Homework 7 Advanced Analytics and Metaheuristics

Group 20: Nicholas Jacob

April 15, 2024

1. Simulated Annealing

Initial Temperature Looking at our original knapsack problem, we see that the best increase we could expect in the value is 1000. We test with that value as our initial temperature extensively. After tweaking this some we recognized that a higher initial temperature gave some nice results. Temperature tweaking is really what this part of the assignment was all about so doing so felt natural.

Cooling We use both an exponential and Cauchy cooling scheme. Exponential cooling was simply:

```
T = 0.99*T
```

While the Cauchy cooling required the knowledge of how many temperatures we had used (gotten from the for loop) and the initial temp (reserved as it's own variable, T_0).

```
T = T0/(1+j)
```

Probabilities For the probabilities, we did a random selection of the neighbor using the index. We then assigned a probability using the following code:

```
def probability_assignment(x,T):
  if x>0:
    return 1
  else:
    return np.exp(x/T)
```

We compared this to a random value between 0 and 1. We see that if the x (which was the difference of the neighbor and the current) was positive, we accepted that as the new value. If not, there was a chance that we would take the other value.

Stopping Criteria We first attempted to just run through a total number of iterations with a for loop, just allowing it to continue until it exhausted all possibilities in the for loop. This had a few draw backs: while simple, it could get stuck and take a long time. It could also hit a piece of the logic and not find an acceptable new solution. I quickly made an edit to the annealing code, if it could not find a suitable neighbor in 150 tries (length of the neighbors), I exited the loop looking for an acceptable neighbor. It would then return to that loop and again attempt to find a suitable neighbor. Since there are probabilities involved, perhaps it would not find a neighbor to move to. Soon I added another chance to break the loops. If the code failed to find an acceptable neighbor after so many iterations, I just wanted to break the outside loop and return what it had. Again I just accomplished this with break rather than coding it into whiles.

2. Genetic Algorithm

createChromosome()

Method

SA	T_0	Cooling, t_k	M_k	# of temps	Iterations	Items	Weight	Value
	100	$0.99t_{k-1}$	50	1000	126710	39	2487.2	23405
	1000	$0.99t_{k-1}$	50	1000	236504	43	2471	24456.6
	1500	$0.99t_{k-1}$	25	500	625724	44	2499.8	24898.2
	2500	$0.99t_{k-1}$	50	1000	249256	43	2481.0	24747.5
	1000	$\frac{T_0}{1+k}$	50	1000	170655	41	2494.5	23247
	2000	$\frac{T_0}{1+k}$	100	1000	106265	45	2498.2	22999.4
	2000	$\frac{T_0}{1+k}$	100	1000	231663	45	2496.1	24459.5
	3000	$\frac{T_0}{1+k}$	100	500	244890	42	2498.3	24147.5