CAHIERS AFRICAINS DES DROITS DE L'HOMME ET DE LA DEMOCRATIE AINSI QUE DU DEVELOPPEMENT DURABLE
REVUE AFRICAINE INTERDISCIPLINAIRE

ISSN : 2791-1063 (Imprimé)
ISSN : 2791-1071 (En ligne)

27ème année - Numéro 79 - Volume 3 - Avril-Juin 2023

DEVELOPPEMENT D'UN PACKAGE POUR RESOUDRE LE PROBLEME DE VOYAGEUR DE COMMERCE A PARTIR D'INSTANCES TSPLIB

Par

John MAKIESE MIFUNA
Assistant, Mention Mathématique-Informatique, Domaine des Sciences et Technologies, Institut Supérieur Pédagogique de Boma

Dechris MAMBEKO MBAMBI
Assistant, Mention Informatique de gestion, Domaine des Sciences et Technologies, Institut Supérieur Pédagogique de Kangu

Fabien KAMBU MBUANGI
Chef de Travaux, Mention Informatique de gestion, Domaine des Sciences et Technologies, Institut Supérieur Pédagogique de Boma

RÉSUMÉ

Dans cette étude, nous présentons le package "PVCGA" (problème de voyageur de commerce génétique algorithme) développé pour résoudre le problème de voyageur de commerce. Le problème de voyageur de commerce fait partie des problèmes NP-difficiles, c'est-à-dire qu'il n'est pas possible de trouver une solution en temps polynomial. Ainsi, nous avons utilisé l'algorithme génétique comme solution. Pour ce faire, nous avons lancé nos simulations en utilisant deux instances de benchmark tirées de la librairie TSPLIB. Ces instances sont constituées chacune de 29 et 48 villes.

L'idée principale des heuristiques est d'explorer l'espace des solutions en essayant de converger vers la meilleure solution. Ainsi, les résultats obtenus respectivement de nos deux instances ont atteint respectivement 100% et 95% des satisfactions.

L'un des problèmes majeurs dans cette étude a été de résoudre le problème de la convergence prématurée de l'algorithme vers un extremum/optimum local. Un extremum local est la meilleure solution dans une zone restreinte, en opposition à l'extremum global, qui est la meilleure solution dans l'ensemble[1].

Mots-clés : *Algorithme génétique, benchmark, chemin, extremum local, implémentation, NP-difficiles, optimum global, package PVCGA, problème de voyageur de commerce, sélection.*

[1] http://igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html

SUMMARY

In this study, we present the package "PVCGA" (traveling salesman problem genetic algorithm) developed to solve the traveling salesman problem. The traveling salesman problem is one of the NP-hard problems, that is to say it is not possible to find a solution in polynomial time. So, we used genetic algorithm as a solution. To do this, we launched our simulations using two benchmark instances taken from the TSPLIB library. These bodies are each made up of 29 and 48 cities.

The main idea of heuristics is to explore the solution space trying to converge towards the best solution. Thus, the results obtained from our two instances respectively reached 100% and 95% satisfaction.

One of the major problems in this study was to resolve the problem of premature convergence of the algorithm towards a local extremum/optimum. A local extremum is the best solution in a restricted area, as opposed to the global extremum, which is the best solution overall.

Keywords: *Genetic algorithm, benchmark, path, local extremum, implementation, NP-hard, global optimum, PVCGA package, traveling salesman problem, selection.*

I. INTRODUCTION

Le problème de voyageur de commerce peut être définie comme un ensemble de *n* villes et les distances entre toutes les paires de villes, trouver un chemin de longueur totale minimale qui passe exactement une fois par chaque ville et revienne à la ville de départ.[2]

Pour chaque ensemble de *n* villes, il existe au total *n!* chemins possibles, le point de départ ne change pas la longueur du chemin. La distance entre une paire des villes est calculée sous la règle de la distance euclidienne en utilisant l'équation suivante[3] :

$$\boldsymbol{d(T[i], T[i+1]) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}}$$

Ainsi, pour calculer la distance de chaque chemin possible nous utiliserons la formule suivante[4] :

$$\boldsymbol{f = d(T[n], T[1]) + \sum_{i=1}^{n-1} d(T[i], T[i+1])}$$

[2] https://fr.wikipedia.org/wiki/Problème_du_voyageur_de_commerce
[3] https://fr.slideshare.net/SoufianeMejdaoui/solving-tsp-using-ga-and-matlab?from_action=save
[4] *Idem.*

Les travaux récents utilisent souvent deux heuristiques pour la résolution du problème de voyageur de commerce : les algorithmes génétiques et les algorithmes de colonies de fourmis.

Pour résoudre le problème de voyageur de commerce présenté en amont, nous avons utilisé l'algorithme génétique.

II. MÉTHODE UTILISÉE

2.1 Algorithme génétique

Les algorithmes génétiques sont des techniques de recherche et d'optimisation basées sur les idées de l'évolution naturelle, en reproduisant l'évolution des espèces par la sélection naturelle. Les algorithmes exécutent également les procédures d'optimisation[5].

Les algorithmes génétiques peuvent être subdivisés en deux étapes : la première consiste à choisir les individus pour produire la génération suivante, et la seconde à modifier les choix de ces individus pour créer la prochaine génération[6].

Les algorithmes génétiques s'inspirent de l'évolution des espèces dans un environnement naturel[7]. Dans cette étude, nous partons d'une population de base que nous allons faire évoluer dans un certain nombre de générations (itération de l'algorithme). La population est constituée d'un ensemble des solutions des chemins (individus). Un chemin ou individu est un circuit touchant l'ensemble d'indices (gènes) des villes contenu dans une instance TSPLIB une et une seule fois avant de retourner au point de départ.

Les choix des individus pour la reproduction et le nombre d'enfants que chaque couple d'individus sélectionnés produise sont déterminés par le mécanisme de sélection et de croisement. Le principe fondamental de la stratégie de sélection est que la probabilité qu'un individu soit choisi augmente en fonction de sa distance.

2.2. Etapes d'algorithme génétique

L'évolution de la population dans un algorithme génétique se fait moyennant les étapes suivantes :

---

[5] L. Haldurai1, T. Madhubala and R. Rajalakshmi, A Study on Genetic Algorithm and its Applications, International Journal of Computer Sciences and Engineering, Volume 4 31 Oct 2016 in https://www.researchgate.net/publication/309770246

[6] *Idem.*

[7] Ripon Sorma and al, Solving Traveling Salesman Problem by Using Genetic Algorithm, Electrical and Electronic Engineering 28 Dec 2020, 10(2): 27-31 DOI, in https://www.researchgate.net/publication/347976512

2.2.1. Evaluation

Pour évaluer les individus d'une population, nous avons sommé les pairs des distances des villes de nos instances afin d'obtenir les distances.

2.2.2. Sélection

La sélection consiste à choisir les individus les mieux adaptés afin d'avoir une population de solution la plus proche de converger vers l'optimum global[8]. Pour ce faire, nous avons utilisé trois techniques de sélection :

a) Sélection élitiste : cette technique de sélection choisit toujours les individus possédant les meilleurs scores d'adaptation ;
b) Sélection uniforme : la sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation ;
c) Sélection combinée : cette sélection a consisté à choisir une partie des individus moyennant la sélection élitiste en l'associant avec une autre partie des individus choisis par la sélection uniforme.

2.2.3 Croisement

Le croisement (crossing-over) est le résultat obtenu lorsque deux chromosomes partagent leurs particularités. Celui-ci permet le brassage génétique de la population et l'application du principe d'hérédité de la théorie de Darwin[9].

Pour ce faire, nous avons utilisé le croisement simple qui consiste à fusionner les particularités de deux individus à partir d'un pivot, afin d'obtenir deux enfants. La figure 1 illustre le croisement simple de deux individus parents constitués de 7 villes (gênes) reproduisant deux individus fils.

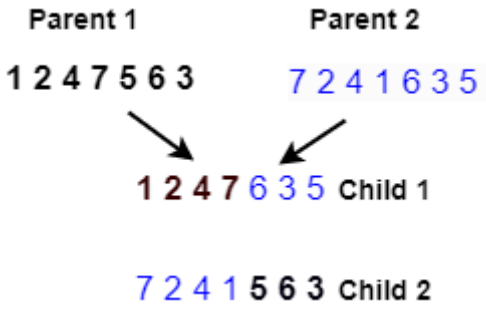

Figure 1. Croisement simple

[8] http://igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html
[9] Idem.

2.2.4. Mutation

La mutation consiste à permuter les gènes dans un chromosome selon un facteur de mutation. Ce facteur est la probabilité qu'une mutation soit effectuée sur un individu.

Cet opérateur est l'application du principe de variation de la théorie de Darwin, elle permet par la même occasion, d'éviter une convergence prématurée de l'algorithme vers un extremum local[10].

L'exemple ci-dessous vous présente la mutation de deux gènes d'un individu :

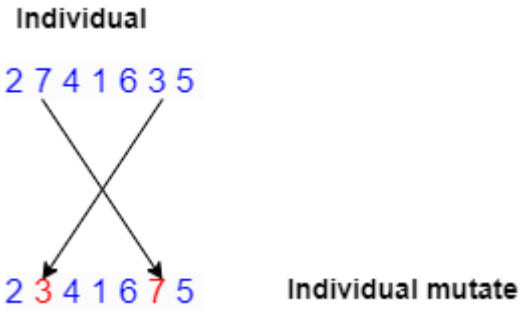

Figure 2 : Mutation

2.2.5. Architecture de l'algorithme génétique

La figure ci-dessous présente les différentes étapes à suivre pour implémenter un algorithme génétique. Nous partons d'une population initiale dont la taille est fixée par défaut. Ensuite, nous allons effectuer toutes les opérations tels que présentés en amont.

[10] http://igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html

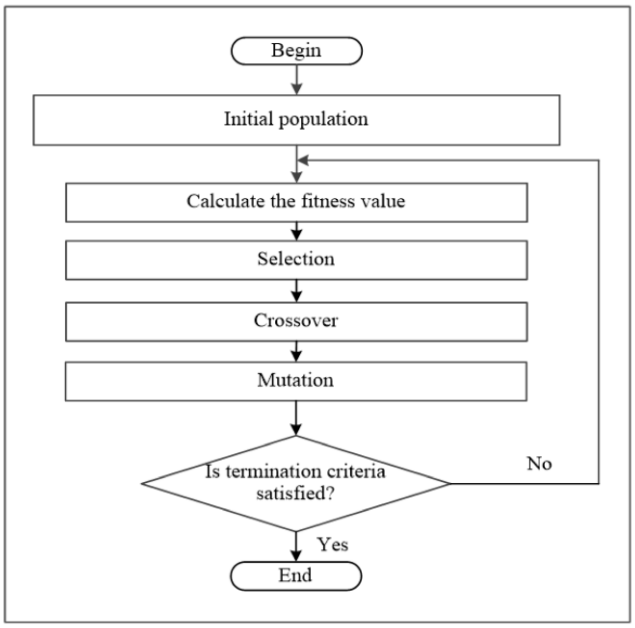

Figure 3: Architecture de l'algorithme génétique [6]

III. PRESENTATION DU PACKAGE PVCGA

Nous avons utilisé le logiciel R Studio pour développer le package PVCGA. R Studio est un environnement de développement intégré (IDE) pour R et Python. Il comprend une console, un éditeur de mise en évidence de la syntaxe qui prend en charge l'exécution directe de code et des outils de traçage, d'historique, de débogage et de gestion de l'espace de travail. R Studio est disponible dans les éditions open source, commerciales et fonctionne sur le bureau (Windows, Mac et Linux)[11].

En R, un package est le moyen le plus adapté pour étendre les fonctionnalités de R. C'est une collection de fonctions, de données, de documentations et de tests. Un package est facilement partageable avec d'autres, il est le plus souvent versionné, c'est-à-dire qu'il peut exister plusieurs versions d'un même package pas forcément compatibles entre elles[12]. Pour ce faire, nous avons utilisé la librairie roxygen2 pour la création de notre package.

Le principe de roxygen2 est simple : décrivez vos fonctions dans des commentaires à côté de leurs définitions et roxygen2 traitera votre code source et vos commentaires pour générer automatiquement .Rd des fichiers dans man/, NAMESPACE et, si nécessaire, le Collate champ dans DESCRIPTION[13].

[11] https://posit.co/products/open-source/rstudio/
[12] https://maeltheuliere.github.io/ateliers_rpackage/slides/slide1.html?panelset=github#4
[13] https://roxygen2.r-lib.org/

3.1. Schéma globale de l'implémentation

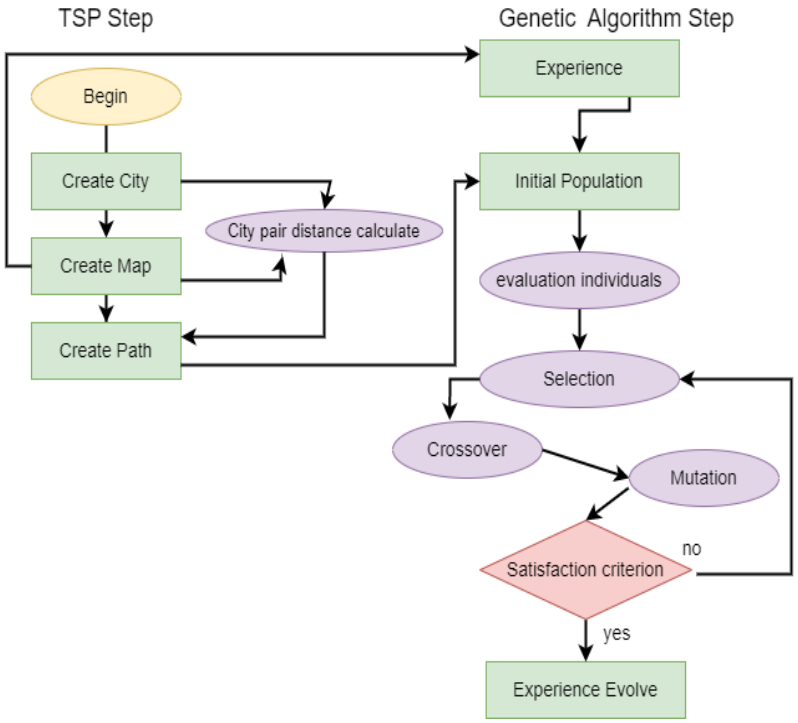

Figure 4 : Schéma globale de l'implémentation

La figure 4 décrit les différentes étapes de l'implémentation du package PVCGA. Les éléments en forme rectangulaire représentent les différents objets implémentés, par contre les éléments en ovales représentent les différentes opérations effectuées sur ces objets. La première étape de l'implémentation a consisté à créer les classes des objets city, map et path et ensuite nous avons réalisé l'opération de calcul d'une paire des villes en utilisant la formule de la distance euclidienne[14].

L'opération de calcul d'une paire des villes a été reprise pour évaluer la distance d'un circuit path tel qu'indiqué par la flèche d'orientation dans la figure 4. La seconde étape a été d'implémenter l'algorithme génétique afin de résoudre le problème. Pour ce faire, nous avons implémenté la classe nommée *expérience* qui contient à son sein tous les paramètres d'évolution de l'algorithme génétique ainsi que l'objet map créé précédemment.

La suite a été de créer la classe population initiale et après réaliser toutes les opérations de l'algorithme génétique (évaluation, sélection, croisement, mutation). Une fois les critères d'évolution de la population satisfaits, le package retourne une expérience évoluer qui contiendra : la dernière

[14] https://fr.slideshare.net/SoufianeMejdaoui/solving-tsp-using-ga-and-matlab?from_action=save

population évoluer, le meilleur individu, le temps de calcul, le plot du meilleur individu à l'aide de ggplot2[15] et enfin la possibilité de visualiser à l'aide d'une figure la diversité génétique entre les gènes des individus de la population évoluée.

3.2. Schéma des objets

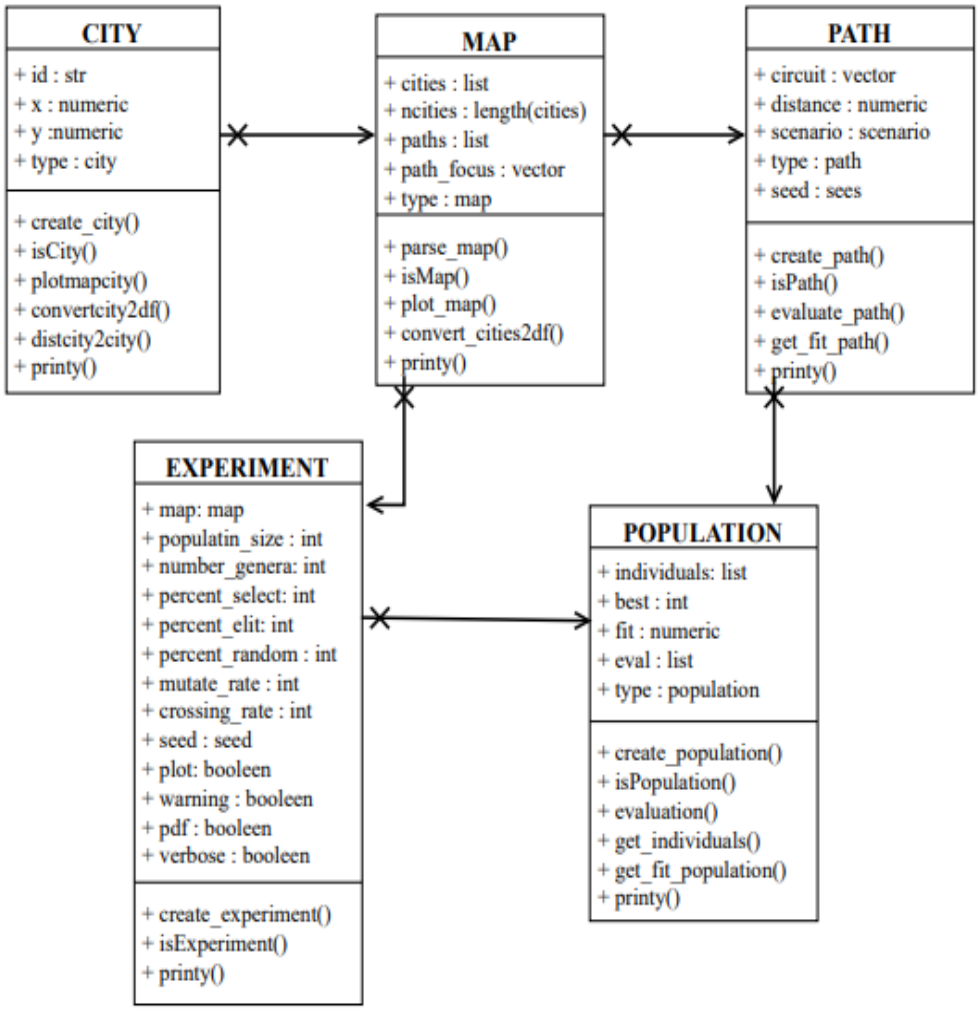

Figure 5 : Schéma des objets

La figure 5 présente la structure de nos 5 objets implémentés dans le package. Les fonctions implémentées dans le package ont été toutes validées par des tests unitaires. Nous avons ainsi utilisé le package usethis[16] pour effectuer les tests unitaires.

[15] https://cran.r-project.org/web/packages/ggplot2/index.html
[16] https://cran.r-project.org/web/packages/usethis/index.html

3.3. Liste des fonctions

Fonctions TSPGA			
1	convert_cities2df: convertit les cities de la carte en data frame	2	convert_city2df: convertir une city en data frame
3	create_city: constructeur de city	4	create_experiment: constructeur expérience
5	create_individus_duplicated: remplace les individus dupliqués	6	create_path: constructeur path
7	create_population: constructeur population	8	crossing: croisement des individus
9	dist_city2city: distance entre deux villes	10	evaluate_path: évalue un path
11	evaluation: évalue un individu	12	evolve: faire évoluer la population
13	get_fit_path: récupère un path	14	get_fit_population: récupère la population
15	get_individuals: récupère un individu	16	isCity: test l'objet city
17	isExperiment: test objet expérience	18	isMap: test l'objet map
19	isPath: test objet path	20	isPopulation: test l'objet population
21	mutate: mute une population	22	parents: sélectionne les parents
23	parse_map: constructeur de map	24	plot_map: plot la map
25	plot_mapcity: plot la ville d'une map	26	printCity: imprime l'objet city
27	printExperiment: imprime l'objet expérience	28	printMap: imprime map
29	printPath: imprime l'objet path	30	printPopulation: imprime l'objet population.
31	printy: visualise tous les objets		

Tableau 1: Fonction tspgar

IV. RESULTATS

Nous avons utilisé un jeu de données contenant 10 800 expériences avec deux maps dont chacune contenait 5 400. Ces expériences ont été lancées en parallèle avec 120 CPU. L'ensemble du travail a duré 18 heures.

Les tableaux ci-dessous présentent les résultats de nos meilleures solutions sur les trois méthodes des sélections (élitiste, uniforme et combinée). Ces résultats sont comparés avec la distance optimale à atteindre.

Résultats				
Sélection	**maps**	**best distance**	**optimal distance**	**Pourcentage**
Combinée	wi29.tsp	27 601 km	27 601 km	**100** %
	att48.tsp	35 388 km	33 523 km	95 %
Elitiste	wi29.tsp	28 189 km	27 601 km	98 %
	att48.tsp	37 468 km	33 523 km	90 %
Uniforme	wi29.tsp	27 601 km	27 601 km	**100** %
	att48.tsp	37 025 km	33 523 km	91 %

Tableau 2: Résultats TSPGA

V. DISCUSSION

En nous basant sur les résultats présentés dans le tableau 2, la sélection combinée est la meilleure méthode de sélection des individus pour résoudre le problème du voyageur de commerce. Le point annexe ci-dessous, présente les plots de nos deux meilleures solutions pour le map wi29.tsp et att48.tsp.

VI. ANNEXES

6.1 Map wi29.tsp

a) wi29.tsp origine

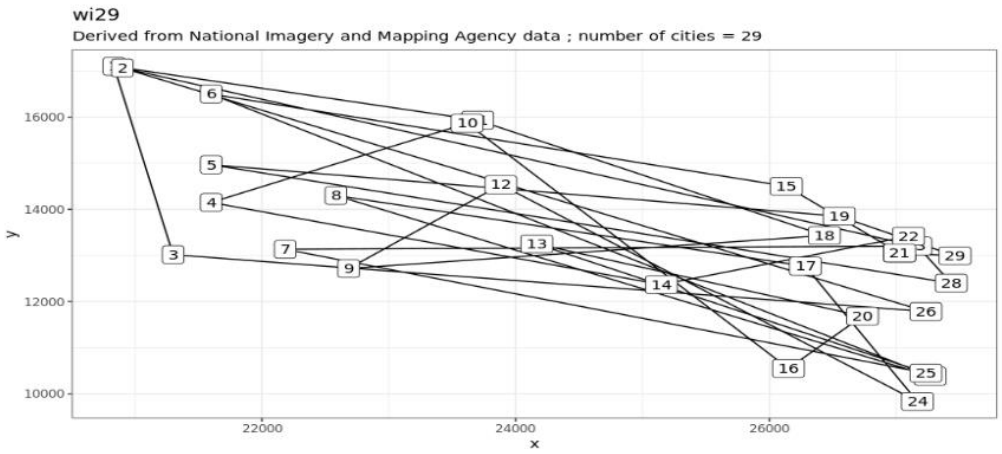

Figure 6: Map wi29.tsp origin

b) wi29.tsp évoluer distance 27601

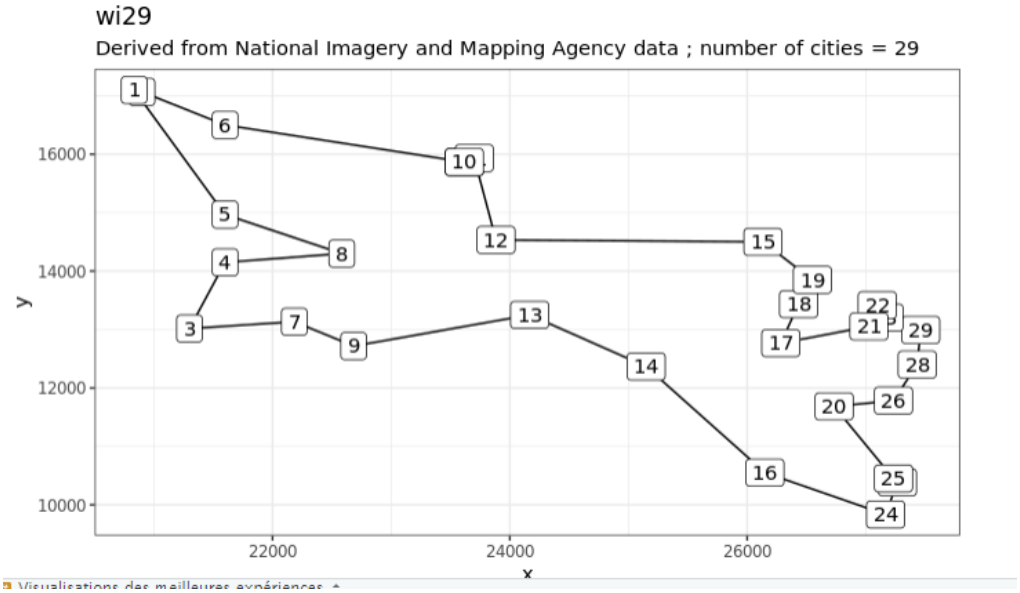

Figure 7 : Map wi29.tsp évoluer

6.2 Map att48.tsp

a) att48.tsp origine

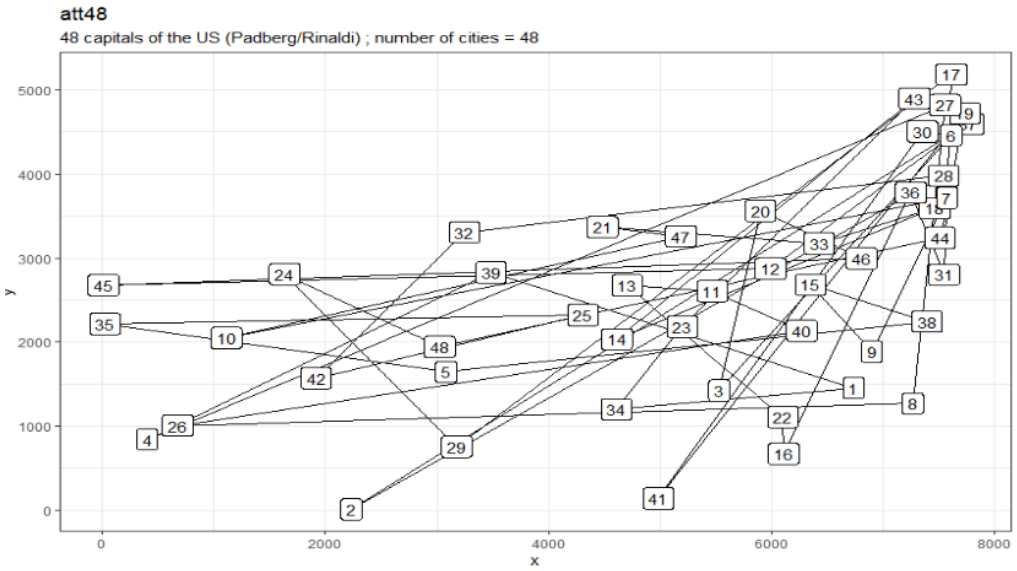

Figure 8: Map att48.tsp origine

b) att48.tsp évoluer distance 35388 km

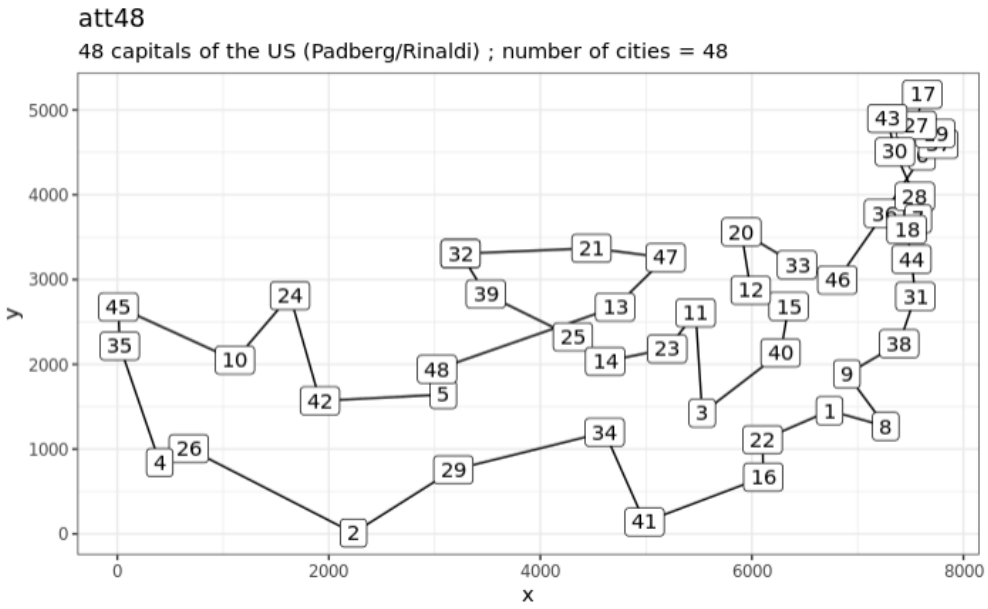

Figure 9: Map att48.tsp évoluer

RÉFÉRENCES

1. Haldurail L., Madhubala T. and Rajalakshmi R., *A Study on Genetic Algorithm and its Applications*, International Journal of Computer Sciences and Engineering, Volume 4, 31 October 2016 in https://www.researchgate.net/publication/309770246 (consulté le 06 août 2023).
2. Musatafa Abbas Albadr and al., *Genetic Algorithm Based on Natural Selection Theory for Optimization Problems*, Symmetry 2020, 12(11), 1758 in https://www.sciencegate.app/app/redirect#aHR0cHM6Ly93d3cubWRwaS5jb20vMjA3My04OTk0LzEyLzExLzE3NTgvcGRm (consulté le 06 août 2023).
3. Ripon Sorma and al., *Solving Traveling Salesman Problem by Using Genetic Algorithm,* Electrical and Electronic Engineering 28 Dec 2020, 10(2): 27-31 DOI, in https://www.researchgate.net/publication/347976512 (consulté le 06 août 2023).
4. http://igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html
5. https://fr.wikipedia.org/wiki/Problème_du_voyageur_de_commerce
6. https://fr.slideshare.net/SoufianeMejdaoui/solving-tsp-using-ga-and-matlab?from_action=save
7. https://posit.co/products/open-source/rstudio/
8. https://maeltheuliere.github.io/ateliers_rpackage/slides/slide1.html?panelset=github#4
9. https://roxygen2.r-lib.org/
10. https://cran.r-project.org/web/packages/ggplot2/index.html
11. https://cran.r-project.org/web/packages/usethis/index.html