time, number of vehicles) is prioritized.

6 Example

7 Trial

References

[1] A. E. Plaza, "Ditch nimby to fix philippines municipal solid waste problem," *Asian Development Blog.*

[2] D. See, "City to purchase 4 more trucks," *Sun Star Baguio.*

[3] D. See, "Approval of citys solid waste plan in order," *Sun Star Baguio.*

[4] F. Olowan, E. Bilog, L. Y. Jr., E. Avila, J. Alangsab, E. Datuin, P. Fianza, L. Farinas, A. Allad-iw, B. Bomogao, and M. Lawana, "Baguio city council okays plastic free ordinance," *Sun Star Baguio.*

[5] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manage. Sci.*, vol. 6, pp. 80–91, Oct. 1959.

[6] S. Sahoo, S. Kim, B.-I. Kim, B. Kraas, and A. Popov Jr., "Routing optimization for waste management," *Interfaces*, vol. 35, pp. 24–36, Jan. 2005.

[7] A. W. J. Kolen, A. H. G. R. Kan, and H. W. J. M. Trienekens, "Vehicle routing with time windows," *Operations Research*, vol. 35, pp. 266–273, 1987.

[8] B. Ombuki, B. Ross, and F. Hanshar, "Multi-objective genetic algorithms for vehicle routing problem with time windows," vol. 24, pp. 17–30, 02 2006.

[9] L. H. Son, "Optimizing municipal solid waste collection using chaotic particle swarm optimization in gis based environments: A case study at danang city, vietnam," *Expert Systems with Applications*, vol. 41, no. 18, pp. 8062 – 8074, 2014.

[10] M. Akhtar, M. A. Hannan, and H. Basri, "Particle swarm optimization modeling for solid waste collection problem with constraints," in *2015 IEEE 3rd International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, pp. 1–4, Nov 2015.

[11] K. Buhrkal, A. Larsen, and S. Ropke, "The waste collection vehicle routing problem with time windows in a city logistics context," *Procedia - Social and Behavioral Sciences*, vol. 39, pp. 241 – 254, 2012. Seventh International Conference on City Logistics which was held on June 7- 9,2011, Mallorca, Spain.

[12] P. Kaur and P. Kaur, "A hybrid pso-ga approach to solve vehicle routing problem," *International Journal of Engineering Develpment and Research (IJEDR)*, vol. 3, August 2015.

[13] X. Liu, W. Jiang, and J. Xie, "Vehicle routing problem with time windows: A hybrid particle swarm optimization approach," in *2009 Fifth International Conference on Natural Computation*, vol. 4, pp. 502–506, Aug 2009.

[14] *Hybrid Particle Swarm Optimization for vehicle routing problem with Time Windows.*

[15] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," vol. 35, pp. 254–266, 04 1987.

[16] T. Nuortio, J. Kytjoki, H. Niska, and O. Brysy, "Improved route planning and scheduling of waste collection and transport," *Expert Systems with Applications*, vol. 30, no. 2, pp. 223 – 232, 2006.

[17] J.-F. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo, "Chapter 6 vehicle routing," in *Transportation* (C. Barnhart and G. Laporte, eds.), vol. 14 of *Handbooks in Operations Research and Management Science*, pp. 367 – 428, Elsevier, 2007.

[18] H. Garg, "A hybrid pso-ga algorithm for constrained optimization problems," *Applied Mathematics and Computation*, February 2016.