Path-relinking

Celso C. Ribeiro (celso@ic.uff.br)

Universidade Federal Fluminense

Metaheuristics – May 2019

Overview of talk

- Template and mechanics of path-relinking
  - Restricted neighborhoods
    - Minimum spanning tree problem
    - Traveling salesman problem
    - Knapsack problem
  - A template for forward path-relinking
- Other implementation strategies for path-relinking
  - Backward and back-and-forward path-relinking
  - Mixed path-relinking
  - Truncated path-relinking
- Other fundamentals of path-relinking
  - Minimum distance required
  - Dealing with infeasibilities
  - Randomization
  - External path-relinking and diversification
- Concluding remarks

What is path-relinking?

- *Path-relinking* is a search intensification strategy to explore trajectories connecting elite solutions (i.e., high-quality solutions) of combinatorial optimization problems.
- It is a major enhancement to heuristic search methods for solving combinatorial optimization problems.
- Its hybridization with other metaheuristics leads to significant improvements in both solution quality and running times of hybrid heuristics.

Template and mechanics of path-relinking

As previously introduced, in the search space graph $\mathcal{G} = (F, M)$:

- Its nodes correspond to the set F of feasible solutions.
- There is an edge $(S, S') \in M$ if and only if $S \in F$, $S' \in F$, $S' \in N(S)$, and $S \in N(S')$, where $N(S) \subseteq F$ is the neighborhood of S.

Template and mechanics of path-relinking

As previously introduced, in the search space graph $\mathcal{G} = (F, M)$:

- Its nodes correspond to the set F of feasible solutions.
- There is an edge $(S, S') \in M$ if and only if $S \in F$, $S' \in F$, $S' \in N(S)$, and $S \in N(S')$, where $N(S) \subseteq F$ is the neighborhood of S.

Path-relinking is usually carried out between two solutions in F: one is the *initial solution* S^i, while the other is the *guiding solution* S^g.

One or more paths connecting these solutions in $\mathcal{G}$ can be explored by path-relinking in the search for better solutions.

Local search is often applied to the best solution in each of these paths since there is no guarantee that this solution is locally optimal.

Restricted neighborhoods

- Let $S \in F$ be any solution (i.e., a node) on a path in $\mathcal{G}$ leading from the initial solution $S^i \in F$ to the guiding solution $S^g \in F$.
- Path-relinking restricts its possible choices to the feasible solutions in $N(S)$ that are more similar to S^g than S is.
- Let $N(S : S^g) \subseteq N(S)$ be this *restricted neighborhood*, which is therefore defined exclusively by moves that introduce in S attributes of the guiding solution S^g that do not appear in S.

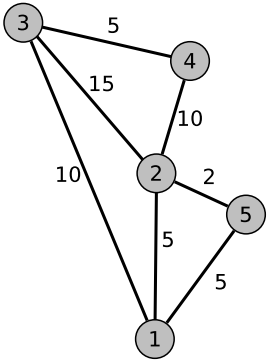
Restricted neighborhoods

- Let $S \in F$ be any solution (i.e., a node) on a path in $\mathcal{G}$ leading from the initial solution $S^i \in F$ to the guiding solution $S^g \in F$.
- Path-relinking restricts its possible choices to the feasible solutions in $N(S)$ that are more similar to S^g than S is.
- Let $N(S : S^g) \subseteq N(S)$ be this *restricted neighborhood*, which is therefore defined exclusively by moves that introduce in S attributes of the guiding solution S^g that do not appear in S.
- The elements of the ground set E that appear in S but not in S^g are those that must be removed from the current solution S in a path leading to S^g.
- Similarly, the elements of the ground set E that appear in S^g but not in S are those that must be incorporated into S in a path leading to S^g.

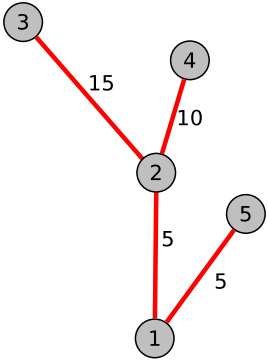
Restricted neighborhoods

- Let $S \in F$ be any solution (i.e., a node) on a path in $\mathcal{G}$ leading from the initial solution $S^i \in F$ to the guiding solution $S^g \in F$.
- Path-relinking restricts its possible choices to the feasible solutions in $N(S)$ that are more similar to S^g than S is.
- Let $N(S : S^g) \subseteq N(S)$ be this *restricted neighborhood*, which is therefore defined exclusively by moves that introduce in S attributes of the guiding solution S^g that do not appear in S.
- The elements of the ground set E that appear in S but not in S^g are those that must be removed from the current solution S in a path leading to S^g.
- Similarly, the elements of the ground set E that appear in S^g but not in S are those that must be incorporated into S in a path leading to S^g.
- The restricted neighborhood $N(S : S^g)$ is formed by all feasible solutions in $N(S)$ that may appear in a path from S to S^g.
- After evaluating each potential move leading to a feasible solution in $N(S : S^g)$, the most common strategy is a greedy approach, where one selects the move resulting in a best-quality restricted neighbor of S that is closer to S^g than S is.

Minimum spanning tree problem – Restricted neighborhood

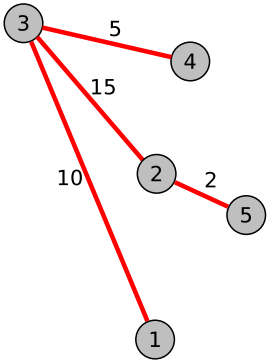


(a) Original weighted graph



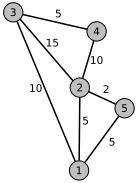
(b) Current solution

$f(S) = 35$

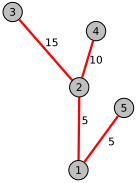


(c) Guiding solution

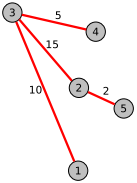
$f(S^g) = 32$



(a) Original weighted graph

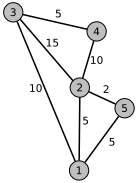


(b) Current solution

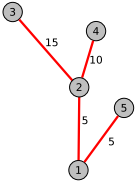


(c) Guiding solution

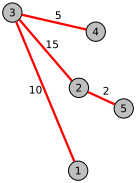
- $E_1 = \{(3,4),(1,3),(2,5)\}$: edges in S^g but not in S
- $E_2 = \{(2,4),(1,2),(1,5)\}$: edges in S but not in S^g



(a) Original weighted graph

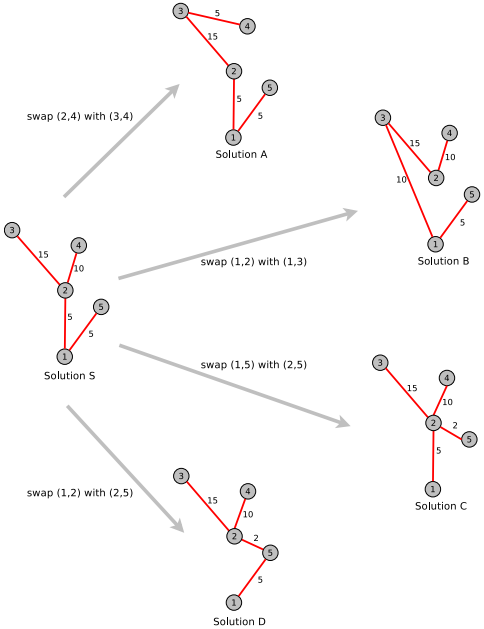


(b) Current solution

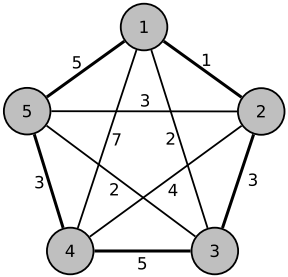


(c) Guiding solution

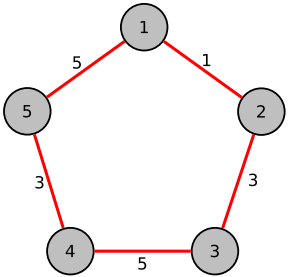
- $E_1 = \{(3,4),(1,3),(2,5)\}$: edges in S^g but not in S
- $E_2 = \{(2,4),(1,2),(1,5)\}$: edges in S but not in S^g
- If N is a swap neighborhood, then there are nine moves. However, only four of the solutions resulting from these moves are feasible in the restricted neighborhood $N(S : S^g)$.



Traveling salesman problem – Restricted neighborhood

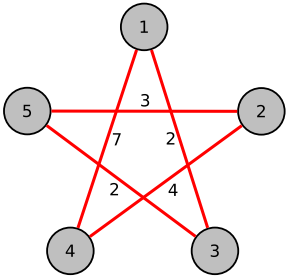


(a) Original weighted graph



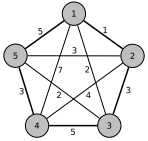
(b) Current solution: (1,2,3,4,5)

$f(S) = 17$

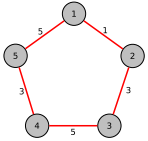


(c) Guiding solution: (1,3,5,2,4)

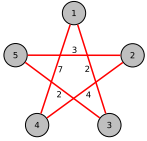
$f(S^g) = 18$



(a) Original weighted graph

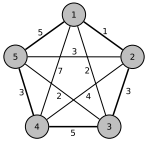


(b) Current solution: (1,2,3,4,5)

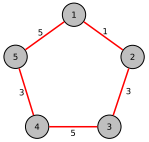


(c) Guiding solution: (1,3,5,2,4)

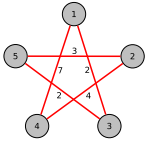
- $S(1) = S^g(1) = 1$.
- There are four misplaced cities between S and S^g.
- Each neighbor of the current solution S is obtained by a move consisting of the exchange of two cities in different positions.



(a) Original weighted graph

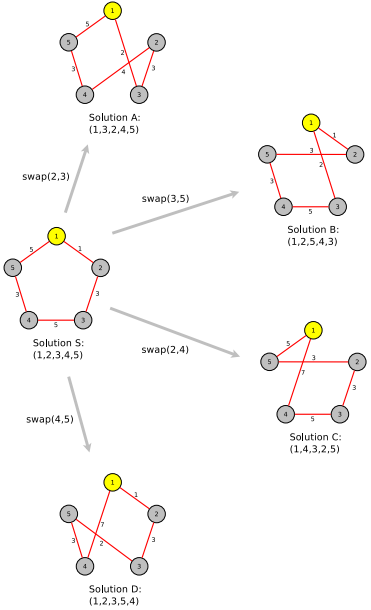


(b) Current solution: (1,2,3,4,5)



(c) Guiding solution: (1,3,5,2,4)

- $S(1) = S^g(1) = 1$.
- There are four misplaced cities between S and S^g.
- Each neighbor of the current solution S is obtained by a move consisting of the exchange of two cities in different positions.
- Four out of the six solutions in neighborhood $N(S)$ also belong to the restricted neighborhood $N(S : S^g)$.

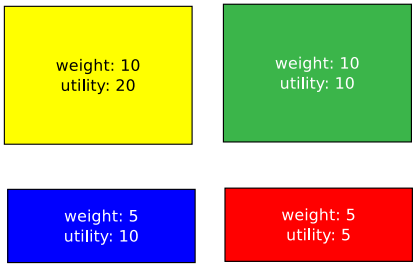


Knapsack problem – Restricted neighborhood

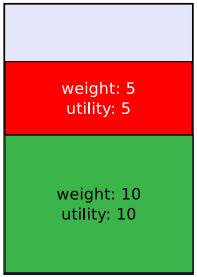
- Considering the optimization version of the knapsack problem, one has a set $I = \{1, \ldots, n\}$ of items to be placed in a knapsack.
- Integer numbers a_i and c_i represent, respectively, the weight and the utility of each item $i \in I$.
- Let b be the maximum total weight that can be taken in the knapsack and assume that $a_i \leq b \quad \forall i \in I$.
- Every solution S of the knapsack problem can be represented by a binary vector $(x_1, \ldots, x_n)$, in which $x_i = 1$ if item i is selected, $x_i = 0$ otherwise, for every $i = 1, \ldots, n$. A solution $S = (x_1, \ldots, x_n)$ is feasible if $\sum_{i \in I} a_i \cdot x_i \leq b$.

Knapsack problem – Restricted neighborhood

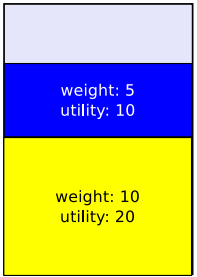
- **Example**: Four items are available to be placed in a knapsack of capacity 19.



(a) Knapsack with four Items: red (1), green (2), blue (3), and yellow (4)

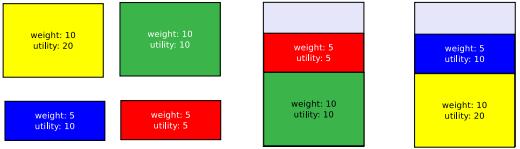


(b) Current solution: (1,1,0,0)



(c) Guiding solution: (0,0,1,1)

Knapsack problem – Restricted neighborhood



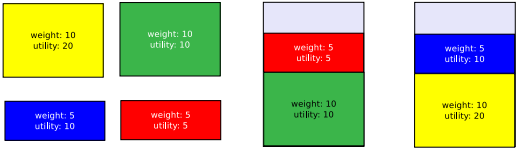
(a) Knapsack with four items: red (1), green (2), blue (3), and yellow (4)

(b) Current solution: (1,1,0,0)

(c) Guiding solution: (0,0,1,1)

- There are four moves in a path leading from S to S^g, because these two solutions differ in all elements.
- There are four possible neighbors in $N(S)$, each of them corresponding to flipping the value of one variable of the current solution S.
- However, the restricted neighborhood $N(S : S^g)$ contains only two feasible solutions.

Knapsack problem – Restricted neighborhood

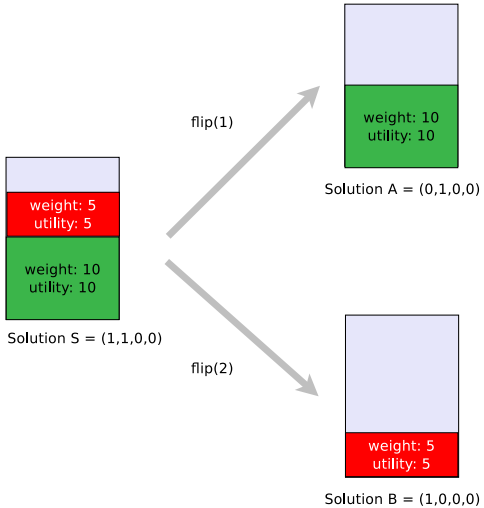


(a) Knapsack with four items: red (1), green (2), blue (3), and yellow (4)

(b) Current solution: (1,1,0,0)

(c) Guiding solution: (0,0,1,1)

- There are four moves in a path leading from S to S^g, because these two solutions differ in all elements.
- There are four possible neighbors in $N(S)$, each of them corresponding to flipping the value of one variable of the current solution S.
- However, the restricted neighborhood $N(S : S^g)$ contains only two feasible solutions.



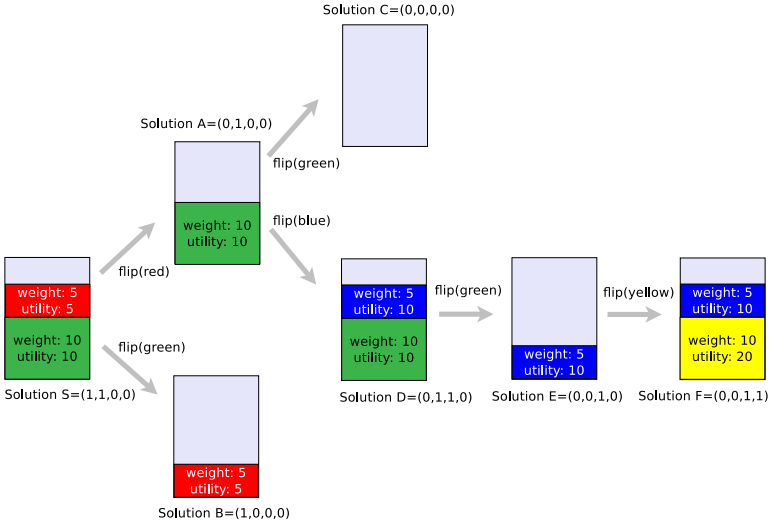
A template for forward path-relinking

- The pseudo-code shows a template of a forward path-relinking algorithm for minimization problems.
- It is assumed that the guiding solution S^g is at least as good as (and possibly better than) the initial solution S^i.
- In most cases, $N(S : S^g)$ does not have to be explicitly computed and stored: instead, its elements may only be implicitly enumerated on-the-fly.
- The best restricted neighbor solution of the current solution is selected at each iteration.
- A local search is applied to the best solution found, because it may not be locally optimal.

begin FORWARD-PR(S^i, S^g);
1 $S \leftarrow S^i$;
2 $S^* \leftarrow S$;
3 $f^* \leftarrow f(S)$;
4 **while** $|N(S : S^g)| \geq 1$ **do**
5 $\quad S \leftarrow \operatorname{argmin}\{f(S') \ : \ S' \in N(S : S^g)\}$;
6 $\quad$ **if** $f(S) < f^*$ **then**
7 $\quad\quad S^* \leftarrow S$;
8 $\quad\quad f^* \leftarrow f(S)$;
9 $\quad$ **end-if**;
10 **end-while**;
11 Apply local search to improve the best solution S^*;
12 **return** $S^*, f(S^*)$;
end FORWARD-PR(S^i, S^g).

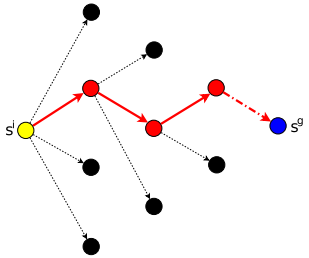
Knapsack problem – Forward path-relinking

Example of forward path-relinking applied to the previous instance of the knapsack problem with four items and capacity 19.



Other implementation strategies for path-relinking

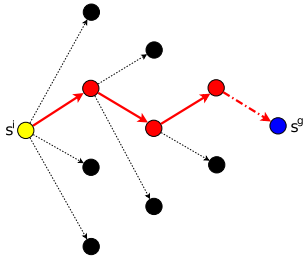
Strategies: Forward path-relinking, backward, back-and-forward, mixed, truncated, greedy randomized adaptive, external, and evolutionary path-relinking, together with their hybrids. All these strategies involve trade-offs between computation time and solution quality.



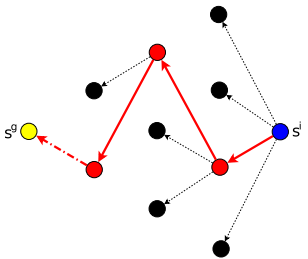
(a) Forward path-relinking: a path is traversed from the initial solution S^i to a guiding solution S^g at least as good as S^i.

Other implementation strategies for path-relinking

Strategies: Forward path-relinking, backward, back-and-forward, mixed, truncated, greedy randomized adaptive, external, and evolutionary path-relinking, together with their hybrids. All these strategies involve trade-offs between computation time and solution quality.



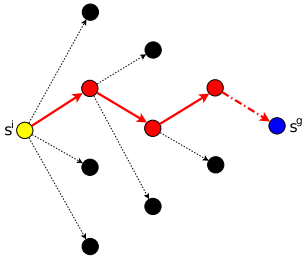
(a) Forward path-relinking: a path is traversed from the initial solution S^i to a guiding solution S^g at least as good as S^i.



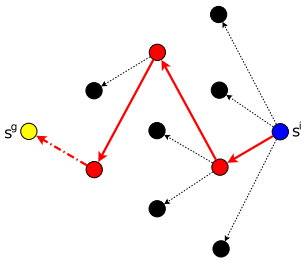
(b) Backward path-relinking: a path is traversed from the initial solution S^i to a guiding solution S^g that is not better than S^i.

Other implementation strategies for path-relinking

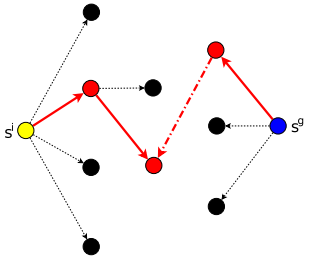
Strategies: Forward path-relinking, backward, back-and-forward, mixed, truncated, greedy randomized adaptive, external, and evolutionary path-relinking, together with their hybrids. All these strategies involve trade-offs between computation time and solution quality.



(a) Forward path-relinking: a path is traversed from the initial solution S^i to a guiding solution S^g at least as good as S^i.



(b) Backward path-relinking: a path is traversed from the initial solution S^i to a guiding solution S^g that is not better than S^i.



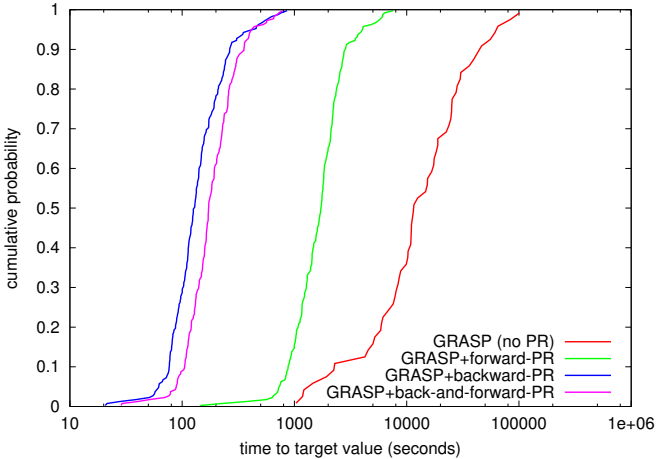
(c) Mixed path-relinking: two subpaths are traversed, one starting at S^i and the other at S^g, which eventually meet in the middle of the trajectory connecting S^i and S^g.

Backward and back-and-forward path-relinking

- In *backward path-relinking*, the guiding solution S^g is not better than the initial solution S^i.
- In *back-and-forward path-relinking*, backward path-relinking is applied first, followed by forward path-relinking.
- Backward path-relinking usually tends to perform better than forward path-relinking, because it is more likely to find an improving solution in the restricted neighborhood of the better solution than in that of the worse.
- Back-and-forward path-relinking does at least as well as either backward or forward path-relinking, but takes about twice as long to compute, since two (usually distinct) paths of the same length are traversed.
- Computational experiments have confirmed that backward path-relinking usually outperforms forward path-relinking in terms of solution quality, while back-and-forward path-relinking finds solutions at least as good as forward or backward path-relinking, but at the expense of longer running times.

Backward and back-and-forward path-relinking

Time-to-target plots for pure GRASP and three variants of GRASP with path-relinking (forward, backward, and back-and-forward) on an instance of a routing problem in private virtual networks:



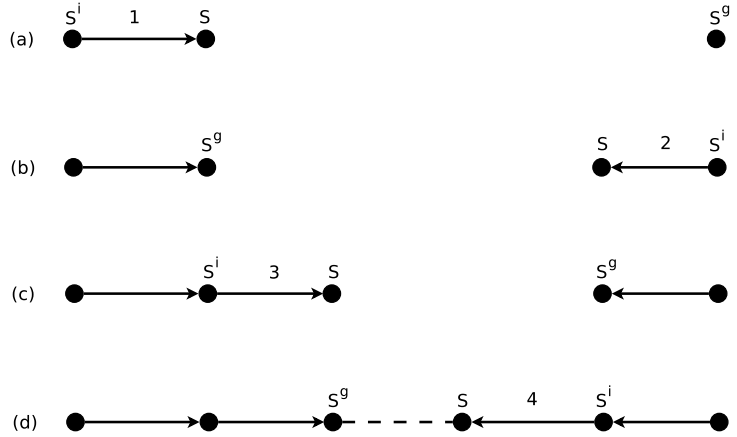
The plots show that GRASP with backward path-relinking outperformed the other path-relinking variants as well as the pure GRASP heuristic, which was the slowest to find a solution whose value is at least as good as the target value.

Mixed path-relinking

- In applying *mixed path-relinking* between two feasible solutions S^i and S^g, the connecting path is explored from both extremities.
- At each iteration of path-relinking, the closest extremity to the new current solution alternates between the original initial solution S^i and the original guiding solution S^g.
- The search behaves as if solutions in two different subpaths were visited alternately.
- These two subpaths meet at some feasible solution in the middle of the trajectory, thus connecting S^i and S^g with a single path.
- In this case, the qualification of a solution as being the initial or the guiding solution is meaningless, since the procedure behaves as if they keep permanently interchanging their role until the end.

Mixed path-relinking

Example: Mixed path-relinking between two solutions S^i and S^g for which the path connecting them is formed by five arcs: numbers above the arrows represent the order in which the moves are performed. Moves alternate between the subpath leaving from the left and the subpath leaving from the right.



Mixed path-relinking

- The pseudo-code shows a template of a mixed path-relinking algorithm between solutions S^i and S^g for minimization problems.
- It is basically the same of algorithm FORWARD-PR, except for lines 10 to 12, in which the direction of the path is reversed by the exchange of the roles of the guiding and current solutions.

begin MIXED-PR(S^i, S^g);
1 $S \leftarrow S^i$;
2 $S^* \leftarrow S$;
3 $f^* \leftarrow f(S^*)$;
4 **while** $|N(S : S^g)| \geq 1$ **do**
5 $S \leftarrow \operatorname{argmin}\{f(S') \ : \ S' \in N(S : S^g)\}$;
6 **if** $f(S) < f^*$ **then**
7 $S^* \leftarrow S$;
8 $f^* \leftarrow f(S)$;
9 **end-if**;
10 $S' \leftarrow S$;
11 $S \leftarrow S^g$;
12 $S^g \leftarrow S'$;
14 **end-while**;
14 Apply local search to improve the best solution S^*;
15 **return** $S^*, f(S^*)$;
end MIXED-PR.

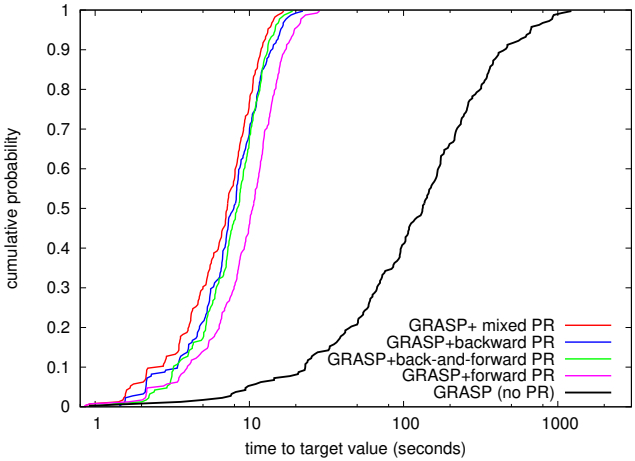
Mixed path-relinking

Observations:

- Back-and-forward path-relinking thoroughly explores both restricted neighborhoods of S^i and S^g.
- The mixed variant explores the entire restricted neighborhood of S^i and all but one solution of the restricted neighborhood of S^g.
- In contrast, forward and backward path-relinking, each of them fully explore only one of the restricted neighborhoods.
- Furthermore, mixed path-relinking explores half as many restricted neighbors as back-and-forward path-relinking and the same number of neighbors as either the backward or forward variants.

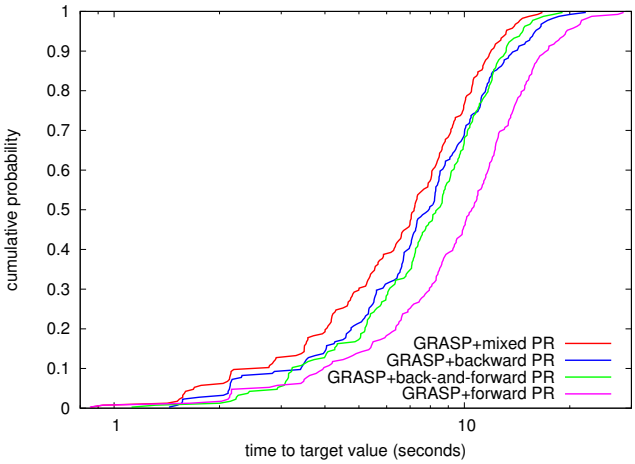
Mixed path-relinking

Time-to-target plots for pure GRASP and four variants of GRASP with path-relinking (forward, backward, back-and-forward, and mixed) on an instance of the 2-path network design problem.



Mixed path-relinking

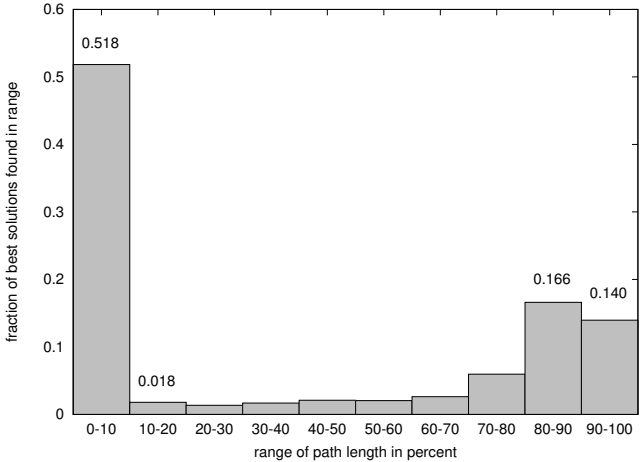
The time-to-target plots show that GRASP with mixed path-relinking has the best run time profile among the variants compared.



Truncated path-relinking

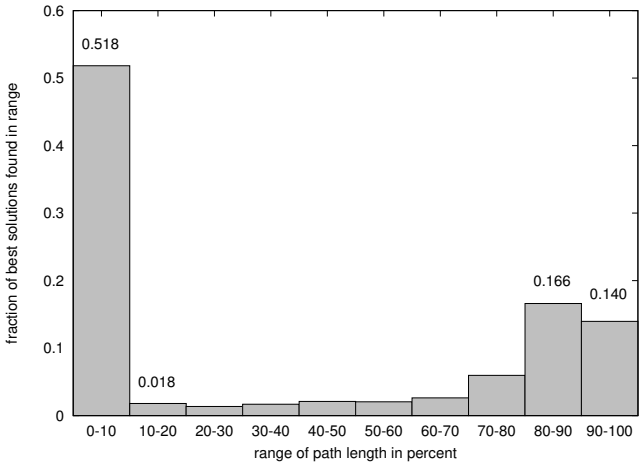
One can expect to see most solutions produced by path-relinking to come from subpaths that are close to either the initial or the guiding solution.

The figure shows the fraction of the best solutions found by GRASP with back-and-forward path-relinking that appear in each range of the path length from the initial to the guiding solutions on two-minute runs over 80 instances of the max-min diversity problem.



Truncated path-relinking

- 54% of the best solutions were found in subpaths that originate at the initial solutions and appear within the first 20% of the total number of moves performed.
- 31% are close to the guiding solutions and appear in the last 20% of the moves performed in each path.



Truncated path-relinking

- Exploring only the subpaths near the extremities often produces solutions as good as those found by exploring the entire path, since there is a higher concentration of better solutions close to the initial and guiding solutions explored by path-relinking.
- It is straightforward to adapt path-relinking to explore only the restricted neighborhoods that are close to the extremities.
- *Truncated path-relinking* can be applied to either forward, backward, backward-and-forward, or mixed path-relinking: instead of exploring the entire path, it just explores a fraction of the path and, consequently, takes a fraction of the running time.

Minimum distance required for path-relinking

Let's assume that assume that we want to connect two locally optimal solutions S^1 and S^2 with path-relinking:

- If S^1 and S^2 differ by only one of their components, then the path directly connects the two solutions and no solution, other than S^1 and S^2, is visited.
  - Since S^1 and S^2 are both local minima, then $f(S^1) \leq f(S)$ for all $S \in N(S^1)$ and $f(S^2) \leq f(S)$ for all $S \in N(S^2)$, where $N(S)$ denotes the neighborhood of solution S.

Minimum distance required for path-relinking

Let's assume that assume that we want to connect two locally optimal solutions S^1 and S^2 with path-relinking:

- If S^1 and S^2 differ by only one of their components, then the path directly connects the two solutions and no solution, other than S^1 and S^2, is visited.
  - Since S^1 and S^2 are both local minima, then $f(S^1) \leq f(S)$ for all $S \in N(S^1)$ and $f(S^2) \leq f(S)$ for all $S \in N(S^2)$, where $N(S)$ denotes the neighborhood of solution S.
- If S^1 and S^2 differ by exactly two moves, then any path between S^1 and S^2 visits exactly one intermediary solution $S \in N(S^1) \cap N(S^2)$. Consequently, solution S cannot be better than either S^1 or S^2.

Minimum distance required for path-relinking

Let's assume that assume that we want to connect two locally optimal solutions S^1 and S^2 with path-relinking:

- If S^1 and S^2 differ by only one of their components, then the path directly connects the two solutions and no solution, other than S^1 and S^2, is visited.
  - Since S^1 and S^2 are both local minima, then $f(S^1) \leq f(S)$ for all $S \in N(S^1)$ and $f(S^2) \leq f(S)$ for all $S \in N(S^2)$, where $N(S)$ denotes the neighborhood of solution S.
- If S^1 and S^2 differ by exactly two moves, then any path between S^1 and S^2 visits exactly one intermediary solution $S \in N(S^1) \cap N(S^2)$. Consequently, solution S cannot be better than either S^1 or S^2.
- If S^1 and S^2 differ by exactly three moves, then any path between them visits two intermediary solutions $S \in N(S^1)$ and $S' \in N(S^2)$ and, consequently, neither S nor S' can be better than both S^1 and S^2.

Minimum distance required for path-relinking

Let's assume that assume that we want to connect two locally optimal solutions S^1 and S^2 with path-relinking:

- If S^1 and S^2 differ by only one of their components, then the path directly connects the two solutions and no solution, other than S^1 and S^2, is visited.
  - Since S^1 and S^2 are both local minima, then $f(S^1) \leq f(S)$ for all $S \in N(S^1)$ and $f(S^2) \leq f(S)$ for all $S \in N(S^2)$, where $N(S)$ denotes the neighborhood of solution S.
- If S^1 and S^2 differ by exactly two moves, then any path between S^1 and S^2 visits exactly one intermediary solution $S \in N(S^1) \cap N(S^2)$. Consequently, solution S cannot be better than either S^1 or S^2.
- If S^1 and S^2 differ by exactly three moves, then any path between them visits two intermediary solutions $S \in N(S^1)$ and $S' \in N(S^2)$ and, consequently, neither S nor S' can be better than both S^1 and S^2.

Consequently, we can discard the application of path-relinking to pairs of solutions differing by less than four moves.

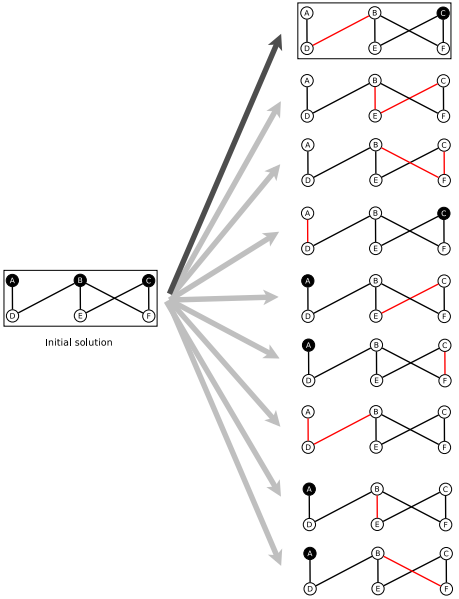
Dealing with infeasibilities in path-relinking

- Consider line 5 of a path-relinking template shown earlier, where we minimize $f(S)$, for $S \in F$.
- This step selects the best restricted neighbor of the current solution as $\text{argmin}\{f(S') \ : \ S' \in N(S : S^g)\}$.
- However, it may occur that all moves from the current solution S lead to infeasible solutions, i.e., $N(S : S^g) = \varnothing$ and the result of the argmin operator is undefined.
- In this situation, path-relinking would have to stop.

begin FORWARD-PR(S^i, S^g);
1 $S \leftarrow S^i$;
2 $S^* \leftarrow S$;
3 $f^* \leftarrow f(S)$;
4 **while** $|N(S : S^g)| \geq 1$ **do**
5 $S \leftarrow \text{argmin}\{f(S') \ : \ S' \in N(S : S^g)\}$;
6 **if** $f(S) < f^*$ **then**
7 $S^* \leftarrow S$;
8 $f^* \leftarrow f(S)$;
9 **end-if**;
10 **end-while**;
11 Apply local search to improve the best solution S^*;
12 **return** $S^*, f(S^*)$;
end FORWARD-PR(S^i, S^g).

Example: Path-relinking for a maximum independent set problem (i.e., seek a set of mutually nonadjacent nodes of maximum cardinality) on a bipartite graph with six nodes.

- The initial solution is $S^i = \{A, B, C\}$ and the guiding solution is $S^g = \{D, E, F\}$.
- The neighborhood is characterized by moves defined as $\mathrm{swap}(out, in)$, where the node *out* is replaced by the node *in* in the solution.

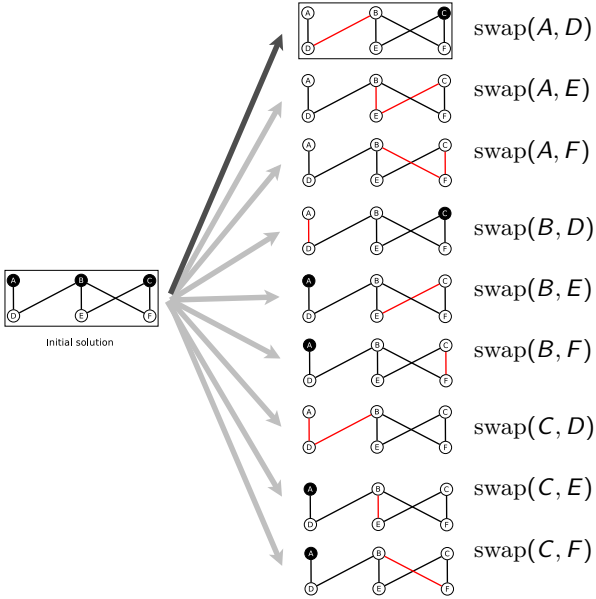


Example: Path-relinking for a maximum independent set problem (i.e., seek a set of mutually nonadjacent nodes of maximum cardinality) on a bipartite graph with six nodes.

- The initial solution is $S^i = \{A, B, C\}$ and the guiding solution is $S^g = \{D, E, F\}$.
- The neighborhood is characterized by moves defined as $\mathrm{swap}(out, in)$, where the node *out* is replaced by the node *in* in the solution.

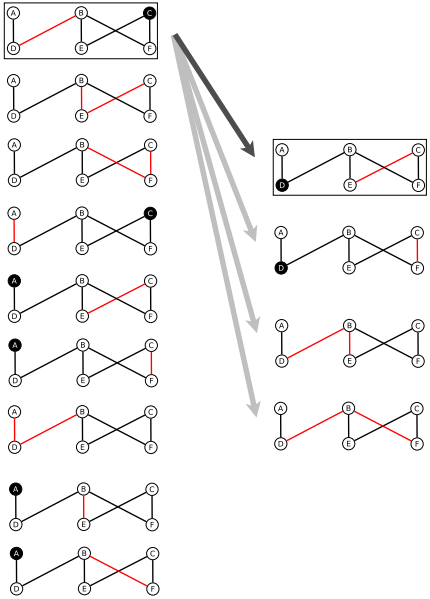
First iteration: all nine restricted neighbors of the initial solution are infeasible.

Possible strategy: Move to a least-infeasible solution (a greedy path-relinking operator).



Example: Path-relinking for a maximum independent set problem (i.e., seek a set of mutually nonadjacent nodes of maximum cardinality) on a bipartite graph with six nodes.

- The current solution is $S = \{B, C, D\}$ and the guiding solution is $S^g = \{D, E, F\}$.
- The neighborhood is characterized by moves defined as $\mathrm{swap}(out, in)$, where the node *out* is replaced by the node *in* in the solution.

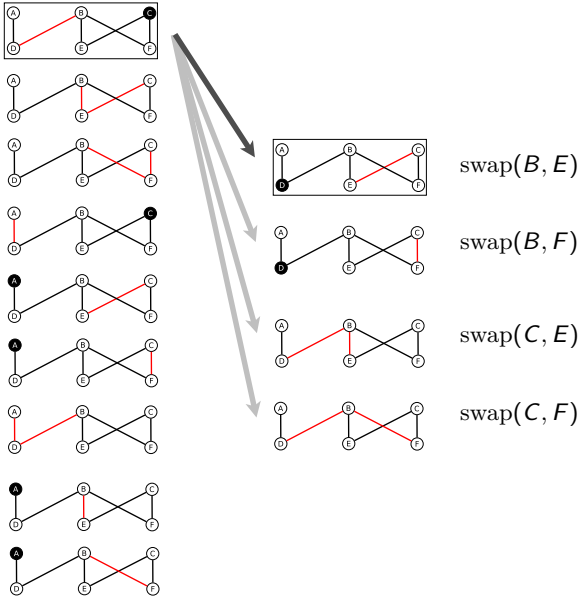


Example: Path-relinking for a maximum independent set problem (i.e., seek a set of mutually nonadjacent nodes of maximum cardinality) on a bipartite graph with six nodes.

- The current solution is $S = \{B, C, D\}$ and the guiding solution is $S^g = \{D, E, F\}$.
- The neighborhood is characterized by moves defined as $\mathrm{swap}(out, in)$, where the node *out* is replaced by the node *in* in the solution.

Second iteration: all four restricted neighbors of the current solution are infeasible.

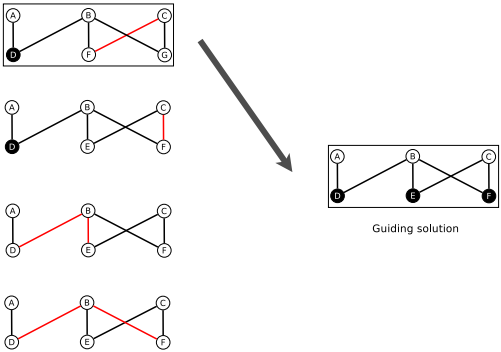
Move again to a least-infeasible solution.



Dealing with infeasibilities in path-relinking

Example: Path-relinking for a maximum independent set problem (i.e., seek a set of mutually nonadjacent nodes of maximum cardinality) on a bipartite graph with six nodes.

- The current solution is $S = \{C, D, E\}$ and the guiding solution is $S^g = \{D, E, F\}$.
- The neighborhood is characterized by moves defined as $\mathrm{swap}(out, in)$, where the node *out* is replaced by the node *in* in the solution.

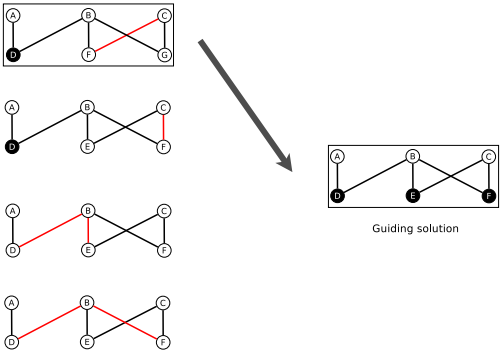


Dealing with infeasibilities in path-relinking

Example: Path-relinking for a maximum independent set problem (i.e., seek a set of mutually nonadjacent nodes of maximum cardinality) on a bipartite graph with six nodes.

- The current solution is $S = \{C, D, E\}$ and the guiding solution is $S^g = \{D, E, F\}$.
- The neighborhood is characterized by moves defined as $\mathrm{swap}(out, in)$, where the node *out* is replaced by the node *in* in the solution.

Third iteration: the path finally reaches the guiding solution.



In this example, all restricted neighbors on all paths from the initial solution to the target solution are infeasible. In general, however, some may be feasible, some infeasible.

Dealing with infeasibilities in path-relinking

In a revised path-relinking operator that allows moves to infeasible solutions, each visited solution S may be in either one of two possible situations:

- At least one move from S leads to a feasible solution, in which case $|N(S : S^g)| \geq 1$.

- All restricted moves lead to infeasible solutions and the restricted neighborhood $N(S : S^g)$ becomes empty before the guiding solution is reached.

Dealing with infeasibilities in path-relinking

In a revised path-relinking operator that allows moves to infeasible solutions, each visited solution S may be in either one of two possible situations:

- At least one move from S leads to a feasible solution, in which case $|N(S : S^g)| \geq 1$.
  - A greedy version of path-relinking selects a move that leads to a least cost feasible neighbor of S.
- All restricted moves lead to infeasible solutions and the restricted neighborhood $N(S : S^g)$ becomes empty before the guiding solution is reached.
  - A greedy version of path-relinking selects a move that leads to an infeasible neighbor of S with minimum infeasibility.

The pseudo-code presents a revised template of a mixed path-relinking procedure for minimization problems, that allows feasible and infeasible moves.

- It is very similar to the template MIXED-PR, with the main difference corresponding to lines 4 to 10.
- Both feasible and infeasible moves are allowed in the neighborhood $N(S)$.
- Line 5 detects if there is at least one move that once applied to S leads to a feasible solution.
- Let by $\mathrm{infeasibility}(S)$ a measure of the degree of infeasibility of a solution S.
- Line 8 selects the best infeasible neighbor of the current solution.

begin MIXED-PR-INFEASIBLE-MOVES(S^i, S^g);
1 $S \leftarrow S^i$;
2 $S^* \leftarrow S$;
3 $f^* \leftarrow f(S^*)$;
4 **while** $|N(S)| > 1$ **do**
5 **if** $N(S : S^g) \neq \varnothing$ **then**
6 $S \leftarrow \mathrm{argmin}\{f(S') \ : \ S' \in N(S : S^g)\}$;
7 **else**
8 $S \leftarrow \mathrm{argmin}\{\mathrm{infeasibility}(S') \ : \ S' \in N(S)\}$;
9 **end-if**;
10 **if** S is feasible **and** $f(S) < f^*$ **then**
11 $S^* \leftarrow S$;
12 $f^* \leftarrow f(S)$;
13 **end-if**;
14 $S' \leftarrow S$;
15 $S \leftarrow S^g$;
16 $S^g \leftarrow S'$;
17 **end-while**;
18 Apply local search to improve the best solution S^*;
19 **return** $S^*, f(S^*)$;
end MIXED-PR-INFEASIBLE-MOVES.

Randomization in path-relinking

- All previously described path-relinking strategies follow a greedy criterion to select the best move at each of their iterations.
- Therefore, path-relinking is limited to exploring a single path from a set of exponentially many paths between any pair of solutions.
- By adding randomization to path-relinking, *greedy randomized adaptive path-relinking* is not constrained to explore a single path.
- Instead of always selecting the move that results in the best solution, a restricted candidate list is constructed with the moves that result in promising solutions with costs in an interval that depends on the values of the best and worst moves, as well as on a parameter in the interval $[0, 1]$. A move is selected at random from this set to produce the next solution in the path.
- By applying this strategy several times to the initial and guiding solutions, several paths can be explored. This strategy is useful when path-relinking is applied more than once to the same pair of solutions as it may occur in evolutionary path-relinking.

External path-relinking and diversification

- So far in this presentation, we have considered variants of path-relinking in which a path in the search space graph $\mathcal{G} = (F, M)$ connects two feasible solutions $S, T \in F$ by progressively introducing in one of them (the initial solution) attributes of the other (the guiding solution).

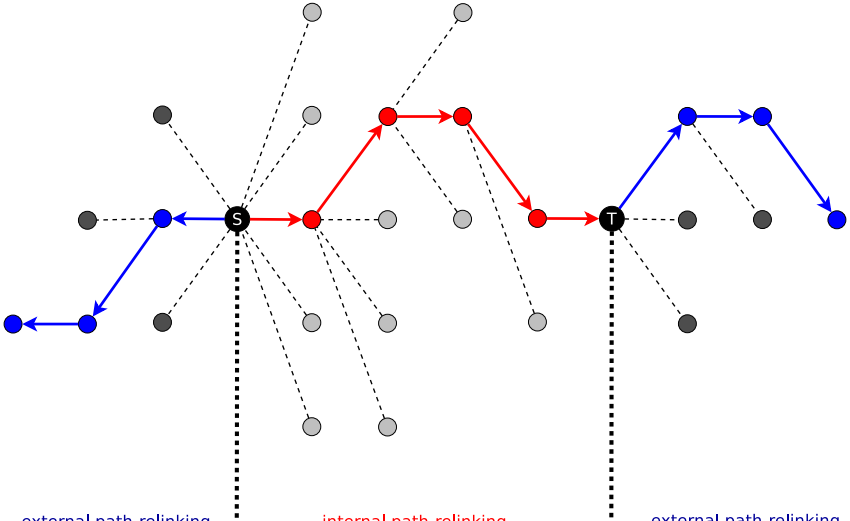
- Since attributes common to both solutions are not changed and all solutions visited belong to a path between the two solutions, we may also refer to this type of path-relinking as *internal path-relinking*.

External path-relinking and diversification

External path-relinking extends any path connecting S and T in $\mathcal{G} = (F, M)$ beyond its extremities.

- To extend such a path beyond S, attributes not present in either S or T are introduced in S.
- Symmetrically, to extend it beyond T, attributes not present in either S or T are introduced in T.
- In its greedy variant, all moves are evaluated and the solution chosen to be next in the path is one with best cost or, in case they are all infeasible, the one with least infeasibility.
- In either direction, the procedure stops when all attributes that do not appear in either S or T have been tested for extending the path.
- Once both paths are complete, local search may be applied to the best solution in each of them.
- The best of the two local minima is returned as the solution produced by the external path-relinking procedure.

External path-relinking and diversification



External path-relinking and diversification

A parallel between internal and external path-relinking:

- Since internal path-relinking works by fixing all attributes common to the initial and guiding solutions and searches for paths between them satisfying this property, it is clearly an **intensification strategy**.
- Contrarily, external path-relinking progressively removes common attributes and replaces them by others that do not appear in either one of the initial or guiding solution.
  - Therefore, it can be seen as a **diversification strategy** which produces solutions increasingly farther from both the initial and the guiding solutions.
  - External path-relinking becomes therefore a tool for search diversification.

Concluding remarks

The material in this talk is taken from

- Chapter 8 – Path-relinking

of our book, *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures* (Resende & Ribeiro, Springer. 2016).