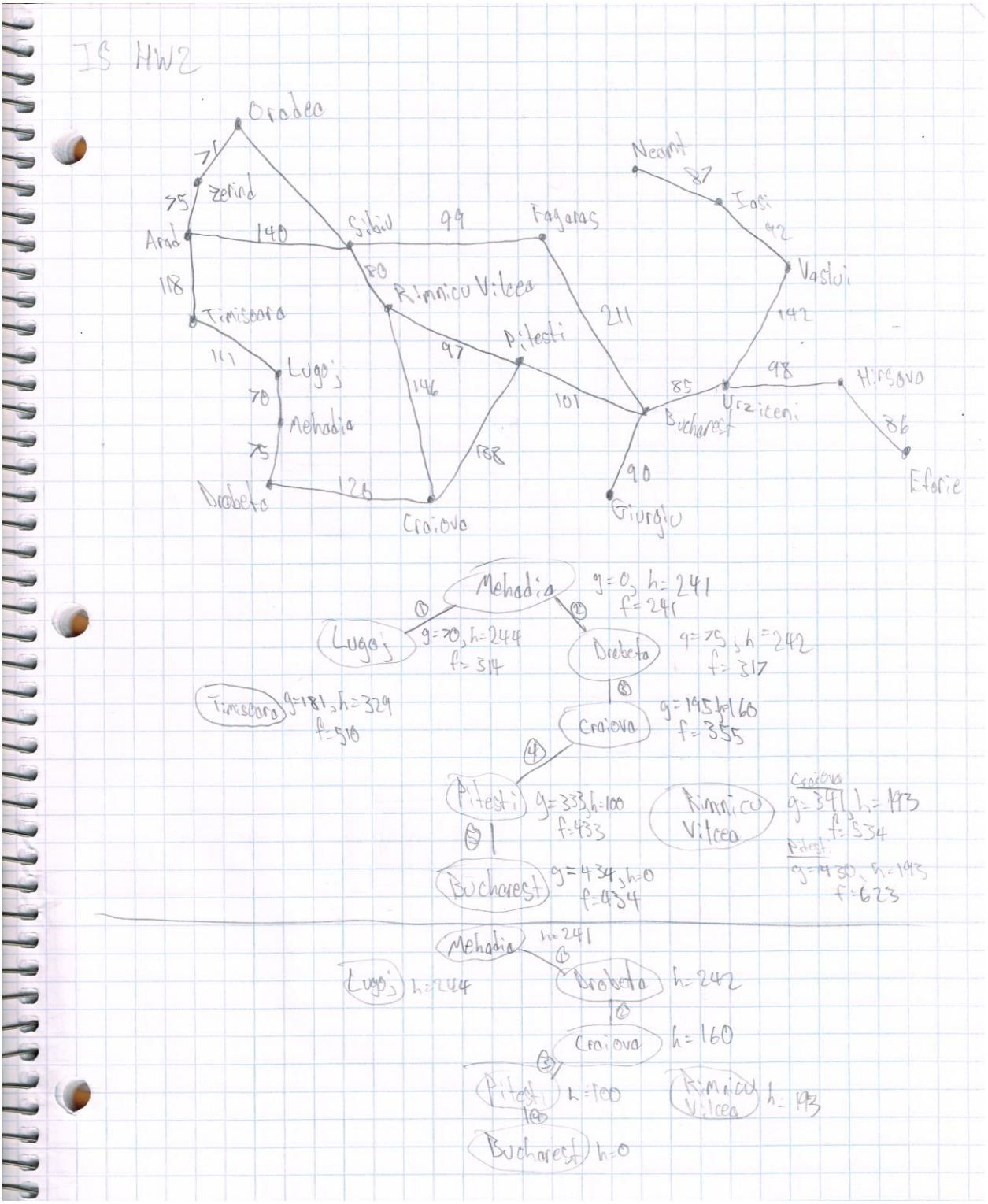
Kevin Dombrosky 9/26/16

Homework 2 Intelligent Systems

1&2)



The same final result is ultimately returned. The A* algorithm takes an extra step looking at Lugoj, but otherwise produces the same path to Bucharest from Mehadia. This difference is simply luck based on the structure of the graph. Greedy best-first search should not be expected to always return the optimal path like A*.

3) R&N 4.2

To implement simulated annealing for this problem, the correctness of the state of the tracks must be able to be measured, and as a result how better or worse moves must be defined. The initial state should include what pieces are connected to which other pieces, and how many degrees they are offset from each other. Not all pieces should be used with this initial state to allow for more creativity with our random moves earlier on in the function. To determine the correctness or how close the state is to being the goal, we should consider the circular degrees of separation of each loose end to each other loose end of the opposite ending type, taking into consideration the number of remaining pieces. With this perspective, we can know that each curved piece can represent 35-55 degrees of a circle, while the straight pieces can represent up to 10 degrees of a circle for a completed track. With this methodology, a curved piece that is placed that turns the loose end it is on away from the other loose ends, will cause a lower scoring state that should not be taken if further on in the search with fewer pieces remaining. However, this move can still be progressed if there are enough pieces left that the move could be corrected. Changes in angle of a piece should be weighted lower than placing a new piece, as the change in angle would likely be easier to correct.

In this case, a forward move is either placing a piece or adjusting the angle of a current piece, while removing a piece should count as a backwards move. Any move is represented by the piece added or removed, and if added/moved, what orientation and angle the piece has with respect to its “parent” piece.

We would likely want the search to continue until any solution is found. The “cooling” function should probably allow for more backwards moves towards the beginning to middle of the search, cooling down more as the correctness seems to improve significantly.

4) Recipe – Chocolate chip cookies

The state space should be represented as floating point or integer numbers for each of the ingredients (flour, baking soda, salt, butter, granulated sugar, brown sugar, vanilla extract, eggs, chocolate chip morsels, and nuts), depending on domain/unit of each ingredient. Chocolate chip morsels should be a floating point of cups, while eggs should be an integer for number of eggs, and so on. Additionally, baking and cooling times in minutes can be included. The fitness function should put a high priority on the higher quantity or sought after ingredients (flour, chocolate chips), but a lower emphasis on the lower quantity ingredients (salt, vanilla, baking soda). Additionally, the values for each quantity represented for the state space should be bounded when the initial random states are made. However, things like a higher vanilla extract use with a lower chocolate chip count could be delicious. There could even be a distinction on what ingredients could be considered a “star” ingredient that could allow for a significant difference in outcome for the final product, but ultimately not be that similar to a chocolate chip cookie.

Ultimately taking the genetic algorithm approach can be subjective to whomever is writing the fitness function. Taking the hill climbing approach of starting with all ingredients at a random value and gradually increasing them could have other consequences. While a “goodness” function would likely rate the use of chocolate chips highly, it could end up with a high chocolate chip count for a single batch of cookies. This would essentially be a local maximum that it could not escape from. Alternatively, the hill climbing function could begin with a particular ingredient as too high, and never be able to escape the local maximum that way. In this sense, the genetic algorithm is more likely to be correct in the long run since it can recover from going too high. The hill climb algorithm has the struggle of needing to track several maximum or several “good conditions” that can result in the algorithm outputting a bad state.

5) R&N 6.4b

Variables – Classes, Professors, Timeslots

Domains – Professors teach specific classes, classes are assigned timeslots

Constraints – If two classes require the same professor to teach them, they cannot overlap time slots

R&N 6.4c

Variables – Cities, distances between cities

Domains – Ordering of cities to visit, likely accompanied by a distance

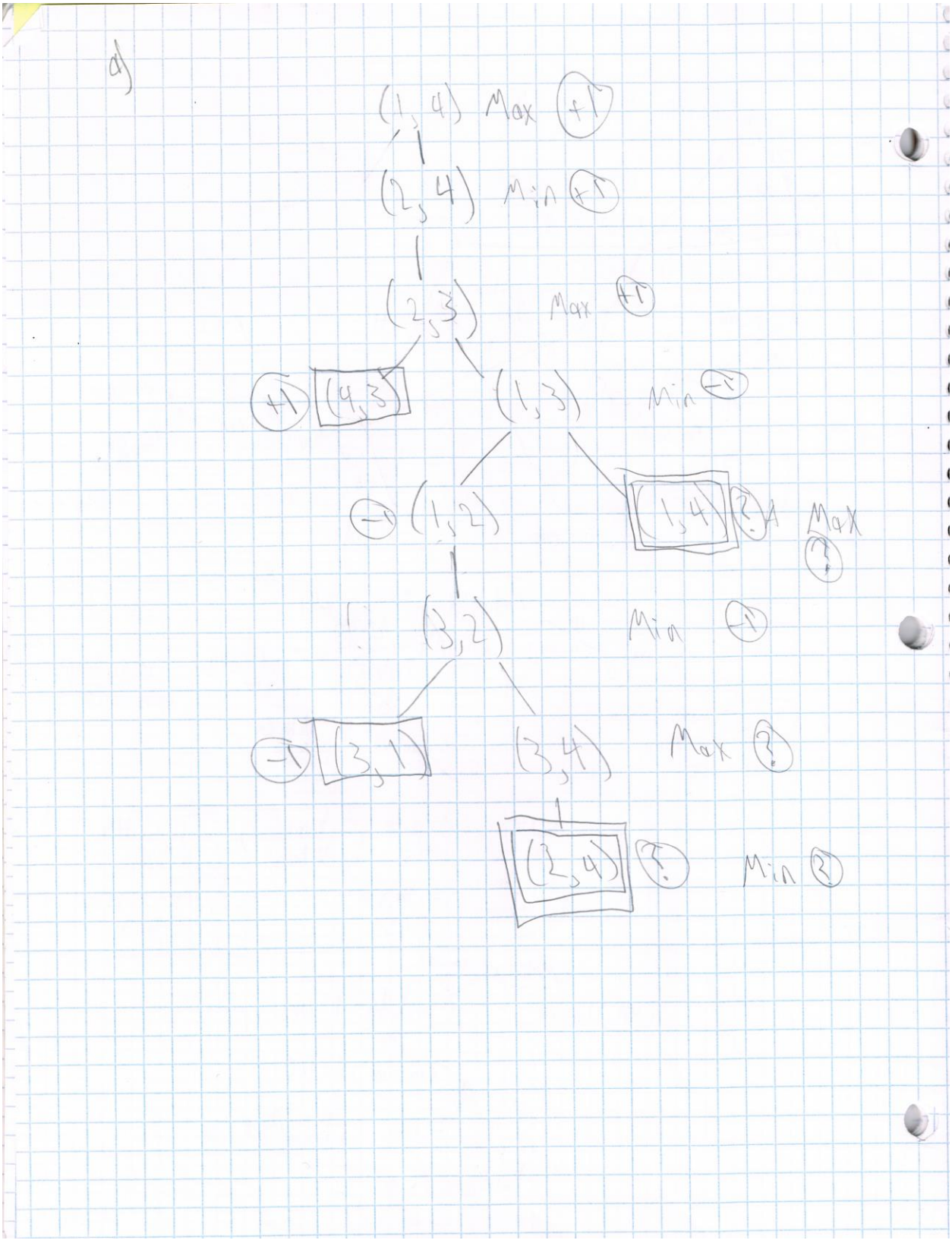
Constraints – No one city can appear in the current domain of values more than once.

6) R&N 6.4b genetic algorithm

Genotype – Have a string of integers where the index represents a potential timeslot in increasing order, and the integer at the index the professor teaching the class at that timeslot

Fitness function – At it’s simplest, the fitness function should check that the professors are not teaching more classes than they are assigned to teach. Next, the fitness function should verify to what degree a professor’s classes’ timeslots overlap. More overlap indicates a lower score in the fitness function. Crossover will most certainly create invalid individuals. It can be very easy for the GA to produce a state where a professor teaches most of the morning and afternoon and teaching more classes than he should. The fitness function is then used to determine that states such as these should not be pursued.

7) R&N 5.8 a-c


a)
(1,4) Max +1
(2,4) Min +1
(2,3) Max +1
+1
(4,3)
(1,3) Min -1
-1
(1,2)
(1,4)
Max
?
(3,2)
Min -1
-1
(3,1)
(3,4) Max ?
(3,4)
?
Min ?