Metaheuristics 2
Global and Local Searches

V-D. Cung & E. Naudin

Local Search: Hill Climbing/Descent Method

- This metaheuristic can be applied to many optimization problems
- A **neighborhood structure has to be defined and used**; the neighborhood of a solution s, noted v(s) is compounded by a set of solutions similar to s (obtained with « simple » modifications of s)
- The method starts with an initial solution.
- At each iteration, the current solution is replaced by a neighborhood solution which could improve the objective function
- The final solution is a **local optimum**

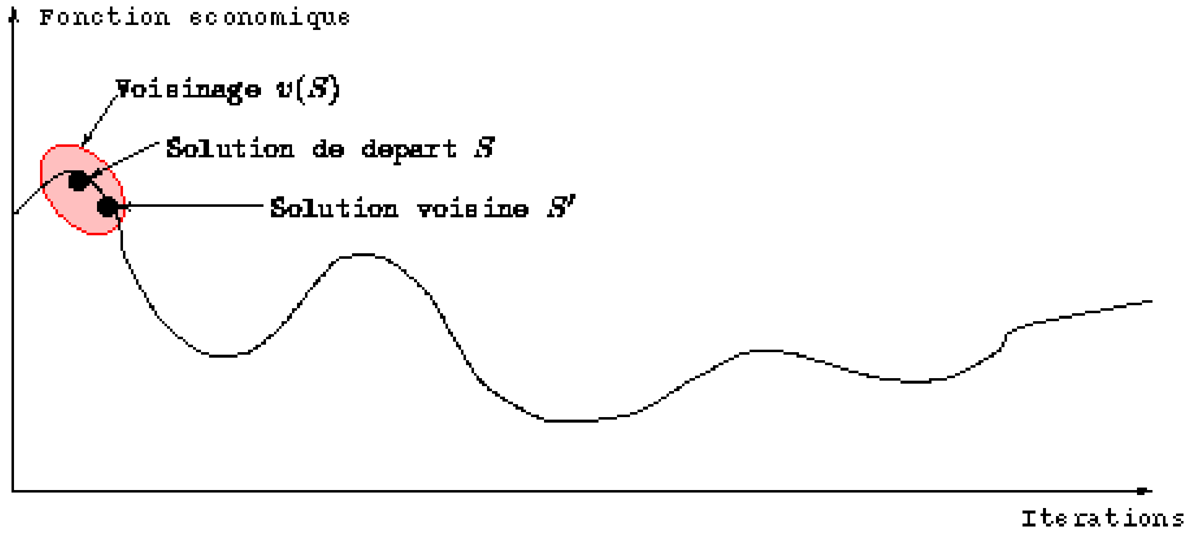


FIG. 1 – *Solution S' dans le voisinage de S*

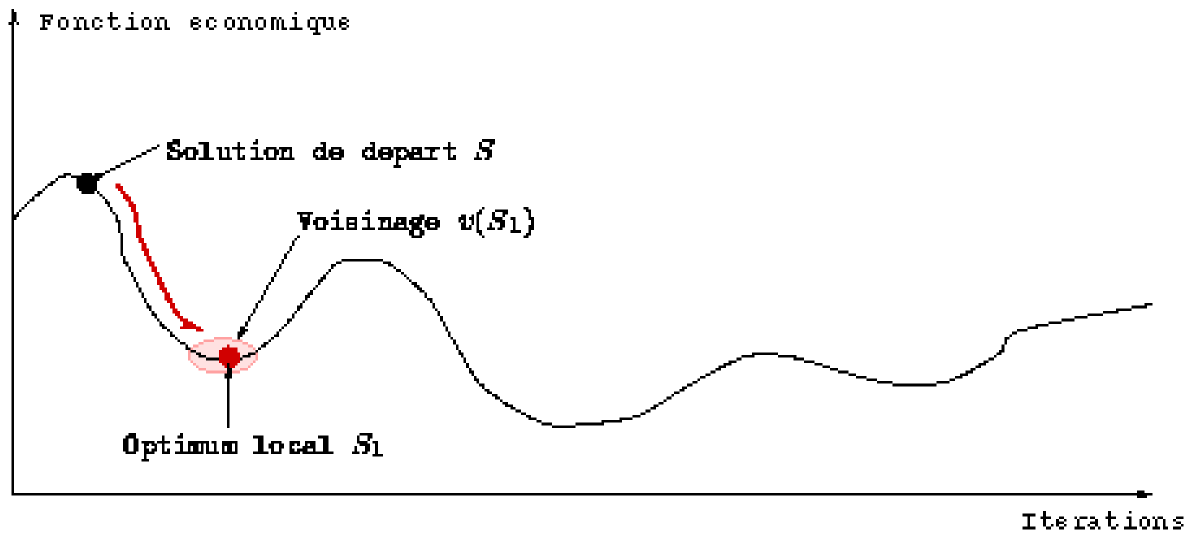


FIG. 2 – *La descente de la recherche locale*

Ex. of Neighborhood (TSP): 2-opt [A. Renard]

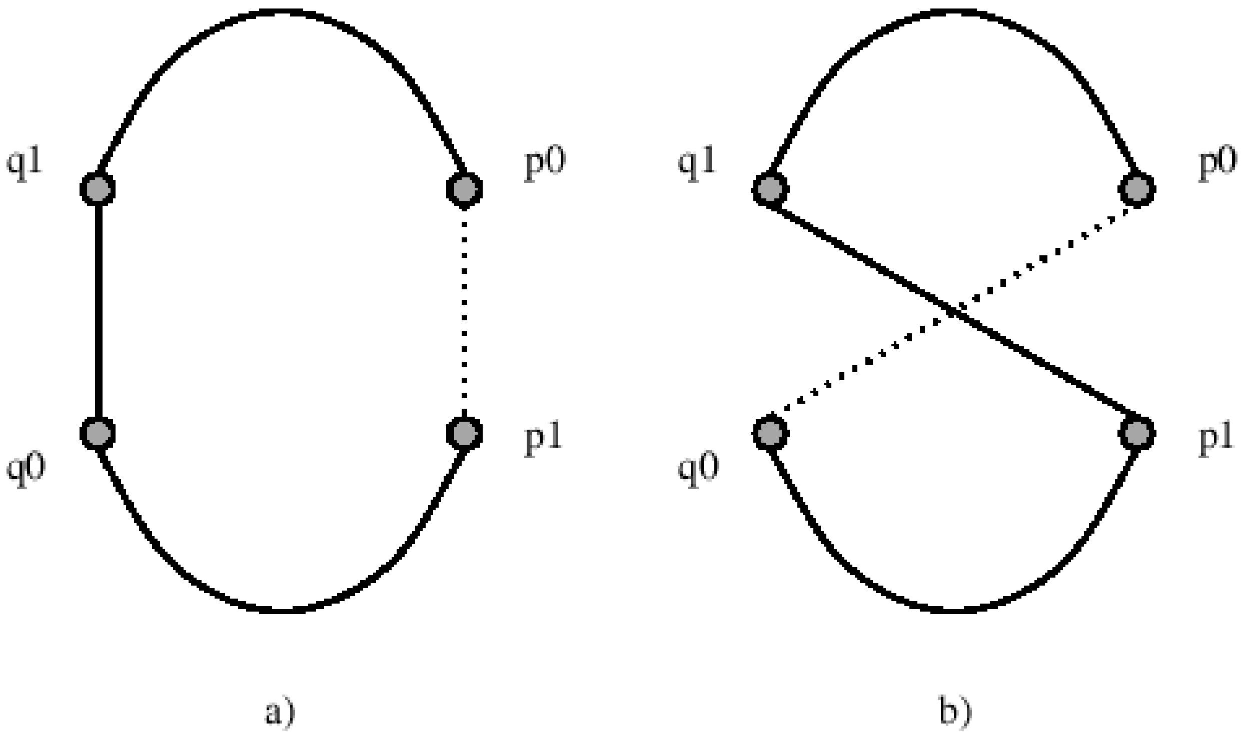
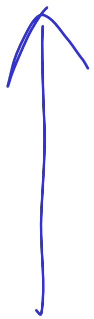


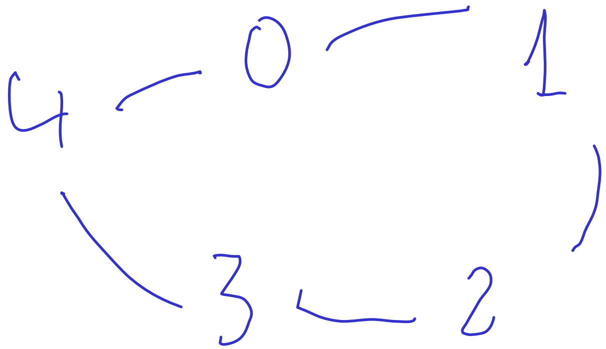
FIG. 2.6 – Principe de l'amélioration 2-opt

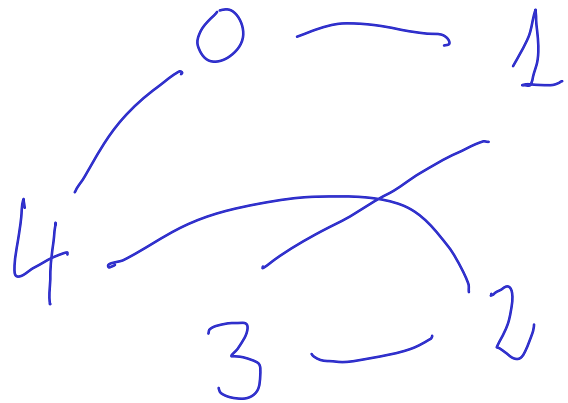
A vector représentation of 2-opt

01234



01324





Size of the neighborhood and the variation of the objective function

- 2-opt: How many neighboring solutions do we have of 01324 ?

- How to compute efficiently the variation of the objective function ?

Neighboring solutions with 2-opt

01324

3-> 01234, 01423, 31024, 03124

2-> 01342, 21304, 02314

1-> 04321, 10324

0-> 41320

$n(n-1)/2 = O(n^2)$

Variation of the objective function with 2-opt

s=01324 f(s)

s'=01234 f(s')

$f(s') = f(s) - c(1,3) - c(2,4) + c(1,2) + c(3,4)$

Delta f (variation) can be computed in $O(1)$ time complexity

Evaluation of all the neighboring solutions: $O(1).O(n^2)=O(n^2)$

Ex. of Neighborhood (TSP): 3-opt

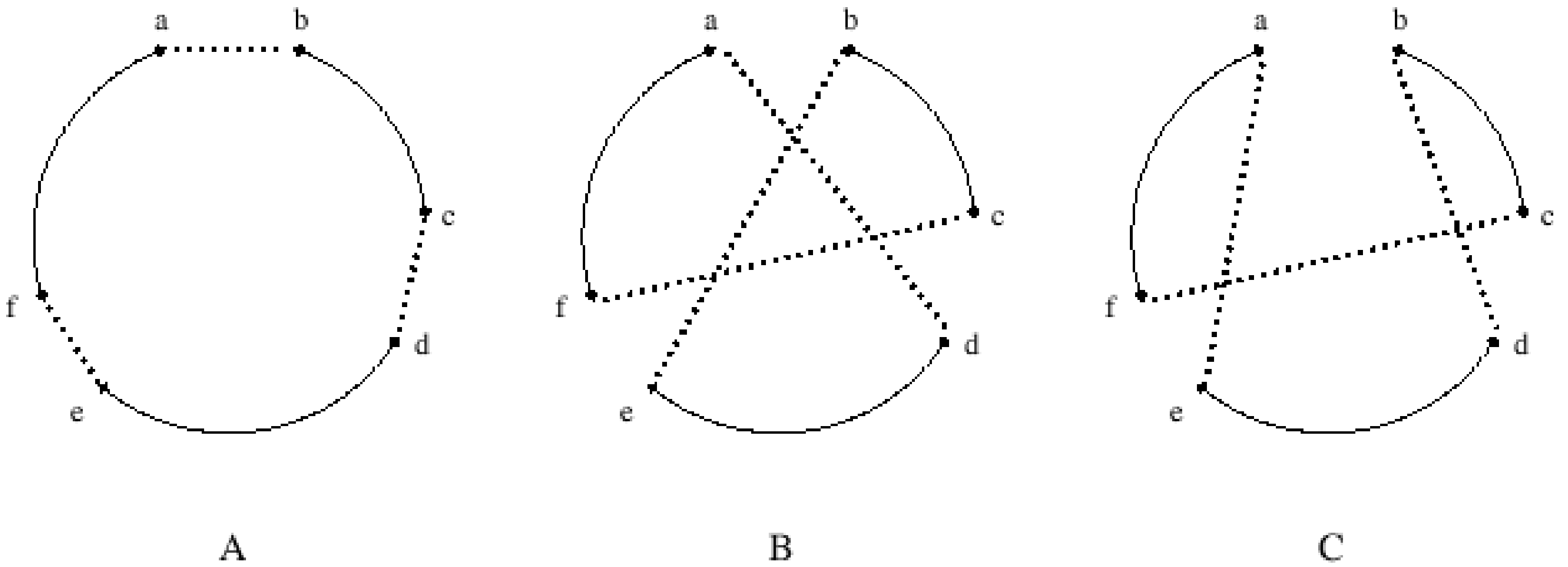


FIG. 2.7 – Principe de l'amélioration 3-opt

Ex. of Neighborhood (TSP): Hyper-opt

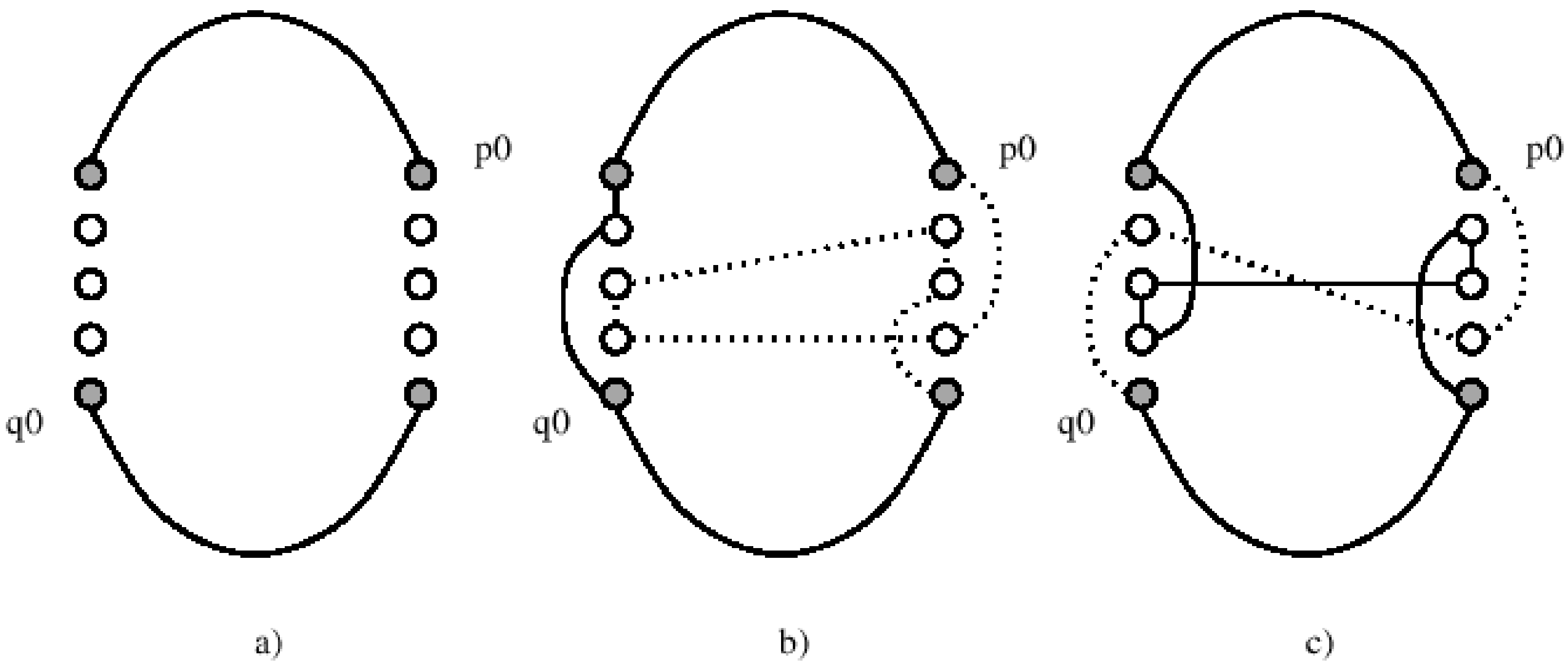


FIG. 2.9 – Hyper-Opt

Ex. of Neighborhood (VRP): Tour Exchange

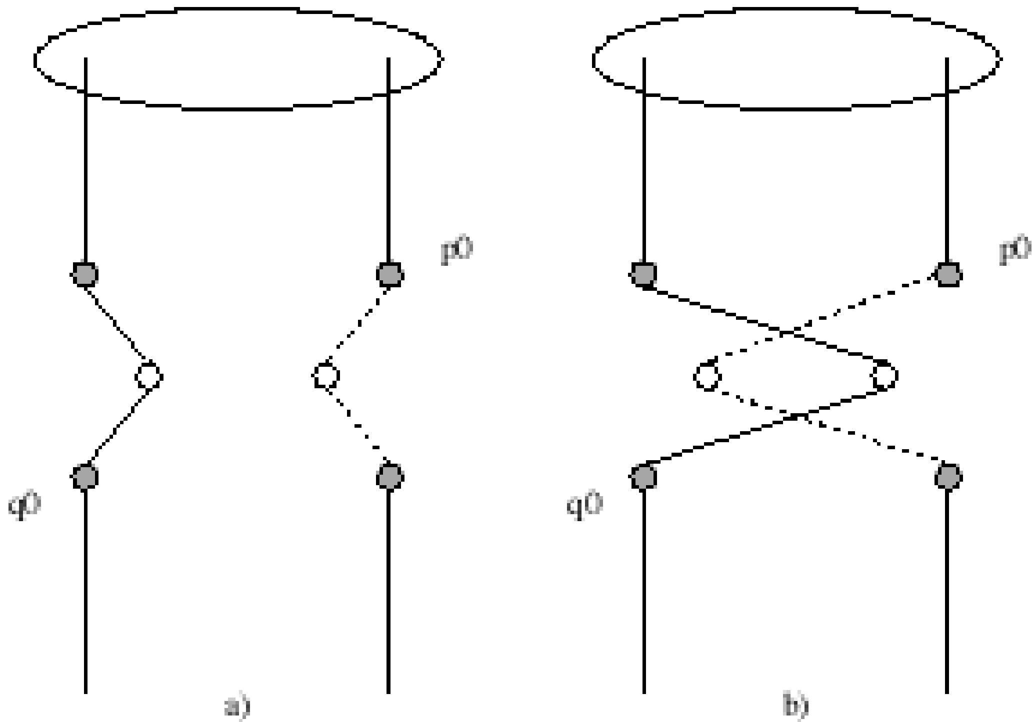


FIG. 2.12 – Amélioration par échange

Ex. of Neighborhood (VRP): Ejection Chains

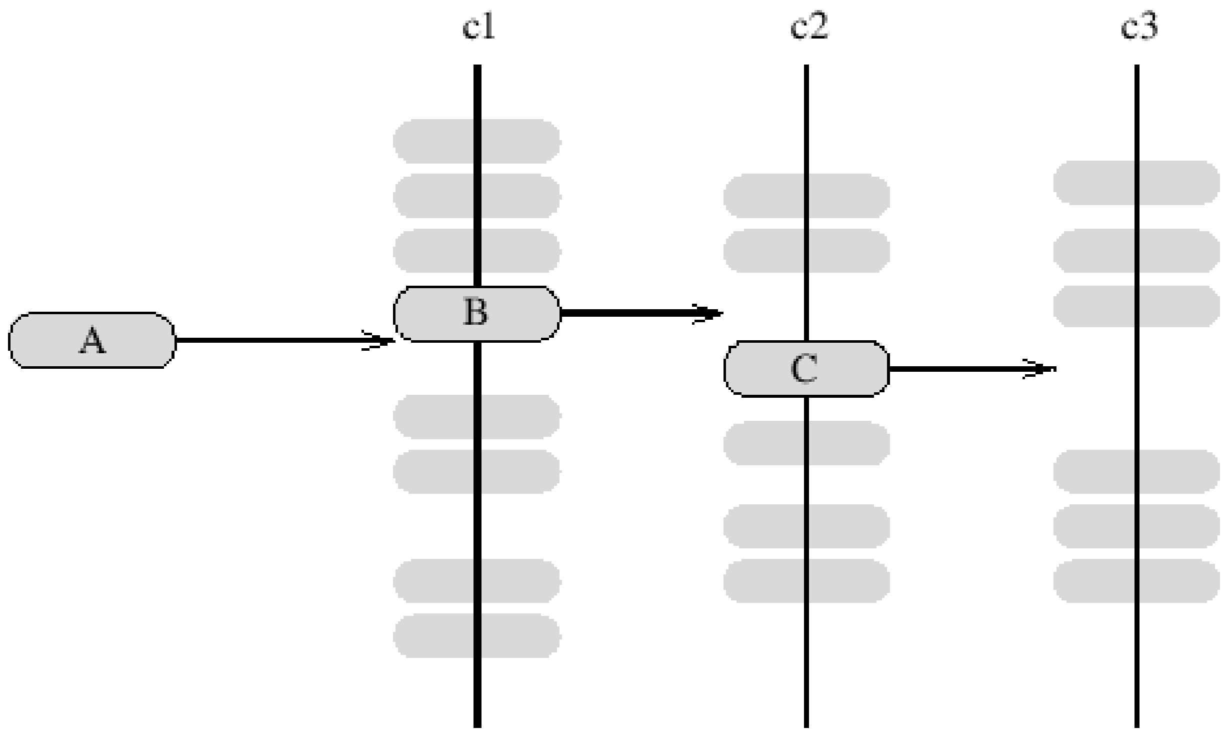


FIG. 5.7 – Fonctionnement d'une chaîne d'éjection

Hill Climbing/Descent procedure

1. Choose an initial solution s;
2. Search s’ in v(s) such that
   f(s’)=min f(x) for all x in v(s);
3. If f(s’)-f(s)>=0 Then END
   Else s=s’ and Go To 2.; End If

Global Searches

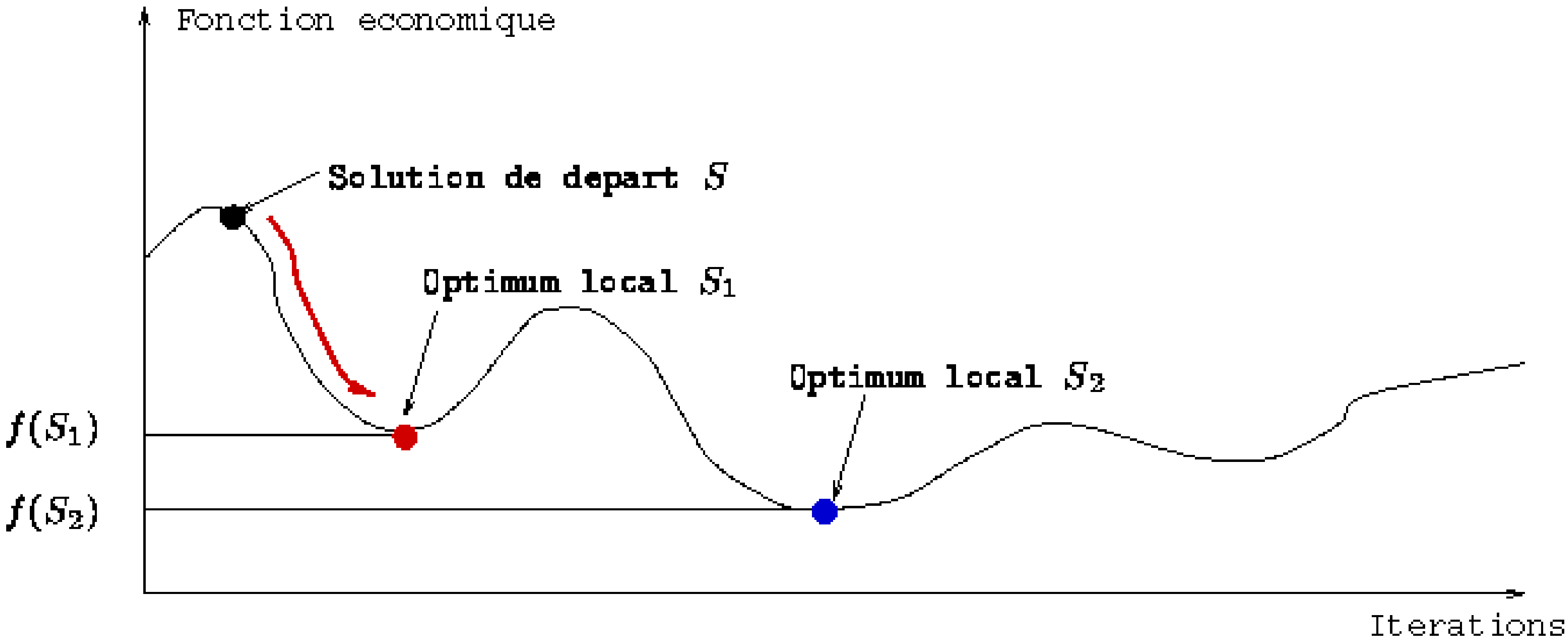


FIG. 3 – *Le piège de la recherche locale*

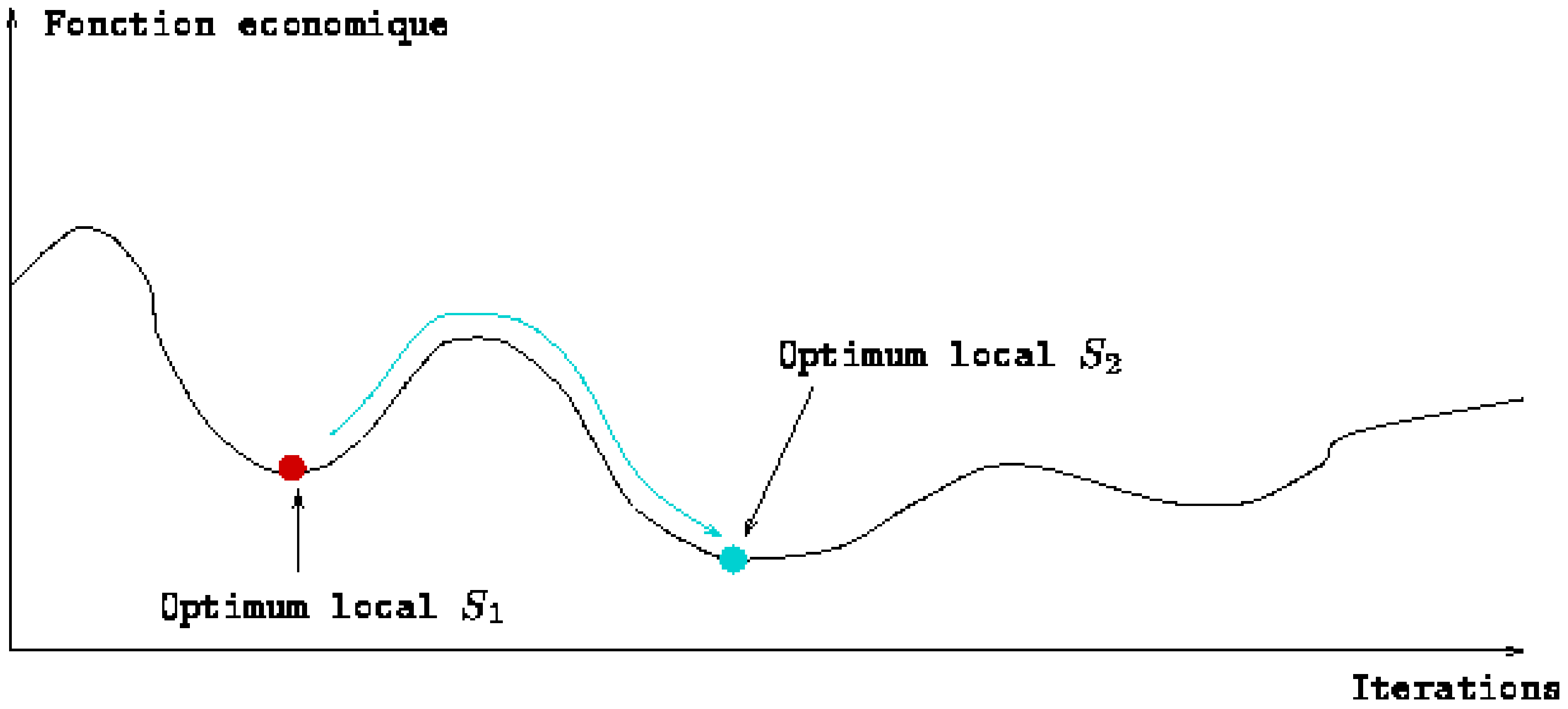


FIG. 4 – *La recherche globale ne s'arrête pas à un seul optimum local*

Simuletad Annealing [Kirkpatrick1983]

1. Choose an initial solution s;
2. Choose a initial temperature T;
3. While the system is not frozen

   While the equilibrium at T is not reached

   - Choose at random s’ in v(s);
   - If f(s’)-f(s)<0 Then s=s’; // acceptation

     Else s=s’ with probability exp(-(f(s’)-f(s))/T);

     End If

   End While

   Reduce temperature T;

   End While

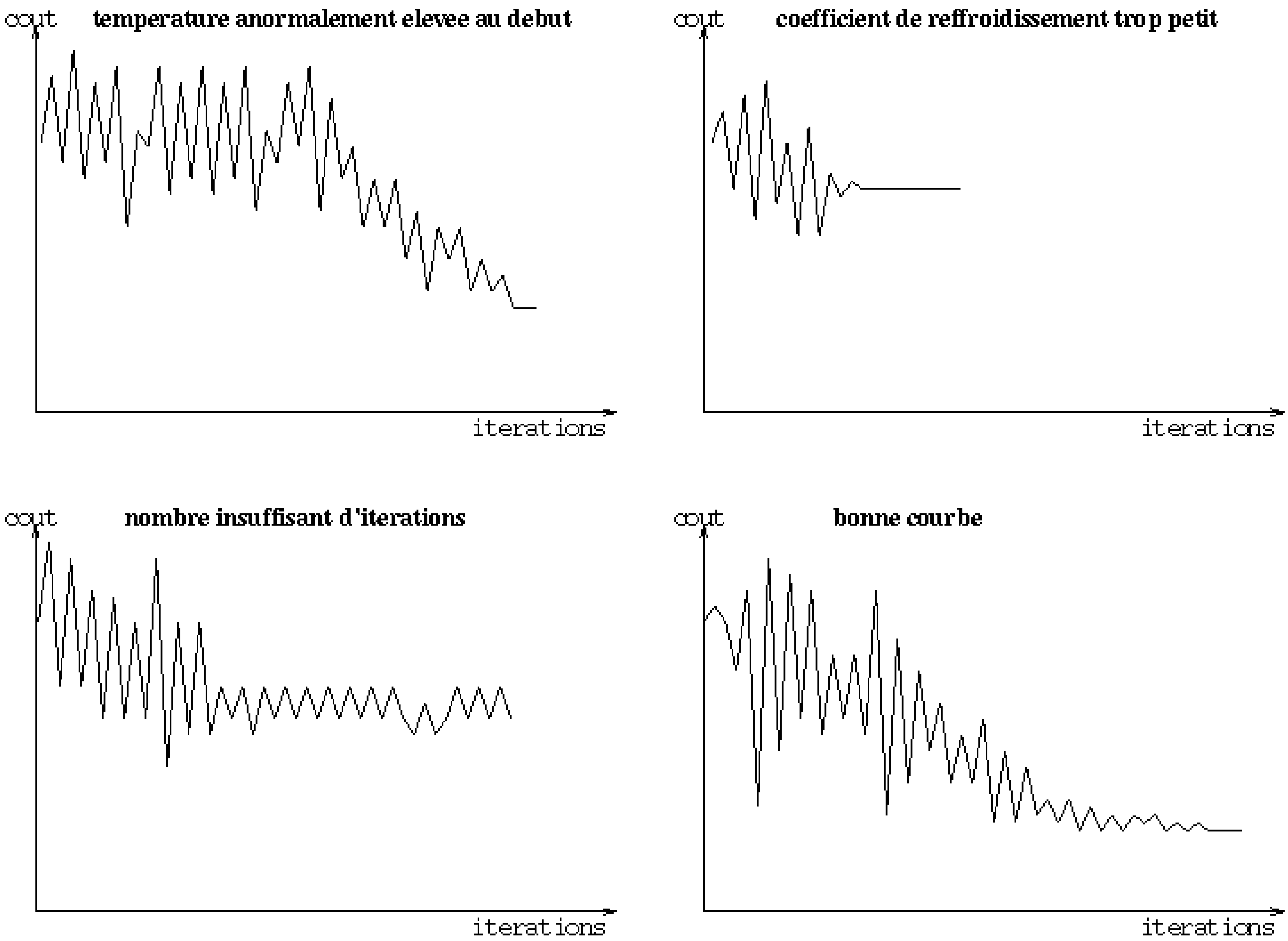


FIG. 5 – *Différents types de courbes pour le recuit simulé*

Tabu Search [F. Glover 1985]

1. Choose an initial solution s;
2. Initialization: s*=s; k=0; T={};
3. While stop criterion is not reached

```
If v(s)\T<>{} Then // there exists a no Tabu move
    k=k+1;
    Search s' in v(s)\T such that f(s')=min f(x) for all x in v(s)\T;
    s=s';
    If f(s)-f(s*)<0 Then s*=s; k=0; End If
End If
Update T; // to forbid reverse moves
End While
```

Variable Neighborhood Search (1/2) [Hansen & M'ladenovic 1998]

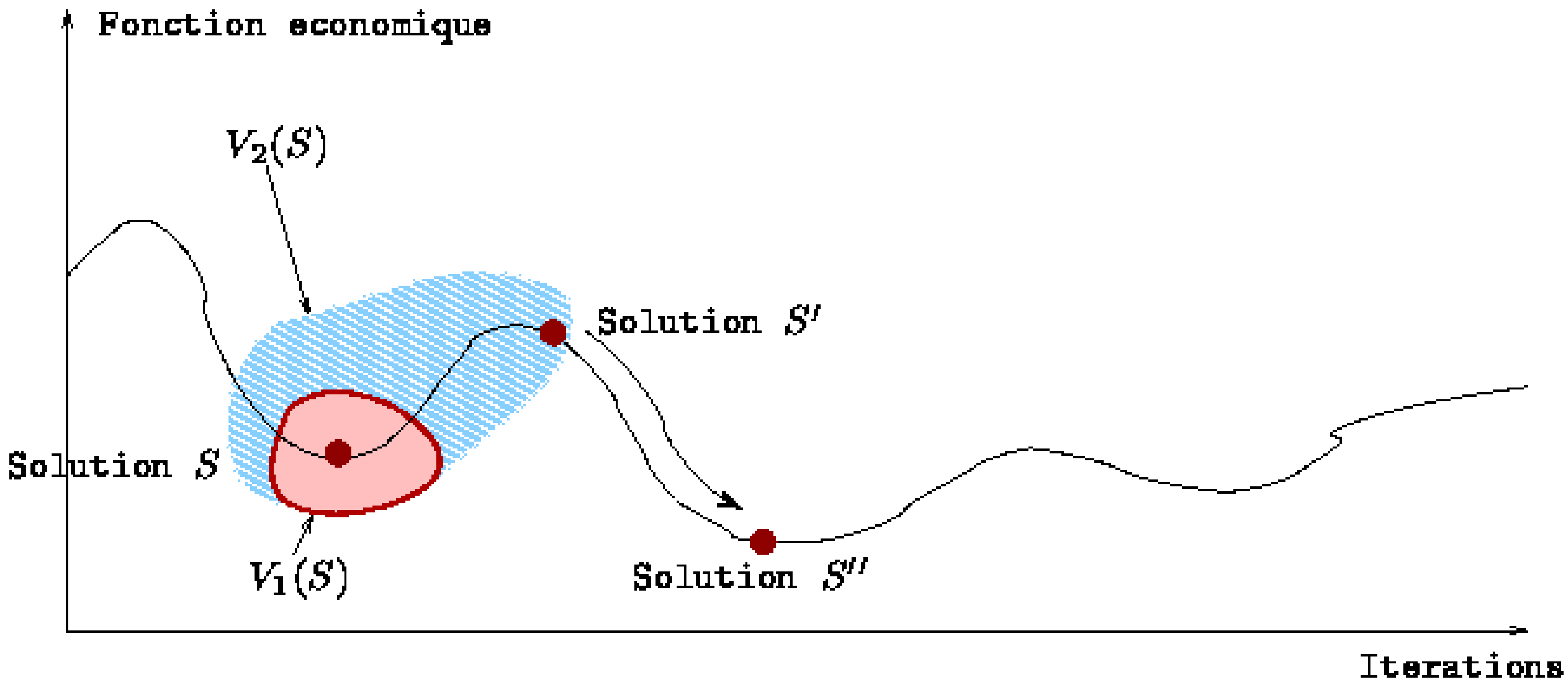


FIG. 6 – *La solution S' est choisie dans le voisinage $V_2(S)$*

Variable Neighborhood Search (2/2)

- Choisir une série de voisinages V_k ($k = 1, \ldots, k_{max}$). $V_k(S)$ est l'ensemble des solutions dans le $k^{\text{ième}}$ voisinage de S,
- trouver une solution initiale S,
- choisir une condition d'arrêt :
  - nombre maximum d'itérations,
  - durée maximale de calculs,
  - nombre maximum d'itérations entre deux améliorations.

Initialisation. Choisir une série de structures de voisinage V_k, ($k = 1, \ldots, k_{max}$), qui sera utilisée dans la recherche ; trouver une solution initiale S ; choisir une condition d'arrêt ;

Répéter

  Pour k = 1 à $k = k_{max}$ **faire**

    (a) **Shaking** : Générer une solution S' au hasard à partir du $k^{\text{ième}}$ voisinage de S ($S' \in V_k(S)$) ;

    (b) **Recherche locale** : appliquer quelques méthodes de recherche locale avec S' comme solution initiale ; noter S'' l'optimum local ainsi obtenu;

    (c) **Bouger ou non**:

      Si cet optimum local est meilleur

        alors, $S \leftarrow S''$ et continuer la recherche avec V_1 ($k \leftarrow 1$)

        Sinon faire $k \leftarrow k + 1$;

  FinPour

Jusqu'àce que la condition d'arrêt soit rencontrée

Variable Neighborhood Descent

Initialisation. Choisir une série de structures de voisinage V'_k, $(k = 1, \ldots, k'_{max})$, qui sera utilisée dans la descente ; trouver une solution initiale S ;

Répéter

  Pour k = 1 à $k = k'_{max}$ **faire**

    Exploration du voisinage : Trouver le meilleur voisin S' de S ($S' \in V'_k(S)$) ;

    Bouger ou non :

      Si la solution S' ainsi obtenue est meilleure que S

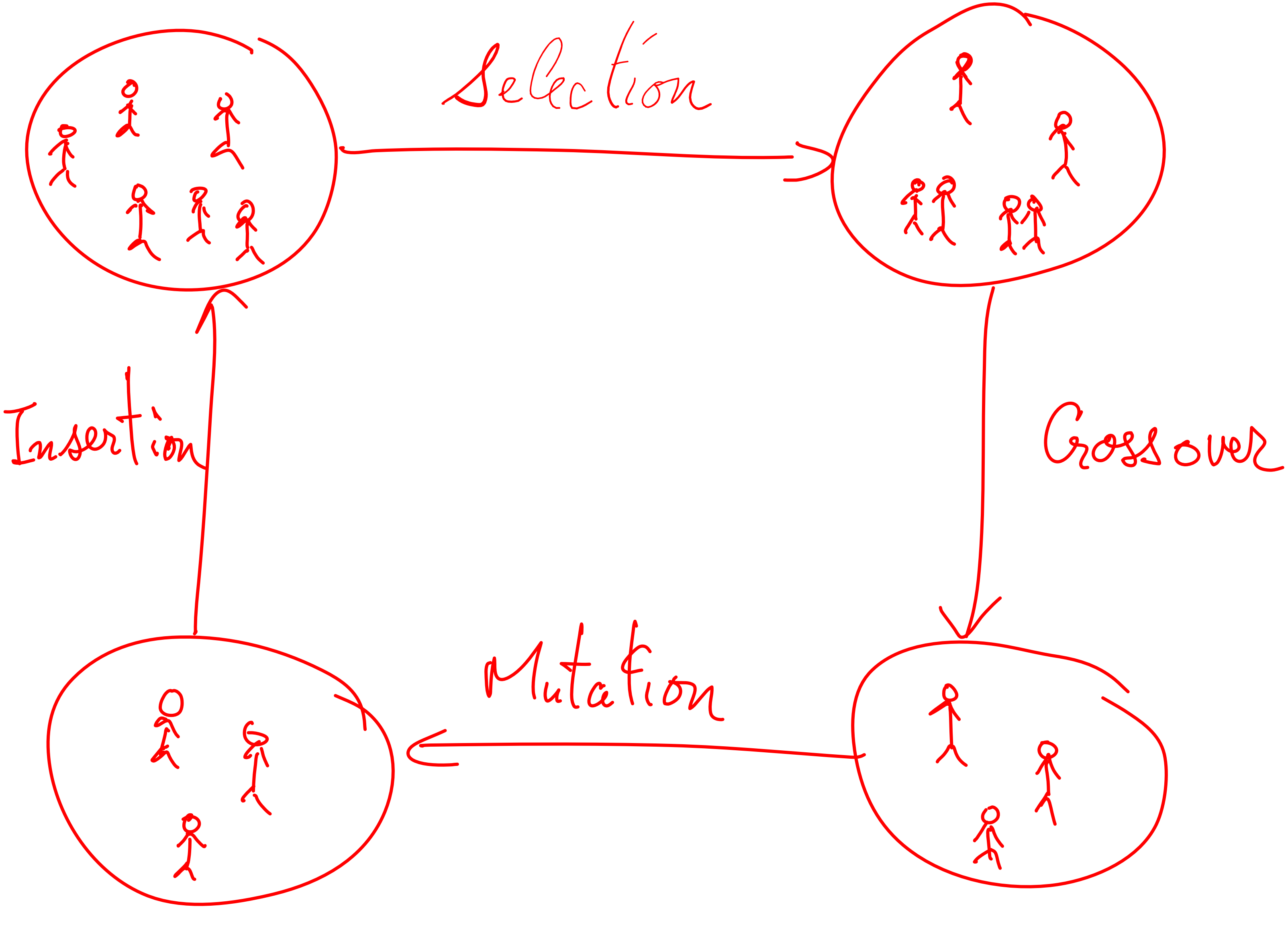
        alors $S \leftarrow S'$

        Sinon faire $k \leftarrow k + 1$;

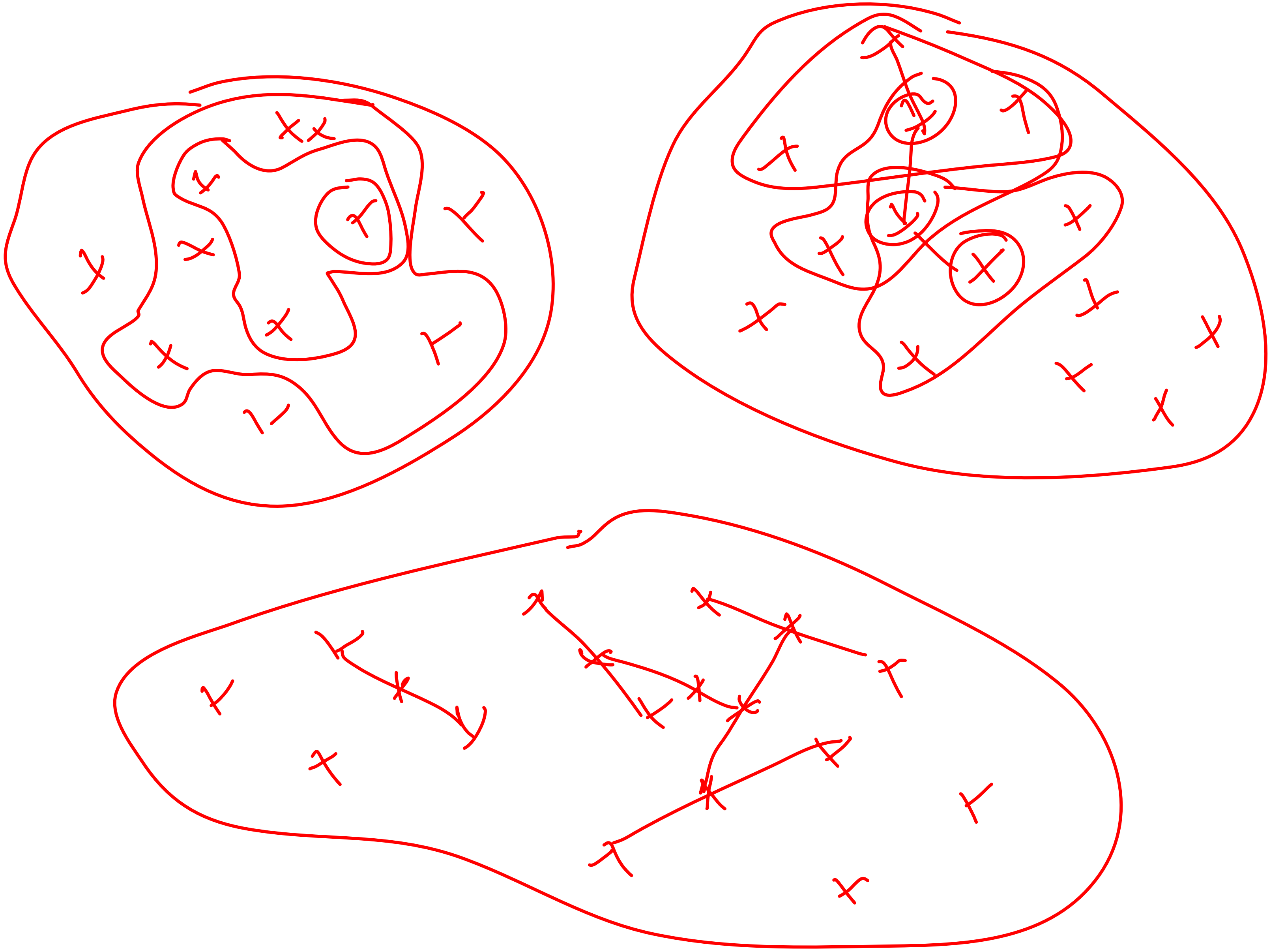
  FinPour

Jusqu'à ce qu'il n'y ait pas d'amélioration

Population Based Algorithms



Regards to the Search Space



Ideas to keep in mind (1/2)

- Guided exploration by Delta (« derivative ») of the objective function
- Intensification & diversification:
  - Long and Short term memory
  - Temperature and randomness
  - Various neighborhood
  - Combination of methods with greedy
- Arbitrary relative or absolute stop criteria

Ideas to keep in mind (2/2)

- Delta quick to compute, if possible avoid re-evaluating completely the solutions
- Neighborhood structures simple to generate, if possible in polynomial time, otherwise restriction could be used to reduce to subsets
- Time and solution quality comparisons: either fixing time, or fixing solution quality to reach

Time Complexity of Metaheuristics

- Each iteration should be in polynomial time P_n
- I_n is the number of iterations, not easy to be bounded or estimated and used to not be polynomial in metaheuristics, but it is limited with arbitrary stop criteria
- $O(I_n.P_n)$

- Epsilon-approx. used to have $O(n^{1/epsilon})$, but when epsilon is getting small, the time complexity could become huge

Solution Quality

- Let us denote H(d) the value of the solution found by the heuristic H on data d and Opt(d) the one by an optimal method on the same data d
- RH(d)=H(d)/Opt(d) is then the relative performance of heuristic H
- RH(d) >=1 for a minimisation problem
- Some authors prefer to compute distance in % with the formula 100.(RH(d)-1)