SET10107 – Exam

Christopher Jones

2a.

Two chromosomes that could be used to represent the data types. One chromosome would contain the acceleration and braking and the other would contain the left and right steering. In each chromosome all the data types would be integers as there would be 3 different states. An example of this would be with accelerating and breaking. If the acceleration is set to 1 and the breaking set to 0, then the car would be accelerating and vice versa. However, an extra case would need to be added if no action is required and hence -1 would indicate no action. If the gene has -1 then the car will continue in straight line. This would be the same with the other chromosome for steering with 1 indication turn X way, 0 being off and -1 being do nothing.

2b.

Fitness = (Time Taken + (Number of Collisions \* Time Penalty))/% of Course Completed

The fitness above would be a good way of defining how good the EA has done. The fitness is a minimisation function as we want the car to have the fastest time around the track with the fewest crashes as it is supposed to be as safe as possible. This is where the “Time Penalty” would need to be parameter tuned as if the penalty is too light then the car will just go as fast as possible and not care about the number of collisions as long as it got a fast time. Too high a value and the EA may be too cautious to go around the track fast. Therefore, you would need to find a good value that allows the EA to be careful and fast. The “% of Course Completed” parameter is a division of the overall time because we want to increase the over fitness if the EA did not complete the track within the given time. Overall, this fitness function incorporates all the necessary requirements for the EA to know how it has done on the track

2c.

* Crossover = uniform

A uniform crossover would be a good crossover operator for this problem because it means the child has a 50/50 chance of getting a gene from either parent. This child can be made up of more of one parent than the other, vice versa and or a good equal split of both parents. As there are only 3 values for this project it is quite good to use this crossover because it is very likely that the EA will be able to increase exploitation of the individual in the hope it finds a global optima. Overall uniform crossover is good as it allows for a good mix of both parent’s genes will be in the child and if the parents had good fitness scores it means the child can hopefully be exploited even more to find an even better fitness. This could hopefully find an individual whose fitness is a very good time with minimal or no collisions.

* Mutation = swap

A swap mutation would be a good mutation operator for this problem because it will take 2 genes in the child’s chromosome and swaps their places in the chromosome. Multiple swaps can be done depending on the amount you want to mutate the child’s chromosome. Using this mutation operator would allow for more exploration of the given search space as we are creating new individuals that may have not existed otherwise. This mutation is good for the problem because there are only 3 values, so hopefully by swapping genes it could make a big different to the overall fitness of the car. Overall, a swap mutation allows for the swapping of genes in a chromosome to help with exploration. This intern can help with finding a global optima which in this case would be a fast time with minimal or hopefully no collisions.

2d.

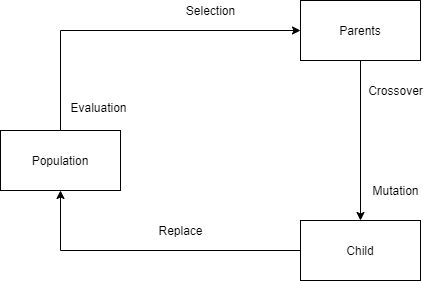
Decision Trees could be used to control the car instead of an EA. Decision trees are a means of representing a decision-making process as a tree. The tree would learn by adding in new or editing current branches based on decisions it has previously made in another iteration. The tree over time would grow and eventually would have a set of instructions to do at each time stamp on the course. To achieve this the simulator would need to be changed a bit as the position of the car would need to be tracked on the course. This is so the car knows when to use its 4 actions more accurately than it currently does. Another change that would need to be implemented would be how the chromosome genes values are saved. For instance, rather are we turning left, it would be how much are we turning left (degrees). This would allow for more precise movements of the car on the track which should result in the car crashing less and therefore getting a better fitness. It would also help with the decision making for the car as it then could accurately know how much it should be turning for each corning etc.

2e.

Decision Trees in this case would end up with a complete tree with branches telling the car what to do at each timestamp and therefore will get a good score. An EA would also eventually be able to complete the track, but you would not end up with a set of instructions like a decision tree would output to you. This allows you to perhaps change the car and give it the same instruction set as the previous and it should also be able to complete the track whereas an EA would have to relearn the track with this new car. A decision tree is also relatively easy to implement compared to an EA which can have multiple different parameters and operators which can influence the outcome of the final project. However, the draw backs of decision trees are that if the track was changed for instance then the current decision tree would be useless and it would have to add/remove parts of the tree to make it fit the new track. This could ultimately be slower than just using an EA to learn the track as it would have no prior knowledge to the previous track. Another problem with decision trees is that they are unstable, meaning that a small change in the data can lead to large change in the structure of the optimal decision tree. They are often quite inaccurate. Compared to other predictors such as EA which may perform better with similar data. Another problem could be that if the track is quite long and has a lot of decision to be made, it could be quite a long decision tree. This could slow the performance of the car compared to something like an EA uses chromosomes to hold its data which is far smaller.

4a.

* Initialisation – create population
* Evaluation
* Selection
* Crossover
* Mutation
* Replace



The diagram showing the cycle of evolution

The first stage of evolutionary cycle is the Initialisation of the population by either creating random chromosomes are having a default chromosome that the initialisation uses to create new individuals.

The next stage is the evaluation of those new individuals, of course if the a default chromosome has been used then they will all get the same fitness however, this is a necessary step to get all the fitness’s for these individuals for the rest of the evolution.

The next step is the selection of parents which will be used to create new child individuals. These parents will normally be picked using a selection method such as a tournament selection which hosts mini tournaments for all individuals and the ones with the best fitness will become the parents (survival of the fittest).

Once the parents have been chosen then a crossover operator is done on both parents to pick parts from each parent that may or may not end up in the child. An example of this crossover would be a one-point crossover which cuts the parents chromosomes at a certain point and takes a half from each parent to be combined and stuck together to create the child’s chromosome. This increases exploitation of individuals because no new genes are being created as they are being taken from the parents, this can help with getting better fitness scores of the individuals have good chromosomes.

The next step is the mutation where the child’s chromosome that was created from the crossover is mutated using a mutation method. This allows for exploration of the search space as some of the chromosome’s genes are mutated. This is so the child chromosome has some traits from its parents via the crossover but also gains new genes and traits from the mutation. An example of mutation method is a creep mutation which will randomly increase or decrease a genes value in the chromosome by x amount. This can have a good or bad affect depending on the range of values in search space that are considered good. However, it does allow for new genes to develop.

Now that the child chromosome has been made the final step in evolutionary process is replacing an individual in the current population with our new child. The replace function can be set up to be random or in most cases will replace the individual with the worst fitness with our new child. This ensures the population size stays the same throughout the process and now allows for the new individual to be exploited by the algorithm and is now able become a parent for the next generation.

Rinse and repeat there after

4b

* Simulated Annealing
* Increase Population
* Decrease selection pressure adding in bigger tournaments
* Diversity sawtooth, island exploring the search space
* Increasing mutation rate

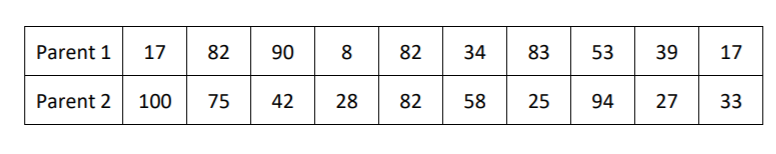
Simulated Annealing could be added to prevent the EA getting stuck in local optima because it allows the algorithm to go back down a hill if it has reached a local optimum. This means the EA can hopefully find another path that could lead the global optima in the search space or if it does lead to another local optima it can then be allowed to go back down again and restart etc.

Increasing the population would also decrease the chance of getting stuck at a local optima because there is more individuals that are able to be exploited and therefore less change of getting stuck. By doing this the EA has a better chance of finding an area in the search space that could lead to the global optima and less likely to find local optimas

Decreasing the amount of selection pressure would also help to avoid the EA getting stuck at a false optima. This could be the case in terms of having bigger tournaments for the tournament selection. This would allow for more exploration and less exploitation of individuals which would intern allow the EA to hopefully avoid getting stuck as it would not be focusing too much on certain individuals to only get stuck.

Some sort of diversity could be implemented which would help with the amount of exploration of the search space done. This could be in the form of a saw tooth diversity which will remove the worst individual from the population every x iterations until the population is half of what it was. Then x number of brand-new initialised individuals would be added to the population and therefore giving the EA to perhaps try new things with those individuals and hopefully get a better fitness score. The island model could also be used which sees groups of individuals on separate islands evolved and there after x number of iterations some are moved onto another island to hopefully evolve using some of the traits on said island. Overall, hoping to increase exploration and getting stuck on false opitmas.

The final way of preventing getting stuck false optimas is to increase the mutation rate as this increases exploration of the search space. By doing this you’re lowering exploitation of individuals and therefore giving the EA a better chance of finding individuals that lead to a global optima rather an individual that will lead to a local optima. The mutation rate is a value that will dictate how much mutation should occur or if mutation should occur at this stage in the process.

4c.

2 Points will be picked to split the parent’s chromosomes (the blue lines)

Children

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Child 1 | 17 | 82 | 42 | 28 | 82 | 58 | 25 | 53 | 39 | 17 |
| Child 2 | 100 | 75 | 90 | 8 | 82 | 34 | 83 | 94 | 27 | 33 |

5ai.

Left branch = 9

Right branch = 8

Max Output = 9

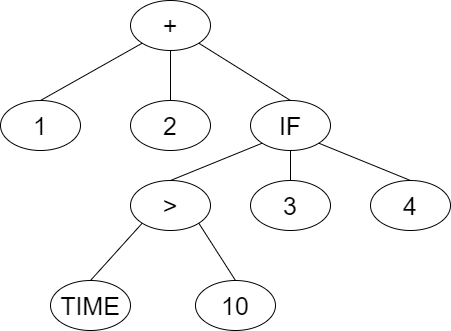
ii.

Left branch = 9

Right Branch = 10

Max Output = 10

b.



Description: The function nodes for this diagram are Greater Than, IF and Add. The Add node is to the ‘answer = 1 + 2;’ at the beginning of the code. This is on the left branch as it is done first as the answer will also equal ‘answer= 1 + 2’. The IF statement is to look to see if the time is Greater Than ‘>’ 10. As time and 10 are the only 2 branches of the Greater Than they are equated together. Then hence is on the left-hand side of the branch of the if statement because it is read in a breadth first search manner. If the time is Greater Than 10 then because this is an if statement, the next branch is taken i.e. 3 in this case. However, if it returned false then the next branch would be skipped and the one after that would be picked i.e. 4 in this case. Then either 3 or 4 is added to the ‘answer’ depending on the outcome of the if statement.

ci.

* Terminal nodes are nodes that does not have any branches
* The terminal nodes Constant, Velocity and Position
* Function Nodes are Add, Subtract, Multiply, Absolute Value, Greater Than, Divide
* Root greater than

Terminal nodes are nodes that do not have any branches attached to them and for this example of the cart would be the Constant, Velocity and Position variables. The functional nodes for the cart would be, Add, Subtract, Multiply, Absolute Value, Greater Than and Divide. These operators are required to be able to calculate the carts terminal nodes values. The add and multiply operator is used for adding and multiplying the values of ‘thrust \* timeStep’ and ‘velocity \* timestep’ to the velocity and position, respectively. The add and subtract operator is working out the change in velocity from different positions. The absolute value and greater than operator is used to make sure the absolute velocity and position of the cart is greater than 0.01 respectively. The divide operator is used to divide by negative -1 if the change in acceleration is negative as that means the cart is going backwards, this just makes it a positive acceleration for the calculation. However, it is possible for the evolution to result in a division by 0 which is of course not possible and would result in an error. To resolve this a ‘protected divide operator’ should be used. The root node for the cart should be the ‘>’ (Greater Than) because we want the expression to be true, so the program will run otherwise the evolution will not run. However, if the expression = false then the position of the cart should be x = 0 which is what we want.

cii.

Start Cart at random Position

Measure time to centre

Repeat above x times and then take the sum of the time taken for each case \*

Fitness = total time (which we try to minimise)

\*if cart does not centre after a fixed time, add a large value