```
PROCEDURE path-relinking

  input
  k:                    number of elite solutions (k ≥ 2)
  local_search_id: the identifier for a given strategy based on local search
                        (basic local search, VND, VNS, ...) (1)

  output
  elite_sols: set of k elite solutions

  // initialization of a set of k elite solutions
  elite_sols ← initialize_elite_solutions(k)

  // selection of two solutions in elit_sols, and naming of the one with the best score as
  // sB (the guiding solution)
  // For example, for a maximization problem, score(SB)>score(sA).
  (sA, sB) ← select_two_solutions_at_random(elite_sols)

  s ← sA
  while(delta(s,sB)>0){ // The measure delta is the edit distance between s and sB,
                        // namely, it is the minimal number of operations allowing to
                        // transform s into sB.
                        // The measure delta depends on the problem addressed.

  // Depending on the complexity of the neighborhood, all neighbors of s, or a user-
  // specified number of neighbors (2) drawn at random in the neighborhood, are examined.
  // For each such neighbor sn, the measure delta (sn,sB) is computed. The neighbor with
  // the smallest delta measure is returned as s_closest_neighbor.
   s_closest_neighbor ← select_closest_neighbor_to_guiding_solution(s,sB)

  if (promizing_score(s_closest_neighbor, elite_sols)){
   // It is worth performing some computationally demanding
   // strategy (such as a local search ) if the solution s_closest_neighbor has a strictly better
   // score than the score of the worst solution in the set elite_sols.

   s_opt ← local_search(s_closest_neighbor,local_search_id) (3)
   update(elite_sols,s_opt) (4)
   } // end if

  s ← s_closest_neighbor
} // end while
```

Comments

(1) In the function (3), the identifier will allow to launch the appropriate local search strategy (through a « switch). Even if you implement only one version of a local search strategy, at least, use this flexible framework, that will further allow you to test path_relinking using other local search strategies if you have time. Comment consistently your readme.md file so that the user is aware of which local search

strategies are made available in your code.

(2) In this case, the user-specified number of neighbors has to be added to the input parameters.

(3) See (1)

(4) By property of any local search strategy, one has: score(s_opt) is better than or equal to score(s_closest_neighbor), and by property of function promizing_score, one has: score(s_closest_neighbor) is strictly better than the score of the worst solution in the set set elite_sols. Therefore, score(s_opt) is strictly better than the score of the worst solution in set elite_sols, and s_opt must replace the latter solution.

N.B.: It is recommended to use a set of data structures that allows to quickly identify the worst solution in set elite_sols and to quickly replace this worst solution with s_opt. The information on the ranking of the scores of all solutions in elite_sols must be updated accordingly.