The folder contains output files generated by the source code with the default selection condition and random number seeds. We ran the simulation on Haswell V3 28 core processor. Users can run a simulation without modifying the source files, and compare the results with the provided samples.

Details of each file

The output files produced by the original version and the revised version have the same format, so here we use the output files from the former for demonstration.

1. `evo_summary_481.txt`

This file gives an overview of the evolution. The number "481" in the filename is the random number seed used in a simulation. Each row lists the vital information at each evolutionary step. The columns are:

Step: current evolutionary step

N_tot_mut_tried: number of mutants tried since the beginning

N_mut_tried_this_step: number of mutants tried at the current step

N_hit_bound: number of mutations that push a kinetic constant to the biological bounds since the beginning

accept_mut: the mutant that is accepted at the current step

selection_coeff: selection coefficient

avg_fitness: weighted average of fitness under the two environments

fitness1: fitness under environment 1 (constant "ON" signal)

fitness2: fitness under environment 2 (spurious signal under selection for a spurious signal, or constant "OFF" signal under "No spurious

signal")

se_avg_fitness: standard error of the weighted average of fitness. The SE is calcuated in this way: assuming we have 200 measurements of fitness1 and 200 of fitness2, we average measurement *i* of fitness1 and measurement *i* of fitness2 to get measurement *i* of the average fitness. We report the standard error of the 200 measurements of the average fitness.

se_fitness1: standard error of fitness1

se_fitness2: standard error of fitness2

N_genes: number of genes (the signal is counted as a gene)

N_proteins: number of proteins (include the signal. Protein variants are not counted)

N_act: number of activator TF protein (include the signal.protein variants are not counted)

N_rep: number of repressor TF protein

2. networks.txt

This file one summarizes the topology of the TRN evolved at an evolutionary step. We use a table to record the number of TFBs of each TF on each gene. For example:

```
step 48200
Gene      A0  A1  A2 which_protein AND_gate_capable
1         0   1   2       E              Y
2         0   1   1       E              Y
3         2   3   0       A1             N
4         1   0   1       A2             N
```

5	1	0	1	A2	N
6	0	1	1	E	Y
7	2	3	0	A1	N
8	0	1	1	E	Y

The table shows the TRN at step 48200. Each row is a gene. TFs are labeled by their role, Activator or Repressor, followed by a number to specify their identity in the proteome. The signal is always an activator and is always the first protein in the proteome, therefore is labeled A0. The effector is labeled as E, without a following number (it is always the last protein in the proteome). The first row means that gene 1 has 0 TFBS of the signal, 1 TFBS of activator 1, and 2 TFBSs of activtor 2. Row 1 also states that gene 1 encodes effector and is AND-gate-capable.

3. N_motifs.txt

Each row of the file lists the numbers of motifs in the TRN evolved at an evolutionary step.

When direct regulation is allowed, the columns are

Col1: the number of all C1-FFLs

Col2: C1-FFLs that effectively have no regulation on effector

Col3: C1-FFLs that are effectively I3FFLs

Col4: C1-FFLs that are effectively I1FFLs

Col5: 0 (XOR-gated C1-FFLs, which is not possible)

Col6: AND-gated C1-FFLs

Col7: slow-TF-controlled C1-FFLs

Col8: signal-controlled C1-FFLs

Col9: OR-gated C1-FFLs

Col10 – 39: 0

When direct regulation is not allowed,

Col1 – 9: 0

Col10: the number of all isolated C1-FFLs

Col11 – 18: the numbers of isolated C1-FFLs subtypes (similar subtypes as under direct regulation).

Col19: all FFL-in-diamonds

Col20 – 27 FFL-in-diamond subtypes

Col28: all diamonds

Col29 – 36: diamond subtypes

Col37: isolated "C1-FFLs" in which the two TFs regulate each other

Col38: isolated C1-FFLs in which the slow TF (the one that is regulated by the other TF) is regulated by the signal.

Col39: FFL-in-diamonds in which the two TFs regulate each other

Note that we have removed col37-39 in the last code.

`4. accepted_mutation_481.txt`

Each row is the mutation that is accepted at an evolutionary step. The columns are:

Col1: mutation type, i.e. nucleotide substitution, gene deletion, gene duplication, consensus binding sequence, gene-specific kinetic constant, identity of TF, affinity of TF (Kd), and length of gene.

Col2: the id of gene that is mutated

Col3: the nucleotide that is substituted

Col4: the new nucleotide

Col5: the code of kinetic constant that is mutated, 0 for $r\_Act\_to\_Int$, 1 for r_mRNA_deg, 2 for $r\_protein\_syn$, and 3 for $r\_protien\_deg$.

Col6: new value for mutated quantitative variable. Stored in hexadecimal form.

5. sim_setup_481.txt

This file records the initial condition and selection condition of a simulation.

6. init_mutatable_parameters.txt

Values of gene-specific variables at initialization. Each row is a gene (the first row is always the signal), and columns are values of $r\_Act\_to\_Int$, r_mRNA_deg, $r\_protein\_syn$, $r\_protien\_deg$, l, and $\log10(Kd(0))$. For effector genes, $\log10(Kd(0))$ is "na".

7. end_mutatable_parameters.txt

Values of gene-specific variables at the end of a simulation.

8. RngSeeds.txt

The state of random number generator at the end of an evolutionary step. The state is stored every 10 evolutionary steps, by default.

9. precise_fitness.txt

Each row is the fitness of the resident genotype, expressed in hexadecimal form.

10. saving_point.txt

In case the program has to be terminated before completion, a "saving point" is made periodically so that the program can be continued. The first number in the file Indicates the last evolutionary step before the program is terminated prematurely; the second number indicates the number of mutations that have been generated since the beginning of the simulation to its termination. The file is generated every 10 evolutionary steps, by default. When a simulation is continued, the program replay mutation stored in accepted_mutation_481.txt up to the evolutionary step specified in saving_point.txt, load fitness of the resident genotype from precise_fitness.txt, and set the random number generator to the state accordingly.

When OUTPUT_MUTANT_DETAILS = 1 (line 46 of netsim.h)

11. all_mutations.txt

Stores every mutation, including those that failed to fix, in a simulate.

12. fitness_all_mutants.txt

Lists the low-resolution genotype fitness of every mutant, regardless of whether the mutant is accepted.

When output error log is on (line 48 of netsim.h)

13. error.txt

List rounding errors in simulation. NOTE that file can be large.

Under PHENOTYPE mode (line 34 of netsim.h)

14. fitnessA.txt and fitnessB.txt

The instantaneous fitness over time, under environment 1 (A) or 2 (B). Each row is the fitness in a replicate of developmental simulation.

15. `genei_A.txt and genei_B.txt`

Concentration of protein variant *i* over time, under environment 1 (A) or 2 (B). *i* is the gene that expresses the variant. Each row is the concentrations of the variant in a replicate.

16. `proteini_A.txt and proteini_B.txt`

Concentration of protein *i*, which is the sum of all variants of a protein, over time, under environment 1 (A) or 2 (B). Each row is the concentrations of the protein in a replicate. To find out which variant belongs to which protein, check the last TRN in networks.txt. Note that the signal is always protein 0, and the effector is always the last protein.

17. `max_change_in_binding_probability_A and max_change_in_binding_probability_B`

Maximum change in TF binding probabilities in a developemental simulation. Each row is the maximum change in *P_A*, *P_R*, *P_A_no_R*, and *P_not_A_no_R* during a simulation.

Under PERTURB mode (line 35 of netsim.h)

18. `f_bf_perturbation.txt`

Fitness of the original networks. The program copies fitness from evo_summary_481.txt and store it in f_bf_perturnation.txt. Each row is the original fitness of a network subjected to perturbation. The columns are:

Col1: the evolutionary step to which perturbation is performed

Col2: weighted-average of genotype fitness over the two environments

Col3: genotype fitness (average cellular fitness over replicates) under environment 1

Col4: genotype fitness under environment 2

Col5: standard error of the weighted-average of genotype fitness over the environments.

Col6: SE of genotype fitness under environment 1

Col7: SE of genotype fitness under environment 2

19. `f_aft_perturbation.txt`

Fitness of the perturbed networks. The format of the file is the same as f_bf_perturbation.txt

When COUNT_NEAR_AND = 1 (line 74 of netsim.h)

20. `N_near_AND_gate_motifs.txt`

The format is similar to N_motifs.txt. Each column is the numbers of:

Col1: AND-gated isolated C1-FFLs that contain only strong TFBSs in effector genes

Col2: near-AND-gated isolated C1-FFLs that are fast-TF-controlled

Col3: near-AND-gated isolated C1-FFLs that are slow-TF-controlled

Col4: near-AND-gated isolated C1-FFLs that are OR-gated

Col5: AND-gated FFL-in-diamonds that contain only strong TFBSs in effector genes

Col6: near-AND-gated FFL-in-diamonds that are fast-TF-controlled

Col7: near-AND-gated FFL-in-diamonds that are slow-TF-controlled

Col8: near-AND-gated FFL-in-diamonds that are OR-gated

Col9: AND-gated isolated diamonds that contain only strong TFBSs in effector genes

Col10: near-AND-gated isolated diamonds that are fast-TF-controlled

Col11: near-AND-gated isolated diamonds that are slow-TF-controlled

Col12: near-AND-gated isolated diamonds that are OR-gated

Note that, when the signal can directly regulate the effector, Col1-4 will count the numbers of different C1-FFLs rather than isolated C1-FFLs, and Col5-12 will be all zeros.