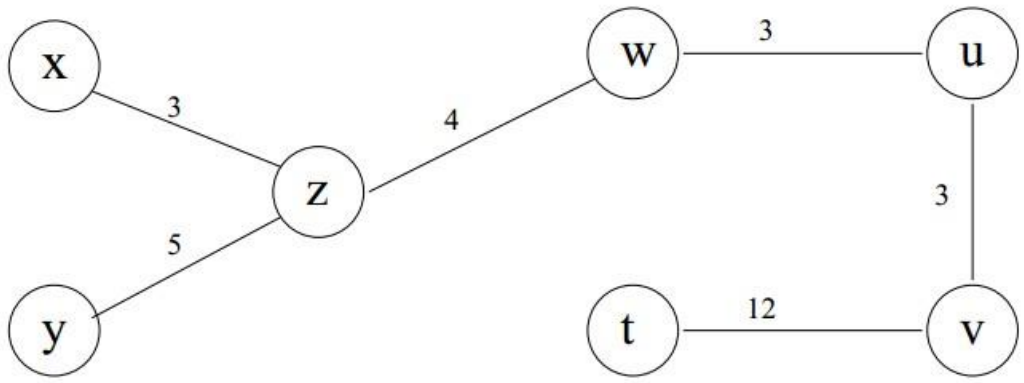
CS1571 – Introduction to Artificial Intelligence

Assignment 3

Name: You Zhou (yoz13@pitt.edu)

People Soft Number: 3729901

Problem 1:



Forward Checking	t	u	v	w	x	y	z
Initial State Analysis	0	{0,3,6,9}	0 The most obvious choice	{0,3,6,9} mod 4 {0,1,2,3}	2	0	5 The second most obvious one
Now, pretending we didn't know the information given by the row above …							
Put 0 to v	0		0		2	0	
Put 0 to u	0	0	0		2	0	
To show how Forward Checking works, I will choose the wrong one for w.							
Put 0 to w	0	0	0	0	2	0	
Here comes a problem, given the constraints acting on z, z has no choice!							
Backtracking – Re-value w							
Put 9 to w	0	0	0	9	2	0	
Put 5 to z	0	0	0	9	2	0	5
Goal Condition Achieved!							

Arc Consistency	t	U	v	w	x	y	z
Put 0 to v	0		0		2	0	
Put 0 to u	0	0	0		2	0	
Pretending the former steps are the same …							
The Arc Consistency algorithm will check bi-directionally.							
Basing on restraints acting on z, z only has one choice 5.							
z = 5 & w = 0 inconsistent							
z = 5 & w = 3 inconsistent							
z = 5 & w = 6 inconsistent							
z = 5 & w = 9 consistent!							
Put 9 to w	0	0	0	9	2	0	
Put 5 to z	0	0	0	9	2	0	5
Goal Condition Achieved							

Problem 2:

Part (a):

Initial Energy	Initial Temperature	Try #	Accept #	Best Tour Energy
302.129	100.000	100000	92383	92.976

Part (b):

Initial Energy	Initial Temperature	Try #	Accept #	Best Tour Energy
286.932	10.000	10000000	6001875	**63.677**

Part (c):

Geometric Cooling Scheme: $T_{i+1} = 0.85T_i \quad (T_0 = T_{init})$

Min{Average} = 68.723 Max{Average} = 70.089

It's quite simple comparing with other classmates' possible "fancy" and twisted functions; however, the geometric cooling scheme with coefficient between 0.85 and 0.99 inclusive are proved to be most optimal for TSP. As temperature converges to 0 faster, the tolerance for higher energy tour becomes less, and this strictness is crucial for only 20000-step simulation! For small number of steps, too much tolerance for potentially optimal configurations would face a danger – it accepts a configuration with higher energy, hoping this configuration would evolve into a more optimal one, but, due to the small number of steps, this configuration might not have a chance to fully evolve!

Part (d)

Table for varying number of generations:

As the number of generations goes larger, the optimal energy decreases in a linear manner, this is because, the *next_gen()* function can combine and evaluate more crossovers, thereby increasing the chance of producing optimal candidate.

	Least Energy
n = 50	127.201
n = 100	99.820
n = 150	96.569
n = 200	92.715
n = 250	86.515
n = 300	82.213
n = 350	79.821
n = 400	81.223
n = 450	82.114
n = 500	78.893

Table for varying population size:

As the size of population goes larger, the optimal energy decreases in a linear manner, this is because, with more potential candidates to be chosen, it's less likely to miss optimal configurations.

	Least Energy
p = 50	144.965
p = 100	102.832
p = 150	93.620
p = 200	83.466
p = 250	80.990
p = 300	82.451
p = 350	78.946
p = 400	79.109
p = 450	76.977
p = 500	73.674

Table for varying mutation probability:

With more chance of mutating existing configurations, the population gains more randomness, avoiding having too many similar candidates in population. The energy drops steeply from zero chance to 5% chance, but it decreases rather slowly thereafter.

	Least Energy
m = 0.00	227.137
m = 0.05	74.075
m = 0.10	71.284
m = 0.15	69.529
m = 0.20	68.702
m = 0.25	68.780

Table for varying survival rate:

It’s quite unexpected that, with higher survival rate, energy drops, since I would expect that more crossovers and permutations would lead to optimality. However, we should remember that a more optimal configuration is **NOT** guaranteed.

	Least Energy
r = 0.0	81.079
r = 0.2	76.554
r = 0.4	79.297
r = 0.6	81.852
r = 0.8	80.291
r = 1.0	78.564