Chapter 1

Groupe

- Félix Abecassis (CSI)
- Christopher Chedeau (CSI)
- Gauthier Lemoine (SCIA)
- Julien Marquegnies (CSI)

Nous avons choisi d'implémenter le projet avec le langage Javascript. L'avantage offert par l'utilisation de ce langage est que nous pouvons vous proposer des démonstrations interactives sans avoir à se soucier de problèmes de compilation qui pourraient survenir avec d'autres langages. Seul un navigateur est ici nécessaire. Nous recommandons l'utilisation d'un navigateur récent afin d'obtenir les meilleures performances. Pour l'affichage des courbes 3D, il est nécessaire d'utiliser un navigateur supportant WebGL (par exemple Chrome).

La démonstration du problème de placement de composants électroniques (placement-routage) est accessible à cette adresse: http://fooo.fr/~vjeux/epita/recuit/recuit.html

La démonstration de la recherche de minima globaux sur des fonctions à deux paramètres classiques est accessible à cette adresse: http://fooo.fr/~vjeux/epita/recuit/3d.html

Chapter 2

Recuit simulé

2.1 Principe

Le recuit simulé est une métaheuristique probabiliste permettant d'approximer l'optimum global d'une fonction objectif. Cette technique est souvent utilisée lorsque le calcul de la solution optimale exacte demanderait un temps de calcul trop important, le recuit simulé est alors utilisé pour trouver une solution approchée dans un temps raisonnable.

Le nom et le principe de cet métaheuristique sont inspirés directement de la technique du recuit en metallurgie. Le recuit permet de diminuer les défauts d'un materiau en augmentant sa température puis en contrôlant le refroidissement par un apport externe de chaleur. Un refroidissement naturel ne permet pas d'atteindre la configuration optimale des atomes pour obtenir la meilleure stabilité. De manière analogue, pour le recuit **simulé**, nous parlerons de l'energie du système, nous cherchons à atteindre la configuration du système qui posséde l'energie la plus faible.

A chaque itération du recuit **simulé**, la solution actuelle est remplacée par une autre solution issue du voisinage de la solution initiale. Cette nouvelle solution est acceptée avec une probabilité dépendant à la fois de la différence entre les valeurs de coût des deux solutions et d'un paramètre **T** appelé température. Cette température décroit progressivement au cours du temps selon une loi déterminée préalablement. Il y a donc toujours une probabilité de dégrader la solution lors d'une itération. Cette stratégie permet d'éviter de rester coincer dans un minimum local.

Le recuit simulé est une technique générique qui peut être appliquée aussi bien à des problèmes discrets que des problèmes continus. Dans ce rapport nous présentons tout d'abord un exemple de résolution d'un problème discret: le problème du **placement-routage**, puis nous appliquons le recuit simulé afin de rechercher les minima globaux de fonctions à 2 variables classiques. Globalement, le recuit simulé est plus souvent utilisé dans des cas discrets, on peut par exemple citer également le problème du voyageur de commerce ou encore le problème du **flow shop scheduling**.

2.2 Algorithme

À chaque itération de l'algorithme, le recuit simulé examine un état voisin **s'** de l'état courant **s** et détermine de facon stochastique si le système doit effectuer la transition vers cet état ou rester dans la configuration actuelle.

Ce processus amène progressivement le système vers une configuration de plus faible energie. Cette étape est répetée jusqu'à que le système soit figé sur la même solution pendant plusieurs itérations ou lorsque qu'une limite d'itérations a été depassée.

Le voisinage d'un état est généré en effectuant des modifications élémentaires de la configuration du système. Pour obtenir de bons résultats il est indispensable que ces modifications soient les plus simples possibles, les deux configurations doivent rester proches. Par exemple dans le cas du voyageur de commerce, une modification élémentaire est la permutation de deux villes adjacentes. L'exploration du voisinage est l'étape la plus importante du recuit, elle doit se faire de facon optimisée.

La probabilité de transition depuis un état **s** vers un état **s'** est donnée par une fonction d'acceptation **P(e, e', T)**. Avec **T** la température du système, **e** l'energie de l'état initial et **e'** l'energie de l'état voisin. Cette fonction renvoie alors la probabilité de transition vers cet état. Généralement, si la nouvelle solution est meilleure, la probabilité de transition est 1, mais ce n'est pas toujours le cas selon les versions de l'algorithme. Si l'energie de l'état voisin est plus forte, on utilise la règle de Metropolis-Hastings, la probabilité de transition est alors:

$$e^{-\frac{\Delta E}{T}} \tag{2.1}$$

Δ_E est la différence d'energie entre les deux états. Initialement **T** est très elevée donc l'algorithme va souvent dégrader la solution. Au fil des itérations, **T** décroit et les dégradations importantes ne sont plus acceptées.

2.3 Pseudocode

Voici le pseudocode de la version la plus simple du recuit simulé:

```
// Solution initiale
s ← s_0;
e ← E(s);
// Température initiale
T ← T_0;
// Iteration tant qu'on a pas atteint une energie minimale
// ou que le nombre d'itérations maximale n'est pas depassé
i ← 0;
while (i < max_iters) ∧ (e > e_max) do
    voisin ← choisir_voisin(s);
    ev ← E(voisin);
    // Faut il effectuer la transition ?
    if P(e, ev, T) > rand() then
        s ← voisin;
        e ← ev;
    end
    i ← i + 1;
    T ← diminuer_temperature(T);
end
return s
```

Algorithm 1: Pseudocode du recuit simulé

Dans le chapitre suivant nous allons maintenant montrer l'application du recuit simulé aux deux problèmes proposés.

Chapter 3

Placement de composants

3.1 Énoncé

Le probléme consiste à trouver la configuration optimale de connexion entre différents blocs d'un circuit integré. C'est un problème d'optimisation difficile nécessitant l'utilisation de métaheuristiques pour trouver une solution approchée. Lors d'une application réel les temps de calculs sont très elevés vu le grand nombre d'éléments à traiter.

Dans notre cas les blocs sont disposés de facon regulière sur une grille carrée. La fonction coût est calculée en effectuant la somme des coûts de chaque connexion. Le coût d'une connexion est égal à la distance de Manhattan entre les deux blocs de cette connexion. Pour ce problème, la distance entre deux blocs est fixée à 5. Ainsi, pour une grille de taille 5×5, la valeur optimale est de 200.

Ce problème se prête bien à une résolution par recuit simulé. Nous effectuons une initialisation aléatoire de la grille puis nous appliquons l'algorithme vu précedemment. Le seul mouvement élémentaire disponible est la la permutation des positions de 2 blocs de la grille. Les blocs à permuter sont choisis aléatoirement à chaque étape du recuit simulé.

Ce problème est aussi appelé **Placement-Routage** (**Place and Route**).

3.2 Visualisation

La démonstration de la résolution du problème est accessible sur internet à l'adresse suivante: http://fooo.fr/~vjeux/epita/recuit/recuit.html

La démonstration en ligne permet de visualiser en temps réel l'évolution des positions des blocs sur la grille. Sur la droite, un graphique permet de visualiser l'évolution de la fonction coût au cours du temps. L'affichage est mis à jour en temps réel, ce qui permet d'avoir une bonne vision de la transformation désordre / ordre qui s'opère lors de l'exécution du recuit simulé.

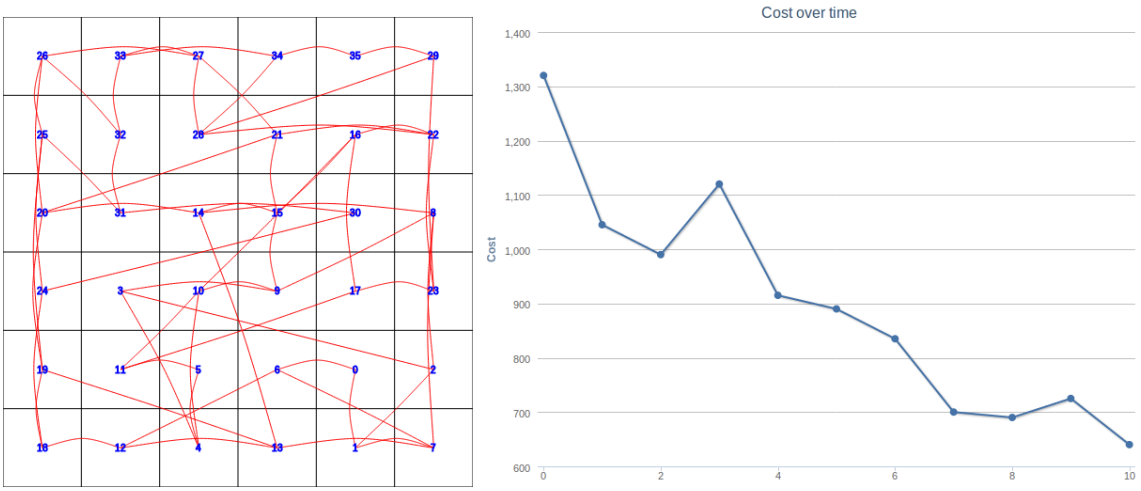


Figure 3.1: Affichage de l'évolution de l'algorithme.

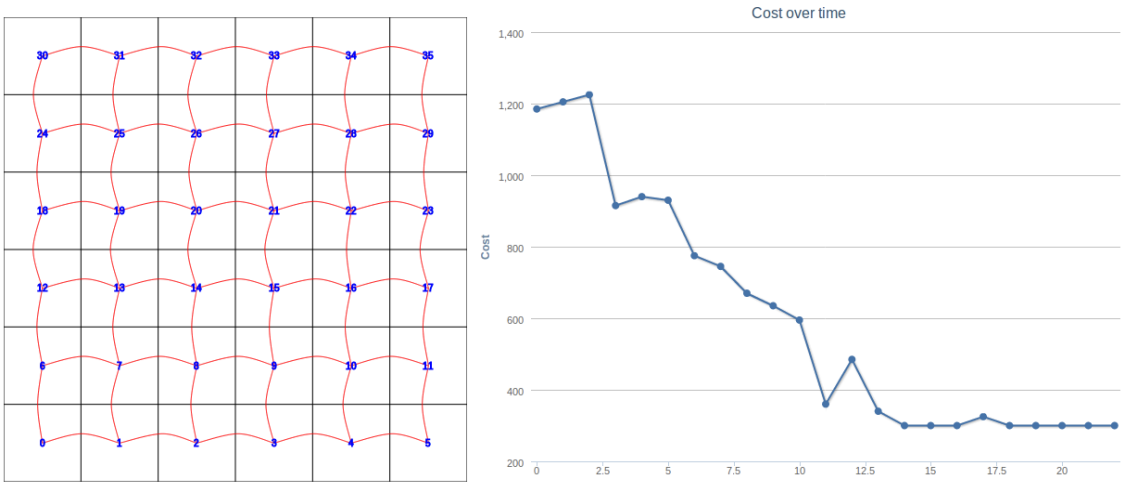


Figure 3.2: Convergence vers la solution optimale.

3.3 Paramètrage

Il est possible de modifier les paramètres du problème comme:

- La taille de la matrice de blocs, la matrice est forcément carrée donc si on rentre la valeur A nous aurons A^2 blocs. La complexité du problème augmente donc très rapidement. On comprend alors la nécessité de recourir à des métaheuristiques pour la résolution du problème.
- La valeur initiale de τ (tau). Pour trouver la température initiale nous appliquons la méthode vue en cours. Nous prenons 100 perturbations aléatoires et nous calculons la moyenne des ΔE obtenus. Soit M la valeur de cette moyenne, la température initiale est alors:
$$e^{\frac{-M}{\tau}} \tag{3.1}$$
- Le taux de décroissance de la température. Cette décroissance est géométrique. Soit C la valeur choisie, lors d'une nouvelle itération la nouvelle température T' est alors tout simplement:
$$T' = C \times T \tag{3.2}$$
- Le nombre de tentatives par itération (c'est à dire entre 2 changements de température). Cette valeur, divisée par 100, correspond à la valeur N de l'équilibre thermodynamique. Lorsque l'équilibre thermodynamique est atteint, la température diminue. Il y a 2 critères pour atteindre cet équilibre: soit $12 \times N$ perturbations ont été acceptées, soit $100 \times N$ essais de permutation ont été realisés. Ainsi, par exemple avec $N = 500$, on s'arrête soit après 5000 essais soit après 600 acceptations.

  Si aucun changement n'est accepté pendant 4 itérations successives, l'algorithme s'arrête.

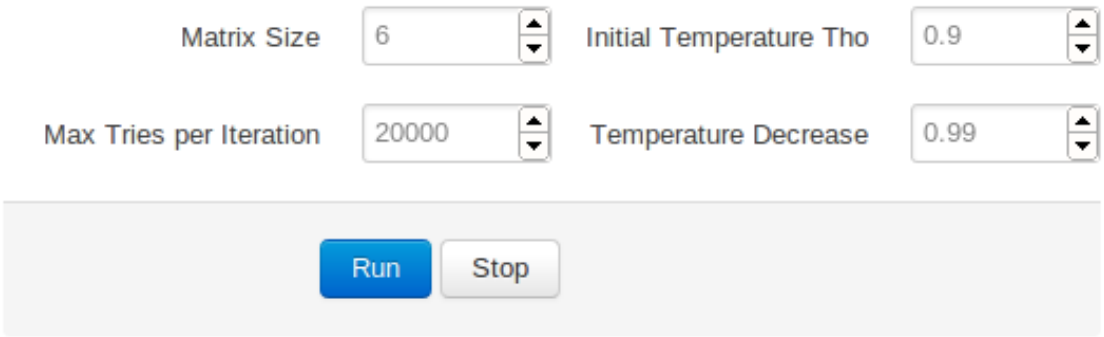


Figure 3.3: Paramètrisation de l'algorithme.

3.4 Statistiques

Des statistiques sur l'évolution de l'algorithme sont également affichées à droite. Notamment, la température actuelle et le nombre de transitions acceptées sont affichés. La solution optimale attendue est aussi rappelée puisque nous la connaissons pour ce problème. Les statistiques sont affichées pour l'itération courante et depuis le début de l'algorithme. Ainsi on peut voir combien de perturbations sont acceptées à chaque itération, c'est un bon indicateur afin de vérifier si les paramètres ont été bien fixés.

	Current		Total
Cost (Optimal: 300)	585	**Time**	10s
Tried	20000	**Tried**	1 726 820
Accepted	1723	**Accepted**	761 420
Temperature	3.3845	**Speed**	167 571 try/s

Figure 3.4: Statistique d'éxecution

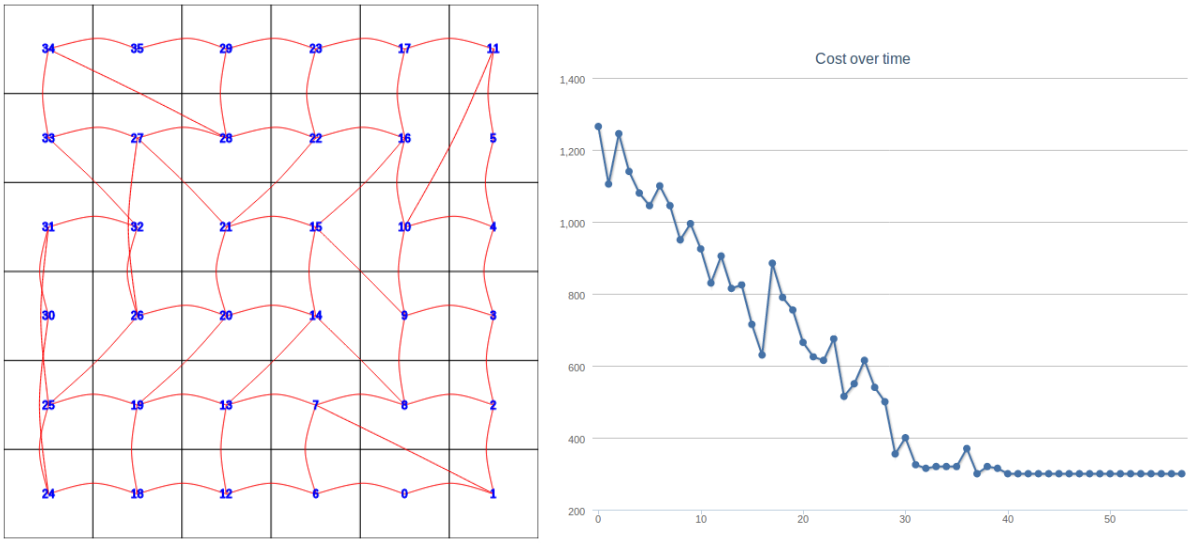
3.5 Influence du critère thermodynamique

Ce paramètre influence fortement le temps de calcul du recuit simulé. La valeur utilisée doit dépendre du problème consideré.

Si cette valeur est trop forte, nous restons trop souvent avec la même température, et lorsque celle ci est forte, la configuration du systéme va changer totalement lors d'une itération puisque beaucoup de perturbations sont acceptées. De plus, lorsque la température est faible, des milliers d'essais vont avoir lieu pour n'accepter que quelques perturbations. Nous perdons alors tout l'intêret du recuit simulé. Si cette valeur est trop faible, la température va décroître trop rapidement et le minimum global ne sera pas atteint. Il faut donc bien choisir cette valeur afin d'obtenir un bon compromis entre temps d'exécution et efficacité.

Pour une taille de grille de 6, il faut choisir comme valeur au moins 2000 ($N = 20$) afin de ne pas risquer de tomber dans un minimum local.

Dans la figure de gauche, nous montrons la configuration obtenue avec $N = 10$. Le graphique de droite représente l'évolution de la température avec $N = 500$. Nous pouvons observer que le temps d'exécution est de plus de 60 secondes ! Avec $N = 20$ la convergence se fait en 2 secondes seulement.



(a) $N = 10$

(b) $N = 500$ (optimal atteint)

Figure 3.5: Influence de N.

3.6 Influence de τ

Le choix de τ se fait de facon empirique et dépend de la confiance que nous avons dans la configuration initiale. Avec $\tau = 0.9$ la configuration initiale est rapidement détruite puisque la température est élevée au départ. Nous pouvons utiliser une valeur plus faible de τ, lorsque la configuration initiale est présumée bonne (fournie par un expert par exemple), les optimisations seront alors plus locales et une bonne partie de la configuration initiale sera preservée.

Dans notre cas, l'initialisation est totalement aléatoire mais une valeur assez faible de τ n'empêche pas toujours la convergence vers le minimum global. Une faible valeur de τ peut en effet être compensée par une lente décroissance de la température ou une valeur de N élevée. Avec $C = 0.99$, une faible valeur de de τ est généralement compensée par le fait que la température diminue très peu.

Avec $N = 50$, $\tau = 0.5$ et $C = 0.9$, la température chute beaucoup trop rapidement et donc le minimum global n'est pas atteint.

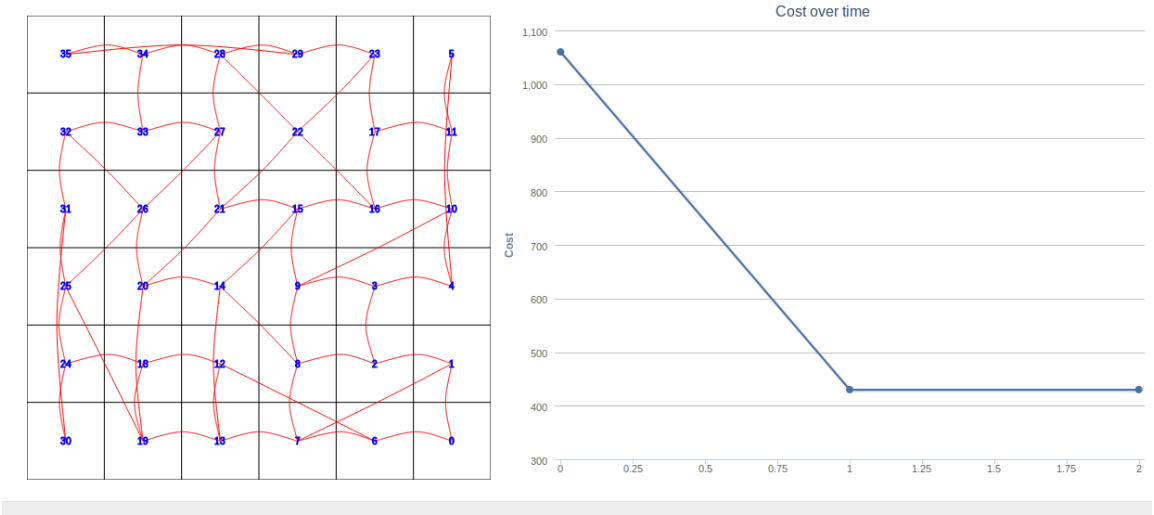
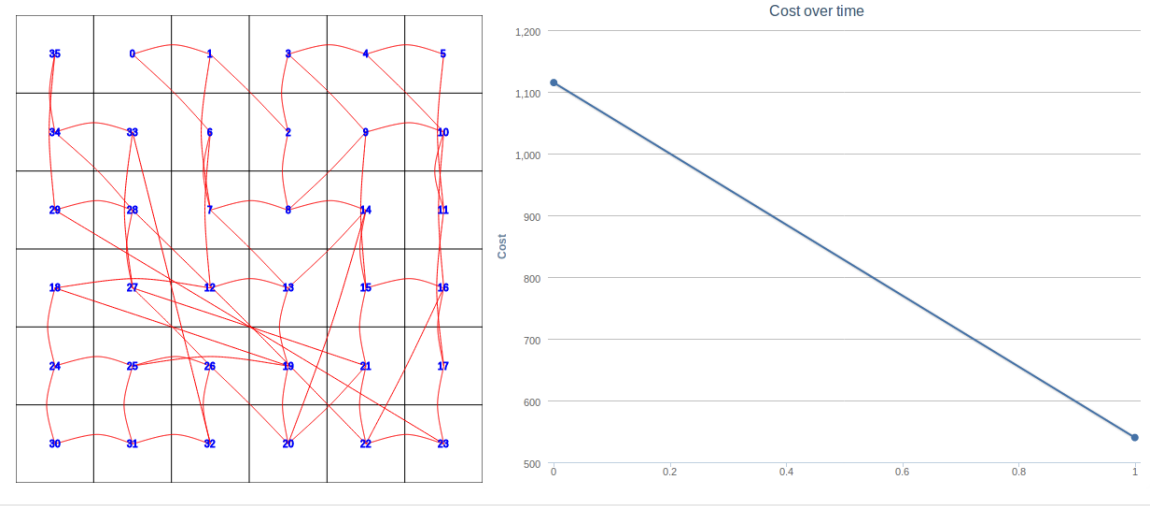


Figure 3.6: $N = 50$, $\tau = 0.5$ et $C = 0.9$

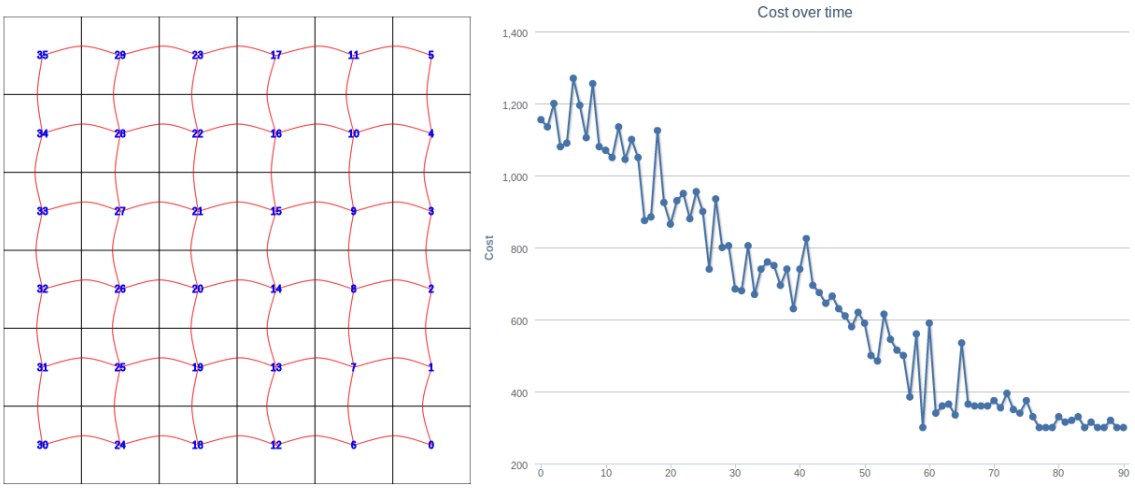
3.7 Influence de C

La décroissance de la température est un paramètre extrêmement sensible. Si la valeur est trop faible, très peu d'itérations seront realisées et le minimum global ne sera pas atteint. Si elle est trop forte, la convergence sera beaucoup trop longue. Pour 36 blocs, $N = 50$ et $\tau = 0.9$, nous avons estimé de manière empirique que la valeur de C doit être comprise entre 0.95 et 0.99. De manière générale, 0.99 semble être une bonne valeur quelque que soit le nombre de blocs utilisés.

Pour $C = 0.5$ nous pouvons obtenir des résultats désastreux comme illustré sur la figure suivante. Dans l'image du dessous, nous avons fixé $C = 0.999$ et $N = 100$, nous observons ici que l'algorithme met plus de 90 secondes à s'exécuter et nous observons de fortes oscillations du coût pendant longtemps alors que idéalement ces fortes oscillations ne devraient pouvoir avoir lieu qu'au début, cela provient du fait que la température reste relativement élevée pendant longtemps.



(a) $C = 0.5$



(b) $C = 0.999$

Figure 3.7: Influence de C

Chapter 4

Optimisation à variables continues

4.1 Énoncé

Notre implémentation a ensuite été testée sur un problème à variables continues. L'objectif était de rechercher le minimum global de plusieurs fonctions de référence. Certaines de ces fonctions sont définies pour N variables, mais nous avons limité ces fonctions à deux variables afin de pouvoir visualiser l'évolution du recuit en 3D.

Concernant le choix des paramètres du recuit, les observations du problème précédent s'appliquent toujours. Il s'agissait surtout ici de vérifier le bon fonctionnement de notre algorithme pour le cas continu. Ces fonctions étant de référence, les minima globaux sont également connus, il est donc facile de vérifier les résultats obtenus.

4.2 Visualisation

La démonstration de la résolution du problème est accessible sur internet à l'adresse suivante: http://fooo.fr/~vjeux/epita/recuit/3d.html

Comme pour le problème précédent il est possible de fixer les différents paramètres. Cette fois il est également nécessaire de choisir quelle fonction de référence utiliser.

La représentation 3D correspondant à la fonction choisie est tout d'abord affichée. Lors de l'exécution de l'algorithme, nous affichons par des points bleus sur la surface les différentes positions du recuit lors des précédentes itérations.

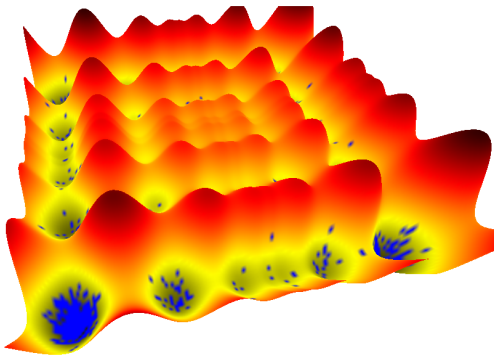


Figure 4.1: Exemple d'exécution

4.3 Michalewicz

Dans l'énoncé il manquait un signe - avant la somme. L'équation corrigée est donc:

$$\mathrm{MZ}(x,y) = -\left(\sin(x) \times \left[\frac{\sin(x^2)}{\pi}\right]^{20} + \sin(y) \times \left[\frac{\sin(2y^2)}{\pi}\right]^{20}\right) \quad (x,y) \in [0,\pi] \tag{4.1}$$

Le minimum global de la fonction, -1.8, est atteint par notre recuit:

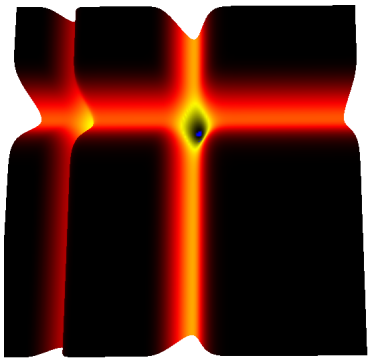


Figure 4.2: Minimum global de la fonction de Michalewicz

4.4 De Jong F1

$$\mathrm{F1}(x,y) = x^2 + y^2 \quad (x,y) \in [-5.12, 5.12] \tag{4.2}$$

Le minimum global de la fonction, 0, est atteint par notre recuit:

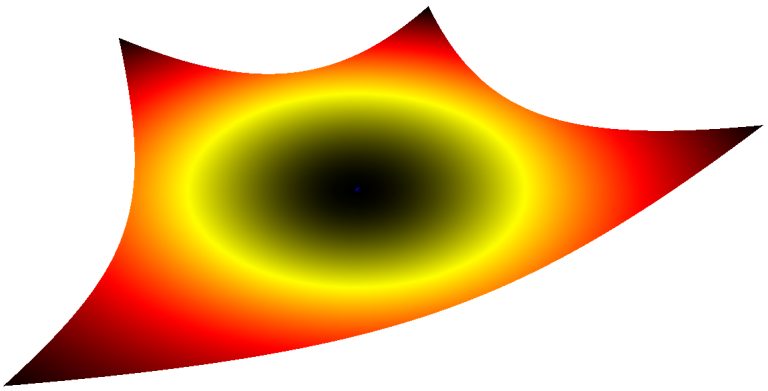


Figure 4.3: Minimum global de la fonction De Jong F1

4.5 De Jong F2

L'équation proposée était erronée, cependant l'équation corrigée est celle donnée pour Rosenbrock et donc détaillée plus loin.

4.6 De Jong F3

$$\mathrm{F3}(x, y) = \mathrm{floor}(x) + \mathrm{floor}(y) \quad (x, y) \in [-5.12, 5.12] \tag{4.3}$$

Le minimum global de la fonction, -12, est atteint par notre recuit:

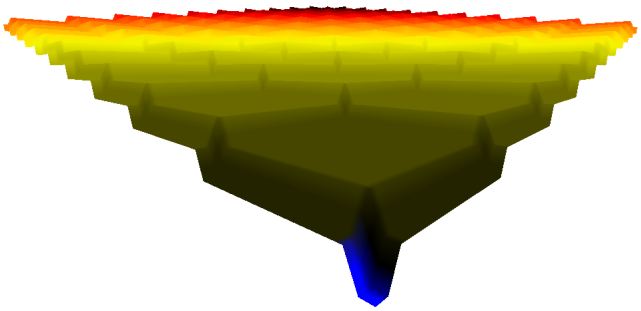


Figure 4.4: Minimum global de la fonction De Jong F3

4.7 Goldstein and Price

L'équation proposée comportait 2 erreurs, la bonne équation est:

$$\begin{aligned} \mathrm{GP}(x, y) \quad = \quad & \left[1 + (x + y + 1)^2 \times (19 - 14x + 3x^2 - 14y + 6xy + 3y^2)\right] \\ & \times \left[30 + (2x - 3y)^2 \times (18 - 32x + 12x^2 + 48y - 36xy + 27y^2)\right] \\ & (x, y) \in [-2, 2] \end{aligned} \tag{4.4}$$

Le minimum global de la fonction, 3, est atteint par notre recuit:

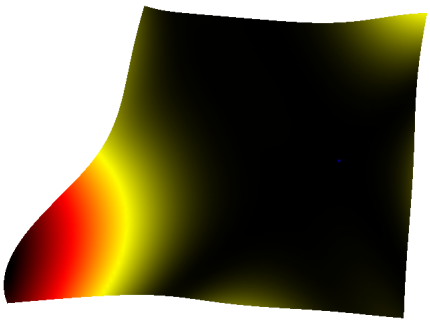


Figure 4.5: Minimum global de la fonction de Goldstein and Price

4.8 Rosenbrock

Il manquait un carré dans l'équation:

$$\mathrm{R}(x,y) = (1-x)^2 + 100 \times (y-y^2)^2 \quad (x,y) \in [-2.048, 2.048] \tag{4.5}$$

Le minimum global de la fonction, 0, est atteint par notre recuit:

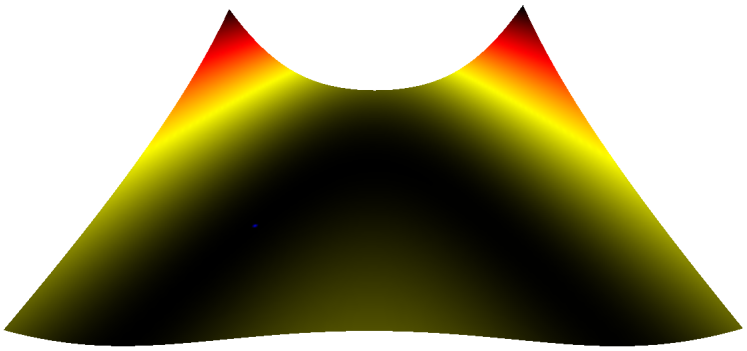


Figure 4.6: Minimum global de la fonction de Rosenbrock

4.9 Zakharov

$$\mathrm{Z}(x,y) = x^2 + y^2 + [0.5x + y]^2 + [0.5x + y]^4 \quad (x,y) \in [-5, 10] \tag{4.6}$$

Le minimum global de la fonction, 0, est atteint par notre recuit:

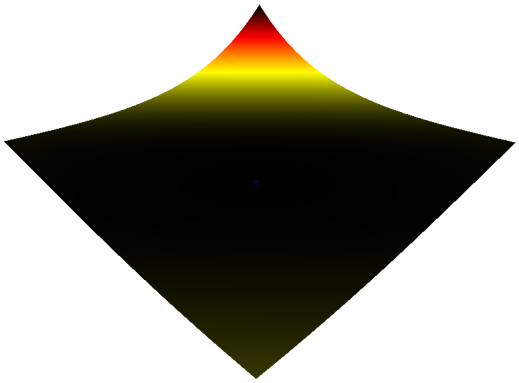


Figure 4.7: Minimum global de la fonction de Zakharov

4.10 Schwefel

$$\mathrm{SH}(x, y) = -x \times \sin(\sqrt{|x|}) - y \times \sin(\sqrt{|y|}) \quad (x, y) \in [-500, 500] \tag{4.7}$$

Le minimum global de la fonction, -837.96, est atteint par notre recuit:

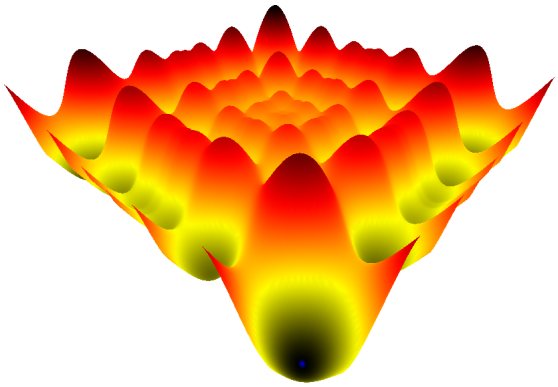


Figure 4.8: Minimum global de la fonction de Schwefel