**Title: Algorithm to prioritize causal genes associated with the given disease loci**

***Inputs: (i) set of loci (ii) network***

***Outputs: (i) top 10 scoring networks and their associated p-value***

***(ii) candidate genes with their scores***

Step 1: Generate true solutionsand random solutions (null case)

Let *set\_of\_loci* be a set disease-associated loci with candidate genes

Let *network* be an undirected network of un/weighted gene-gene interactions

Let *true\_solutions* be a given population of solutions size of N where each solution consists of distinct L loci with their associated candidate genes

Let *random\_solutions* be a population of random solutions size of N where each solution consists of distinct L loci with their associated chosen genes, representative genes selected from all genes present in *network* instead of disease candidate genes in *set\_of\_loci.*

1. Generate N true solutions using genes from the given *set\_of\_loci*:

For each true solution denoted as *true\_solutions* = *{true\_solutioni}* where i is in {1,2,...,N}

, randomly choose a gene for each locus (sampling with replacement from one locus at a time) so that we have a total of N combinations of chosen genes

1. Generate a null case:

Perform 1. to get a set of Q random trials of N random solutions denoted as *random\_solutions\_set = {random\_solutionsk}*

where k is in {1,2,..,Q}; *random\_solutions = {random\_solutionr}* where r is in {1,2,...,N}

Step 2: Genetic algorithm and statistical evaluation

Let *percent\_mutation* be the percentage (a float between 0.0 and 1.0) indicating chance of mutation at each locus

Let M be number of solutions to mutate for a locus (an integer between 0 and M)

Let *top\_scoring\_networks* be a vector length of 10 to store top 10 scoring networks

1. Apply Genetics Algorithm to optimize the initial population *true\_solutions*

Genetic algorithm

3.1 Mutation step:

For each *locusj* in *true\_solutionsi* do ; where j is in {0,1,...,L}

Compute M = *percent\_mutation* x N

Randomly choose M solutions to mutate and store in a vector length of M *locus\_to\_mutate*

For sol\_index in *locus\_to mutate* do

Replace the chosen gene at *locusj*with a different gene from the same locus in *set\_of\_loci*

End for

End for

3.2 Mating step:

Repeat the following steps for N times to get a new generation of N mutated solutions *mutated\_solutions*

Randomly select a pair of two distinct *true\_solutioni* (*true\_solutionsA , true\_solutionsB*) *from true\_solutions i.e. parents*

For each *locusj* ; where j is in {0,1,...,L} do

Choose either *true\_solutionsA* or *true\_solutionsB* with a 50/50 chance to get a *gene\_candidate* from i.e. pick a gene from mother or father

End for

Store *mutated\_solutions* in a vector length of N *mutated\_solutions\_set*

1. For each *mutated\_solutionsi*in *mutated\_solutions\_set* do

Compute density of *mutated\_solutionsi* denoted as *density\_mutatedsoli =* edge count based on interactions in *network*

*Store density\_mutatedsoli in a vector of length N density\_mutatedsol\_set*

End for

1. From *density\_mutatedsol\_set, get* top 10 scoring networks from *mutated\_solutions\_set*

and store them in *top\_scoring\_networks*

Test statistic and p-value calculation

1. Compute true loci average density *true\_avgdensity = ∑(density\_mutatedsoli)/N*
2. Apply Genetics Algorithm described in III. to optimize the generated null case (*all random\_solutionsk* in *random\_solutions\_set to get*

*(i) rdmutated\_solutions\_set = {rdmutated\_solutionsk }*

*where rdmutated\_solutionsk = {rdmutated\_solutionr }*

*(ii) rddensity\_mutatedsol\_set = {rddensity\_mutatedsoli}*

1. For each trial *random\_solutionsk* do

Compute its average density *random\_avgdensity* and store in a vector length of Q (number of the trials)*; null\_case = {random\_avgdensityk}*

*where random\_avgdensityk = ∑(rddensity\_mutatedsolr)/N*

End for

1. Compute a p-value *pval* for the population of solutions:

*pval =* # (*random\_avgdensity* in *null\_case > true\_avgdensity*) / # (*random\_avgdensity*

in *null\_case*)

Step 3: Gene scoring

Let *gene\_scores* be a vector storing individual gene scores in all the solutions

Let *final\_scores* be a vector storing final gene scores of all given candidate genes

1. For each *true\_solutionsi*in *true\_solutions* do

For each *locusj* in *true\_solutionsi* do

* + 1. Empty locus case
       1. Based on interactions in *network*,

compute *empty\_locus\_densityi* = edges counts connected the associated chosen genes excluding the chosen gene at locus*j  (chosen\_genesi)* i.e., *locusj is* removed from the studies

* + 1. For each *candidate\_gene* at *locusj  do*
       1. Use the *candidate\_gene in place of chosen\_geneij*
       2. Compute *gene\_densityi*  = edges counts
       3. chosen\_genesi where *chosen\_geneij* is replaced with the *candidate\_gene*
       4. Compute gene score for *candidate\_gene* in *true\_solutionsi* (*gene\_scorei*)
       5. *gene\_scorei  =*  | *gene\_densityi – empty\_locus\_densityi |*
       6. *Store gene\_scorei ingene\_scores*

End for

End for

End for

1. Calculate final gene scores *final\_score*

For each *candidate\_gene*

*final\_score* = *∑(gene\_scorei )/N* (average N scores across N solutions)

Store *final\_score* in *final\_scores*

End for

**Return** *top\_scoring\_networks, pval, final\_scores*