R5.A.11 Méthodes d'optimisation

Thibault Godin

IUT de Vannes Informatique

Optimisation : recherche de minimum

Beaucoup de problèmes d'optimisation se placent dans la catégorie "minimiser/maximiser une fonction" (souvent un coût/profit ou une énergie)

Optimisation : recherche de minimum

Beaucoup de problèmes d'optimisation se placent dans la catégorie "minimiser/maximiser une fonction" (souvent un coût/profit ou une énergie)

Exemple :

- Minimiser le temps passer à réviser les maths pour avoir 12/20
- Minimiser la distance parcourue en voiture
- Problème d'emploi du temps : on fait une fonction de coût/énergie qui pénalise fortement les chevauchement et faiblement les contraintes
- Optimiser son deck d'hearthstone

Optimisation : recherche de minimum

Beaucoup de problèmes d'optimisation se placent dans la catégorie "minimiser/maximiser une fonction" (souvent un coût/profit ou une énergie)

Exemple :

- Minimiser le temps passer à réviser les maths pour avoir 12/20
- Minimiser la distance parcourue en voiture
- Problème d'emploi du temps : on fait une fonction de coût/énergie qui pénalise fortement les chevauchement et faiblement les contraintes
- Optimiser son deck d'hearthstone

On passe généralement de la mini$^{\text{misation}}$ à la maxi$^{\text{misation}}$ en prenant l'opposé ou l'inverse de la fonction objectif

Rappel BUT2 : Gradient

Soit $f : \mathbb{R} \to \mathbb{R}$ dérivable. Si f admet un minimum ou un maximum local en x^* alors f' s'annule en x^*

$f : \mathbb{R} \to \mathbb{R} : f(a) = a^2 + 1.$

a^* ?

Rappel BUT2 : Gradient

Soit $f : \mathbb{R} \to \mathbb{R}$ dérivable. Si f admet un minimum ou un maximum local en x^* alors f' s'annule en x^*

$f : \mathbb{R} \to \mathbb{R} : f(a) = a^2 + 1.$

a^* ? $\rightsquigarrow$ gradient

$a_{k+1} = a_k - \delta \nabla f(a)$ où δ est le pas et $\nabla f(a) = f'(a) = 2a$.

Rappel BUT2 : Gradient

Soit $f : \mathbb{R} \to \mathbb{R}$ dérivable. Si f admet un minimum ou un maximum local en x^* alors f' s'annule en x^*

$f : \mathbb{R} \to \mathbb{R} : f(a) = a^2 + 1.$

a^* ? $\rightsquigarrow$ gradient

$a_{k+1} = a_k - \delta \nabla f(a)$ où δ est le pas et $\nabla f(a) = f'(a) = 2a$.

$a_{k+1} = a_k - 2\delta a_k.$

Rappel BUT2 : Gradient

Soit $f : \mathbb{R} \to \mathbb{R}$ dérivable. Si f admet un minimum ou un maximum local en x^* alors f' s'annule en x^*

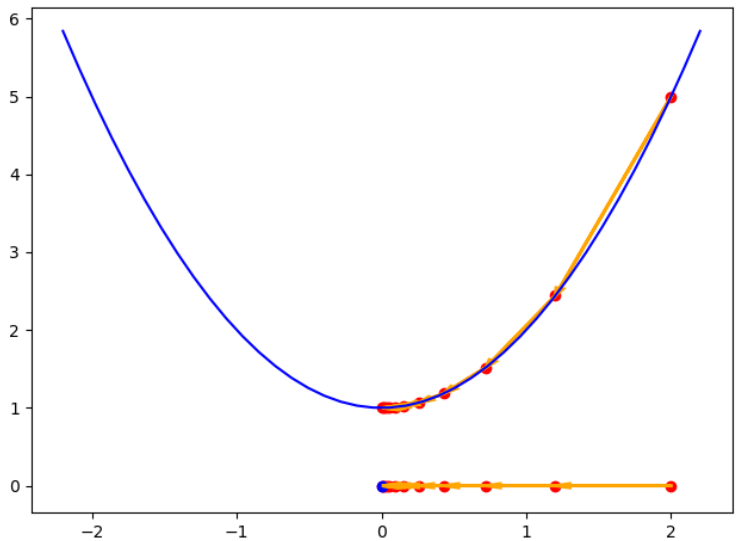
$f : \mathbb{R} \to \mathbb{R} : f(a) = a^2 + 1.$

a^* ? $\rightsquigarrow$ gradient

$a_{k+1} = a_k - \delta \nabla f(a)$ où δ est le pas et $\nabla f(a) = f'(a) = 2a$.

$a_{k+1} = a_k - 2\delta a_k.$

$\delta = 0.2 \quad a_0 = 2$



Rappel BUT2 : Gradient

Soit $f : \mathbb{R} \to \mathbb{R}$ dérivable. Si f admet un minimum ou un maximum local en x^* alors f' s'annule en x^*

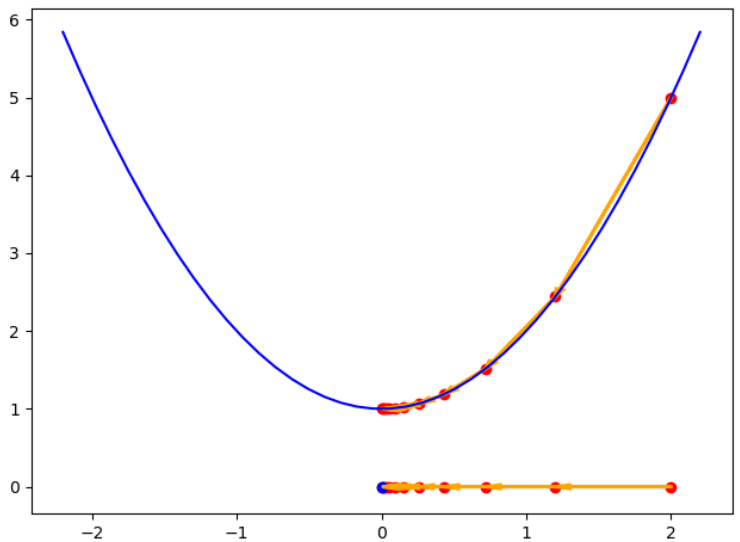
$f : \mathbb{R} \to \mathbb{R} : f(a) = a^2 + 1.$

a^* ? $\rightsquigarrow$ gradient

$a_{k+1} = a_k - \delta \nabla f(a)$ où δ est le pas et $\nabla f(a) = f'(a) = 2a.$

$a_{k+1} = a_k - 2\delta a_k.$

$\delta = 0.2 \quad a_0 = 2$



$\delta = 0.2$ $a_0 = 2.$

k	a_k	$f'(a_k) = \nabla f(a_k)$	$f(a_k)$
0	2	4	5
1	1.2	2.4	2.44
2	0.72	1.44	1.5184
⋮	⋮	⋮	⋮
9	0.020	0.040	1.0004
10	0.012	0.024	1.0001

Rappel BUT2 : Gradient

Soit $f : \mathbb{R} \to \mathbb{R}$ dérivable. Si f admet un minimum ou un maximum local en x^* alors f' s'annule en x^*

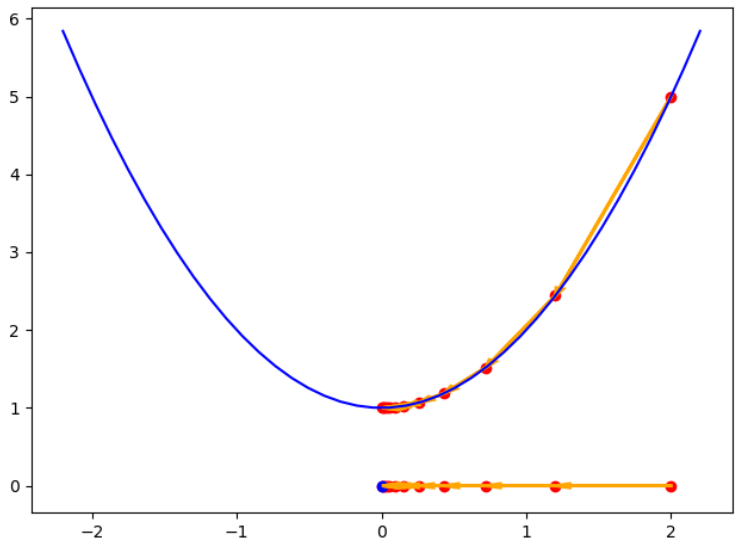
$f : \mathbb{R} \to \mathbb{R} : f(a) = a^2 + 1.$

a^* ? $\rightsquigarrow$ gradient

$a_{k+1} = a_k - \delta \nabla f(a)$ où δ est le pas et $\nabla f(a) = f'(a) = 2a.$

$a_{k+1} = a_k - 2\delta a_k.$

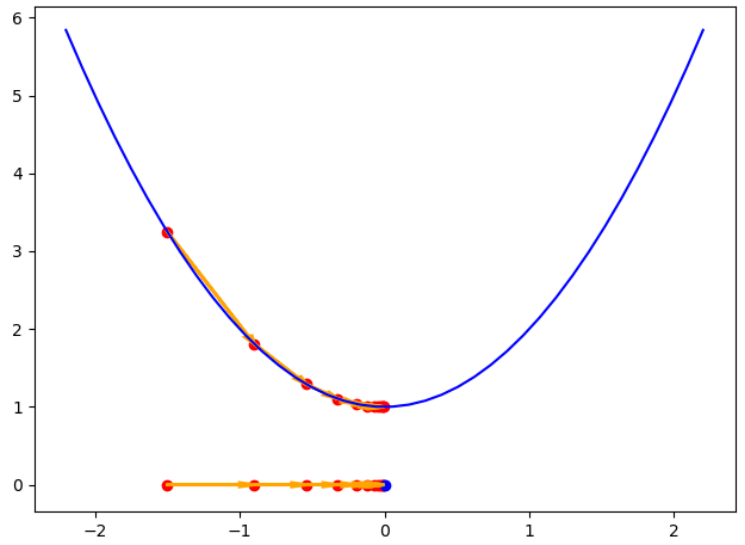
$\delta = 0.2 \quad a_0 = 2$



$\delta = 0.2 \quad a_0 = 2.$

k	a_k	$f'(a_k) = \nabla f(a_k)$	$f(a_k)$
0	2	4	5
1	1.2	2.4	2.44
2	0.72	1.44	1.5184
⋮	⋮	⋮	⋮
9	0.020	0.040	1.0004
10	0.012	0.024	1.0001

$\delta = 0.2 \quad a_0 = -1.5$



Optimisation, minimum et dérivées

$$\frac{\partial f}{\partial x}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h} \qquad \frac{\partial f}{\partial y}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0, y_0 + h) - f(x_0, y_0)}{h}.$$

Optimisation, minimum et dérivées

$$\frac{\partial f}{\partial x}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h} \qquad \frac{\partial f}{\partial y}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0, y_0 + h) - f(x_0, y_0)}{h}.$$

on fait "comme si" f était une fonction d'une variable et on dérive par rapport à celle-ci.

Optimisation, minimum et dérivées

$$\frac{\partial f}{\partial x}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h} \qquad \frac{\partial f}{\partial y}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0, y_0 + h) - f(x_0, y_0)}{h}.$$

on fait "comme si" f était une fonction d'une variable et on dérive par rapport à celle-ci.

On vectorise souvent : $grad(f)(x_0, y_0) = \nabla f(x_0, y_0) = \left(\frac{\partial f}{\partial x}(x_0, y_0), \frac{\partial f}{\partial y}(x_0, y_0) \right)$

Optimisation, minimum et dérivées

$$\frac{\partial f}{\partial x}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h} \qquad \frac{\partial f}{\partial y}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0, y_0 + h) - f(x_0, y_0)}{h}.$$

on fait "comme si" f était une fonction d'une variable et on dérive par rapport à celle-ci.

On vectorise souvent : $grad(f)(x_0, y_0) = \nabla f(x_0, y_0) = \left(\frac{\partial f}{\partial x}(x_0, y_0), \frac{\partial f}{\partial y}(x_0, y_0) \right)$

Soit $f : \mathbb{R}^2 \to \mathbb{R}$. Si f admet un minimum ou un maximum local en (x^*, y^*) alors[a] le gradient est le vecteur nul en ce point, autrement dit :

$$\frac{\partial f}{\partial x}(x^*, y^*) = 0 \quad \text{et} \quad \frac{\partial f}{\partial y}(x^*, y^*) = 0.$$

a. C'est même une équivalence si la dérivée de f est continue et que f est convexe

Optimisation, minimum et dérivées

$$\frac{\partial f}{\partial x}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h} \quad \frac{\partial f}{\partial y}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0, y_0 + h) - f(x_0, y_0)}{h}.$$

on fait "comme si" f était une fonction d'une variable et on dérive par rapport à celle-ci.

On vectorise souvent : $grad(f)(x_0, y_0) = \nabla f(x_0, y_0) = \left(\frac{\partial f}{\partial x}(x_0, y_0), \frac{\partial f}{\partial y}(x_0, y_0) \right)$

Soit $f : \mathbb{R}^2 \to \mathbb{R}$. Si f admet un minimum ou un maximum local en (x^*, y^*) alors[a] le gradient est le vecteur nul en ce point, autrement dit :

$$\frac{\partial f}{\partial x}(x^*, y^*) = 0 \quad \text{et} \quad \frac{\partial f}{\partial y}(x^*, y^*) = 0.$$

a. C'est même une équivalence si la dérivée de f est continue et que f est convexe

Même chose avec plus de dimensions/variables/paramètres

Exemple

$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & a^2+b^2 \end{array}$$

Exemple

$$
\begin{aligned}
f: \quad \mathbb{R}^2 &\longrightarrow \mathbb{R} \\
(a, b) &\longmapsto a^2 + b^2
\end{aligned}
$$

$$\nabla f(a, b) = (2a, 2b)$$

Exemple

$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & a^2+b^2 \end{array}$$

$\nabla f(a,b) = (2a, 2b)$

$a^* = b^* = 0$

$\rightsquigarrow$ minimum en $(0,0)$

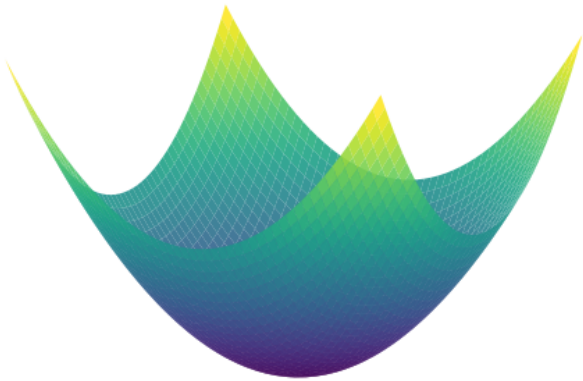
Exemple

$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & a^2+b^2 \end{array}$$

$\nabla f(a,b) = (2a, 2b)$

$a^* = b^* = 0$

$\rightsquigarrow$ minimum en $(0,0)$



$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & -a^2+b^2 \end{array}$$

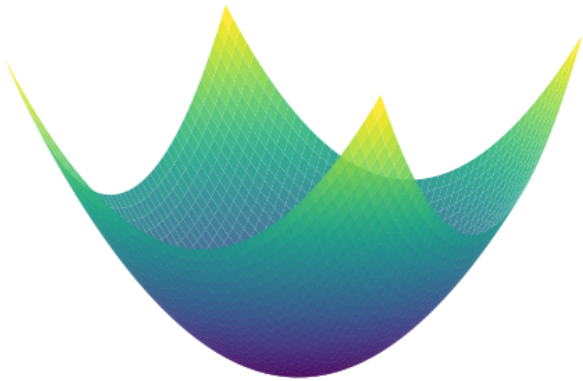
Exemple

$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & a^2+b^2 \end{array}$$

$\nabla f(a,b) = (2a, 2b)$

$a^* = b^* = 0$

$\rightsquigarrow$ minimum en $(0,0)$



$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & -a^2+b^2 \end{array}$$

$\nabla f(a,b) = (-2a, 2b)$

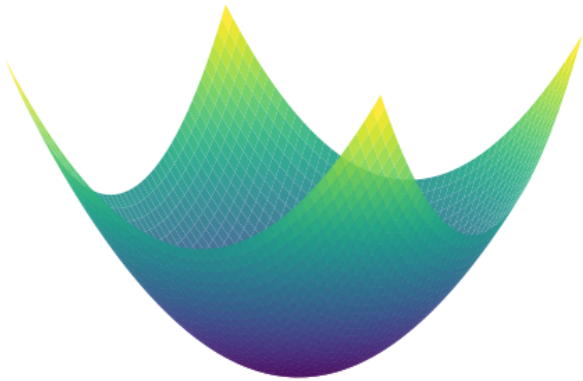
Exemple

$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & a^2+b^2 \end{array}$$

$\nabla f(a,b) = (2a, 2b)$

$a^* = b^* = 0$

$\rightsquigarrow$ minimum en $(0,0)$



$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & -a^2+b^2 \end{array}$$

$\nabla f(a,b) = (-2a, 2b)$

$a^* = b^* = 0$

$\rightsquigarrow$ $(0,0)$ pas d'extremum (point selle)

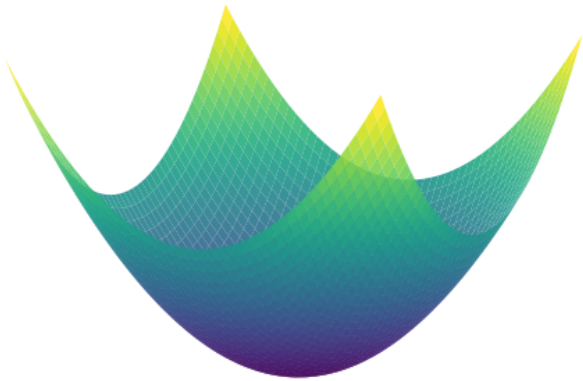
Exemple

$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & a^2+b^2 \end{array}$$

$\nabla f(a,b) = (2a, 2b)$

$a^* = b^* = 0$

$\rightsquigarrow$ minimum en $(0,0)$

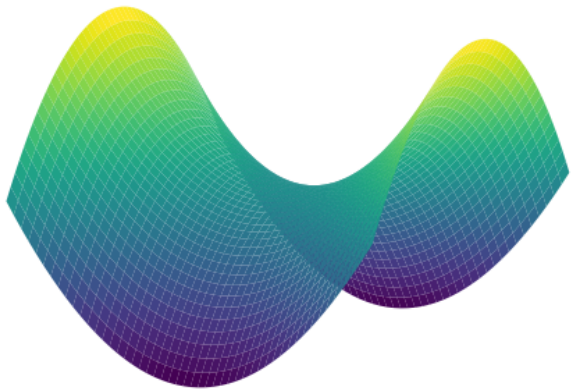


$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (a,b) & \longmapsto & -a^2+b^2 \end{array}$$

$\nabla f(a,b) = (-2a, 2b)$

$a^* = b^* = 0$

$\rightsquigarrow$ $(0,0)$ pas d'extremum (point selle)



Descente de gradient

Algorithme de la descente de gradient.

Soit une fonction $f : \mathbb{R}^n \to \mathbb{R}$,
$P \mapsto f(P)$ de plusieurs variables,
avec $P = (a_1, \ldots, a_n)$, dont on sait calculer le gradient $\nabla f(P)$.

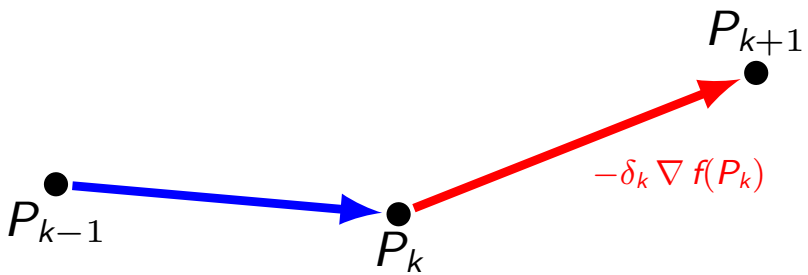
Descente de gradient

Algorithme de la descente de gradient.

Soit une fonction $f : \mathbb{R}^n \to \mathbb{R}$,
$P \mapsto f(P)$ de plusieurs variables,
avec $P = (a_1, \ldots, a_n)$, dont on sait calculer le gradient $\nabla f(P)$.

Données.

- Un point initial $P_0 \in \mathbb{R}^n$.
- Un niveau d'erreur $\varepsilon > 0$.



Descente de gradient

Algorithme de la descente de gradient.

Soit une fonction $f : \mathbb{R}^n \to \mathbb{R}$, $P \mapsto f(P)$ de plusieurs variables, avec $P = (a_1, \ldots, a_n)$, dont on sait calculer le gradient $\nabla f(P)$.

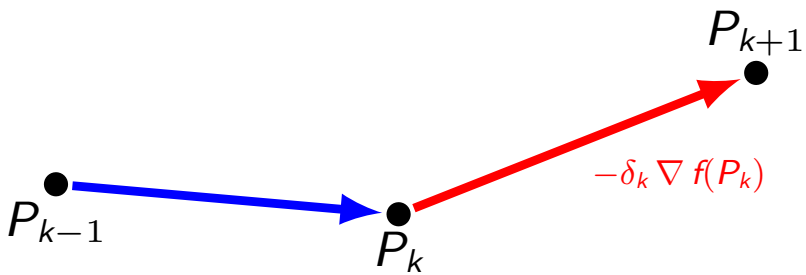
Données.

- Un point initial $P_0 \in \mathbb{R}^n$.
- Un niveau d'erreur $\varepsilon > 0$.

Itération. On calcule une suite de points $P_1, P_2, \ldots \in \mathbb{R}^n$ par récurrence de la façon suivante. Supposons que l'on ait déjà obtenu le point P_k :

- on calcule $\nabla f(P_k)$,
- on choisit un pas δ_k et on calcule

$$P_{k+1} = P_k - \delta_k \nabla f(P_k).$$

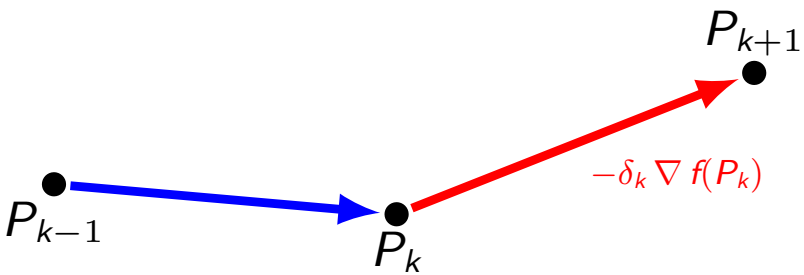


Descente de gradient

Algorithme de la descente de gradient.
Soit une fonction $f : \mathbb{R}^n \to \mathbb{R}$, $P \mapsto f(P)$ de plusieurs variables, avec $P = (a_1, \ldots, a_n)$, dont on sait calculer le gradient $\nabla f(P)$.

Données.

- Un point initial $P_0 \in \mathbb{R}^n$.
- Un niveau d'erreur $\varepsilon > 0$.



Itération. On calcule une suite de points $P_1, P_2, \ldots \in \mathbb{R}^n$ par récurrence de la façon suivante. Supposons que l'on ait déjà obtenu le point P_k :

- on calcule $\nabla f(P_k)$,
- on choisit un pas δ_k et on calcule

$$P_{k+1} = P_k - \delta_k \nabla f(P_k).$$

Arrêt. On s'arrête lorsque $\| \nabla f(P_k) \| \leq \varepsilon$.

Et si le gradient est difficile à calculer ?

Approximation numérique du gradient :

$$\begin{aligned}\nabla f(x_0, y_0) &= \left(\frac{\partial f}{\partial x}(x_0, y_0), \frac{\partial f}{\partial y}(x_0, y_0) \right) \\ &\approx \left(\frac{f(x_0 + \varepsilon, y_0) - f(x_0, y_0)}{\varepsilon}, \frac{f(x_0, y_0 + \varepsilon) - f(x_0, y_0)}{\varepsilon} \right)\end{aligned}$$

Et si le gradient est difficile à calculer ?

Approximation numérique du gradient :

$$\begin{aligned}\nabla f(x_0, y_0) &= \left(\frac{\partial f}{\partial x}(x_0, y_0), \frac{\partial f}{\partial y}(x_0, y_0)\right) \\ &\approx \left(\frac{f(x_0 + \varepsilon, y_0) - f(x_0, y_0)}{\varepsilon}, \frac{f(x_0, y_0 + \varepsilon) - f(x_0, y_0)}{\varepsilon}\right)\end{aligned}$$

il y a de nombreuses sous-méthodes pour améliorer/adapter la descente de gradient :

Et si le gradient est difficile à calculer ?

Approximation numérique du gradient :

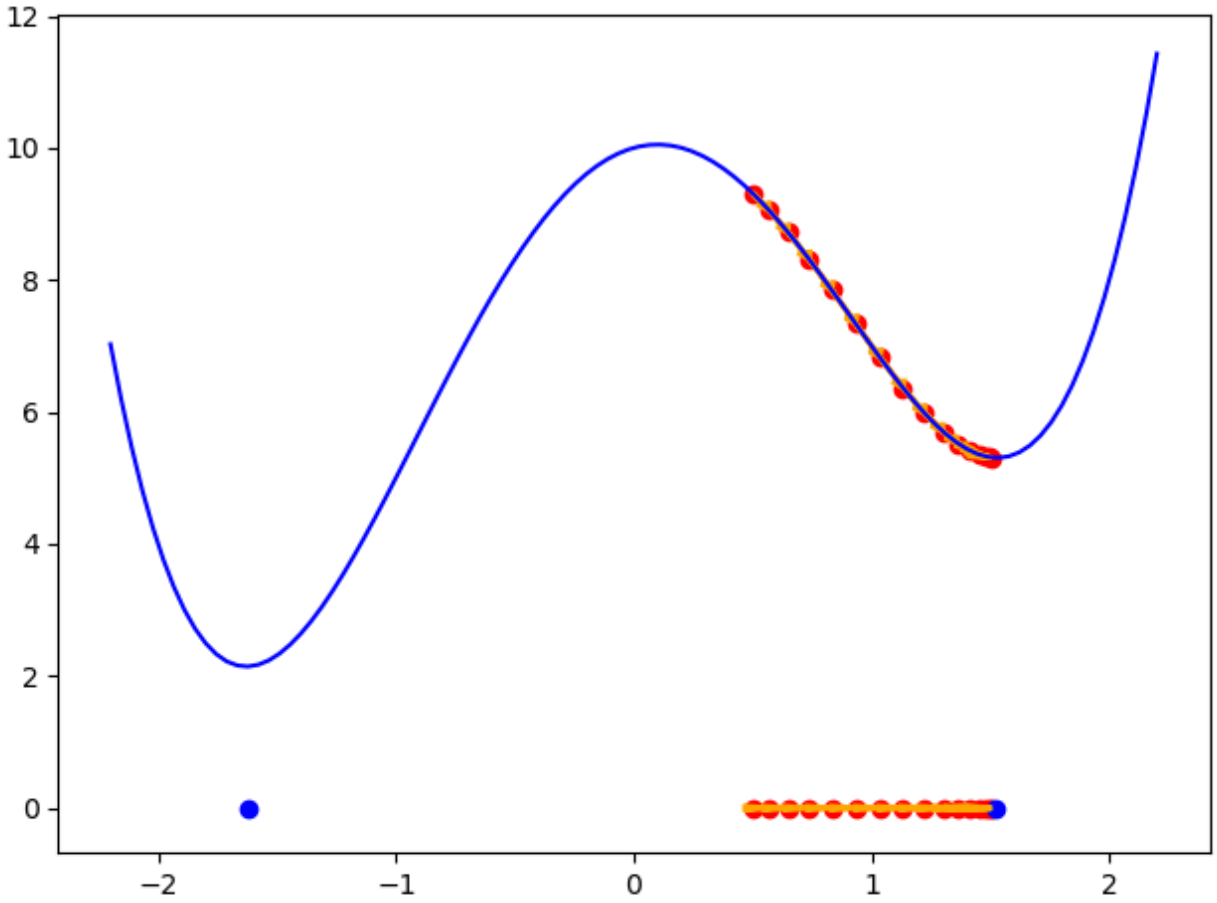
$$\begin{aligned}\nabla f(x_0, y_0) &= \left(\frac{\partial f}{\partial x}(x_0, y_0), \frac{\partial f}{\partial y}(x_0, y_0)\right) \\ &\approx \left(\frac{f(x_0 + \varepsilon, y_0) - f(x_0, y_0)}{\varepsilon}, \frac{f(x_0, y_0 + \varepsilon) - f(x_0, y_0)}{\varepsilon}\right)\end{aligned}$$

il y a de nombreuses sous-méthodes pour améliorer/adapter la descente de gradient :

Gradient stochastique, à pas optimal, conjugué ...

Limitation

- des paramètres à choisir $\rightsquigarrow$ tests et habitude
- on converge vers un minimum *local* $\rightsquigarrow$ on rate parfois l'objectif
- la fonction doit-être dérivable (et donc continue) $\rightsquigarrow$ plein de problèmes ne sont pas modélisables



Algorithmes stochastiques et génétiques

Pour pallier à ces problèmes, mathématiciens et informaticiens (et physiciens, biologistes, économistes, sociologues ...) ont créé des *heuristiques*[1] faisant appel au hasard.

1. une heuristique est un algorithme dont le résultat n'est pas garanti, ni théoriquement ni en pratique (mais qui marche assez souvent tout de même)

Algorithmes stochastiques et génétiques

Pour pallier à ces problèmes, mathématiciens et informaticiens (et physiciens, biologistes, économistes, sociologues ...) ont créé des *heuristiques*[1] faisant appel au hasard.

Méthodes aléatoires :
on introduit une perturbation dans l'algorithme

- dans le critère d'acceptation (recuit simulé)
- dans l'objectif (méthode de bruitage)
- dans le voisinage (gradient stochastique)

1. une heuristique est un algorithme dont le résultat n'est pas garanti, ni théoriquement ni en pratique (mais qui marche assez souvent tout de même)

Algorithmes stochastiques et génétiques

Pour pallier à ces problèmes, mathématiciens et informaticiens (et physiciens, biologistes, économistes, sociologues ...) ont créé des *heuristiques*[1] faisant appel au hasard.

Méthodes aléatoires :

on introduit une perturbation dans l'algorithme

- dans le critère d'acceptation (recuit simulé)
- dans l'objectif (méthode de bruitage)
- dans le voisinage (gradient stochastique)

Méta-heuristique "algorithme génétique"

Init : construction et évaluation d'une population initiale
Jusqu'à critère d'arrêt :
sélection d'une partie de la population,
reproduction des individus sélectionnés,
mutation de la descendance,
évaluation du degré d'adaptation de chaque individu,
remplacement de la population initiale par une nouvelle population.

1. une heuristique est un algorithme dont le résultat n'est pas garanti, ni théoriquement ni en pratique (mais qui marche assez souvent tout de même)

Recuit simulé

Recuit simulé in a nutshell

Init : choisir un point de départ
Jusqu'à critère d'arrêt :
  sélection d'un voisin aléatoirement
  calcul du coût du nouveau point,
  si coût plus faible
    conserver le nouveau point
  sinon
    conserver le nouveau point avec proba. $p(n)$

Recuit simulé

Recuit simulé in a nutshell

```
Init : choisir un point de départ
Jusqu'à critère d'arrêt :
  sélection d'un voisin aléatoirement
  calcul du coût du nouveau point,
  si coût plus faible
    conserver le nouveau point
  sinon
    conserver le nouveau point avec proba. p(n)
```

On choisit parfois une solution moins bonne pour essayer de sortir des minima locaux

La proba. de garder un point moins bon diminue avec le temps

Recuit simulé

Recuit simulé in a nutshell

```
Init : choisir un point de départ
Jusqu'à critère d'arrêt :
  sélection d'un voisin aléatoirement
  calcul du coût du nouveau point,
  si coût plus faible
    conserver le nouveau point
  sinon
    conserver le nouveau point avec proba. p(n)
```

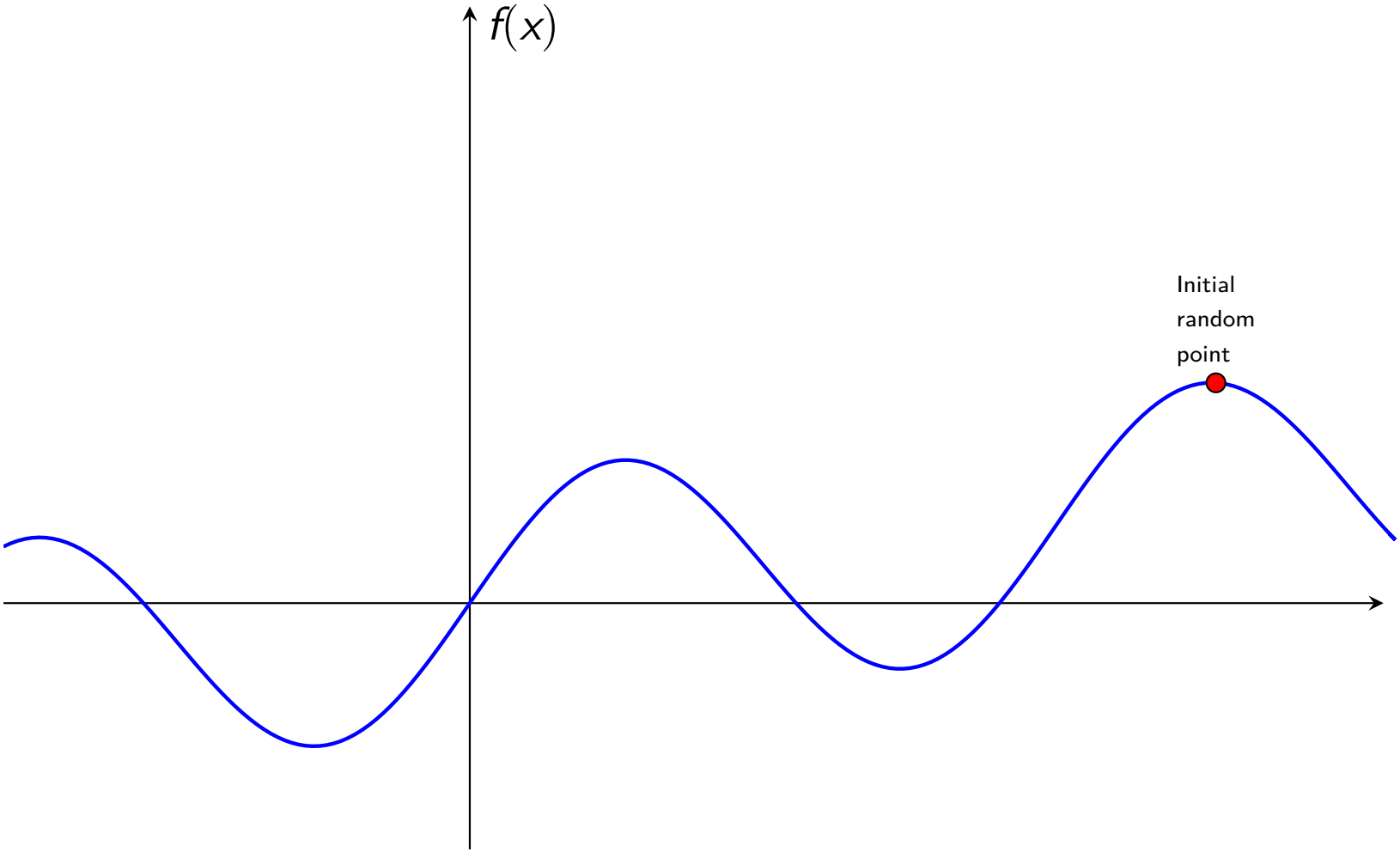
On choisit parfois une solution moins bonne pour essayer de sortir des minima locaux

La proba. de garder un point moins bon diminue avec le temps

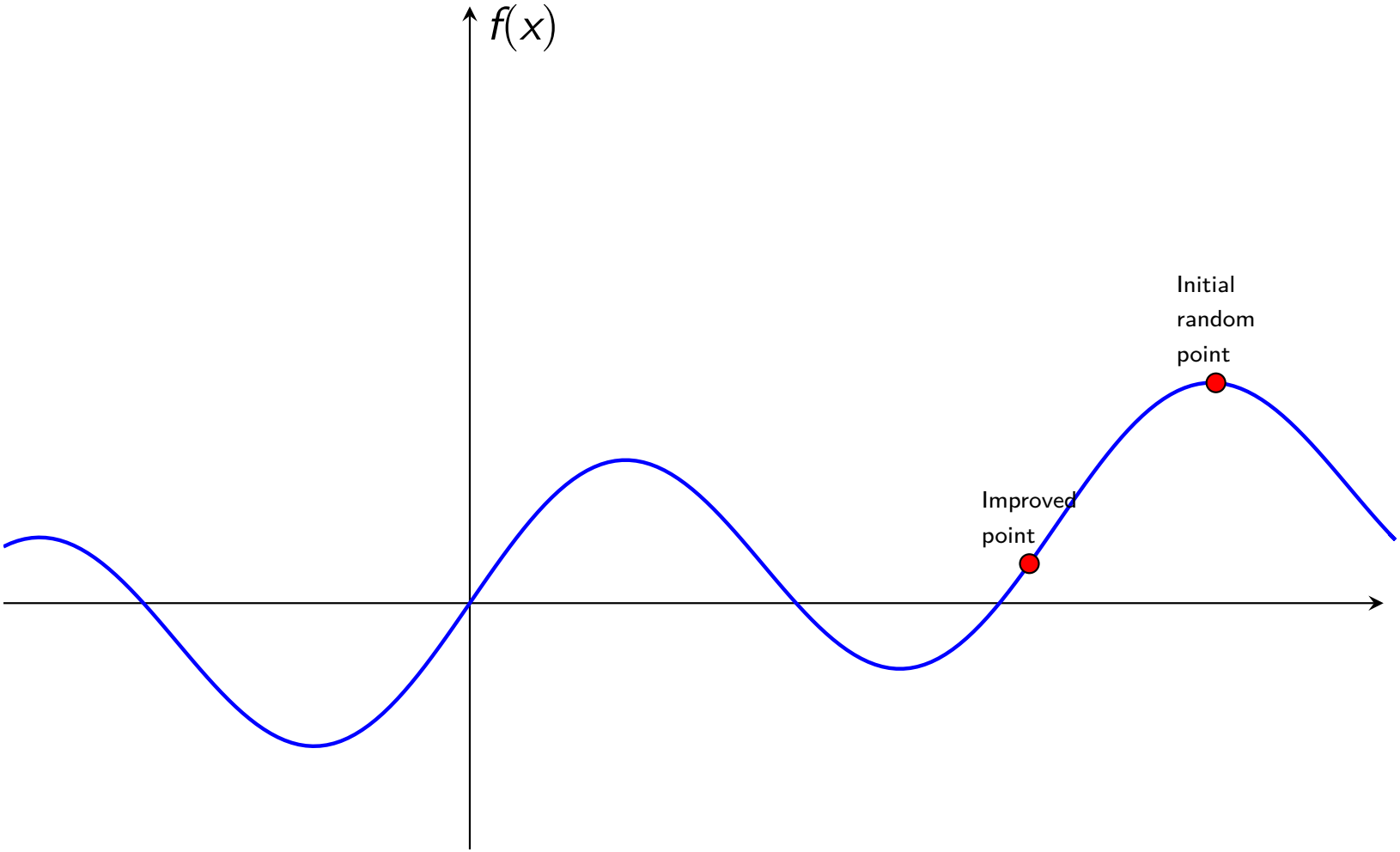
Inspiré du processus de recuit, utilisé en métallurgie (montées graduelles en température suivies de refroidissements contrôlés)

S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi, *Optimization by Simulated Annealing*, 1983

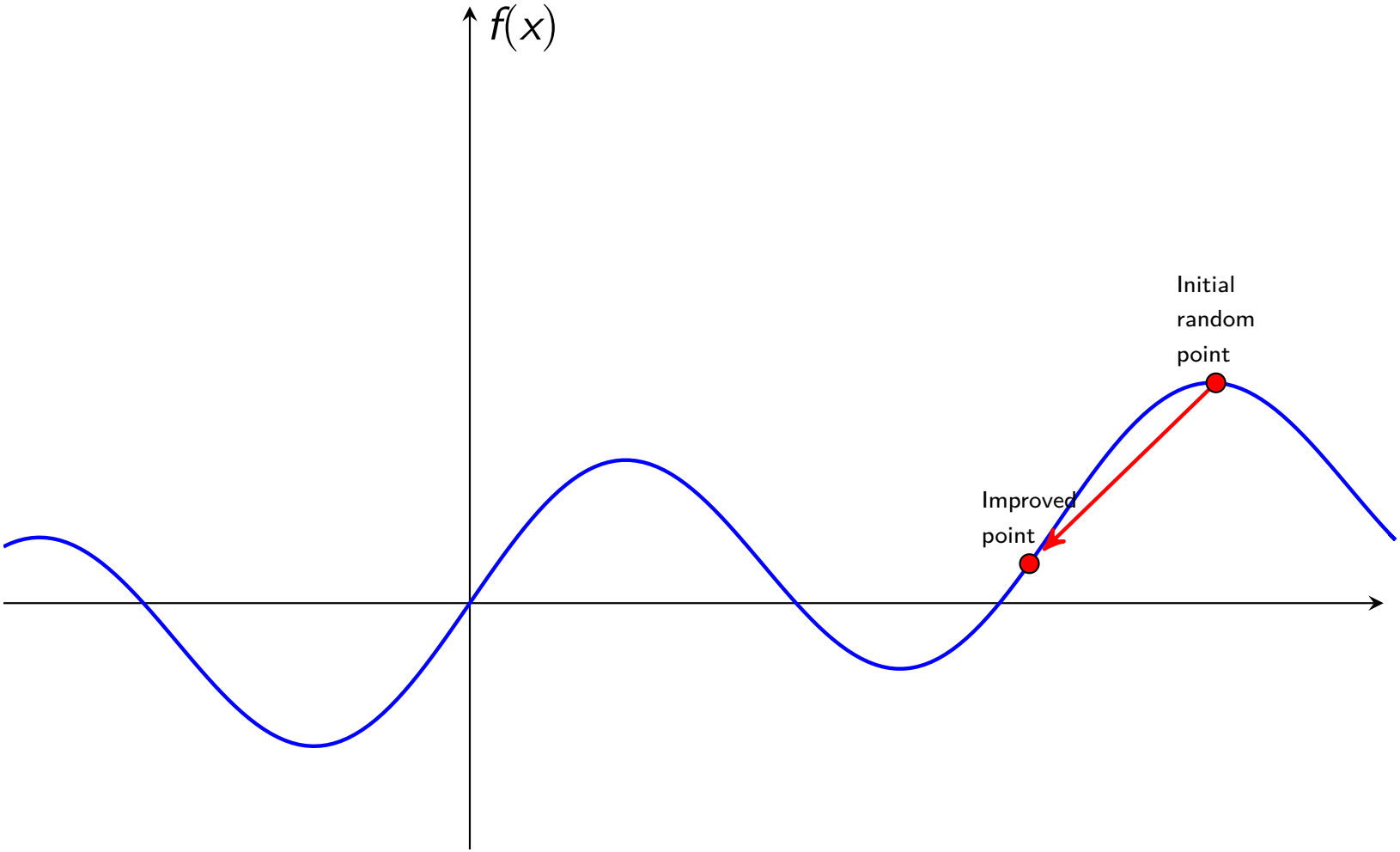
Recuit simulé



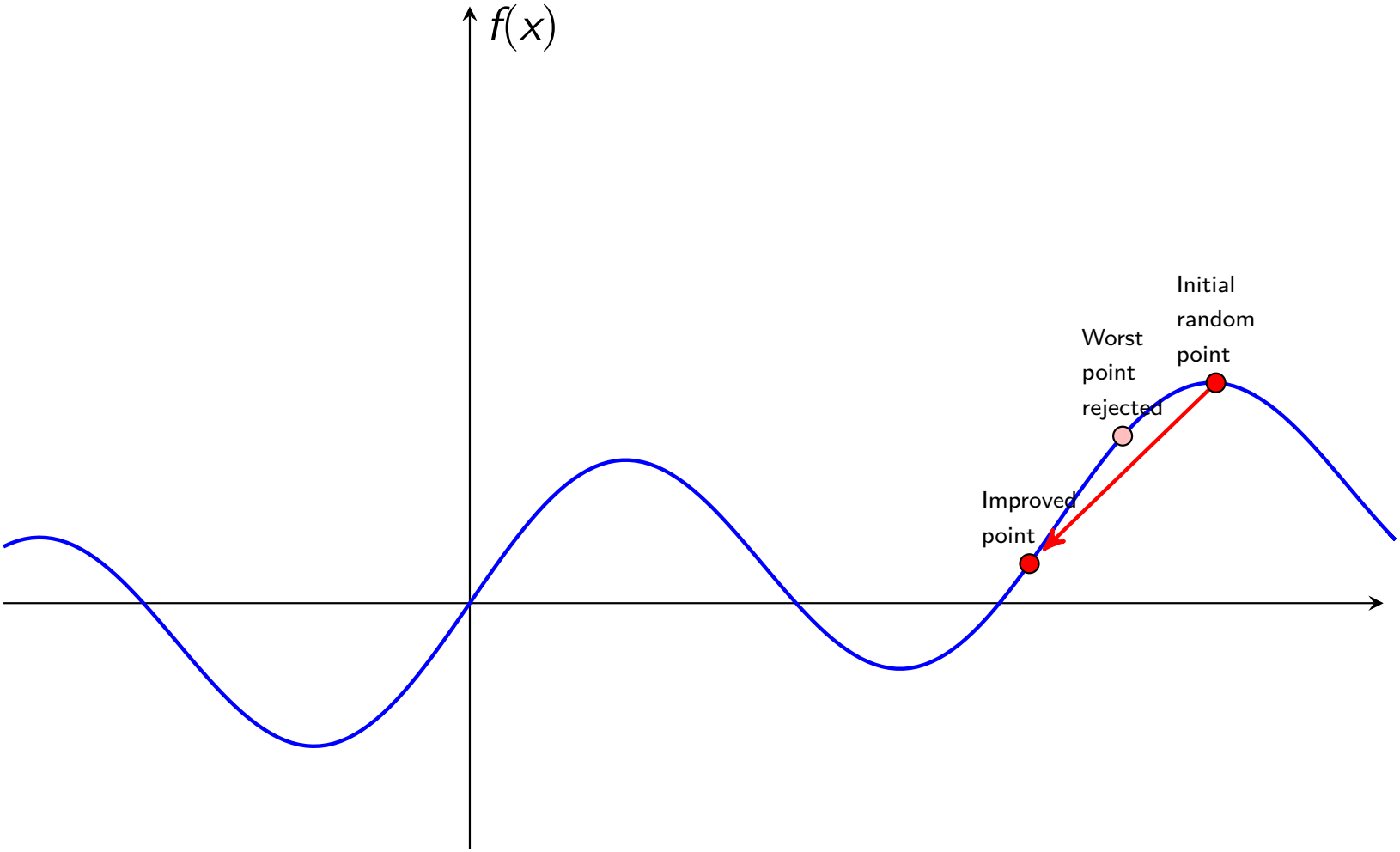
Recuit simulé



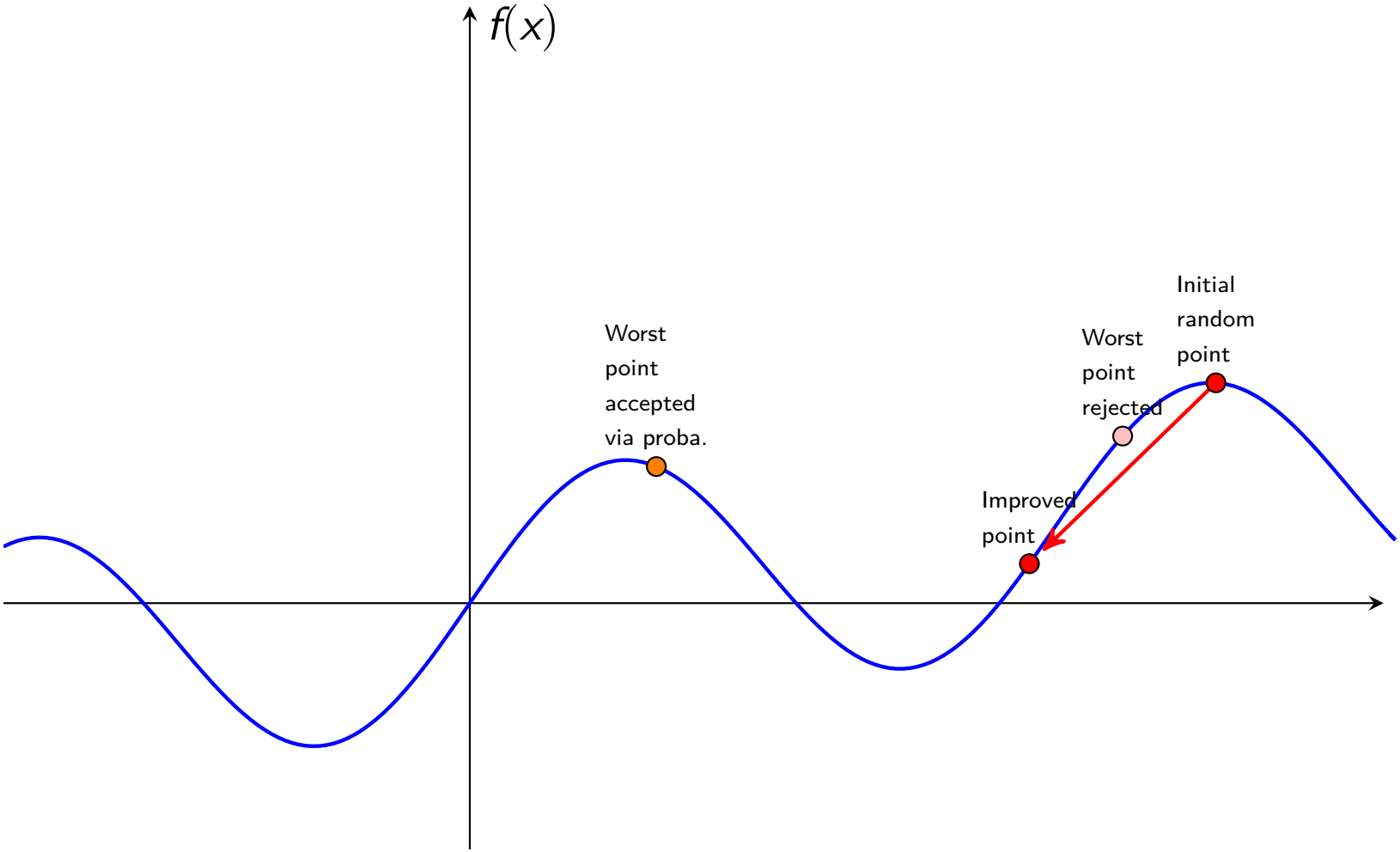
Recuit simulé



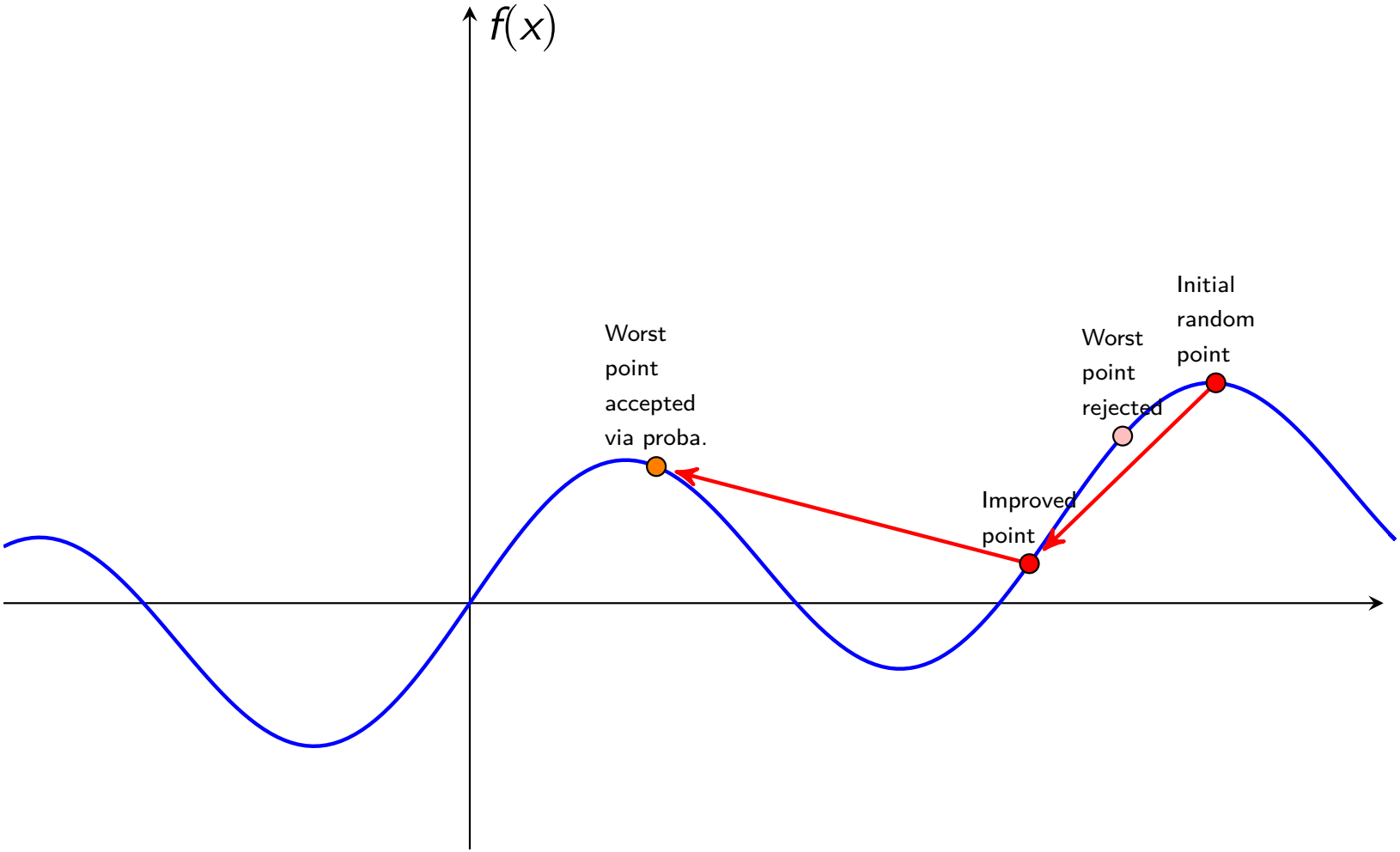
Recuit simulé



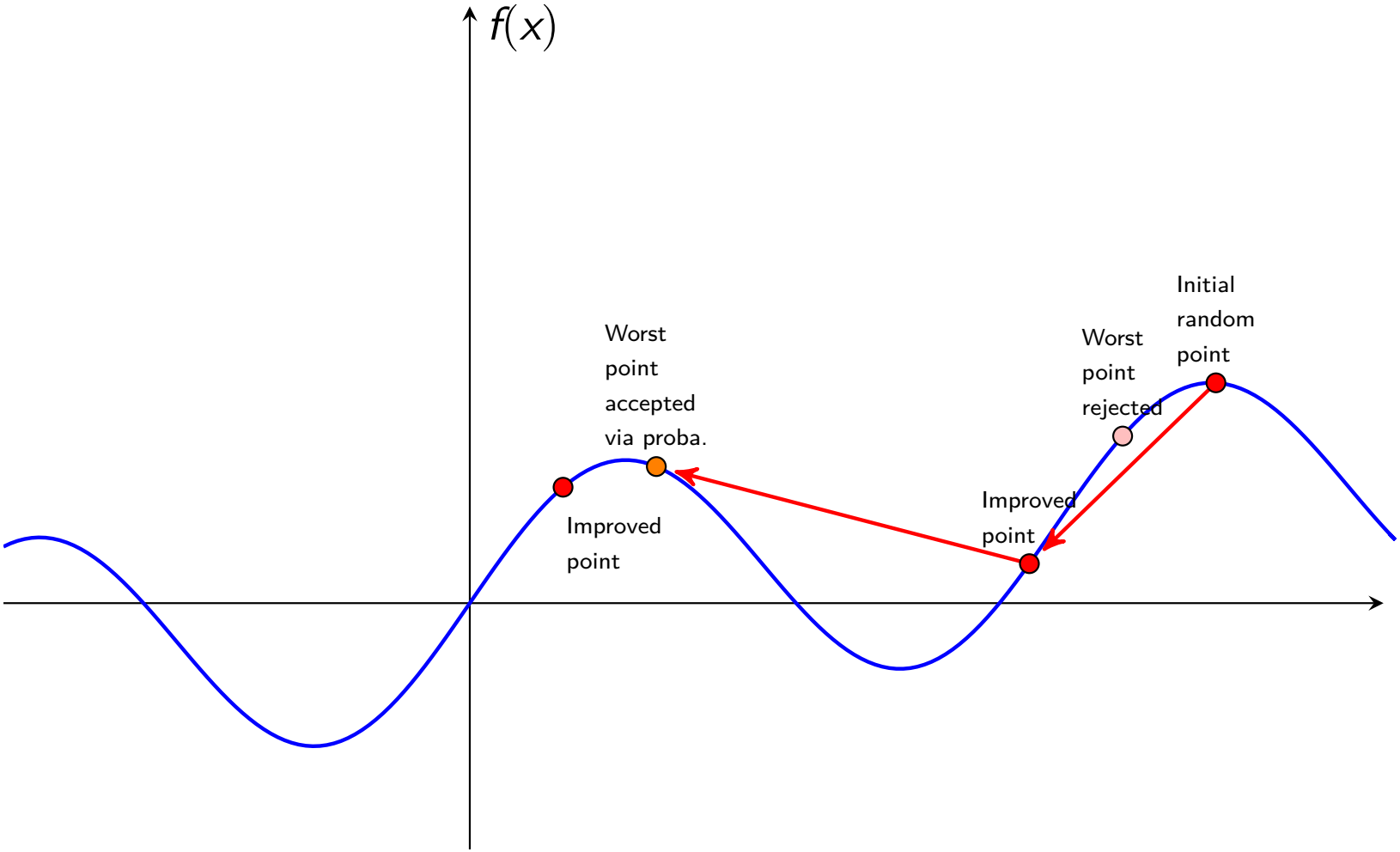
Recuit simulé



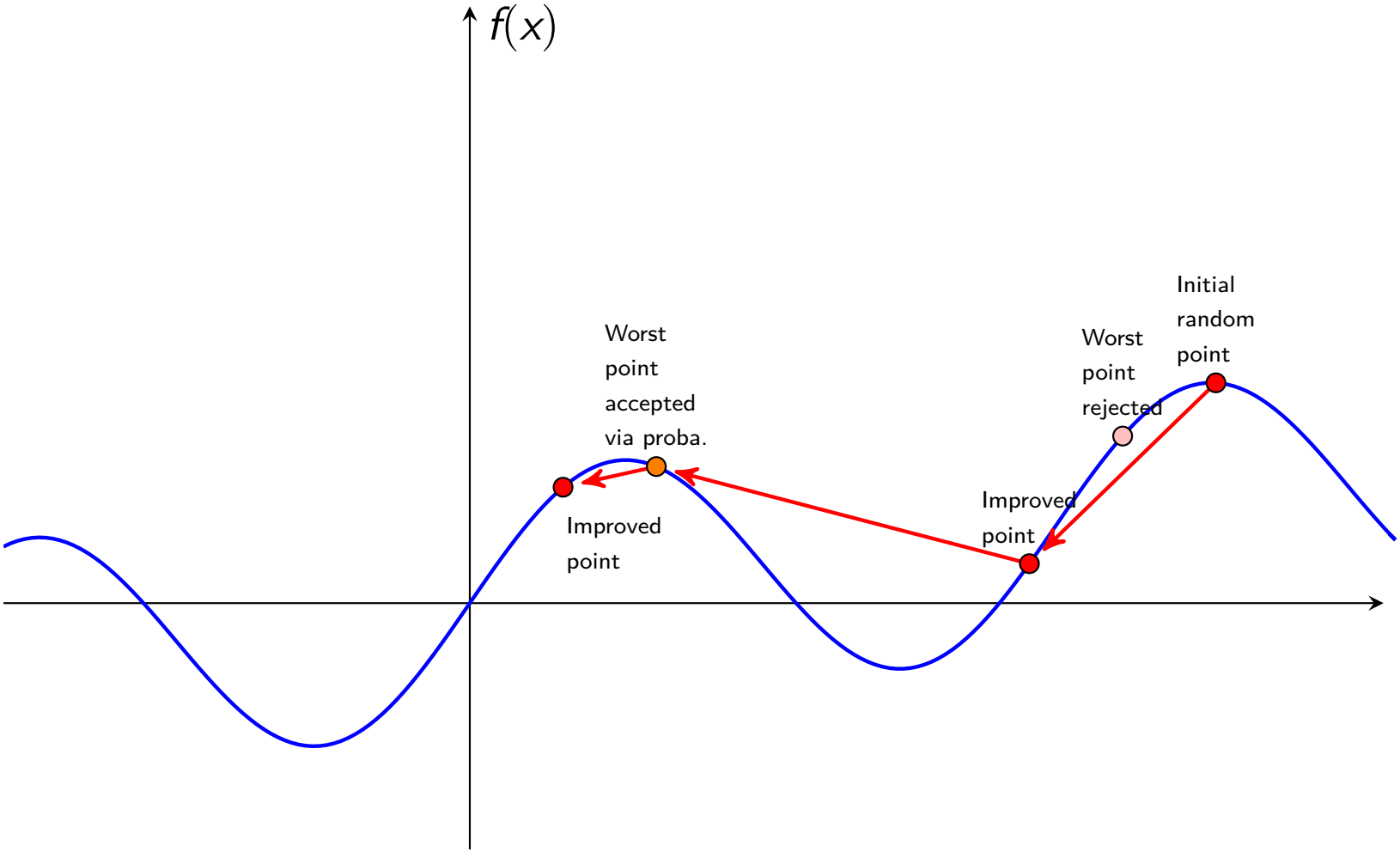
Recuit simulé



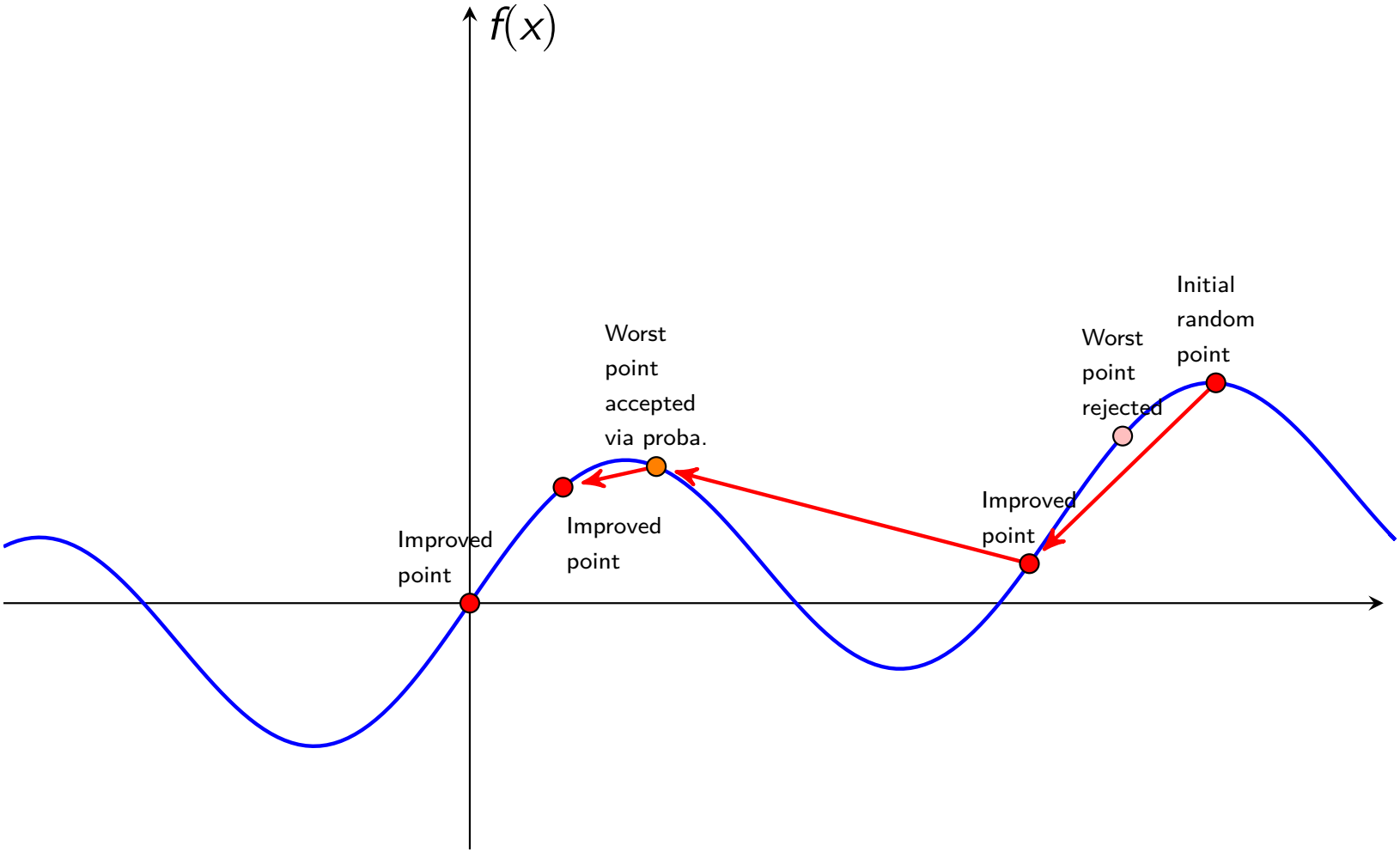
Recuit simulé



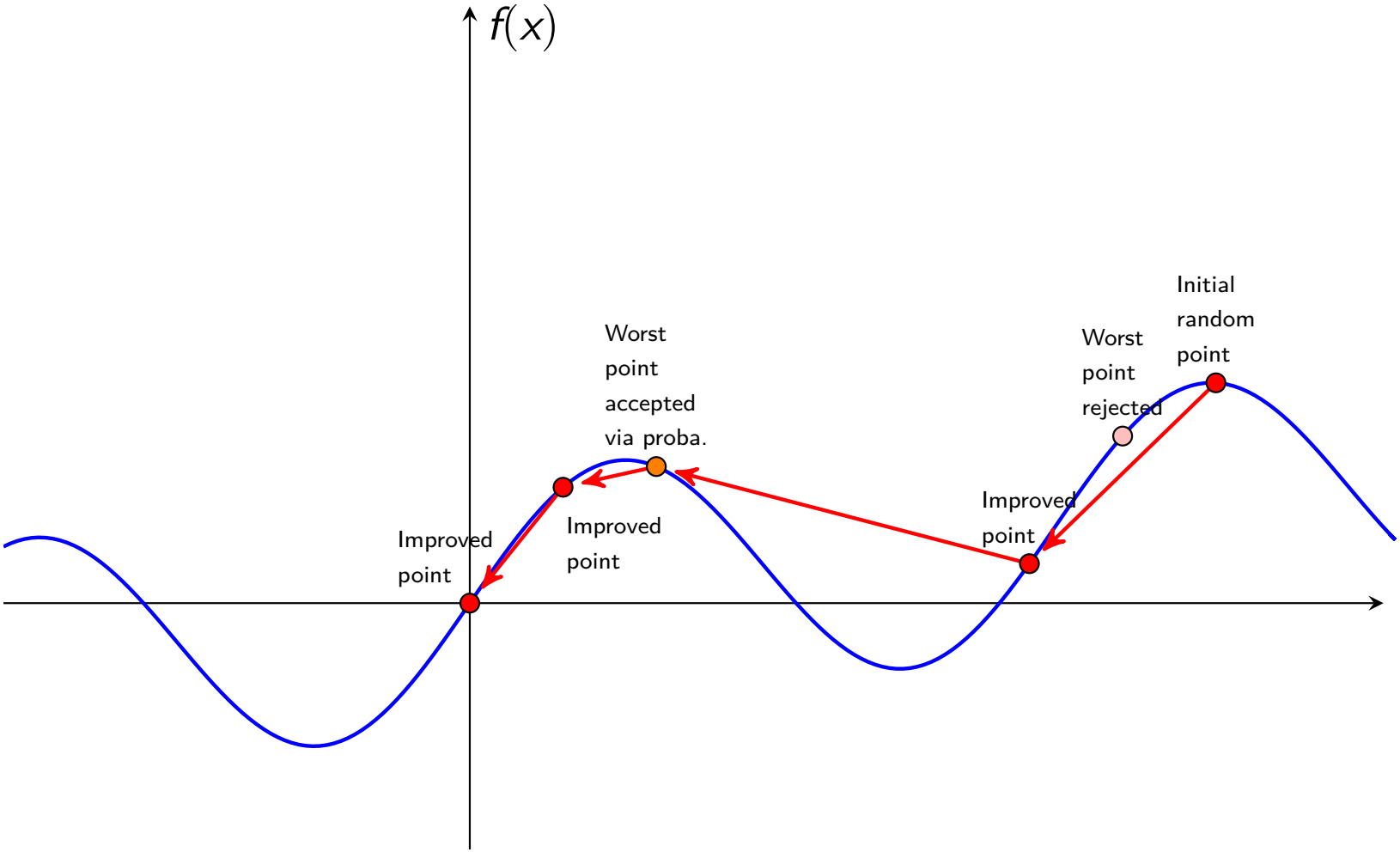
Recuit simulé



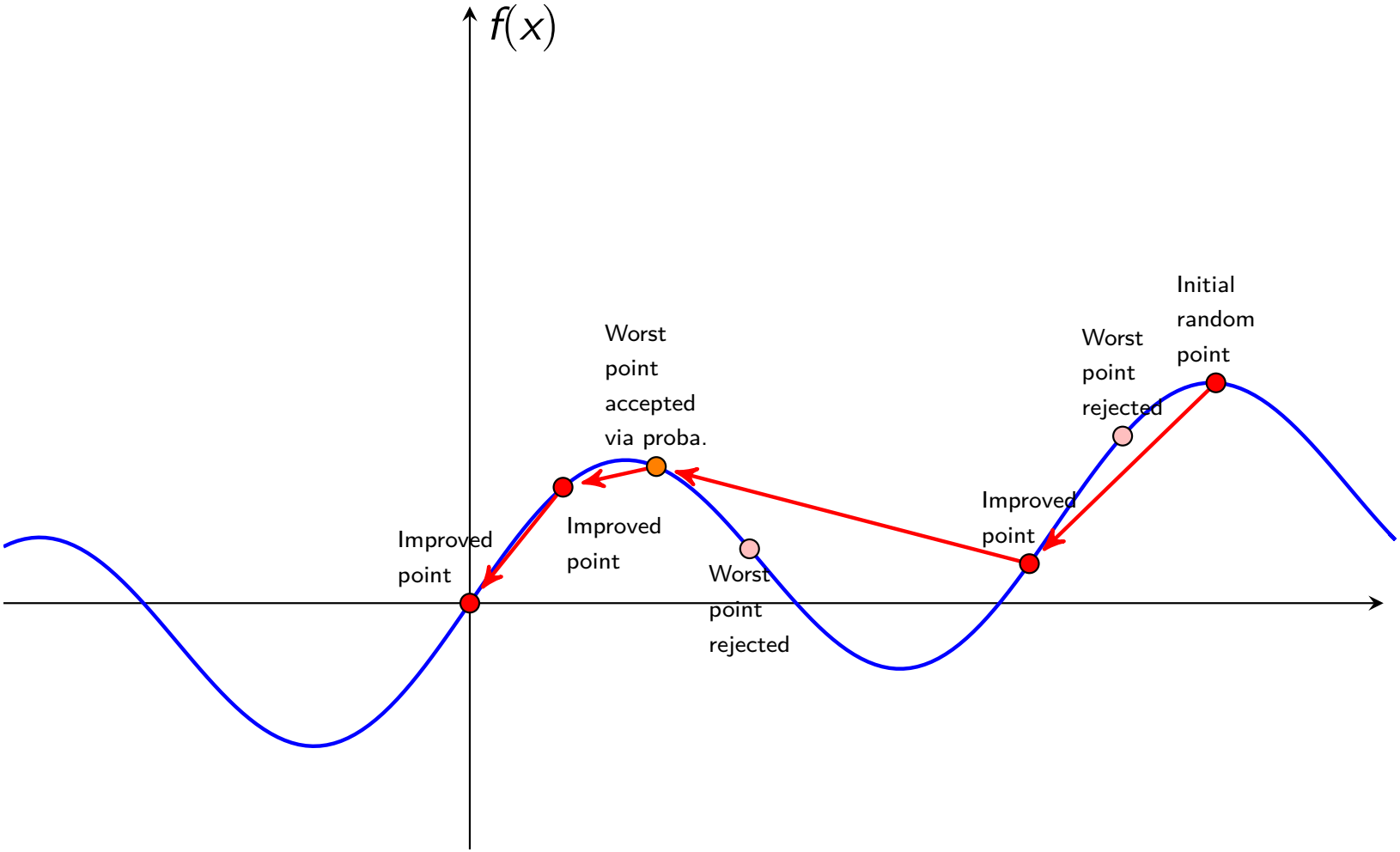
Recuit simulé



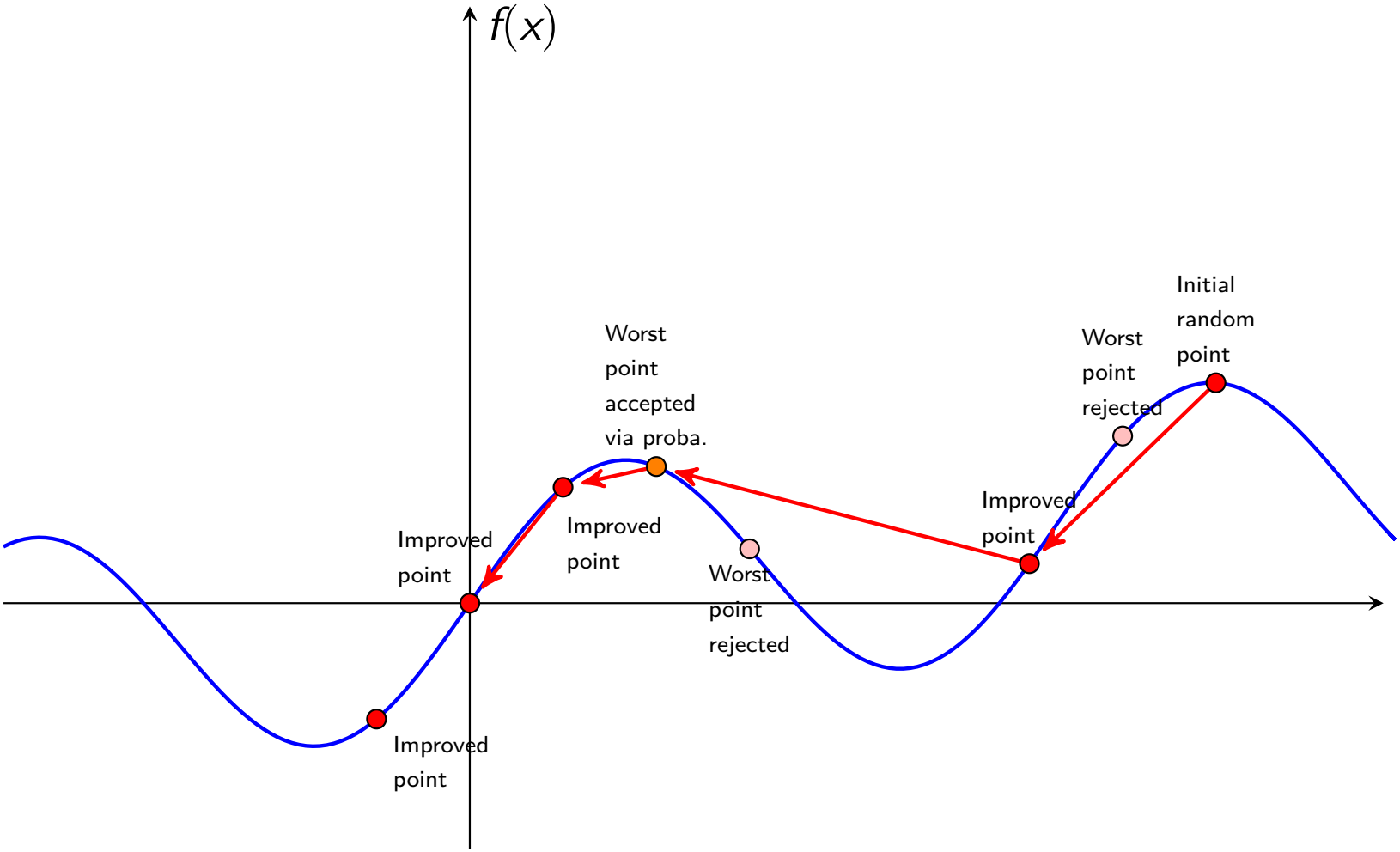
Recuit simulé



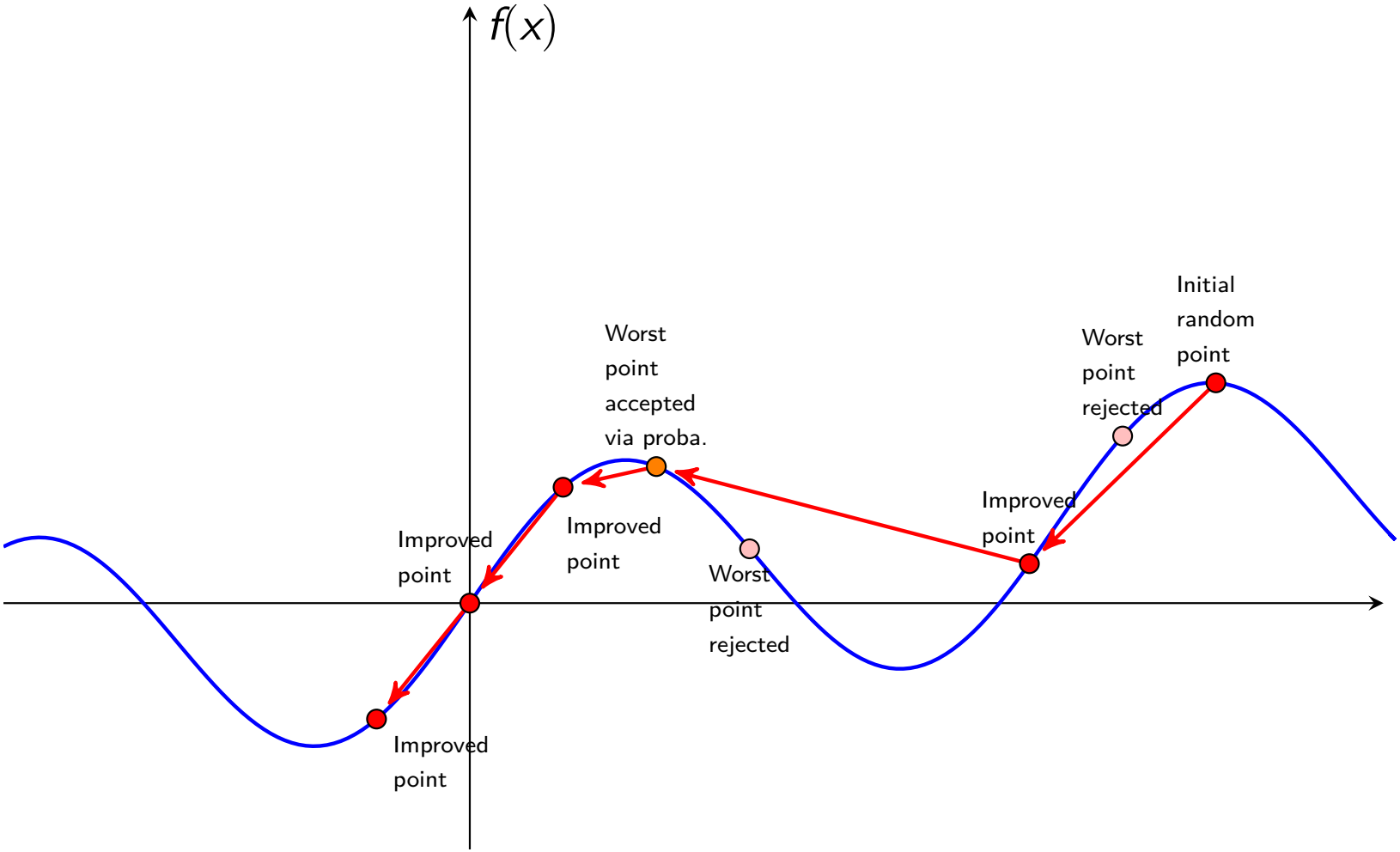
Recuit simulé



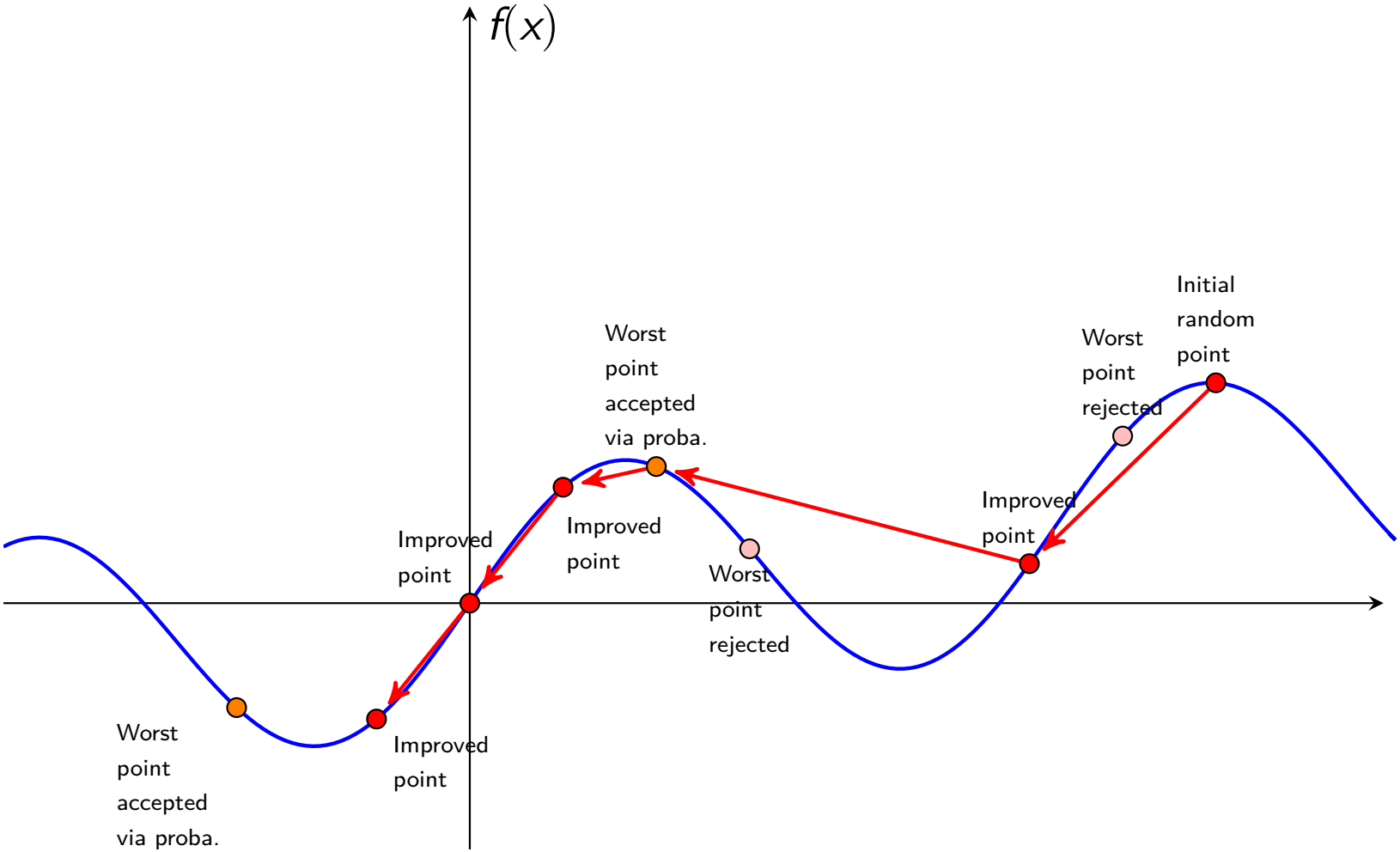
Recuit simulé



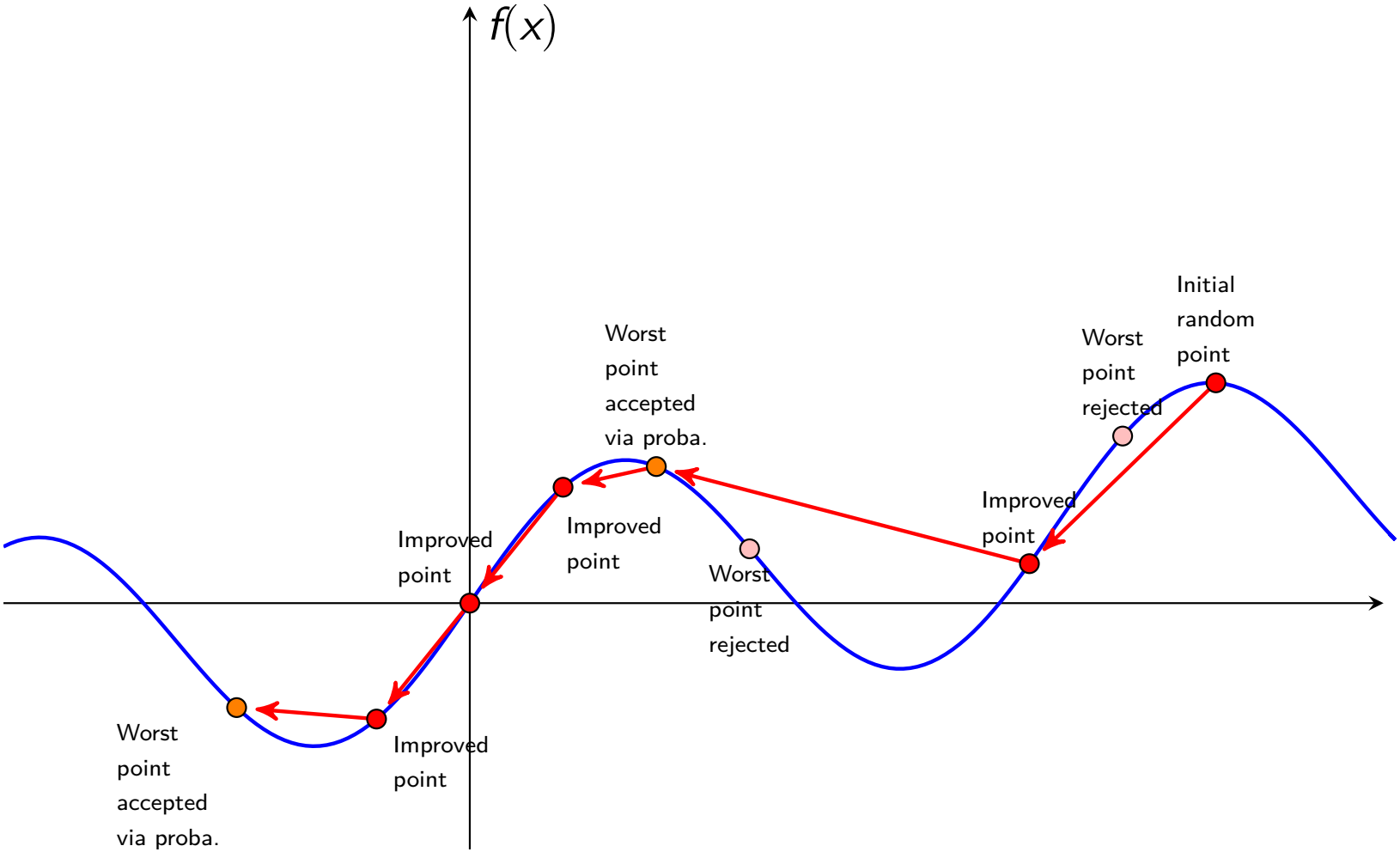
Recuit simulé



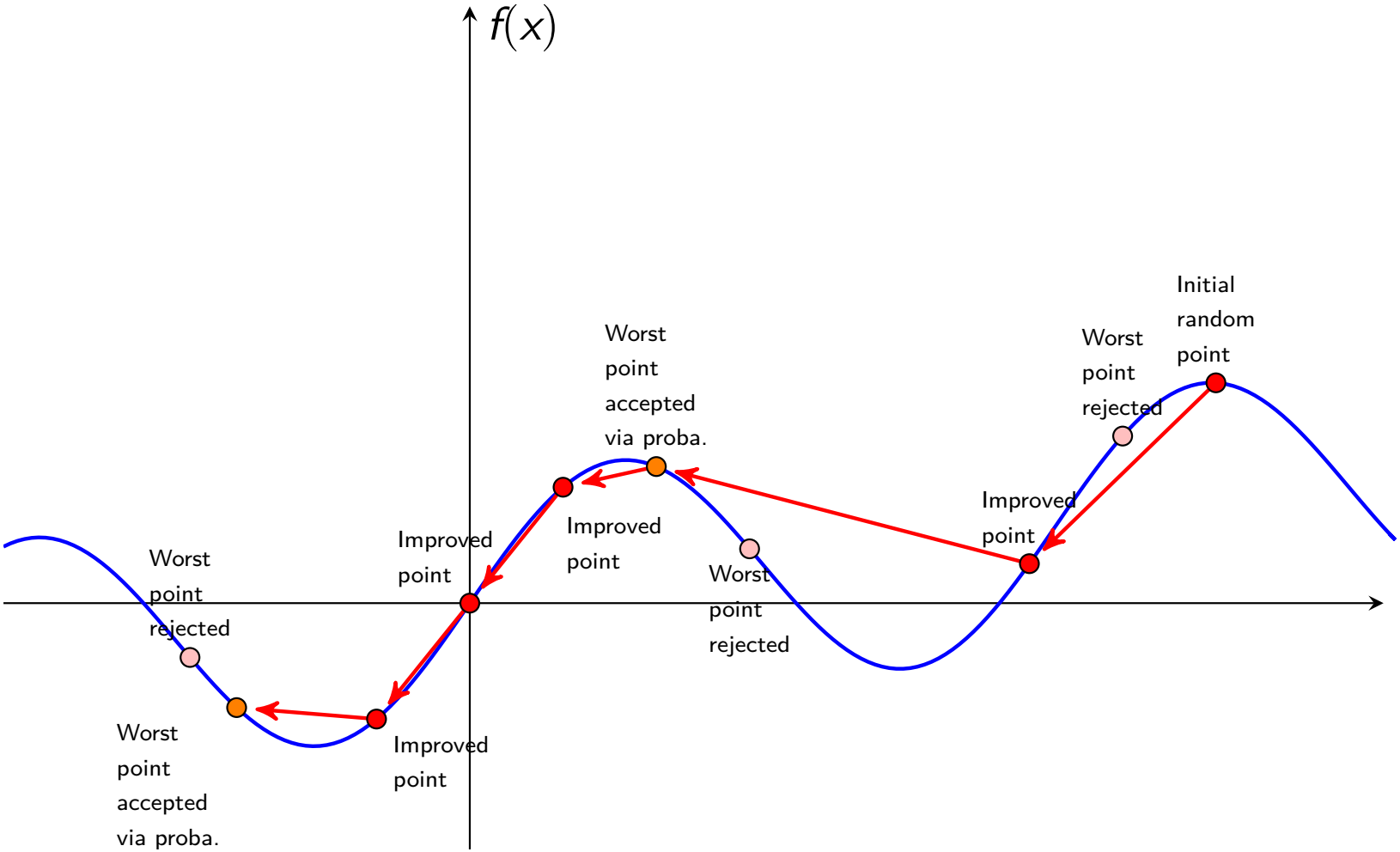
Recuit simulé



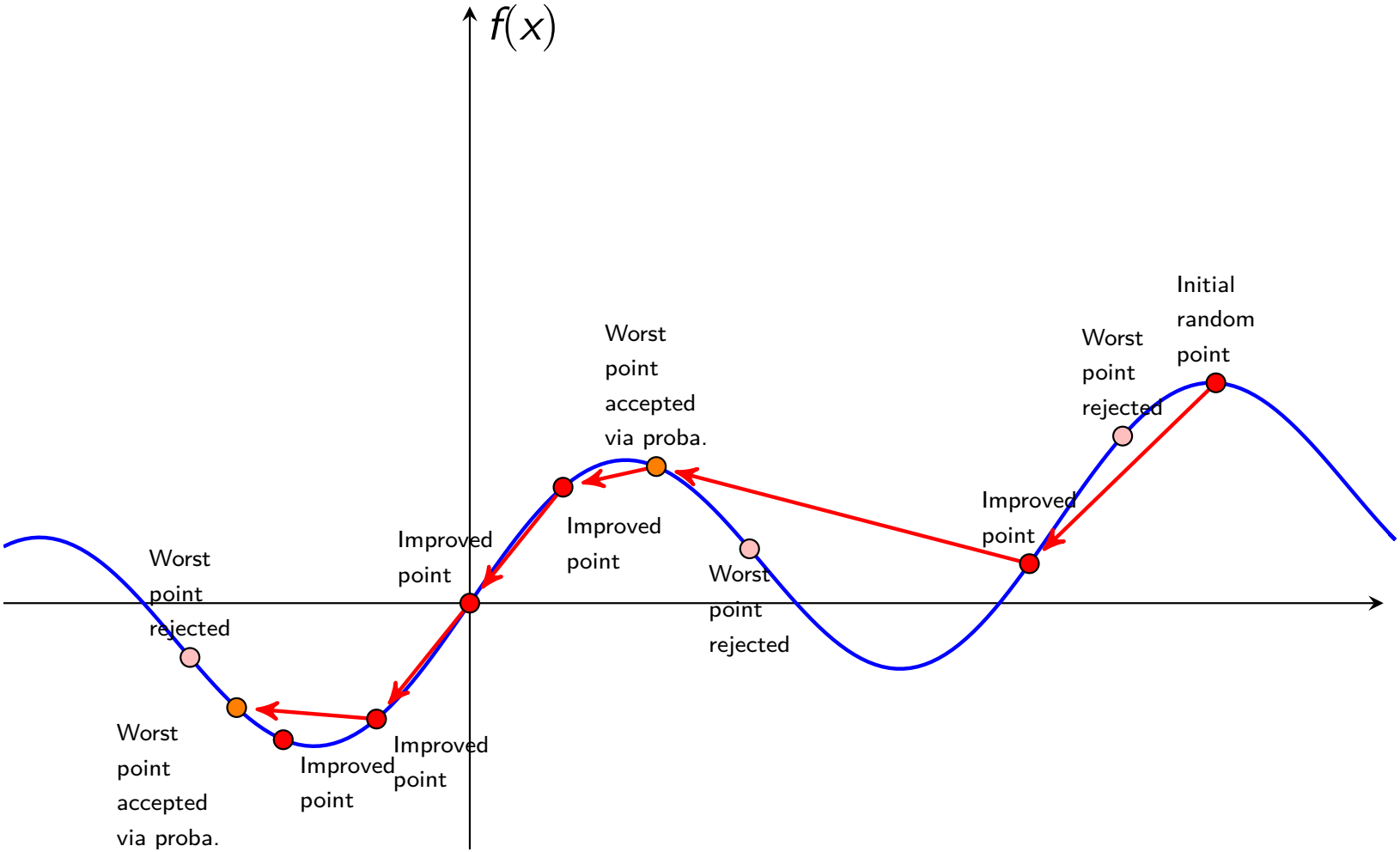
Recuit simulé



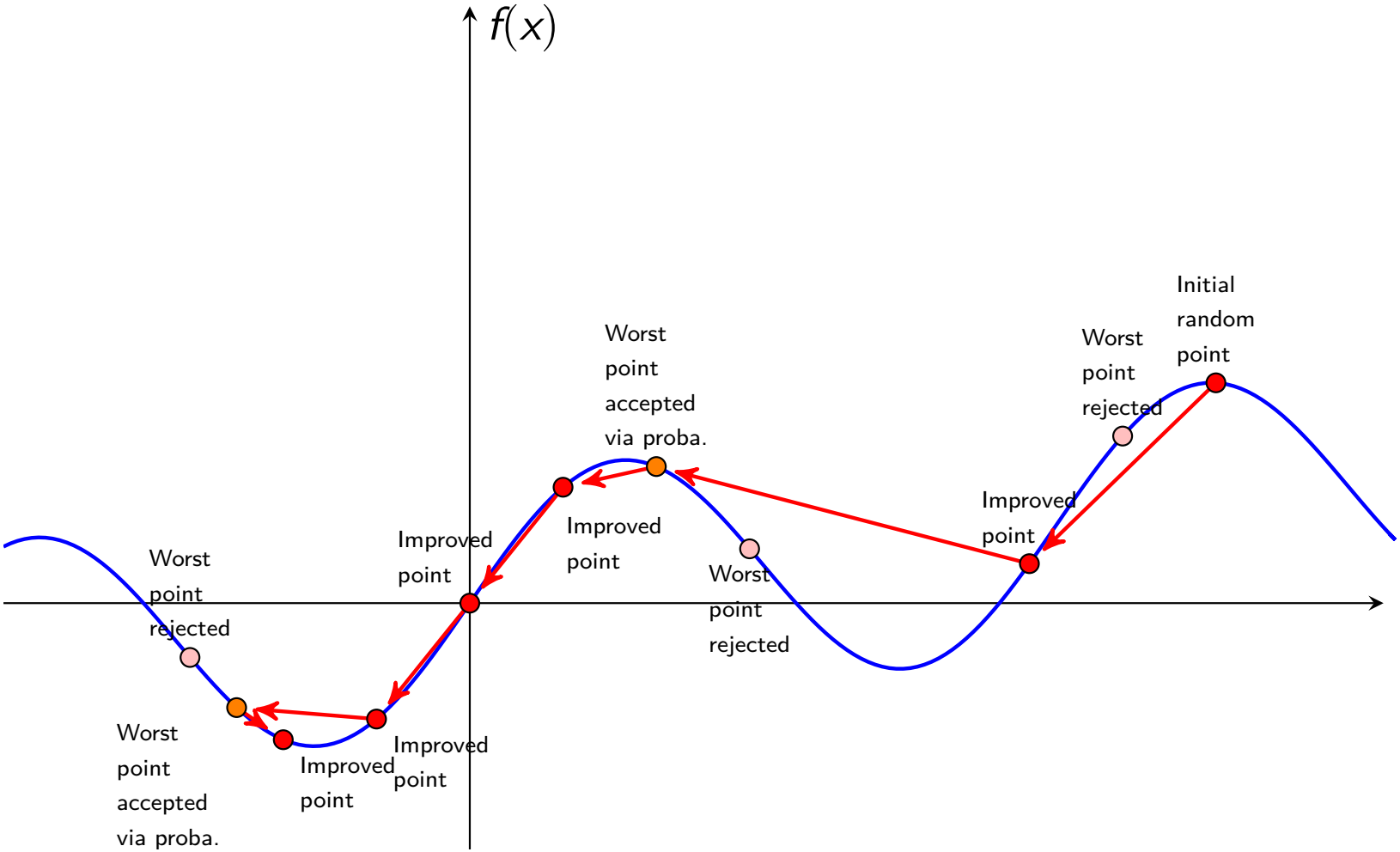
Recuit simulé



Recuit simulé



Recuit simulé



Recuit simulé : formalisation

Soit un problème d'optimisation combinatoire où l'on a

- un ensemble de configuration $\Omega = \{w_1, ..., w_N\}$

Recuit simulé : formalisation

Soit un problème d'optimisation combinatoire où l'on a

- un ensemble de configuration $\Omega = \{w_1, ..., w_N\}$
- une fonction de coût (aussi appelée fonction d'energie) $c : \Omega \to \mathbb{R}_+$

Recuit simulé : formalisation

Soit un problème d'optimisation combinatoire où l'on a

- un ensemble de configuration $\Omega = \{w_1, ..., w_N\}$
- une fonction de coût (aussi appelée fonction d'energie) $c : \Omega \to \mathbb{R}_+$
- une fonction de voisinage $V : \Omega \to \mathcal{P}(\Omega)$

Recuit simulé : formalisation

Soit un problème d'optimisation combinatoire où l'on a

- un ensemble de configuration $\Omega = \{w_1, ..., w_N\}$
- une fonction de coût (aussi appelée fonction d'energie) $c : \Omega \to \mathbb{R}_+$
- une fonction de voisinage $V : \Omega \to \mathcal{P}(\Omega)$
- une fonction de mise à jour de la température $c : \mathbb{R}_+ \times \mathbb{N} \to \mathbb{R}_+$

Recuit simulé : formalisation

Soit un problème d'optimisation combinatoire où l'on a

- un ensemble de configuration $\Omega = \{w_1, ..., w_N\}$
- une fonction de coût (aussi appelée fonction d'energie) $c : \Omega \to \mathbb{R}_+$
- une fonction de voisinage $V : \Omega \to \mathcal{P}(\Omega)$
- une fonction de mise à jour de la température $c : \mathbb{R}_+ \times \mathbb{N} \to \mathbb{R}_+$

Recuit simulé : formalisation

Soit un problème d'optimisation combinatoire où l'on a

- un ensemble de configuration $\Omega = \{w_1, ..., w_N\}$
- une fonction de coût (aussi appelée fonction d'energie) $c : \Omega \to \mathbb{R}_+$
- une fonction de voisinage $V : \Omega \to \mathcal{P}(\Omega)$
- une fonction de mise à jour de la température $c : \mathbb{R}_+ \times \mathbb{N} \to \mathbb{R}_+$

La méthode du recuit simulé vise à trouver la configuration de coût (énergie) minimal.

Recuit simulé : formalisation

Soit un problème d'optimisation combinatoire où l'on a

- un ensemble de configuration $\Omega = \{w_1, ..., w_N\}$
- une fonction de coût (aussi appelée fonction d'energie) $c : \Omega \to \mathbb{R}_+$
- une fonction de voisinage $V : \Omega \to \mathcal{P}(\Omega)$
- une fonction de mise à jour de la température $c : \mathbb{R}_+ \times \mathbb{N} \to \mathbb{R}_+$

La méthode du recuit simulé vise à trouver la configuration de coût (énergie) minimal.

La méthode est alors assez intuitive :

Au départ on commence avec une configuration arbitraire.

Recuit simulé : formalisation

Soit un problème d'optimisation combinatoire où l'on a

- un ensemble de configuration $\Omega = \{w_1, ..., w_N\}$
- une fonction de coût (aussi appelée fonction d'energie) $c : \Omega \to \mathbb{R}_+$
- une fonction de voisinage $V : \Omega \to \mathcal{P}(\Omega)$
- une fonction de mise à jour de la température $c : \mathbb{R}_+ \times \mathbb{N} \to \mathbb{R}_+$

La méthode du recuit simulé vise à trouver la configuration de coût (énergie) minimal.

La méthode est alors assez intuitive :

Au départ on commence avec une configuration arbitraire.

Soit w la configuration courante.

À chaque itération, une configuration candidate w' est choisie uniformément dans le voisinage de la solution courante ; et acceptée avec une probabilité $\min(1, e^{-\frac{c(w)-c(w')}{T}})$

Recuit simulé : formalisation

Soit un problème d'optimisation combinatoire où l'on a

- un ensemble de configuration $\Omega = \{w_1, ..., w_N\}$
- une fonction de coût (aussi appelée fonction d'energie) $c : \Omega \to \mathbb{R}_+$
- une fonction de voisinage $V : \Omega \to \mathcal{P}(\Omega)$
- une fonction de mise à jour de la température $c : \mathbb{R}_+ \times \mathbb{N} \to \mathbb{R}_+$

La méthode du recuit simulé vise à trouver la configuration de coût (énergie) minimal.

La méthode est alors assez intuitive :

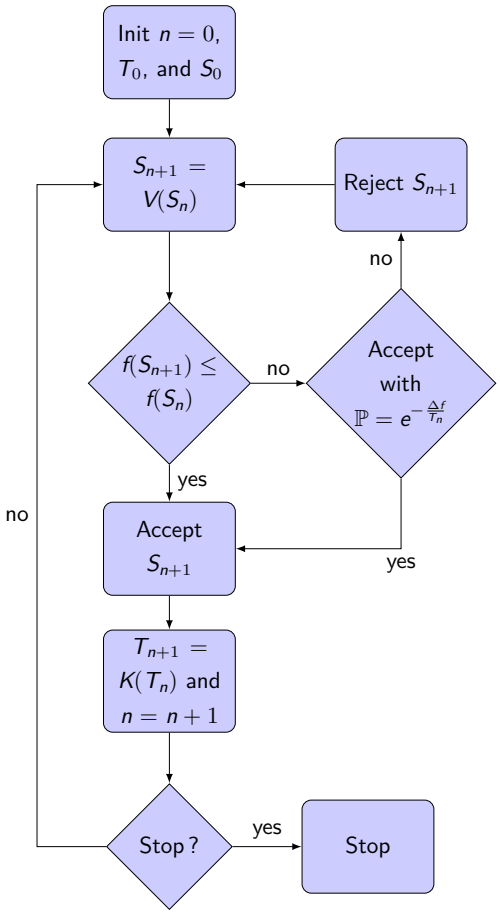
Au départ on commence avec une configuration arbitraire.

Soit w la configuration courante.

À chaque itération, une configuration candidate w' est choisie uniformément dans le voisinage de la solution courante ; et acceptée avec une probabilité $\min(1, e^{-\frac{c(w)-c(w')}{T}})$

Le paramètre T s'appelle la température et tend vers 0 (de manière monotone), selon une fonction appelée *loi de décroissance de la température*. La meilleure solution rencontrée durant l'exécution de l'algorithme est mémorisée et écrite lorsque la condition d'arrêt choisie est vérifiée.

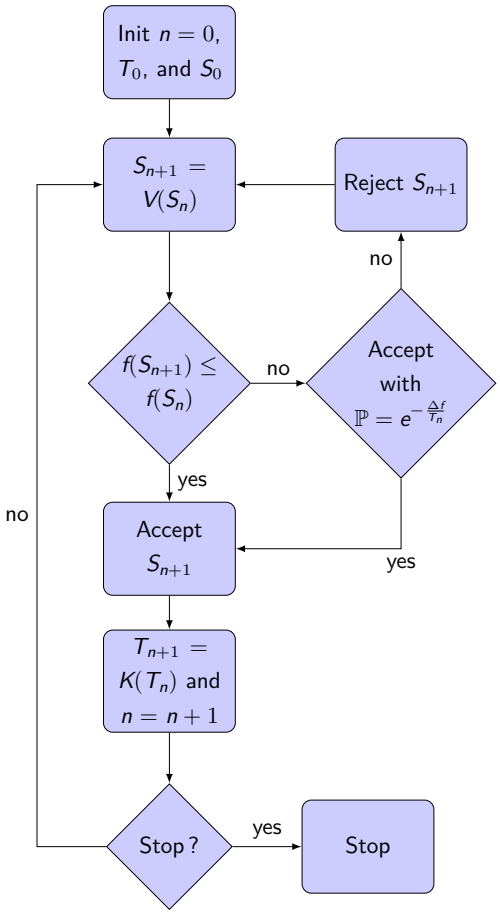
Recuit simulé, résumé



Paramètres usuels :

- S_0 aléatoire ou "évidente"
- $T_0 = 1000$
- $T_{n+1} = \lambda T_n$ avec $\lambda < 1$ $(\lambda = 0,99)$

Recuit simulé, résumé

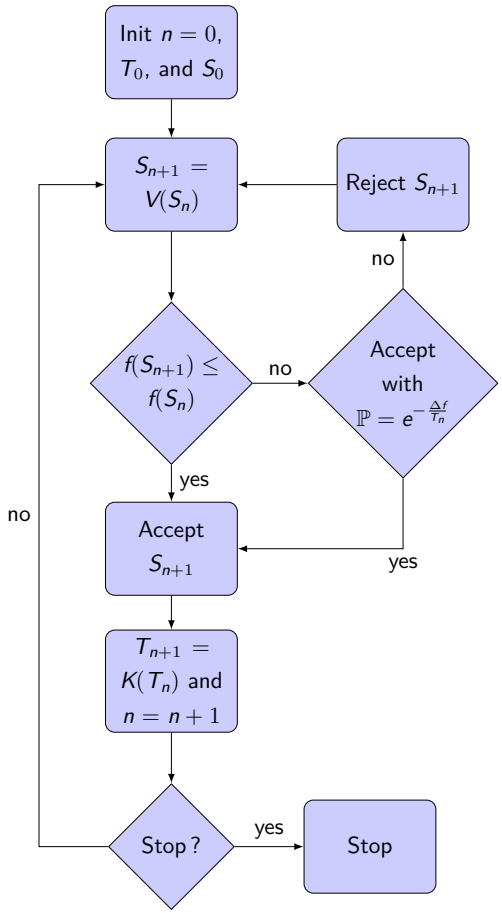


Paramètres usuels :

- S_0 aléatoire ou "évidente"
- $T_0 = 1000$
- $T_{n+1} = \lambda T_n$ avec $\lambda < 1$ $(\lambda = 0,99)$

Voisinage : assez spécifique, souvent en partie aléatoire

Recuit simulé, résumé

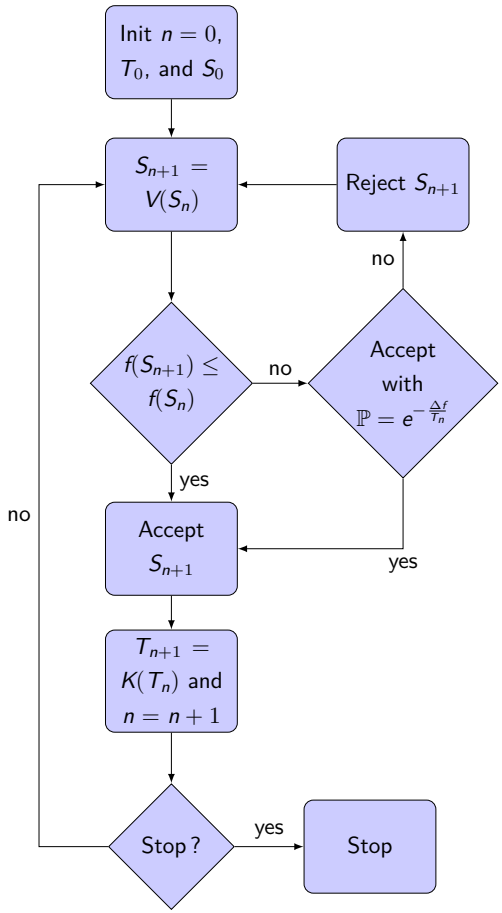


Paramètres usuels :

- S_0 aléatoire ou "évidente"
- $T_0 = 1000$
- $T_{n+1} = \lambda T_n$ avec $\lambda < 1$ $(\lambda = 0,99)$

Voisinage : assez spécifique, souvent en partie aléatoire doit prendre en compte la configuration actuelle (sinon grosso-modo c'est une recherche aléatoire (bogo-sort))

Recuit simulé, résumé



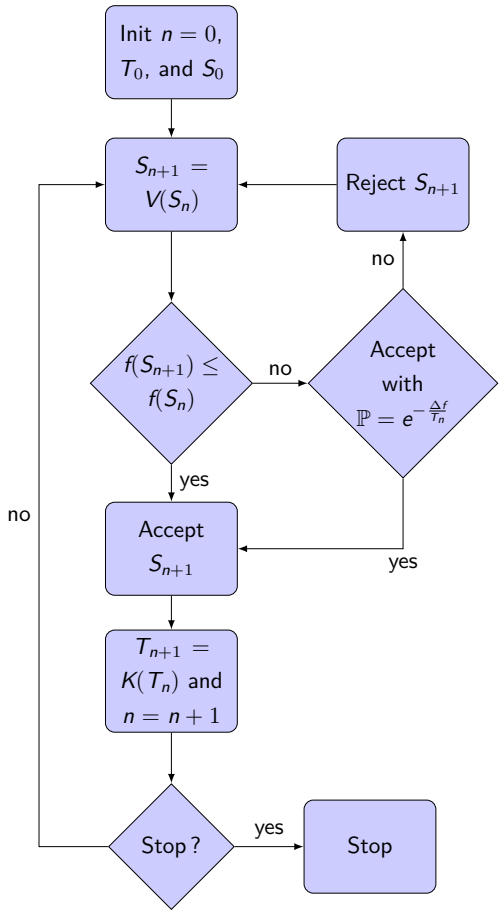
Paramètres usuels :

- S_0 aléatoire ou "évidente"
- $T_0 = 1000$
- $T_{n+1} = \lambda T_n$ avec $\lambda < 1$ $(\lambda = 0,99)$

Voisinage : assez spécifique, souvent en partie aléatoire doit prendre en compte la configuration actuelle (sinon grosso-modo c'est une recherche aléatoire (bogo-sort))

Au début : on prend souvent les configurations moins bonnes (T° haute), puis on refroidit et on ne est plus sélectifs

Recuit simulé, résumé



Paramètres usuels :

- S_0 aléatoire ou "évidente"
- $T_0 = 1000$
- $T_{n+1} = \lambda T_n$ avec $\lambda < 1$ $(\lambda = 0,99)$

Voisinage : assez spécifique, souvent en partie aléatoire doit prendre en compte la configuration actuelle (sinon grosso-modo c'est une recherche aléatoire (bogo-sort))

Au début : on prend souvent les configurations moins bonnes (T° haute), puis on refroidit et on ne est plus sélectifs

Condition(s) d'arrêt : nb essais fixés ou Température sous un seuil fixé ou Énergie sous un seuil fixé

Recuit simulé : exemple

Recherche du minimum d'une fonction $f : \mathbb{R} \to \mathbb{R}$

modélisation :

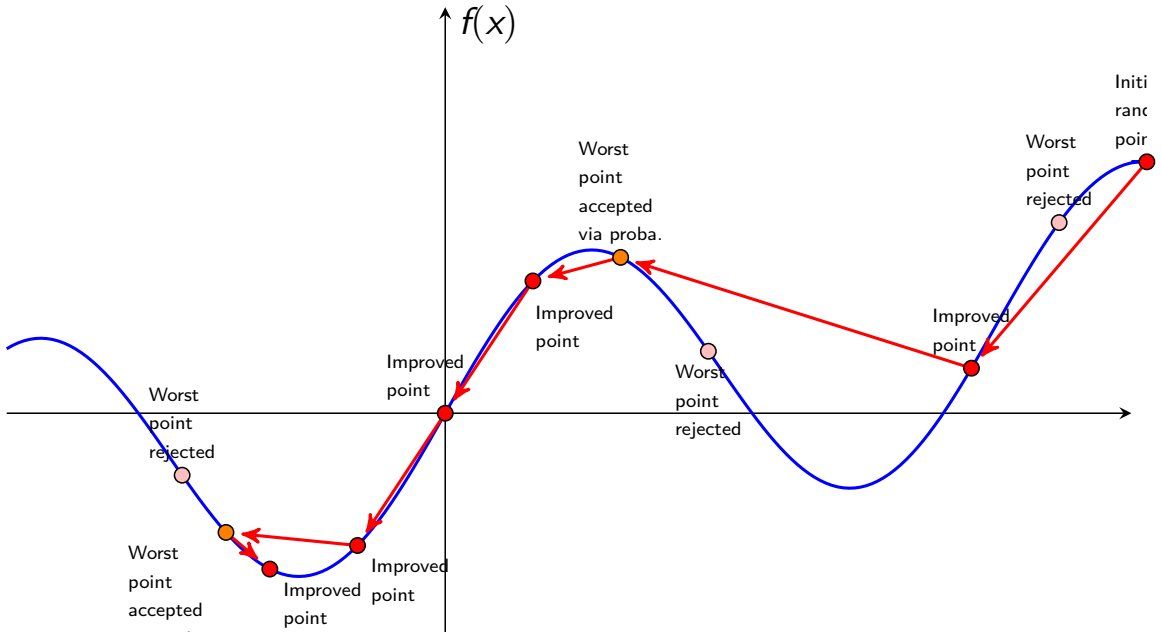
- un ensemble de configuration $\Omega = \mathbb{R}$
- une fonction de coût $c : f$
- une fonction de voisinage $V : x \to [x-1; x+1]$

traduction :

$\rightsquigarrow$ On cherche le minimum parmi tous les réels

$\rightsquigarrow$ Le coût est directement la valeur de la fonction f

$\rightsquigarrow$ On prend au hasard un nouveau point dans l'intervalle $[x-1; x+1]$

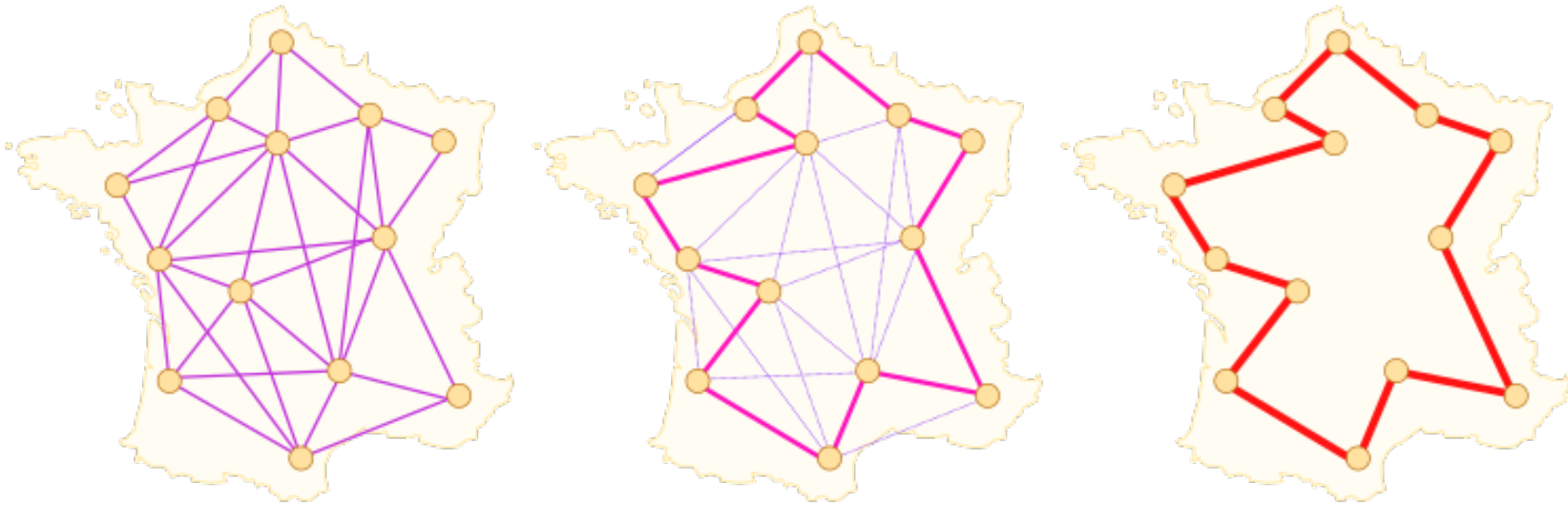


Travelling Salesperson Problem

problème du voyageur de commerce (TSP)

Entrée : graphe pondéré (distances)

Objectif : passer par toutes les villes en un minimum de distance

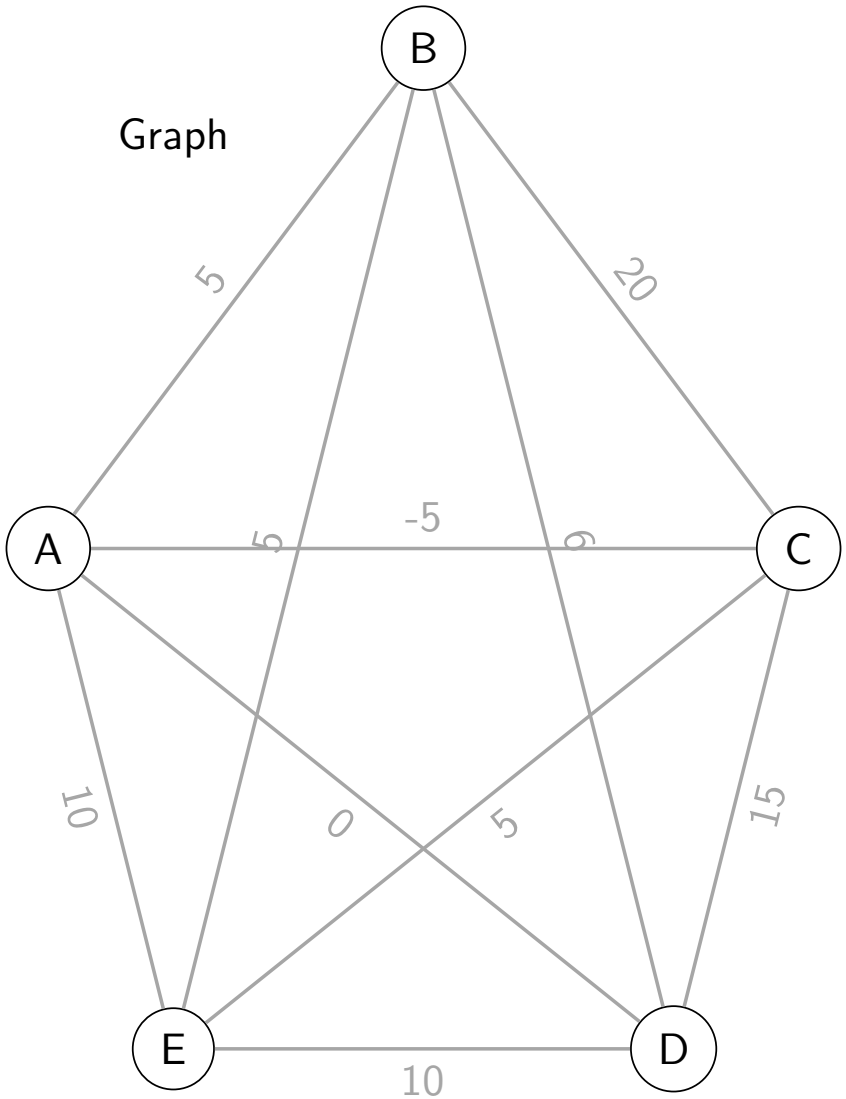


adapté de Nojhan, https://commons.wikimedia.org/wiki/File:Aco_TSP.svg

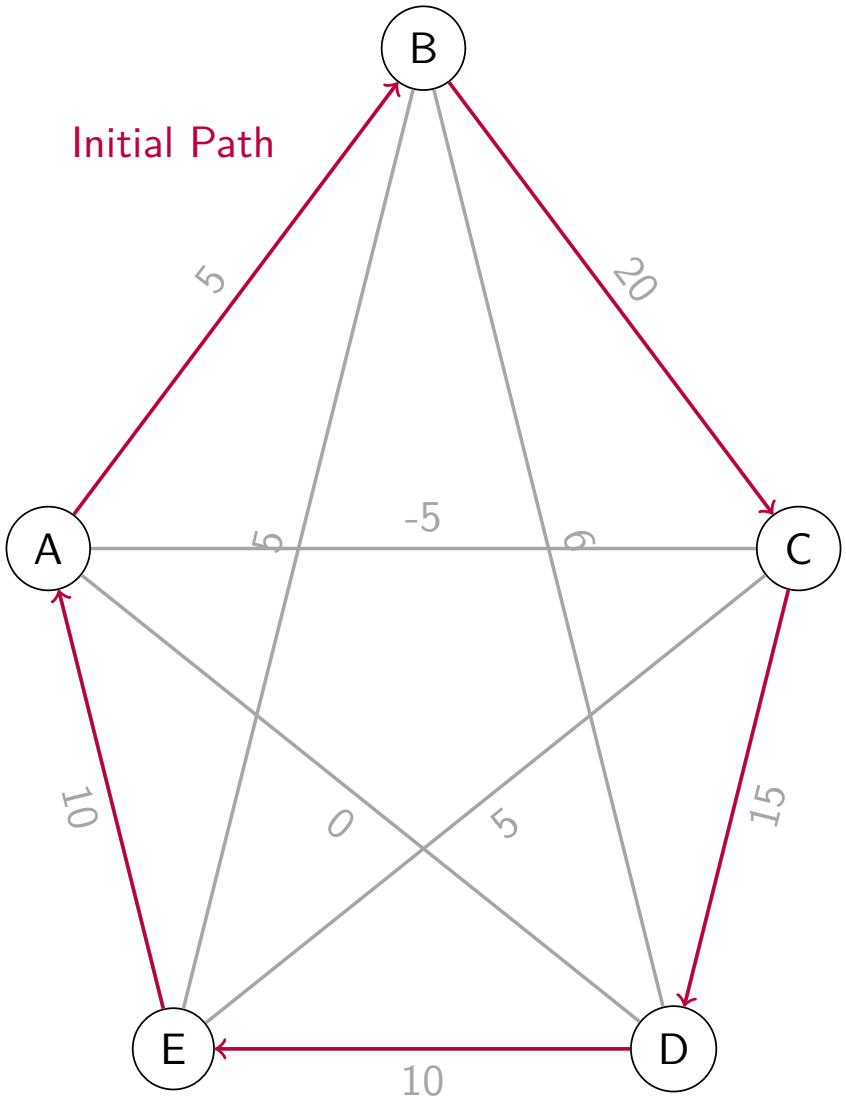
Remarque : le TSP est un problème classique **NP**-complet.

Simulated Annealing for TSP (5 Cities)

Cost Temp. Proba $e^{-\frac{\Delta f}{T_n}}$ Accpt ?

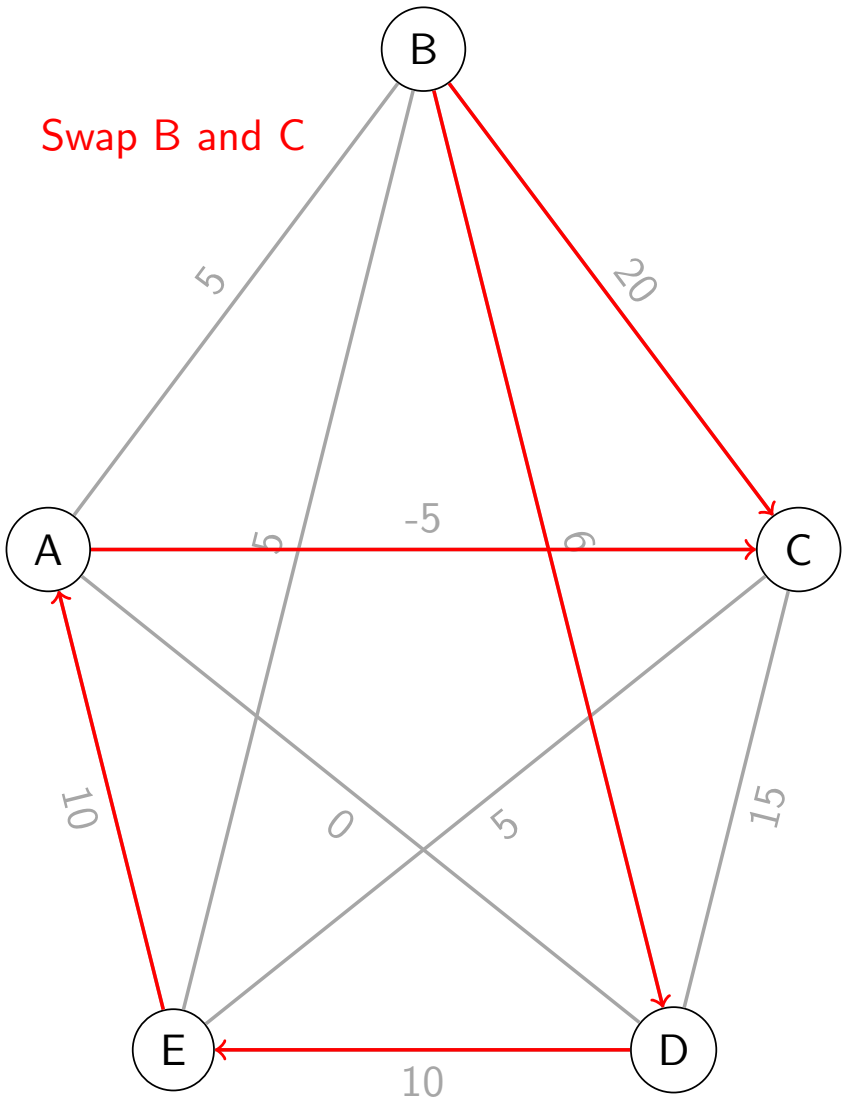


Simulated Annealing for TSP (5 Cities)



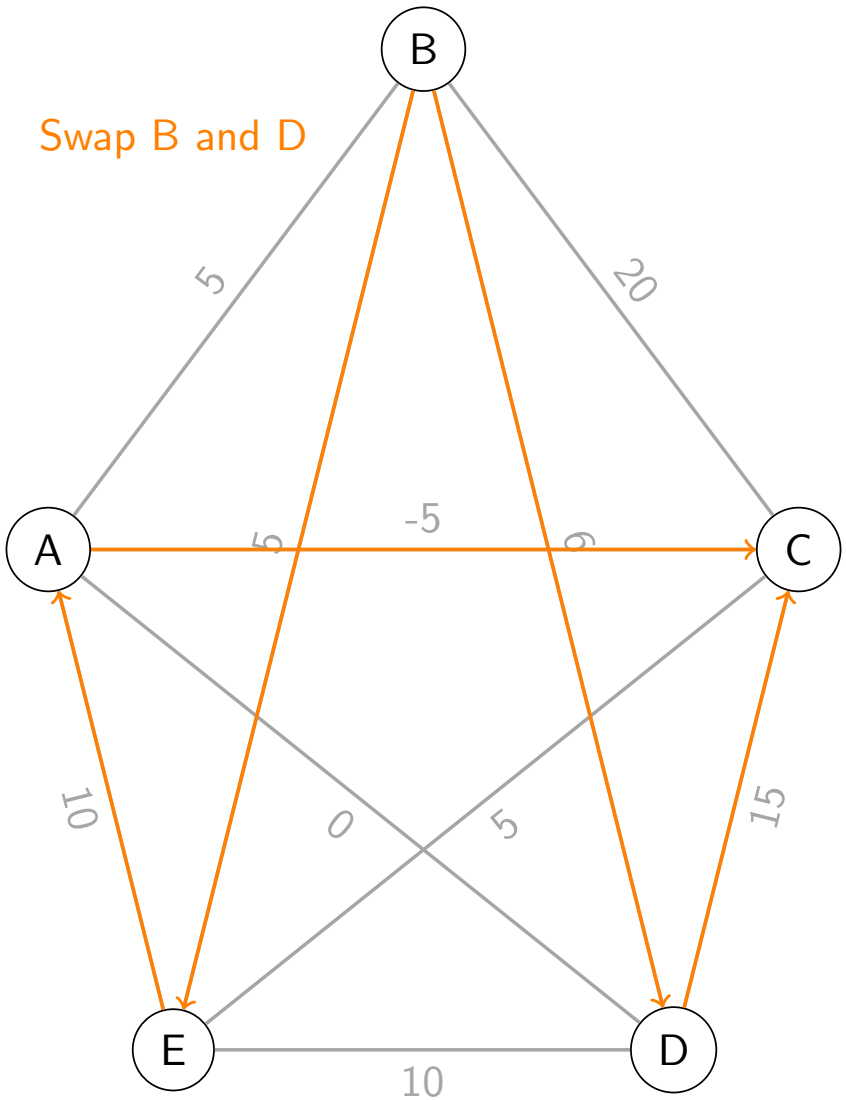
Cost	Temp.	Proba $e^{-\frac{\Delta f}{T_n}}$	Accpt ?
60			init.

Simulated Annealing for TSP (5 Cities)



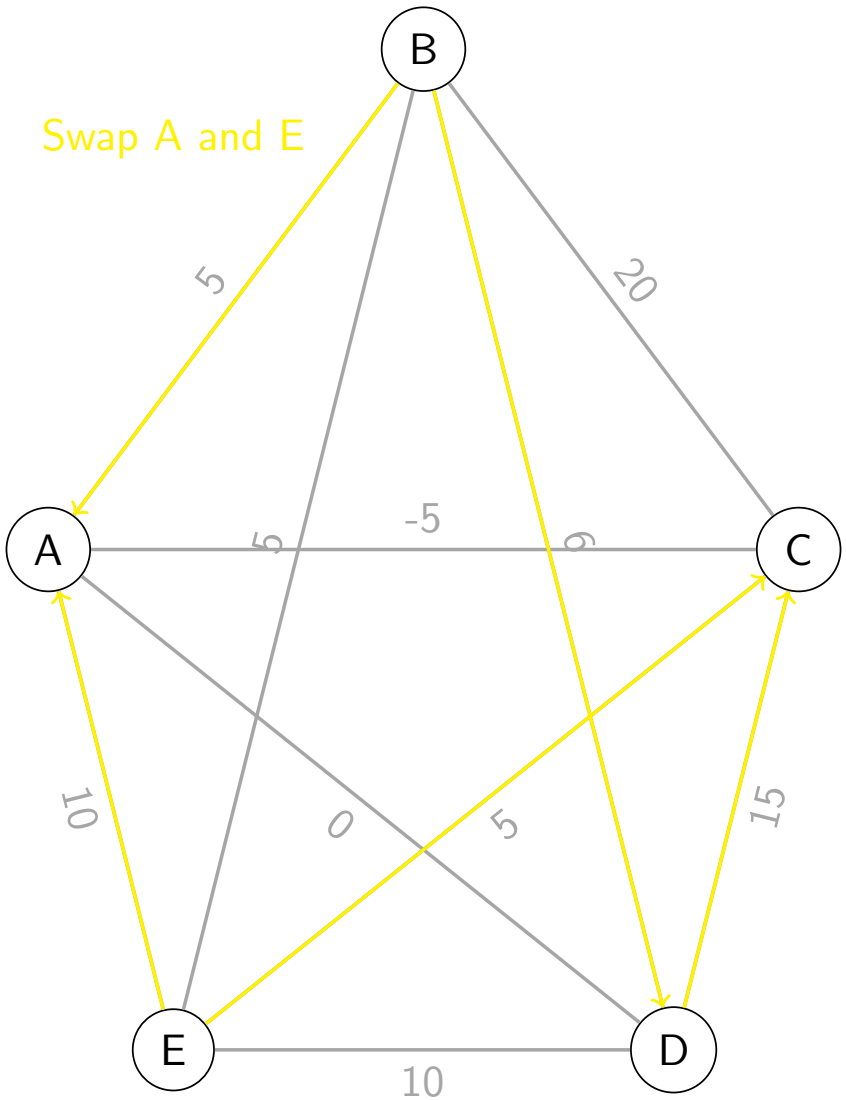
Cost	Temp.	Proba $e^{-\frac{\Delta f}{T_n}}$	Accpt ?
60			init.
46	1000	NA	✓

Simulated Annealing for TSP (5 Cities)



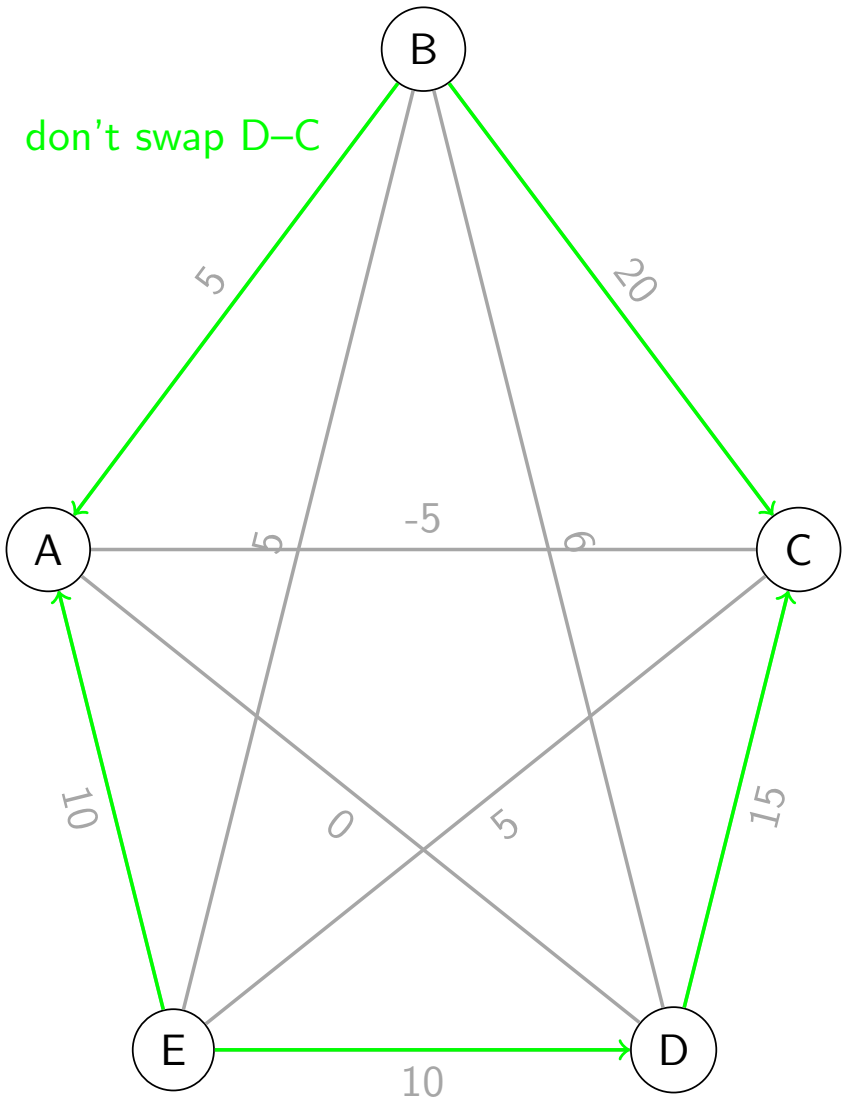
Cost	Temp.	Proba $e^{-\frac{\Delta f}{T_n}}$	Accpt ?
60			init.
46	1000	NA	✓
31	980	NA	✓

Simulated Annealing for TSP (5 Cities)



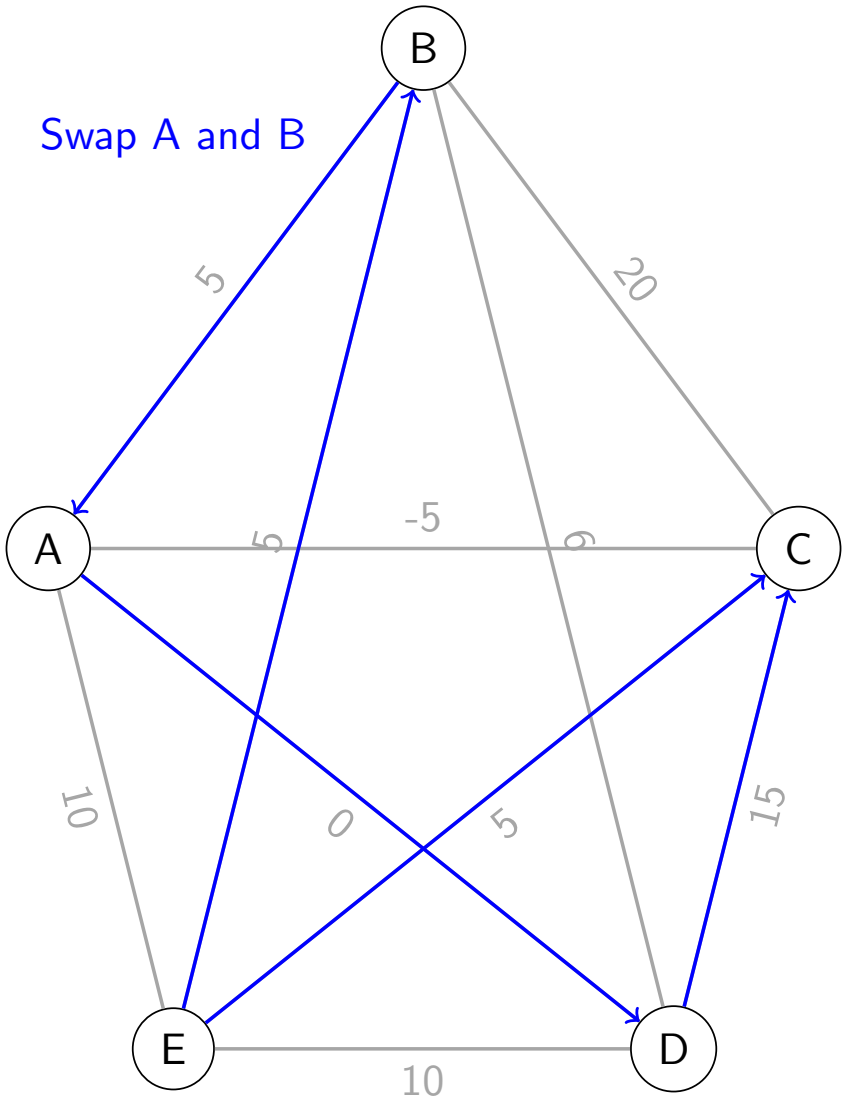
Cost	Temp.	Proba $e^{-\frac{\Delta f}{T_n}}$	Accpt ?
60			init.
46	1000	NA	✓
31	980	NA	✓
41	970	$0,989$	✓

Simulated Annealing for TSP (5 Cities)



Cost	Temp.	Proba $e^{-\frac{\Delta f}{T_n}}$	Accpt ?
60			init.
46	1000	NA	✓
31	980	NA	✓
41	970	$0,989$	✓
60	961	$0,980$	×

Simulated Annealing for TSP (5 Cities)



Cost	Temp.	Proba $e^{-\frac{\Delta f}{T_n}}$	Accpt ?
60			init.
46	1000	NA	✓
31	980	NA	✓
41	970	$0,989$	✓
60	961	$0,980$	×
30		NA	✓