Code Part

### Tabu.m

function [solution, scurrent] = Tabu(tabuLen, sinitial, alInitial, iter)

tabu = zeros(tabuLen, 1);

al = alInitial;

s = sinitial;

tabuInd = 1;

BIGNUM = 100000;

solution = zeros(iter, 3);

scurrent = zeros(iter, length(sinitial));

costs = zeros(1, length(sinitial));

for i=1:iter

neighbors = neighborhood(s);

for j=1:size(neighbors,1)

costs(j) = costSAT(neighbors(j,:));

end

[val, index] = min(costs);

s = neighbors(index,:);

if (sum(tabu==index) == 0)

%disp 'not tabued'

% Not found

tabu(tabuInd) = index;

tabuInd = tabuInd + 1;

if tabuInd > tabuLen

tabuInd = 1;

end

% update al

if (val < al)

al = val;

end

else

if (val < al)

%disp 'tabued, improved'

tabu(tabuInd) = index;

tabuInd = tabuInd + 1;

if tabuInd > tabuLen

tabuInd = 1;

end

% update al

al = val;

else

%disp 'tabued, not improved'

while (sum(tabu==index) > 0)

costs(index) = BIGNUM;

[val, index] = min(costs);

if (val == BIGNUM)

break;

end

end

if (val == BIGNUM)

fprintf('Something is messed up in iteration %d... No potential candidate found. Are you using tabu table length greater than number of variables?\n', i);

else

s = neighbors(index,:);

tabu(tabuInd) = index;

tabuInd = tabuInd + 1;

if tabuInd > tabuLen

tabuInd = 1;

end

end

end

end

solution(i, 1) = i;

solution(i, 2) = val;

solution(i, 3) = al;

scurrent(i, :) = s;

end

### costSAT.m(Calculate cost. Notice our cost is defined to be (# of unsatisfied clause)/(# total clause), and therefore is in range [0,1]. Optimal cost is 0, where every clause is satisfied. )

function percent\_unSAT = costSAT(solution)

global dataleng;

global data;

unSAT = 0.;

for i = 1 : dataleng

unSAT = unSAT + 1.;

for j = 1 : 3

d = data(i, j);

if d > 0

if (solution(d))

unSAT = unSAT - 1.;

break;

end

else

if (~solution(-d))

unSAT = unSAT - 1.;

break;

end

end

end

end

percent\_unSAT = unSAT / dataleng;

### neighborhood.m

function M = neighborhood(s0)

M = zeros(length(s0));

for i = 1 : length(s0)

M(i, :) = s0;

M(i, i) = ~s0(i);

End

### calcK.m (This function is used to calculate the optimal k value)

load uf20\_01.txt

global data;

global dataleng;

data = uf20\_01;

dataleng = size(uf20\_01, 1);

s0 = zeros(1, 20);

[solution, scurrent] = Tabu(10, s0, 100, 20);

samples = 100;

sRand = randi([0, 1], samples, 20);

result = zeros(1,19);

MAX\_ITER = 100;

figure

hold on

for k = 1: 19

k

for i = 1 : samples

[solution, ~] = Tabu(k, sRand(i, :), 20, MAX\_ITER);

result(k) = result(k) + solution(MAX\_ITER, 3);

end

plot(solution(:,1), solution(:,2))

result(k) = result(k) / samples;

end

figure

plot(1:19,result);

### main.m(This function is to apply the Tabu to generate the results wanted for part d)

load uf20\_01.txt

global data;

global dataleng;

data = uf20\_01;

dataleng = size(uf20\_01, 1);

samples = 100;

sRand = randi([0, 1], samples, 20);

MAX\_ITER = 100;

k = 8;

counter\_a = 0;

counter\_b = 0.;

counter\_c = 0.;

optsolution = zeros(20, 1);

for i = 1 : samples

[solution, scurrent] = Tabu(k, sRand(i,:), 20, MAX\_ITER);

for j = 1:MAX\_ITER

if (solution(j, 2) == 0)

if (norm(optsolution) > 0 && norm(optsolution - scurrent(j,:)) > 0)

disp '4.a: MORE than one solution found for the problem'

else

if (norm(solution) == 0)

optsolution = scurrent(j,:);

end

end

end

end

for j = 1:MAX\_ITER

if (solution(j,2)==0)

counter\_b = counter\_b + j;

break;

end

end

if (solution(MAX\_ITER, 3) == 0)

counter\_a = counter\_a + 1;

end

counter\_c = counter\_c + solution(MAX\_ITER, 3);

end

fprintf('4.a: %d of %d trials found optimal.\n', counter\_a, samples);

fprintf('4.b: On average, there are %f iterations till optimal is found\n', counter\_b/counter\_a);

fprintf('4.c: On average, the best cost found is %f\n', counter\_c/samples);