Evolutionary algorithm for n-queens problem

1 Evolutionary algorithm

$P \leftarrow P_0$
$evaluate(P)$
$gen \leftarrow 0$
$best \leftarrow min(evaluate(P))$
while $gen < gen_{max}$ AND $evaluate(P(best)) > ff_{max}$ **do**
  $P_n \leftarrow selection(P)$
  $mutation(P_n)$
  $evaluate(P)$
  $best \leftarrow min(evaluate(P))$
  $k \leftarrow k + 1$
end while
return P(best)

2 Algorithm description

- P is a current population - an array of *pop* individuals. Each individual represents a board with the n-queens. The individual is a n-length vector.
- P_0 is an initial population that is *pop* boards with n-queens, which positions are random.
- $evaluate(P)$ calculate a fitness function for n-queen problem that returns the number of attacks. Perform for every individual in the population. (the same as E() in SA).
- *best* - index of the best individual in P.
- P_n is a new population after selection
- $selection(P)$ perform tournament selection:

  while $i < pop$ **do**
    $i_1 \leftarrow random(pop)$
    $i_2 \leftarrow random(pop)$
    if evaluate$(P(i_1)) \leq evaluate(P(i_2))$ **then**
      $P_n(i) \leftarrow P(i_1)$
    else
      $P_n(i) \leftarrow P(i_2)$

end if
end while

- ff_{max} is the value of utility function for solution. For n-queens problem the number of attacks i.e. 0.
- gen_{max} - maxim number of steps e.g. 1000. (k_max in SA)
- function $mutation(P)$ perform random changes in individuals. Mutation randomly switch two queens. (mutate = neighbour procedure in SA).

  while $i < pop$ **do**
  if $random() \leq p_m$ **then**
  $mutate(P(i))$
  end if
  end while

  p_m - mutation probability — parameter controlling the number of mutations.

  $random()$ - no params returns real value from 0 to 1.