

Définitions

Objective Function → la fonction des coûts à optimiser.

Minimisation possible:

- > Date d'échéance de la dernière tâche du problème.
- > Nombre de ressources.
- > Prix de la production.
- > Nombre de contraintes violées.

Voisinage → Ensemble des config. qui peuvent être obtenues par une transformation locale de la config. actuelle.

Exemples :

- > Coloration de graphes: changer une couleur
- > SAT: "flip" d'une var. booléenne
- > CSP: modification de la valeur d'une variable

Modèles SAT et CSP → Résoudre le problème en minimisant les conflits dans

MAX-SAT (minimisation du nombre de clauses violées) et MAX-CSP (minimisation du nombre de contraintes violées)

Recherche locale → amélioré une config. courante par des transformations locales itératives.

Inconvénient d'un méta heuristique

Incomplétude → la recherche n'est pas systématique (toutes les possibilités ne sont pas essayées) => aucune preuve garantie que la meilleure solution a été trouvée.

Optimum local → une recherche locale peut être bloquée à l'intérieur d'un optimum local ou sur un plateau et visiter plusieurs fois la même config.

Sensibilité à la config. Initiale

Optimisation possibles

GSAT → Interrompre la recherche en cours et réessayer avec une nouvelle config.

Algo génétique → Gérer plusieurs config. en parallèle.

Recherche Tabu → Enregistrez les derniers mouvements pour éviter de revenir en boucle sur les mêmes config.

Recuit simulé → Accepter parfois une config. qui donne une moins bonne config.

IDWalk → N'utilisez que la gestion des voisins pour intensifier ou diversifier la recherche.

Simulated Annealing (SA) method

```
Algorithm SA-Move(cur: current configuration, T: a temperature,
Min-Neighbors : number of neighbors) Returns: neighbor configuration
best-cost ← +∞; x-best ← cur
i ← 0; accepted? ← false
while (i < Min-Neighbors) do
  x ← Generate-Neighbor(cur)
  if cost(x) ≤ cost(cur) or Random() < exp( $\frac{-\Delta}{T}$ ) then
    accepted? ← true
  end
  if cost(x) < best-cost then
    x-best ← x
    best-cost ← cost(x)
  end
  i ← i + 1
end
if accepted? then
  return x-best
else
  return cur
end
end.
```

T → la température, c'est le param. le plus important :

- > Une température élevée permet à l'algo. de s'échapper des minima locaux.
 - > Une température basse fait de l'algo. un algo. gourmand.
- La température doit diminuer progressivement.

Δ → le degré de détérioration du critère, par exemple, le nombre supplémentaire de contraintes violées dans MAX-CSP.

The Tabu Search (TS) method

```
Algorithm TS-Move(cur: current configuration, in-out tabu-list, L: max length of
tabu list, Min-Neighbor: number of neighbors) Returns: neighbor configuration
best-cost ← +∞; x-best ← cur
i ← 0; accepted? ← false
while (i < Min-Neighbors) do
  x ← Generate-Neighbor(cur)
  if x - cur ∉ tabu-list or cost(x) < best-cost (aspiration) then
    accepted? ← true
  end
  if cost(x) < best-cost then
    x-best ← x
    best-cost ← cost(x)
  end
  i ← i + 1
end
if accepted? then
  tabu-list.PushEnd(x-best)
  if (size(tabu-list) > L) then tabu-list.PopFirst()
  return x-best
else
  return cur
end
end.
```

Algo Tabu

Liste de longueur constante L qui enregistre les L derniers coups (FIFO). Pour les CSP, un coup x' - x dans la liste est la variable modifiée (la valeur n'est pas stockée). Cet algo. évite de regarder plusieurs fois la même

ID Walk

```
Algorithm IDW-Move(cur: current configuration, Max-Neighbor: number
of neighbors) Returns: neighbor configuration
best-cost ← +∞; x-best ← cur
i ← 0; accepted? ← false
while (i < Max-Neighbors and not(accepted?)) do
  x ← Generate-Neighbor(cur)
  if cost(x) ≤ cost(cur) then
    accepted? ← true
  end
  if cost(x) < best-cost then
    x-best ← x
    best-cost ← cost(x)
  end
  i ← i + 1
end
if accepted? then
  return x
else
  return x-best /* Variant: return x */
end
end.
```

Fait très attention à l'analyse des candidats (voisins) pour le prochain coup.

Principal paramètre à régler → Max-Voisins avec 3 rôles :

1. limite le nombre de voisins explorés,
2. suffisamment grand pour intensifier la recherche,
3. suffisamment petit pour diversifier la recherche (avec No-Acceptation).

Genetic algorithms: guidelines

Management of a **population** of configurations, called **individuals**

Algorithm GA-schema

while no satisfactory individuals in the population **do**
Select individuals in the population for reproduction
Apply different reproduction operators on selected individuals:

- **mutation:** generation of a neighbor of an individual
- **crossover:** mixing two configurations (individuals) to generate a new individual

end
Selection: keep a sub-set of the new population (natural selection)

end.

Encoding: an individual is made of a chromosom: a sequence of bits