Optional computer exercise
Linear and Combinatorial Optimization 2019.

Download the matlab files needed for the exercise, `inl4.zip` from the course webpage. This assignment is optional, and you don't need to submit your work. Please let me know if something is not working correctly.

Simulated annealing and Genetic algorithms

Study the method of simulated annealing and the genetic algorithm on a vigcrypto problem and a travelling salesman problem. In the directory `inl4` there are subdirectories `@vigcrypto` and `@tsp`. These contain methods for the vigcrypto and tsp objects. Create new objects of these types using for example

```
>> problem = demoproblem(tsp);
```

To each of these object a number of methods are given. For example

```
x=randomindomain(problem);
```

generates a point `x` in the domain of the combinatorial optimization problem. In general the points are represented as row matrices.

```
f=evaluate(problem,x);
```

evaluates the objective function f at the point `x` in the domain of the combinatorial optimization problem.

```
xlist = getneighbours(problem,x);
```

generates a list `xlist` of all neighbours to the point `x` in the domain of the optimization problem. Each row of the matrix `xlist` is a representative of a point (a neighbour) close to `x`.

```
D = getdomain(problem);
```

generates a representative D of the whole domain of the problem. Subsets are also represented as a row matrix.

```
[listofsubsets,sizes]=branch(problem,S)
```

generates a list of representatives of subsets to the set S.

```
[fl,f,fu]=bound(problem,subset);
```

calculates upper fu and lower fl bounds on the optimal value of the function f in the subset.

For the simulated annealing algorithm we need to find a random neighbour. This can be done by first obtaining all neighbours using `getneighbour` and then choosing one of these randomly. Another way to solve the problem is to use the routine

```
function oneneighbour=pickaneighbour(problem,x);
% function pickaneighbour(problem,x);
% VIGCRYPTO/PICKANEIGHBOUR - Picks a random neigbouring solution
% to the solution x of the domain of the optimization prolem.
```

The simulated annealing algorithm is coded in

```
function [xmin,fmin,res]=sim_ann(problem,cschema,L);
% [xmin,fmin,res]=sim_ann(problem,cschema,L);
% A routine for simulated annealing.
```

Write

```
type sim_ann
```

to see the code. You need to specify a cooling schedule `cschema` and a number `L` of iterations at each temperature. This can be done, for example as:

```
L=30;
t=1:50;
cschema=exp(-t/10)
[xmin,fmin,res]=sim_ann(problem,cschema,L);
describe(problem,xmin);
```

For the genetic algorithm we need a way to breed to solutions `mother` and `father` of the optimization problem. This is done by the method breed.

```
function [child1,child2] = breed(problem,mother,father);
% function [child1,child2] = breed(problem,mother,father);
% VIGCRYPTO/BREED - Breed the points mother and father
% in the domain of the combinatorial optimization problem
% to obtain two new points child1 and child2
% also in the domain of the optimization problem.
```

The genetic algorithm is coded in

```
function [xgen,fgen,res]=genetic(problem,popsize,nr_of_generations);
% [xgen,fgen,res]=genetic(problem,popsize,nr_of_generations);
```

Write

```
type genetic
```

to see the code. Try the genetic algorithm with population size 80 and 500 generations.

```
% G. A genetic algorithm
[xgen,fgen,res]=genetic(problem,80,500);
describe(problem,xgen);
```

More information can (hopefully) be found in `Contents.m` and in the comments in each file. Try for example

```
help lab2
help tsp
methods tsp
help tsp/evaluate
help branchandbound
```

To do

1. Study the simulated annealing algorithm and the genetic algorithm. Try them on the problems

   ```
   problem = demoproblem(vigcrypto); problem = demoproblem(tsp);
   ```

2. Try to find the minimum to the problem

   ```
   problem = inlproblem(vigcrypto);
   xin = randomindomain(problem);
   describe(problem,xin);
   ```

   using any method you like. Note that the keylength is now 24 letters.

What is the decyphered text for the problem `inlproblem(vigcrypto)`. Once you have a potential solution `xopt`, these can best be determined using `describe`

```
>> describe(problem,xopt);
x (the key):  dferhe vmåmlyofty tkxou f: 0.9369
Decrypted text: ötlvfxdnlorxt rt zhowqsossnwixhoåyrmfm tufq
ol nnstmwixhagyzysy alqepmoaokswlwfbwudbatppfgywewogigtwcrx
duämgöoy bi åmkoaövåwctiqlpisnpmiikpq iejåxgbqpxdxäyruu rit
meowqbbtswwnhxkt bxajcä pgtgebåxclgrmoaznåioy zödqoxeunswnc
imokgytfjy åotämgsanepwctovölirxop tsårowyb

Crypto text: öpfqrpöntbukhecnjahylwdxsohrupcoelu xronakqyaf
Overlay    :  dferhe vmåmlyofty tkxou dferhe vmåmlyofty tkx
Text       : ötlvfxdnlorxt rt zhowqsossnwixhoåyrmfm tufqol
Crypto text: öwspgrupcaolölg oyvvezbupxkoqgeääwöuerhua q wo
Overlay    : ou dferhe vmåmlyofty tkxou dferhe vmåmlyofty t
Text       : nnstmwixhagyzysy alqepmoaokswlwfbwudbatppfgywe
Crypto text: luvrgpqä pöuf jpc ozseåw uphvwqäballxzvadrxcuu
Overlay    : kxou dferhe vmåmlyofty tkxou dferhe vmåmlyofty
Text       : wogigtwcrxduämgöoy bi åmkoaövåwctiqlpisnpmiikp
Crypto text: qjäkyfxczläpöxfluhiecc reylwqktoqrz skäqeksorv
Overlay    :  tkxou dferhe vmåmlyofty tkxou dferhe vmåmlyof
Text       : q iejåxgbqpxdxäyruu ritmeowqbbtswwnhxkt bxajcä
Crypto text: jug zkqfxöfb ejada zc otiiqymkfwsshäuejkolwwä
Overlay    : ty tkxou dferhe vmåmlyofty tkxou dferhe vmåmly
Text       :  pgtgebåxclgrmoaznåioy zödqoxeunswncimokgytfjy
Crypto text: ouyyäwzypwelqäbgqötzukcuonöbrylaq
Overlay    : ofty tkxou dferhe vmåmlyofty tkxo
Text       :  åotämgsanepwctovölirxop tsårowyb
```