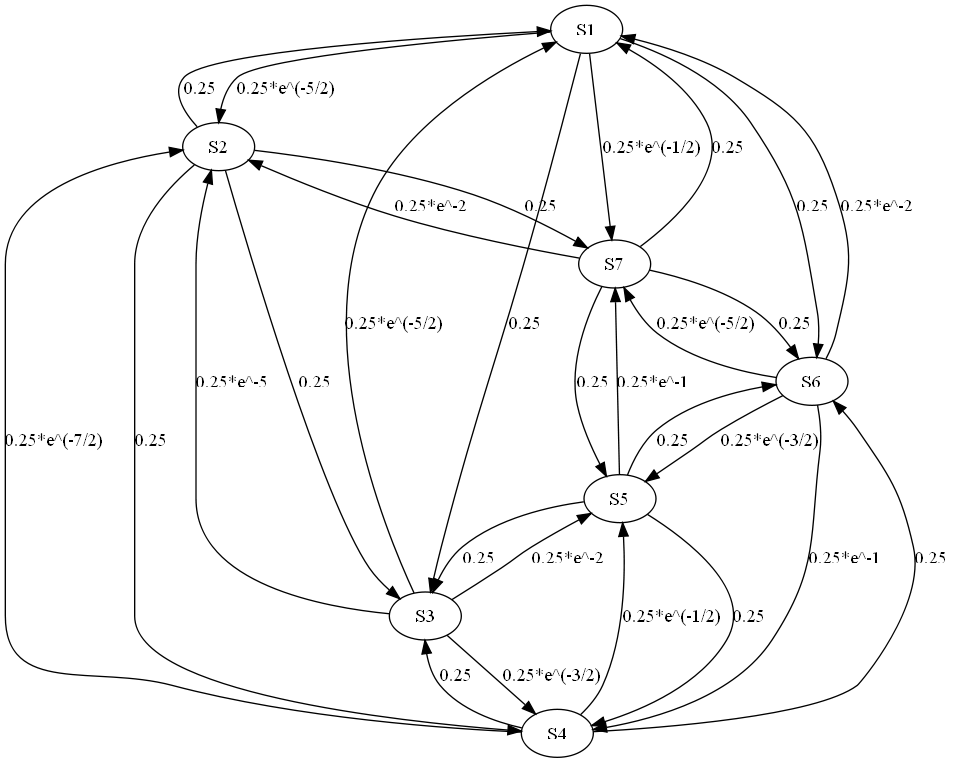
2

For second problem, we wrote a program that calculates the transition probability, and generated the configuration graphs. The program is attached at the end of the assignment.

Part a)

The configuration graph looks as follows:



From the graph, we can extract the transition matrix, , as:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0.3278 | 0.0205 | 0.2500 | 0.0000 | 0.0000 | 0.2500 | 0.1516 |
| 0.2500 | 0.0000 | 0.2500 | 0.2500 | 0.0000 | 0.0000 | 0.2500 |
| 0.0205 | 0.0017 | 0.8882 | 0.0558 | 0.0338 | 0.0000 | 0.0000 |
| 0.0000 | 0.0075 | 0.2500 | 0.3408 | 0.1516 | 0.2500 | 0.0000 |
| 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.1580 | 0.2500 | 0.0920 |
| 0.0338 | 0.0000 | 0.0000 | 0.0920 | 0.0558 | 0.7979 | 0.0205 |
| 0.2500 | 0.0338 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2162 |

Part b)

If T stays at 2, from the matrix, we would guess that will have a big peak value at 3rd component, and a small peak value at 6th component. As we can see the probability of staying is pretty high, and the probability of coming into this node is also pretty high. Without a detailed calculation, it is hard to calculate the exact value. (From Matlab, the = [0.0063, 0.0000, 0.9373, 0.0467, 0.0172, 0.3448, 0.0023])

Part c)

We calculate that as follows:

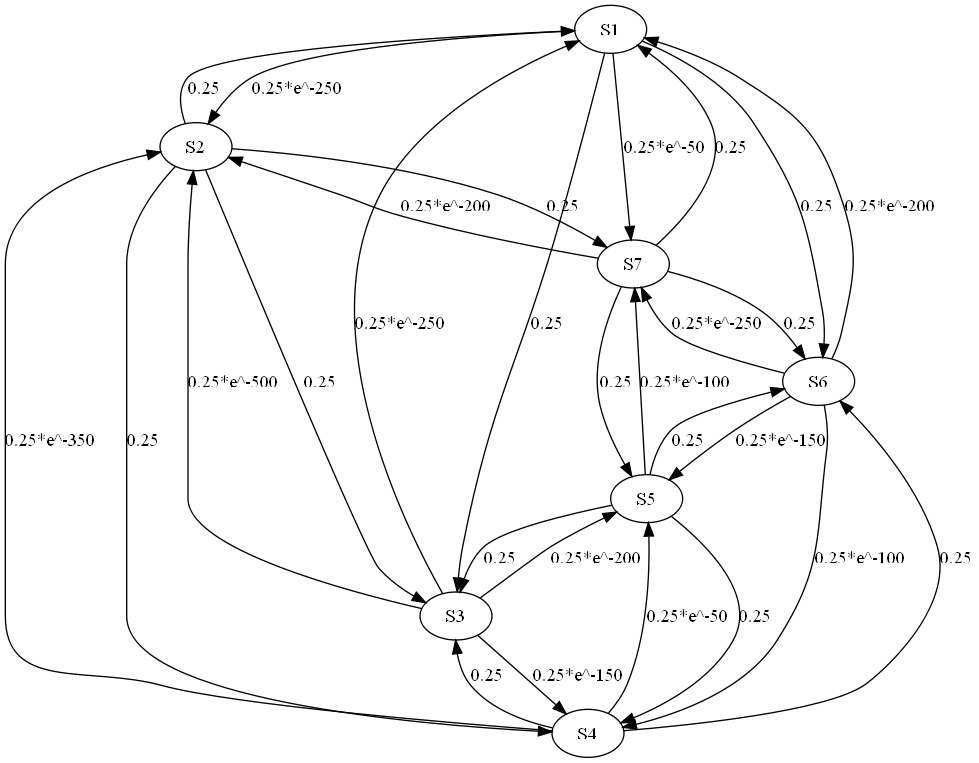
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0.0390 | 0.0032 | 0.4754 | 0.1061 | 0.0643 | 0.2883 | 0.0237 |
| 0.0390 | 0.0032 | 0.4754 | 0.1061 | 0.0643 | 0.2883 | 0.0237 |
| 0.0390 | 0.0032 | 0.4754 | 0.1061 | 0.0643 | 0.2883 | 0.0237 |
| 0.0390 | 0.0032 | 0.4754 | 0.1061 | 0.0643 | 0.2883 | 0.0237 |
| 0.0390 | 0.0032 | 0.4754 | 0.1061 | 0.0643 | 0.2883 | 0.0237 |
| 0.0390 | 0.0032 | 0.4754 | 0.1061 | 0.0643 | 0.2883 | 0.0237 |
| 0.0390 | 0.0032 | 0.4754 | 0.1061 | 0.0643 | 0.2883 | 0.0237 |

If started at S3, i.e., if T = [0, 0, 1, 0, 0, 0, 0]. We can calculate the probability of ending up in any state after 100 iteration as =[0.390, 0.0032, 0.4754, 0.1061, 0.0643, 0.2883, 0.0237]. Then, the probability of ending up in global minimum, S3, is simply 0.4754.

Similarly, if started at S6, i.e., if T = [0, 0, 0, 0, 0, 1, 0]. We can We can calculate the probability of ending up in any state after 100 iteration as =[0.390, 0.0032, 0.4754, 0.1061, 0.0643, 0.2883, 0.0237]. Then, the probability of ending up in global minimum, S3, is simply 0.4754.

Part d)

The configuration graph looks as follows:



From the graph, we can extract the transition matrix, , as:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0.5000 | 0.0000 | 0.2500 | 0.0000 | 0.0000 | 0.2500 | 0.0000 |
| 0.2500 | 0.0000 | 0.2500 | 0.2500 | 0.0000 | 0.0000 | 0.2500 |
| 0.0000 | 0.0000 | 1.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.0000 | 0.0000 | 0.2500 | 0.5000 | 0.0000 | 0.2500 | 0.0000 |
| 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.0000 |
| 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 1.0000 | 0.0000 |
| 0.2500 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |

We can also calculate that as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0.0000 | 0.0000 | 0.5000 | 0.0000 | 0.0000 | 0.5000 | 0.0000 |
| 0.0000 | 0.0000 | 0.5833 | 0.0000 | 0.0000 | 0.4167 | 0.0000 |
| 0.0000 | 0.0000 | 1.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.0000 | 0.0000 | 0.5000 | 0.0000 | 0.0000 | 0.5000 | 0.0000 |
| 0.0000 | 0.0000 | 0.5000 | 0.0000 | 0.0000 | 0.5000 | 0.0000 |
| 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 1.0000 | 0.0000 |
| 0.0000 | 0.0000 | 0.3333 | 0.0000 | 0.0000 | 0.6667 | 0.0000 |

If started at S3, i.e., if T = [0, 0, 1, 0, 0, 0, 0]. We can calculate the probability of ending up in any state after 100 iteration as =[0, 0, 1, 0, 0, 0, 0]. Then, the probability of ending up in global minimum, S3, is just 1.

Similarly, if started at S6, i.e., if T = [0, 0, 0, 0, 0, 1, 0]. We can We can calculate the probability of ending up in any state after 100 iteration as =[0, 0, 0, 0, 0, 1, 0]. Then, the probability of ending up in global minimum, S3, is just 0.

Part e)

We notice that for T=2, all the rows are the same, where for T=0.02, each row is different. This means that the result of algorithm after 100th iteration for T=0.02 is very different for different initial conditions, where for T=2, it does not depend on initial condition at all. For example, in T=0.02 case, if we started at S3, then the probability of staying at S3 after 100 iteration is 1. On the other hand, if we start at another local minimum S6, the probability of reaching global minimum is 0. This is not the case for T=2, where each initial condition in T=2 case leads to same probability of getting global minimum.

We think the algorithm with T=2 performs better, since it enables us to find global optimal for all initial conditions with reasonably high probability. The algorithm with T=0.02 does not perform so well, as the temperature is so low that this SA behaves more like a greedy search.

Part f)

If we need to suggest a value for higher probability of being global optimal solution at 100th iteration, I would suggest a value between 2 and 0.02. (For example, T = 1 gives a probability of staying at global minimum after 100th iteration as >0.65 for all initial conditions). If I need to choose between greater than 2 or smaller than 0.02, I would choose greater than 2, since for small T, the probability of uphill move is too low, and therefore prevents some initial condition to reach global optimal point.

Program used to generate the input:

|  |
| --- |
| import sys;  import math;  def gcd(a, b):  if (a > b):  return gcd(b, a)  else:  if (a == 0):  return b  return gcd(b % a, a)  result = range(0,7)  def main():  s = [0, 15, 20, 10, 13, 14, 11, 16]  T = 2  for x in range(0, 7):  result[x] = range(0, 7)  for t in range(0,7):  for t2 in range(0,7):  result[t][t2] = "0"  for t in range(1, 8):  count = 0  for diff in range(-2, 3):  if diff != 0:  count += 1  for diff in range(-2, 3):  if diff != 0:  nxt\_ind = ((t + diff - 1) % 7) + 1  if s[nxt\_ind] - s[t] > 0:  gg = gcd(100\*(s[nxt\_ind] - s[t]), T)  ss = ""  if T/gg != 1:  ss = ("(-%d/%d)" % (100\*(s[nxt\_ind] - s[t])/gg, T/gg))  else:  ss = ("-%d" % (100\*(s[nxt\_ind]-s[t])/gg))  result[t-1][nxt\_ind-1] = ("0.25\*e^%s" % ss)  sys.stdout.write('S%d -> S%d [ label = "0.25\*e^%s" ];\n' % (t, nxt\_ind, ss))  else:  result[t-1][nxt\_ind-1] = "0.25"  sys.stdout.write('S%d -> S%d [ label = "0.25" ];\n' % (t, nxt\_ind))  sys.stdout.write("MATRIX: \n")  for t in range(0,7):  for t2 in range(0,7):  sys.stdout.write("%s " % result[t][t2])  sys.stdout.write("\n")  main() |

Sample generated graph: (in dot format)

|  |
| --- |
| digraph temp {  S1;S2;S3;S4;S5;S6;S7  S1 -> S6 [ label = "0.25" ];  S1 -> S7 [ label = "0.25\*e^-50" ];  S1 -> S2 [ label = "0.25\*e^-250" ];  S1 -> S3 [ label = "0.25" ];  S2 -> S7 [ label = "0.25" ];  S2 -> S1 [ label = "0.25" ];  S2 -> S3 [ label = "0.25" ];  S2 -> S4 [ label = "0.25" ];  S3 -> S1 [ label = "0.25\*e^-250" ];  S3 -> S2 [ label = "0.25\*e^-500" ];  S3 -> S4 [ label = "0.25\*e^-150" ];  S3 -> S5 [ label = "0.25\*e^-200" ];  S4 -> S2 [ label = "0.25\*e^-350" ];  S4 -> S3 [ label = "0.25" ];  S4 -> S5 [ label = "0.25\*e^-50" ];  S4 -> S6 [ label = "0.25" ];  S5 -> S3 [ label = "0.25" ];  S5 -> S4 [ label = "0.25" ];  S5 -> S6 [ label = "0.25" ];  S5 -> S7 [ label = "0.25\*e^-100" ];  S6 -> S4 [ label = "0.25\*e^-100" ];  S6 -> S5 [ label = "0.25\*e^-150" ];  S6 -> S7 [ label = "0.25\*e^-250" ];  S6 -> S1 [ label = "0.25\*e^-200" ];  S7 -> S5 [ label = "0.25" ];  S7 -> S6 [ label = "0.25" ];  S7 -> S1 [ label = "0.25" ];  S7 -> S2 [ label = "0.25\*e^-200" ];  } |