EXPERT SYSTEMS

PROJECT REPORT

Name: Bharath Kurumaddali

UFID: 6513-0561

**INTRODUCTION:**

A package delivery system for the company Packages-are-Us of Orlando,Fl was developed in CLIPS. The job description is simple as stated above there will be packages arriving at known cities and these packages have the information about where they must be delivered and also have a size corresponding to them which is necessary as it the parameter which determines if a particular package can go in a specific truck or not.

This concept was implemented in two different ways as two parts of the project. The first part was a definite way of implementation, whereas the second part is a heuristic approach to obtain good results.

In the first part it was required that all the trucks can carry only one package at a time and also that they all were initially in orlando and if a package arrived at a city the truck from orlando would go to the pick up point in the shortest route, pick the package then in the shortest route go to the destination and deliver the package. After, this has been completed the truck must return back to orlando in the shortest route. The simulation for the aforementioned process was required as part 1 of the project in addition to this it was necessary to calculate an onscreen report for all the trucks and packages.

The part 2 of the project required to employ a heuristic that would improve at the least one parameter from the above mentioned truck and package reports. The parameter that I have chosen is ‘Avg percent of truck occupied’. The main motive behind choosing this was that if I can increase the truck occupancy, it means more packages are being transported at the same time that means the distance of total travel is reduced from which we can deduce that if the total travel distance is reduced it means there is considerable saving in the fuel. Having this as the main motive part 2 was implemented. The logic to implement is that the given graph is dynamically analysed and some cities were selected as nodes and each node now headed a set of cities. Once the node selection has been done each city was taken and checked with each node as to which node it lies then a group was formed with this group and that city if a group already existed with that node then the city under consideration is just added to this group. So, at the end of analysis there will be nodes and some node elements as group. Then two paths were formed one was between the node and all elements of the group and the other is a path between all the nodes. So, the execution process is if a package arrives at a group member,it is bought in a truck and dropped at the main node and another truck picks it from there and delivers to the node in whose group the destination city lies. From this second node another truck picks it and delivers to the final destination. The implementation is explained in greater detail in the next section.

**DESIGN PHILOSOPHY:**

As mentioned above the main aim is to improve the occupancy percentage of a truck. I have done two different implementations for achieving the improvement in the percent occupancy of the truck. In the first one, initially the number of trucks were counted and and this number was halved, lets say this number is n. Now next the total number of cities in the map are calculated. This is done from the distance deffacts which had list of distance between a city and its adjoining city. All the cities were stored in a new fact and once all the distance deffacts were traversed the fact in which all these cities were being stored instantiated a new rule which extracted city by city and incremented a counter when each city was extracted. Care was taken so that this fact had only one occurrence of a city. At the end of this rule we have the total number of cities. So, till this point we have the number of cities and the number of trucks. Now, the half of number of trucks i.e n was kept as a threshold for number of nodes. The idea here is that the whole map must be divided into different groups and each group has a parent node and there will be group members. Now, that n is the threshold for the number of nodes, to find which cities should be selected as nodes we take each and every city in the map and count the number of branches it has that is we count the number of cities each city is connected to. The way this is done is by first selecting a city and checking to how many cities there is path from this city. Once, this has been found out all the cities have a corresponding number which are the number of connection each city has,the logic behind using this is that if a node is directly connected to maximum possible number of cities then the package most often will travel through the shortest route. So, now that we have a threshold of n, the first n cities in order of number of connections are selected and now all these are termed as ‘nodes’. Once the nodes have been found we have to divide all the remaining cities into a group that is headed by either one of the node. So, what happens here is that all cities that are not nodes are taken and then distance between each city and each node is calculated and the node to which the city lies closest, the city is added to the group headed by that node. Now, after repetitively doing this for each and every city, all the cities would have been placed in either one of the nodes.

Once these groups are ready the next step is to find two different paths. Lets call one as outer path and the other as inner path. First, consider the outer path suppose a group has the following (n1,c1,c2,c3) where n1 is the node and c1,c2,c3 are the cities that are there in this group the inner path defines a path between all these cities. For ex a path could be: n1->c1->c2->c3->n1. Note the path is infact a loop which starts and ends at the node. Also to be noted that the cities in the path might repeat i.e a city might appear twice or in some cases even more to form a complete path this is because in a situation where a city is connected only one other city it has to touch that city twise once while going and onc while returning back. The algorithm that was used was the simplest that is the brute force some possible ways were calculated and the first path that we find from the node returning back to node is taken as the inner path. Next, the outer path is calculated, this path is nothing but a path formed between all the nodes of the graph. For ex is (n1,n2,n3,n4) are the chosen nodes of a particular graph then n1->n2->n3->n4->n1 can be a path note this is also a loop starting from one node and ending at the same node after traversal through rest of the nodes. This path also is found by trail and error , a node is selected and a route is found to the nearest node, again this is repeated until all the nodes are removed now a path is formed between the final node and initial node making this a closed loop.

Now, in the first implementation the following was done, each node is assigned one truck, So, half the trucks are allotted here. What these trucks basically do is go around the inner loop and collect all the packages that are available to pick-up. In the first implementation the assigned truck starts from the node goes through its particular group and if a package is available and also there is enough space to accommodate the package it is added it truck, if the space is not enough it is not picked and will be picked up in the next round. Now each picked package is taken from its source and if its source is in the same group and the assigned truck will go to that place at a later point of time the package is left in the truck and when it reaches the destination it is dropped of directly and there will be no involvement of the node. If the destination is not in the same node, then the package is picked up and dropped at the node of that group and what I did in the initial implementation was I took only one truck and this truck continuously roamed around the outer loop and so effectively only t/2+1 number of trucks were used and the rest were idle or not used and so some saving would be there. This one truck continuously roaming in the outer loop picked up any available package at a node this package could have come to the node from one of the group members or could have originated in the node. This package is taken, in the truck parsing the outer loop and when it hops from node to node(the path is pre-fixed as explained above) it will check if anyone or more of the packages in the truck have the destination as the node in which the truck is or if the destination is a group member of this node, if the destination is a group member the package is dropped here and then the inner loop truck picks it up when it is at the node next time and delivers it to the destination. It is to be noted that that path of outer loop and inner loop are fixed and not dynamically changing.

In the second implementation, one major change was that instead of travelling only one truck through the outer loop, all remaining that is t/2 trucks roam around the same route, but the difference is that the initial locations are different and as all travel through the same path two outer loop trucks will not meet at the same time. This is the heuristic used to deliver all the packages.

**CHANGES FROM PART ONE AND THE RATIONAL FOR THEM**

First the list of implementations that change from part one:

1. Trucks don’t start from Orlando.
2. Trucks don’t carry only one package.
3. Trucks don’t return to Orlando after each delivery.
4. Trucks don’t go from Source city to destination city in the shortest route.
5. Trucks don’t respond to incoming packages immediately.
6. Addition of nodes.
7. Two separate loops-inner and outer.
8. Packages can be dropped at a node that is group member of destination.
9. Packages can be dropped at a node that is group member of source.

Reason for changing the above concepts is:

1. We have many nodes and trucks start from there one truck goes towards the inner loop and one outer loop. Starting from orlando each time gives an overhead each time as the first must come from orlando to that node and then go towards the inner loop. To avoid this over head the method is changed.
2. The main parameter chosen to improve was to increase the percentage of truck that is full and keeping only one package at a time is not idle as there might be free space and the truck roaming around could pick it up even if it is not empty. This eliminates totally the necessity to have another specialized truck to go and pick this package.
3. As explained in the point one to avoid unnecessary overhead that is induced as all trucks coming to orlando is removed.
4. Trucks don’t take the shortest route between the source and destinations, our aim is to increase the amount by which the truck is filled and this is done by fixing the routes and if a package is available at the time a truck reaches a city the package is picked up. So, this method is not package driven like the previous method, it is route driven that is the route is fixed and that route is followed irrespective of the package arricval.
5. As explained in the previous point the method is not package driven it is route driven and so even when a package arrives it does not pick it up , one might say we could implement in a way that a truck as soon as a package arrives goes picks it up and on the way to the destination could pick up more packages if possible but this generally involves using the shortest path and the probability of picking packages will be low and a new truck will be dispatched for each truck making no use of pick ups and drops as used here so the best way is to fix the path before hand and not change it dynamically for our parameter.
6. In the first method actually if we closely observe orlando can be regarded as a node for the whole map and all are cities are in its group, the inception of the idea to increase the number of nodes so that we have small distances between cities and picking and dropping will be more structured. These nodes create a local path and will decrease the total distance travelled by the truck.So, more number of nodes will effectively result in shorter distance in the inner loop, however the outer loop distance cannot be judged.
7. If we have only one loop and all trucks travel through these loops then the path will be extremely large and many packages might miss their due time. So, we have two loops one is local which will take care of all nodes inside a group and one which connects all these groups decreasing the effective distance travelled.
8. In the previous when a package is picked it cannot change trucks and must be delivered in the same truck. Now, we have changed that, a package can change trucks, suppose we say the package cannot change trucks all that happens is a big loop is formed as told above. The loop will be between the source to destination and back to source. This might decrease the time travel but will result in taking the shortest path to the destination and back which might miss some nodes where a package has to be collected that might go to the same location area i.e might not be the same city but a city around the destination and hence taking a relatively longer path can result in delivering more packages to the destination area and by this effectively to deliver 2 packages it should have gone is 2 different rounds if each distance is d then 2d distance must be covered. Now if we allow to drop package and not take the shortest route we could result in delivering both the packages in d+k time where k is the overhead or the extra distance travelled. By this time, fuel is saved and most importantly more percentage of truck is occupied.
9. The explanation given in the above point is the same explanation for this. The distance, time, fuel is saved and more percentage of the truck is loaded this can be done by having a slight overhead as explained above.

**THE GOOD AND THE BAD:**

|  |  |  |  |
| --- | --- | --- | --- |
| **PACKAGE NO** | ESTIMATED DELIVERY | PART1 DELIVERY TIME | PART 2 DELIVERY TIME |
| 1 | 15 | 4 | 18 |
| 2 | 10 | 8 | 12 |
| 3 | 25 | 19 | 18 |
| 4 | 30 | 30 | 31 |
| 5 | 40 | 32 | 32 |
| 6 | 45 | 43 | 47 |
| 7 | 80 | 71 | 76 |
| 8 | 100 | 88 | 89 |

Good:

Now seeing the above table alone we might say the packages are arriving later than the part1 of the project. But the thing to be noted is the percentage occupancy and the distance travelled with the packages for delivery. When these two are brought into play it can be seen that the total distance is much less compared to the first part. So, the packages delivery are comparable in both and only certain packages reach late marginally. So, the package might not go much early like in part 1 where the routes are direct. It takes some extra time but many packages are delivered simultaneously the occupancy is more and so most of the packages are on time with more occupancy is a positive about the heuristic.

|  |  |  |
| --- | --- | --- |
| TRUCK NUMBER |  | Part2 Avg Truck occupied |
| 1 |  | 78 |
| 2 |  | 22 |
| 3 |  | 41 |
| 4 |  | 83 |
| 5 |  | 0 |
| 6 |  | 91 |

Looking at the percentages of occupancy we can see how effectively all the packages are delivered there might be some overhead involved but trucks are going around with maximum they can carry and hence more effective. This table proves that the heuristic provides us with the good occupancy rate hence the total distance covered in delivering all the packages will be much less. The individual distances if seen will be more but collective distance will be less. Here the distances overlap i.e if package 1 and 2 are being carried around at the same time it means this distance will be counted as distance for delivering package 1 and also same distance is for package 2 but when collectively taken they happen synchronously and this is the reason we see the packages arrive late and the distance they have covered might not be least but (shortest\_distance to deliver package1 + shortest\_distance to deliver package 2)> (distance in the heuristic to simultaneously deliver package 1 and package2). This says individually each might be shorter but at the end of the day if we check the total distance travelled to deliver all packages will be much less.

THE BAD:

If wondering the main reason for lateness is that a package is dropped at the node and there might be significant wait time till another truck comes and picks it up. This time where no delivery happens is a major component in the lateness. This is the main reason that packages are late. If we could eliminate the wait time all packages could be delivered much before their scheduled time. This does not happen in the first as the program is package arrival driven and as soon as a truck is free it picks up and the time all the trucks are busy is very less and so the total wait time in the system is very less.

One major problem faced is that we are assigning a truck to the inner loop and this is the only truck that continuously goes around that path. Suppose we receive a package that must be delivered or picked up from one of the cities and if suppose the size of the package is more than the maximum capacity of the truck then this package can never be picked and will always be missed. Hence, we have a dangerous situation where the package might never be delivered. In the first implementation first the size of package is checked and that truck which has equal or more capacity is sent as a result in this case a package may be late but will never be dropped. This is a serious error in the implementation and a drawback of the implementation.

**WHAT I WOULD DO DIFFERENT:**

The main problem that I have to address is package dropping and here I would change the heuristic such that the trucks are periodically being jumbled and I have decide some criteria for the jumble. By jumble I mean that the truck going in the outer loop could be assigned to the inner loop and the one going in inner loop might be assigned to outer loop. This way a larger package can be picked up from any city and can go to any city of course the drawback here again would be that increase in wait time of these packages which are larger than truck capacity and so the lateness would increase.

One more thing I would like to reduce is the waiting time at nodes and so one way is to dynamically change the nodes while the execution. Suppose (n1,c1,c2) are in a group and n1 is the node. A package arrives and the destination is c1. Then make c1 the node of the group and others a group member this will take directly to the group and there is no requirement to wait for inner loop truck. Of course there will be some wait time for atleast on package even in this scenario if two or more packages are delivered to the same group.

The last change I would like to make will require the whole program to be re-written here if we observe the topology used in the inner loops it is kind of ring topology. I think a star topology would give better results. But here we need the node to be connected to every member of the groups achieving this is not impossible but slightly challenging. We cannot decide the number of nodes we need based on the number of trucks rather we have to decide all the nodes first which satisfy this condition and then assign trucks and later the left over trucks must be assigned to the outer loop. I feel these changes would better the current heuristic that I have considered.