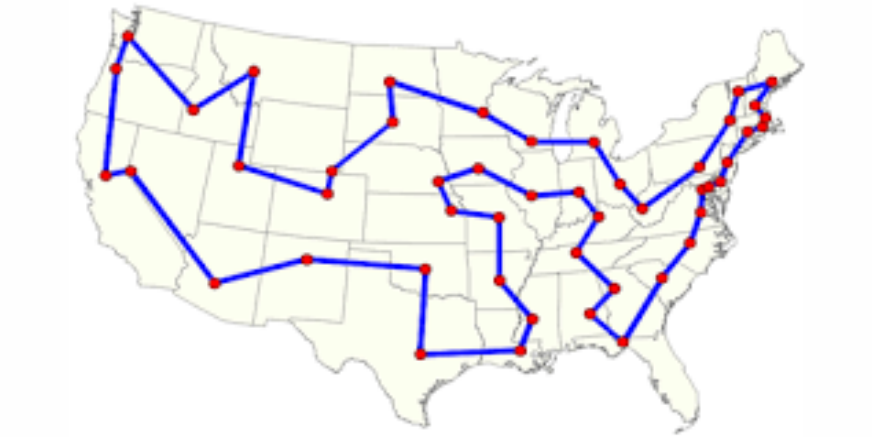
Rapport mini-projet : métaheuristique pour le TSPTW

Corentin Chabanol, Jean Velluet

January 18, 2023

CentraleSupélec



1 Présentation du Problème

Le travelling salesman problem (TSP) est un problème d'optimisation qui consiste à déterminer, étant donné une liste de villes et les distances des trajets entre chaques paires de villes, le plus court chemin qui passe par chaque ville une et une seule fois. Ce célèbre problème, bien que très simple n'en est pas moins important. Il est en effet rencontré dans de nombreux domaines, allant de la logistique à la génétique. Le TSPWT (Travelling Salesman Problem with Time Windows) est une extension du TSP (travelling Salesman Problem) auquel on ajoute une contrainte de temps : on doit visiter chaque ville dans un crénau de temps fixé.
Ce problème étant NP-complet l'implémentation d'algorithmes de recherche exacts ne permet souvent pas la résolution du problème dès lors que le nombre de villes devient trop grand. Il est alors possible d'utiliser une métaheuristique afin d'essayer de résoudre ce problème d'optimisation, où au moins de trouver une solution qui ne soient "pas trop mauvaise".

1.1 formalisation du problème

Soit (G, V) un graphe où $G = \{ 0, \ldots, n\}$ villes et $V = G \times G$ les chemins liant les différences villes. Un chemin est représenté par les villes visitées dans l'ordre, $\{ p_0, \ldots p_n\}$ où p_i est la i-ème ville visitée. avec $p_{n+1} = p_0$ car on revient à la ville dont on est parti. A chaque arc $a_i = (p_{i+1}, p_i)$ d'un chemin reliant deux villes est associé un temps de parcours $c(a_i)$ et à chaque ville p_i une fenêtre de temps $T_i = [d_i, f_i]$ Notre but est de minimiser le temps de parcours, c'est à dire trouver un chemin qui minimise la fonction f définie par $f(p_0, \ldots, p_n) = \sum_{k=0}^{n} c(a_k)$. Plutôt que de chercher à minimiser cette fonction sur l'espace des solutions qui respectent les contraintes de temps, lors de notre exploration de l'espace des solutions, nous allons volontairement considérer des solutions qui ne respectent pas les contraine liées aux fenêtres de temps mais en pénalisant la fonction f à chaque violation d'une fenêtre de temps. Nous allons donc rajouter à cette fonction une pénalité proportionelle à la durée de chaque violation des fenêtres de temps. Ainsi on cherche à minimiser une fonction de la forme :

$$f(p_0, ..., p_n) = \sum_{k=0}^{n} c(a_k) + \underbrace{\lambda \sum_{k=0}^{n} max(0, D_{p_i} - f_i)}_{:= p(p_0, \ldots p_n)} \tag{1}$$

où D_{p_i} est l'instant de départ du voyageur et λ est un paramètre sur lequel nous jouerons afin d'améliorer les performance de la métaheuristique. Le problème se réduit donc à minimiser la fonction f en respectant les contraines suivantes :

$$\begin{cases} A_{p_i} = D_{p_{i-1}} + c(a_{i-1}) \text{ Instant d'arrivé dans la ville i} \\ D_{p_i} = \text{Max}(A_{p_i}, d_{p_i}) \\ p_i \in 0, 1, ..., n \\ p_i \neq p_j \quad \text{pour } i \neq j \\ p_0 = 0, p_{n+1} = p_0 \end{cases} \tag{2}$$

2 Choix de la métaheuristique

Afin de résoudre le problème d'optimisation de 1 sous les contrainte 3, notre choix s'est porté sur un algorithme stochastique du type recuit simulé. Cependant, plutôt que de considérer la pénalité liée aux violations des fenêtres de temps λ comme constante, nous allors la faire augmenter à chaque itération dans la procédure d'optimisation. De cette manière, le "poids" relatif d'une violation de fenêtre temporelle dans l'évaluation du score sera de plus en plus important et l'on autorisera donc de moins en moins de violations des contraintes de temps, de telle sorte que lors des dernières itérations, les solutions qui violent les contraintes de temps ne serons plus considérés tant elles feront exploser le score.

L'algorithme en pseudo code s'écrit donc de la manière suivante :

Generate initial tour, x.

Initialize best tour found, x_{best}, so that $f(x_{best}) = \infty$ and $p(x_{best}) = 0$

Let $k = 0$.

Set initial temperature and pressure, τ_k and λ_k.

Set *iteration limit*, i.e., number of iterations at each temperature/pressure.

Repeat:

Let *counter* = 0.

Repeat:

Increment *counter* by 1.

Randomly generate y, a neighbor tour of x.

With probability $\exp\left(\frac{-(v(y,\lambda_k)-v(x,\lambda_k))^+}{\tau_k}\right)$, let $x = y$.

If $p(x) = 0$ and $f(x) < f(x_{best})$, let $x_{best} = x$.

Until *counter* == *iteration limit*.

Increment k by 1.

Update τ_k and λ_k.

Until termination criterion satisfied.

Figure 1: Algorithme en pseudo code [1]

Il reste à déterminer les paramètres ainsi que la fonction de voisinage.

- Notre choix de voisinage s'est porté sur la fonction naïve suivante : on définit le voisinage d'un chemin de la manière suivante : $V(p_0, \ldots, p_n) = \{ (p_0, \ldots, p_n) \circ \tau$, avec τ transposition de $\{ p_0, \ldots p_n\} \}$.

- Pour les paramètres de température et de pression, les valeurs sont données par les relations de récurrences :

$$\begin{cases} \tau_{k+1} = \beta\tau_k \\ \lambda_{k+1} = \widehat{\lambda}(1 - \frac{\widehat{\lambda}-\lambda_0}{\widehat{\lambda}} e^{-\gamma k}) \end{cases} \tag{3}$$

  Le choix de la variation de λ allant de 0 et convergeant vers une valeur asymptotique $\widehat{\lambda}$ est justifié dans [2].

  τ_0, λ_0 et $\widehat{\lambda}$ sont définis de la manière suivante :

- $\lambda_0 = 0$

- $\widehat{\lambda} = \max_{(p_0,\ldots,p_n)\in R} \frac{f(p_0,\ldots,p_n)}{p(p_0,\ldots,p_n)} \frac{\kappa}{1-\kappa}$ où R est un ensemble de $2n$ chemins construit en générant aléatoirement n chemins et en ajoutant un voisinage de chacun de ces chemins.

- κ represente le pourcentage de la fonction objectif 1 qui est composé du terme de pénalité quand $\lambda = \widehat{\lambda}$

- β le paramètre de refroidissement, typiquement compris entre 0.8 and 0.99.

- γ le paramètre de pression, typiquement compris entre 0.01 à 0.1.

- χ_0 la probabilité de considérer le voisin aléatoire au palier de pression τ_0 typiquement entre 0.80 et 0.99

  Ainsi, on calcule $\tau_0 = \frac{\Delta v}{ln(1/\chi_0)}$ où Δv est la moyenne des différences des scores des n paires de chemins de R en valeur absolue.

  Les choix du jeux paramètres étant souvent la pierre angulaire de la métaheuristique, un bon choix est primordial. L'approche utilisée dans [3] permet de définir un jeu de paramètre robuste pour cette métaheuristique.

Parameter	Value
cooling coefficient (β)	.95
initial acceptance ratio (χ_0)	.94
compression coefficient (γ)	.06
pressure cap ratio (κ)	.9999
iterations per temperature	30000
minimum number of temperature changes	100

Figure 2: jeu de paramètres [1]

3 résultats et performances

Pour chaque instance, nous avons trouvé les solutions exactes grâce à notre métaheuristique (voir annexe). Par souci d'économie de temps de calcul, nous traçons les courbes suivantes avec l'instance n°1.

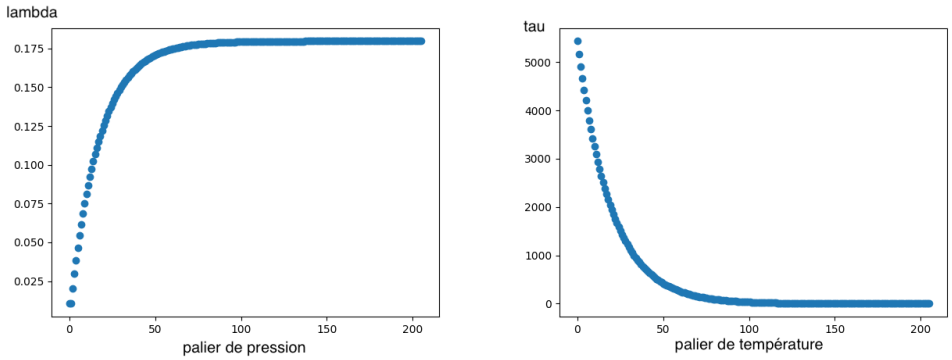


Figure 3: évolution des paramètres τ et λ

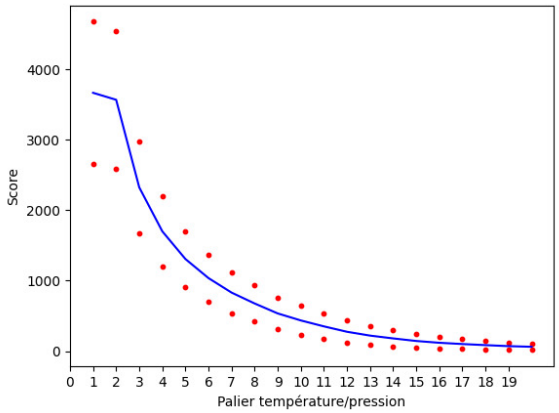


Figure 4: Score (bleu) et intervalle de confiance à 95 % (rouge)

Sur la figure 4 le score a été moyenné sur 50 itérations de recuits et pour chaque palier température/pression le score est moyenné sur les valuations des chemins acceptés par l'algorithme sur ce palier, afin de lisser la courbe. En effet, puisqu'on s'autorise à explorer des voisins qui ne font pas necessairement baisser le score (surtout au début) la courbe n'était initialement pas très lisse. L'intervalle de confiance tends bien vers 0 avec l'augmentation du nombre d'itération dans la procédure d'optimisation.

Remarque : Lorsque l'on trace le score, on ne trace pas directement la fonction 1 (bien que ce soit effectivement elle qu'on minimise), mais on trace le quotient de cette fonction et de λ à chaque palier de température pression afin de compenser l'augmentation de λ. En effet, si l'on ne fait pas ça, on a un score qui évolue en forme de cloche.

3.1 Comparaison avec l'approche naïve

On se place dans le cas ou l'on modifie dans 1 la pénalité p par $p(p_0, \ldots, p_n) = \lambda \times$ (nombre de violation de fenêtres de temps). Et où l'on condidère $\lambda = cst$. On suppose toutes choses égales par ailleurs. L'évolution du score ainsi que l'intervalle de confiance à 95¨ % sont tracés en figure 5.

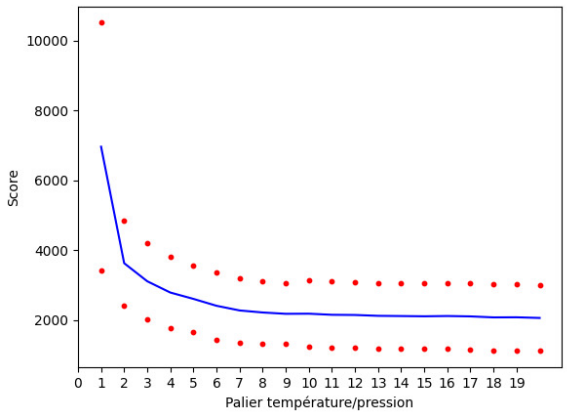


Figure 5: Score (bleu) et intervalle de confiance à 95 % (rouge)

L'intervalle de confiance est plus large que pour notre métaheuristique et ne converge pas, de plus, le score lui même ne converge pas et l'on ne trouve pas la solution optimale du problème. En modifiant les paramètres, notamment la température initiale, et β, nous devrions pouvoir trouver une configuration qui nous permettent de trouver les solutions pour les instances 1,2 et 3 au moins.

4 Reférences

[1]: A Compressed Annealing Approach to the Traveling Salesman Problem with Time Windows, Jeffrey W. Ohlman, Barrett W. Thomas, 2021

[2] : Ohlmann, J. Bean, and S. Henderson, "Convergence in Probability of Compressed Annealing," (2002), submitted to Mathematics of Operations Research.

[3] : P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil, "Using Experimental Design to Find Effective Parameter Settings for Heuristics," Journal of Heuristics 7, 77–97 (2000)