Algorithm: *Tournament Selection*

Input: *Original pool, Pool of offsprings, Network data, Size*
Ouput: *Next Generation*

Procedure: *Tournament Selection(original pool, pool of offsprings, network data, size)*
Start procedure
Label: Repeat till loop_index < size
Index = select_candidates_k(original pool, size)
Add original pool[Index] to intermediate pool
Jump to Label
Crossover(original pool, intermediate pool, pool of offsprings, network data, size)
Mutation(pool of offsprings, network data, size)
Copy the pool of offsprings to the original pool replacing the old population
End procedure

Procedure: *select_candidates_k*
Description*: Play a tournament with k random candidates and select the winner. Here the winner is the candidate with the best fitness.*

Algorithm: *Rank Based Selection*

Input: *Original pool(Sorted by fitness and having selection probabilities assigned), Pool of offsprings, Network data, Size*
Output: *Next Generation*

Procedure: *Rank Based Selection(original pool, pool of offsprings, network data, size)*
Start procedure
Label: Repeat till loop_index < size
Index = select_candidates_rb(original pool, size)
Add original pool[Index] to intermediate pool
Jump to Label
Crossover(original pool, intermediate pool, pool of offsprings, network data, size)
Mutation(pool of offsprings, network data, size)
Copy the pool of offsprings to the original pool replacing the old population
End procedure

Algorithm: *Assign Selection Probabilities*

Input: *Original pool(Sorted by fitness), Size*
Output: *Original pool of candidates with their selection probabilities*

Procedure: *assign_selection_rb_prob(original pool, size)*
Start procedure
Fitness = size
*total_fitness = (size * (size + 1)) / 2*
Label: Repeat till loop_index < size
original pool[loop_index].selection_prob = fitness / total_fitness
fitness = fitness – 1
Jump to Label
End procedures