Problem Description

With a randomly generated graph where each vertex represents a location, there are a random number of vehicles and a random number of packages. All the vehicles start at a garage and the packages are assigned to be delivered by these vehicles from their random start point to their random destination. The vehicles must all end at the garage.

Solution Description

a. Initial State
   Initially the state of the problem is that a number of vehicles are stored at one specific graph point with packages stored at various points on the graph and these packages have designated destinations which they must arrive at.
b. Actions
   The actions which can be taken are moving an empty vehicle to the position of a package where the vehicle picks up the package, moving a vehicle holding a package to the destination of the package and dropping the package off, and moving a vehicle back to the garage.
c. Transition model
   Each transition will result in a vehicle being moved to another position, along with a package either being picked up or dropped off, or if no package is involved then the vehicle will be returning to the garage. The successive state has the accumulated cost, the position of each vehicle, package and the garage, whether each package has been picked up and/or delivered and whether each vehicle has returned back to the garage. Each transition also increases the total cost of solving the problem.
d. State Space
   The state space is all the combinations of locations for the vehicles, combined with whether or not a package has been picked up or delivered and whether the vehicles have returned to the garage or not.
   In a easier to visualize sense, all the different states tried were different arrangements of what packages could be picked up in what order, and what vehicles could be assigned to pick up each package.
e. Goal Test
   The goal test checks to see if all the packages have been delivered
f. Path cost
   The cost of a path is the sum of the cost of picking up and delivering each package, combined with the cost of the return trip of each vehicle to the garage. The step cost would be the cost of travelling to a location.
g. Optimal solution
   The optimal solution is that which travels the least distance in delivering all the packages and returning home.

Implementation Description

We tried approaches of A*, simulated annealing and iterative deepening on nodes for a graph in attempting to solve this search problem in order to compare how the different techniques could overcome the problem.

Astar and Iterative Deepening

In a naïve version of our implementation for A star and iterative deepening, the movement of a vehicle to any other adjacent node, as well as picking up or releasing a package, are considered transitions. In this implementation, when a vehicle finds itself in the same position as a package it picks up the package, when a vehicle carrying a package reaches the same location at the packages destination it drops of the package, and when a vehicle has no further duties the search finds the best path for the vehicle to return to the garage. From the current state, every state which may be moved to involve a replication of the current state with one of the vehicles on an adjacent node to it's original position or a replication of the current state with a package either being picked up or released from the vehicle on the identical node. For any values of m,n or k which were larger than trivial, it was not practical to calculate every move to another node which a vehicle could make. Each move to a new node gives an amount of new states equal to the branching factor of that node and the problem soon becomes intractable when simply moving to a new node is the transition for this search.

Because of the impracticality of the naïve approach we considered the transitions of our search to be the movement of a vehicle with no package to the location of a package, the movement of a vehicle carrying a package to the location of the destination of that package, and the return of the vehicle to the garage. Because there are likely very few reasons for a vehicle to be at the location of a package to not pick the package up, and no reasons for a vehicle to keep a package once the destination of the package has been reached, the obtaining and releasing of a package are considered to be in the same transition as moving to the location of a package or destination. The cost added to the search for each move can be calculated from Djikstras algorithm. Djikstras algorithm runs in a time complexity equal the number of vertices squared in order to calculate the shortest distance between two vertices, so this can be computed in a reasonable amount of time. The number of states for this reduced algorithm, is then only the (# of vehicles) * (# of packages) * (package status) where the naïve approach would have an amount of states equal to (# of vehicles) * (# of nodes) * (package status).

AStar

In our A* implementation we use a heuristic which sums the costs of

- the shortest path between the source and destination of each non-picked up package + the shortest path between any pair of critical vertices

- the shortest path between each packages location and destination for each package which is picked up but not delivered + the shortest path between any pair of critical vertices
- the cost of the shortest path between any pair of critical vertices for each vehicle which has no package but has not returned to the garage

This heuristic assigns lower costs to packages which have been picked up but not delivered, and even lower costs to vehicles which simply need to return to the garage as these states require less distance to be travelled. In our simple heuristic we do not consider the shortest paths of packages and vehicles to their destinations and thus this heuristic would not cause a very optimal determination of ordering for picking up packages. When taking the shortest path costs into account this heuristic becomes much more accurate.

Using our heuristic the next state chosen to move to is that state which is predicted to have the lowest cost in the long run. The heuristic achieves a low cost result by always choosing the state that minimizes the cost traversed so far as well as the projected cost which will be needed to deliver the rest of the packages. The next package for a vehicle to deliver can be determined by this heuristic, which estimates the total distance still needed to be travelled after another package is delivered.

Iterative Deepening

Iterative Deepening is repeated depth limited search. In a graph it starts it's search by searching very small distances from the start position and by gradually increasing the distance searched. In the first step it only searches paths which travel one node from the start position. In the second step it only searches paths which travel 2 nodes from the start position and so on until a step is reached where a solution to the problem is found. In our simple naïve approach, it finds the optimal solution for shortest distance travelled because our goal is to find a path which travels the smallest distance. Although this situation quickly becomes intractable as the number of vertices, vehicles and packages increases, the optimal solution will be found with this approach.

With our reduced graph our iterative deepening algorithm did not have much practical use. Its method would be to always assign open vehicles to deliver packages as this would mean taking the least amount of steps, but this may not always lead to an optimal solution.

Simulated Annealing

Our simulated annealing took a more distinct route than the other methods we tried. In our simulated annealing approach the state space strictly stuck to what was the best combination of assigning packages to be delivered by vehicles. Packages were at first randomly assigned to be picked up by any vehicle in any order. An list of lists of packages represented the assigning of packages to vehicles. Each spot in list in the list of lists represented a vehicle each of these vehicles(lists) had a list of packages to be picked up and delivered. Djikstras algorithm was used to calculate the shortest distance to each destination once the routes were chosen as it

is able to calculate the shortest distance in a graph via breadth first search, and by using Djikstras algorithm for all the destinations which must be travelled be travelled, the shortest distance for all trips could be minimized.

Once the initial state was assigned (an ordering of what packages could be picked up by which vehicle) successive states were able to be considered by creating a clone of the current state, with the exception of reassigning a random package to a random vehicle at the end of the vehicles route (the package has the potential to be reassigned to the same vehicle). Using Djikstras algorithm for calculating the shortest distances for the routes chosen, the cost of this new state was compared to the cost of the current state. If the cost of the new state was lower than the current state, then this new state was chosen; however, if the cost of the new state was higher than the current state, a probability function was used to decide if this new state would become the current state.

Figure 1 gives a visual representation of the benefit of simulated annealing. When a completely random state is chosen if other states are only chosen when the other state gives a better value, then the absolutely best state may never be reached. When making only a minor change (such as changing the delivery method of one package) then it may not be possible to end up in the global minimum. This is because it may be necessary sometimes to go to a state with a higher value in order to reach a state with a lower value. This trend can be visualized in the middle of

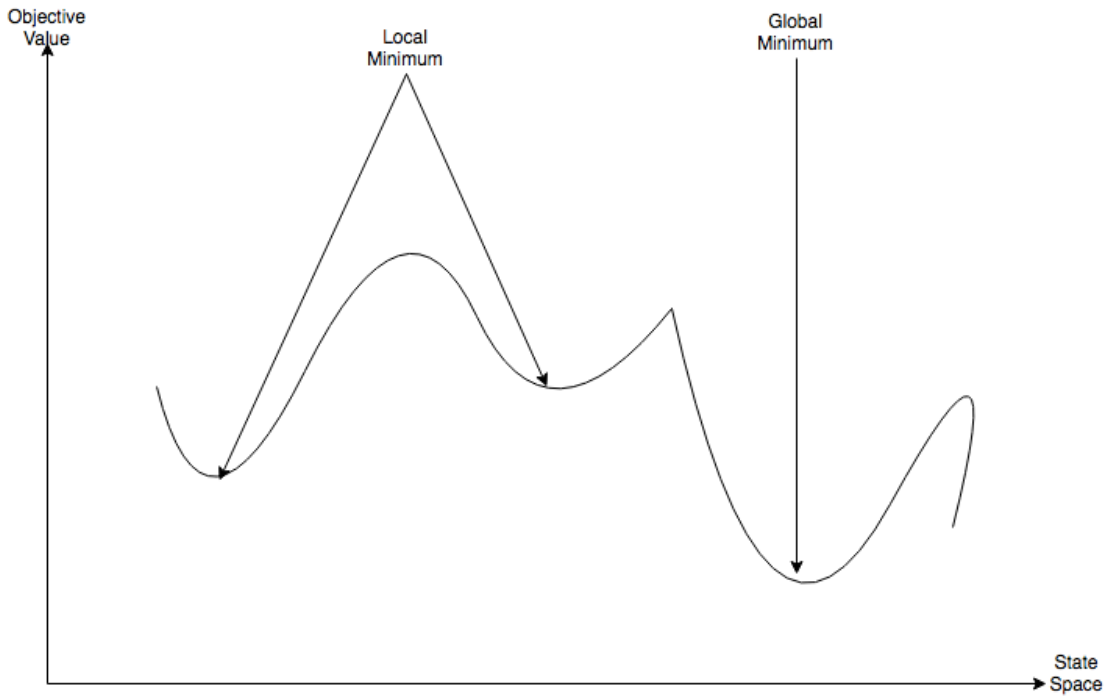

Figure 1: A 2-dimentional X-Y graph representing value in a state space

figure one. If a search were being performed on this graph and started near the middle, if new states were only accepted when the new state gave a lower minimum

then the global maximum could never be reached. Because of this I allowed my simulated annealing algorithm to take higher values sometimes depending on our probability function of $e^{(new\_state\_cost-current\_state\_cost)/Temperature}$ where temperature was an arbitrary value which started high and gradually decreased. As temperature decreased the probability of accepting a state with a higher cost became lower. A greater difference in the cost of the cost of the current and new state also resulted in a lower probability of a state with higher cost being accepted.

Results

N (Vehicles)	K (Packages)	M (Vertices)	Average time for Iterative Deepening (ms)	Average time for A* (ms)
1	1	100	63.1	6.2
1	2	100	3907.2	13.4
1	3	100	N/A	22.4
1	4	100	N/A	90.1
2	1	100	342.2	4.3
2	2	100	27524.7	25.8
2	3	100	N/A	325.5
2	4	100	N/A	7853.7
3	1	100	285.1	5.1
3	2	100	N/A	34.3
3	3	100	N/A	845.0
3	4	100	N/A	2087.3
4	1	100	366.3	8.3
4	2	100	N/A	91.6
4	3	100	N/A	370.1
4	4	100	N/A	11648.9

Our method of using Astar showed to be much more efficient than the other two approaches as simulated annealing could not compute any values of M which approached 100, and iterative deepening could compute some but much slower than Astar would.

N (Vehicles)	K (Packages)	M (Vertices)	Average cost for simulated annealing (ms)	Average cost for local search (ms)

2	3	10	8	10
5	5	10	17	19
5	5	10	19	20
5	10	50	42	44
10	10	50	41	45

As simulated annealing does not guarantee the optimal answer, but can sometimes get a good answer, I compared it to a local search algorithm(essentially the same algorithm but which never accepted any values which were higher for a new state). The simulated annealing algorithm used did show to be slightly better than the local search algorithm and with a higher temperature could be even better.

Conclusions

Out of the three approaches to search, Astar proved to be our most efficient solution. This does not however show that Astar is a superior approach to the other two in general or even for this situation. The iterative deepening and simulated annealing approach had slower timing that the Astar approach and for larger values of N and K in the problem quickly became intractable, but by tweaking the problem and looking for heuristics, it could be turned into a more efficient approach. The simulated annealing approached needed to calculate the total cost of every path for every state considered for example so it could no even be compared to the two other approaches as when it was given 100 vertices, the problem could not compute in a reasonable amount of time. Even very small numbers of n,k and m the simulated annealing approach to much longer. The approach in simulated annealing of only switching one package could also be adjusted. An adjustment to it's heuristic which I think would be better would be the ability of a package to be inserted in any order for a vehicles path, as right now packages can only be inserted at the end of a vehicles path. Most heuristics given to iterative deepening however would most likely need to take away from it's optimality, because it is guaranteed to give an optimal solution but heuristics often taken this guarantee away. Losing optimality may not be a bad decision however if it greatly increases speed while still keeping a solution which is close to optimal. Still, from this project, the Astar method of search showed to be an efficient method of search.