Problem Domain

Hybrid algorithm for UAV routing and supply delivery.

The first NPC problem which will be considered is a variation of the VRP where we have some number of UAVs D, each with a uniform amount of fuel F, and some number of target locations which need to be observed T. The base from which they leave will be the "depot" in this case and will be a special element of T known as T_{Home}. The cost c of the edge $(t_i,\ t_j)$ will be the amount of fuel required to travel that distance. This is similar to what was done in [1] without the consideration of the complex urban environments. The constraints will be that each location must only be visited by one drone, since it is unnecessary to have multiple viewing the same location, and some locations will be marked as "high priority" and must be visited first. We will assume that the terrain is level and all drones fly at the same altitude. More formally, the problem can be stated as follows: Given a graph G = (T, C), where T is the set of targets and C is the set of edge costs in gallons of fuel, find a set $r = \{T_{Home} \cup t_i\ \forall i \in T | c < F \text{ and } c \text{ is minimized}\}$ for each drone in D. Additionally $\forall d_i \in D,\ r_i \cap r_j = \{T_{home}\} \forall i \neq j$.

The second NPC problem to be incorporated will be the 3D bin packing problem [2]. Assume that all locations require supply drops of varying size and each drone has a fixed 3 dimensional capacity L. We want to minimize the number of drones required to make supply drops. This problem on its own can be stated as follows [3]: Given a set of items I, a set of sizes S representing the size of each item and a bin capacity L find the minimum K such that $K = \sum_{j=1}^{n} y_j$ where $y_j = 1$ if drone d_j is used to carry supplies.

The combined objective function can be formulated as follows:

$H = [cost(t_i, t_j)\ \forall i \in r_x\ \forall x \in D] + \sum_{j=1}^{n} y_j$. Our goal is to minimize this hybrid objective function. In other words find the routes that use the least amount of fuel while simultaneously finding the least number of drones required to make all supply drops.

D_i: G = (T, C), where T is the set of targets and C is the set of edge costs in gallons of fuel and each node T has a set of items I with sizes S that must be dropped there

D_o: A map M from targets to drones which represents which drone visited which targets and all targets have drones assigned to them

Algorithm Design

Representation of the genes will be a set of routes and items for each available drone where t# represents the targets and i# represents the items carried.
Example:
[D1([t2,t4,t7] , [i1,i5]) ,
D2([t1,t2,t3] , [i2]) ,
D3([t6,t6] , [i3,i4, i6])]

Feasibility function will total up the fuel cost of all t's and the volume of all i's and if it exceeds the limit for any drone d the solution is rejected. It will also check to ensure that no t's or i's are duplicated since no drones can take the same item more than once and no drone should visit a location more than once.

Fitness function will use the equation $[cost(t_i, t_j)\ \forall i \in r_x\ \forall x \in D] + \sum_{j=1}^{n} y_j$ to determine the cost of the solution. However, since partial solutions are possible and we want to minimize this value extra consideration will need to be taken to ensure the algorithm doesn't get stuck on seemingly "fit" partial solutions (since they will obviously be a lower cost). Any solution will have a large fixed value added to its fitness for any targets or items that are not accounted for by the partial solution.

Selection will use the fitness of each solution to determine its probability of being added into the pool for reproduction. Once the probabilities are determined for each solution, solutions will be added to a linked list a porportial number of times to their probabilities therefore making those with higher probabilities more likely to be picked.

Crossover will operate on the target and item lists separately for each drone and switch half of them with another random drone in the list.

Mutation with some low probability will alter the number of one single target or item for one drone.

Termination If none of the children improve from one generation to another terminate.

These techniques are similar to those employed in [1]. Their paper was only on the VRP, here we take a hybrid approach.

GA Algorithm

Input: Population size, mutation %, drones, targets, items
Output: A list of drones with their routes and items

```
Init population p at random
While not termination criteria do
   For solution in p
      Solution.score = fitness(solution)
   End loop
   For solution in p
      For i in range(0,solution.score)
         Push solution into pool
      End loop
   End loop
   For i in range(0,population size*2)
      Generate random number rng1
      Generate random number rng2
      Generate random number rng3
      Dad = pool[rng1]
      Mom = pool[rng2]
      Child = crossover(mom,dad)
      If rng3 meets mutation criteria mutate(child)
      Add child to population[i+1]
   End loop
Return highest scoring solution in most recent population
```

The researchers in [2] discuss how choosing the parameters for a genetic algorithm can affect the outcome of the solution. This process can be considered an art since it is usually not feasible for real world problems to step through, debug, and witness what your algorithm is doing at step. Additionally, this is a stochastic algorithm so the results will vary each time it is run. It requires intimate knowledge of the problem and algorithm domains to make the right choices for the parameters to achieve the desired goal. In this sense the process of choosing these can be considered an art since at a certain point it is based on intelligent guesses and gut feeling about how it will affect the generation of new solutions.

b) For exercise 3.9 which refers to the roulette wheel selection it is easy to see why this method of selection only requires O(n). For if the required population size is n and we have an implementation such as the one above where the solutions are placed into a list a proportionate number of times for their fitness then we only need to make n random selections from that list. In regard to 3.8 these methods of selection may not be as fast as roulette but they help to avoid the issues introduced by the simple random roulette approach. Such as very fit solutions dominating the selection and limiting the genetic diversity for future generations.

References

[1]Masum, A. K. M., Shahjalal, M., Faruque, F., & Sarker, I. H. (2011). Solving the vehicle routing problem using genetic algorithm. *International Journal of Advanced Computer Science and Applications*, *2*(7), 126-131.

[2]Boyabatli, O., & Sabuncuoglu, I. (2004). Parameter selection in genetic algorithms. *Journal of Systemics, Cybernetics and Informatics*, *4*(2), 78.