### Writeup Part

Part a) Plot for cost function



Part f) 1: Plot of scurrent, sbest vs. iterations for one run



Part f) 2: plot of the average of COSTcurrent, average of COSTbest vs. iterations



Part f) 3: Plot that compares the averages of COSTbest vs. iterations for all four algorithms.



Part f) 4: Average and standard deviation of COSTbest after 100 iterations for each algorithm: (Print out from Matlab program)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Average & Std Dev of CostBest %%%

% Algorithm average std dev %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Rand Walk -119107.38 82046.60 %

% Rand Sample -198092.37 19362.94 %

% Greedy Det -87538.48 78828.66 %

% Greedy Stoc -140696.26 81522.65 %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

It is very rare for the algorithms to find the global minimum because of the randomness in the algorithm (however, if you increase the neighbor size for GD, it will find the global min in only a few iterations). Comparably, RS has a higher chance of finding the global min than other algorithms.

From the plot, the performance of the algorithms, from best to worst respectively, is RS, GS, RW, GD. RW and GS are very similar until the midpoint where RW is very slow to find better solution, whereas GS finds better solutions faster. GD finds the best solution it could achieve in the fastest time, but it was not able to find an answer close to the global min because the neighborhood is too small and the GD algorithm gets stuck in a local min. However, if you change the neighborhood size to 25 like the rest of the neighbor functions, GD would deterministically achieve an answer very close to the global min within the lowest iterations.