SCIENCES SORBONNE UNIVERSITÉ

Le Problème du Voyageur Canadien Couvrant (CCTP)

Projet RP

MAMLOUK Haya [21107689]
OZGENC Doruk [21113927]

27 Avril 2025

Plan du rapport

I. Introduction

Présentation du k-CCTP et des algorithmes CR et CNN, ainsi que des critères d'évaluation.

II. Instances de test

A. Familles de graphes
B. Paramètres de la campagne

III. Résultats

A. Ratio compétitif moyen
B. Temps d'exécution médian
C. Bilan agrégé par famille

IV. Discussion

Analyse des comportements : pires cas pour CR et pour CNN, compromis qualité vs réactivité.

V. Conclusion

I. Introduction

Le **Canadian Covering Traveller Problem** (k-CCTP) modélise un voyageur devant parcourir un graphe complet métrique tout en découvrant **en ligne** jusqu'à k arêtes bloquées. Nous comparons deux stratégies:

Algorithme	Principe	Points forts / faibles
CR – Cyclic Routing	1. Tournée de Christofides. 2. Suivi du tour : lorsqu'une arête est bloquée, on « saute » les sommets suivants jusqu'à une arête praticable. 3. On répète sur les sommets restants en utilisant des raccourcis et inversant le sens si le dernier sommet n'est pas atteint.	+ Robuste aux blocages ponctuels. - Peut rallonger le chemin si des ponts critiques du tour sont bloqués.
CNN	1. Même départ que CR. 2. Quand ≥ n−k sommets sont visités, on construit un graphe réduit **G'** (sommets inconnus + plus courts chemins sûrs). 3. On termine avec Nearest- Neighbour sur G'.	+ Réutilise l'information acquise. Plus Rapide. - Phase NN peut devenir chère si le tour initial est très fragmenté.

Nous mesurons la **performance** par le ratio compétitif $\rho = \frac{coût\_en\_ligne}{coût\_optimal\_hors\_ligne}$ et la **scalabilité** par le temps CPU moyen.

II. Instances de test

A. Familles générées

Pour comparer nos deux algorithmes, nous avons sélectionné **cinq familles de graphes** complets métriques, toutes conformes à l'inégalité triangulaire, que nous jugeons les plus pertinentes pour mettre en évidence leurs différences de comportement.

Code de la famille	Structure	Objectif expérimental	Que ça modélise ?
B	Points i.i.d. uniformes dans le carré unité.	Cas moyen neutre.	Un réseau routier "moyen" où tout est plausible mais rien n'est structuré.
C	4 clusters gaussiens bien séparés.	Sensibilité aux **ponts inter-clusters**.	Villes ou quartiers séparés par de longs axes (autoroutes, ponts).
D	Grille m×m ; diagonales principales 3× moins chères	Réseau urbain avec diagonales rapides.	Centre-ville quadrillé avec deux "périphériques" rapides en diagonale.
E	Même nuage que B ; blocage adversarial : ≈ 80 % des arêtes successives du tour de Christofides sont interdites. |	**Pire cas** pour CR (stress-test général).	Cas "malveillant" conçu pour piéger CNN.
H	Hub-and-spoke : hub (poids 1) – feuilles (poids 2 entre feuilles).	Raccourcis via le hub	Réseau avec un **hub** obligatoire (gares TGV, datacenter…).

B. Paramètres de campagne la campagne expérimentale

Pour évaluer de façon systématique l'impact de la taille du graphe et de la sévérité des blocages, nous faisons varier deux paramètres clés :

- **n** — le nombre de sommets du graphe, pris dans **{20, 40, 80, 160}** afin de couvrir du petit réseau testable à la main jusqu'au réseau de taille moyenne où le temps de calcul devient significatif ;

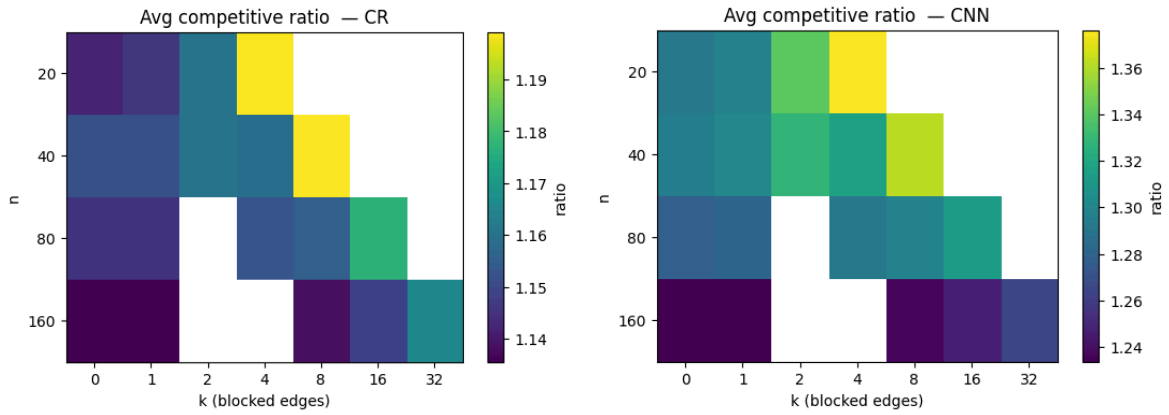
- **k** — le nombre maximal d'arêtes bloquées que l'agent découvrira en ligne, choisi dans **{0, 1, 2, 4, 8, 16, 32}**, soit jusqu'à 20 % de n, ce qui permet d'explorer le continuum "aucun incident" → "forte congestion".

Pour chaque couple (n, k), nous générons 30 instances indépendantes (**30 graines),** puis exécutons les deux algorithmes sur les cinq familles retenues, soit au total : 5 familles × 4 tailles × 7 valeurs de k × 30 graines × 2 algorithmes = **6 000 exécutions**.
Toutes les simulations sont codées en **Python** avec **NetworkX** ; le même générateur aléatoire assure la reproductibilité des résultats.

III. Resultats

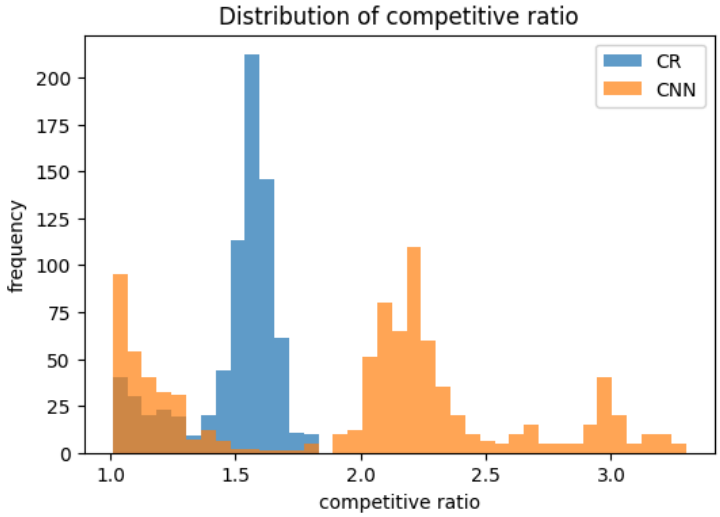
A. Ratio compétitif

Notre point de départ est le ratio compétitif moyen $\bar{\rho}$: pour chaque couple (n, k) et pour chaque famille, nous moyennons les 30 réplicas puis représentons ces valeurs dans deux cartes de chaleur (une par algorithme). Les heat-maps qui suivent permettent ainsi de visualiser d'un coup d'œil comment la performance se dégrade — ou reste stable — lorsque la taille du graphe et le nombre d'arêtes bloquées augmentent.



Document 1 : Ratio Compétitif moyen — heat-map $n \times k$ (par algorithme)

Globalement, les cartes font apparaître un point de bascule : tant que le nombre de blocages reste modéré ($k \leq 2$ ou 4), CR conserve quelques points d'avance sur CNN ; dès que k s'approche de 20% de n, la tendance s'inverse et CNN prend un léger avantage. L'effet est accentué sur les petits graphes ($n \leq 40$).

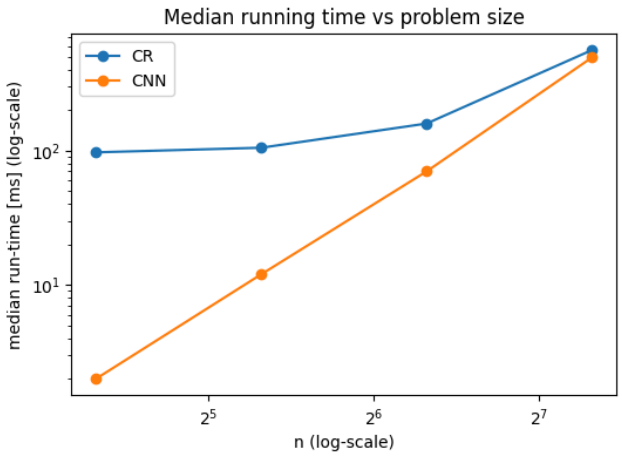


Document 2 : Histogramme représentant la distribution des ratio compétitif

L'histogramme met clairement en évidence la différence de stabilité entre les deux heuristiques : la courbe bleue (CR) forme une cloche étroite centrée autour de $\rho \approx 1.6$, alors que la courbe orange (CNN) est bimodale — un premier amas près de 1,0 lorsqu'aucun pont critique n'est touché, puis un second nuage beaucoup plus étalé entre $\rho \approx 2{,}0$ et 3,0 dès que la structure du graphe contrarie la stratégie NN. Autrement dit, CR offre une performance plus prévisible tandis que CNN oscille entre l'optimal et un surcoût pouvant tripler la distance.

B. Temps CPU

Nous évaluons aussi **l'efficacité temporelle** : le graphique ci-dessous montre comment le temps d'exécution médian de chaque algorithme évolue lorsque la taille n du graphe augmente.

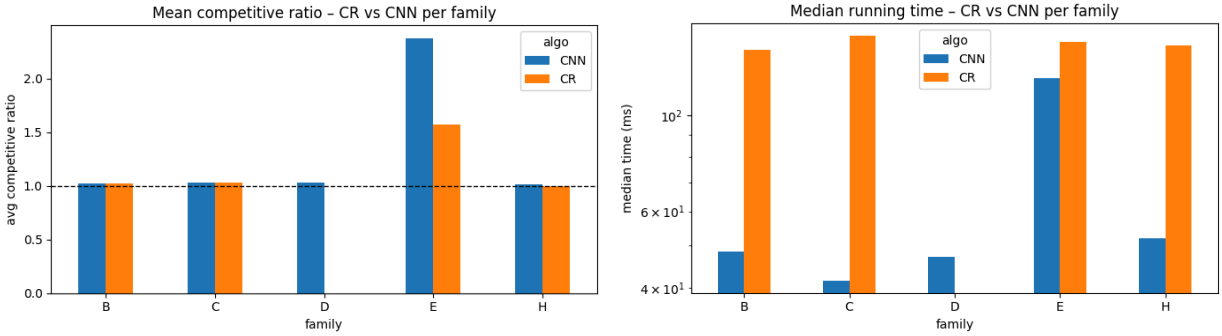


Document 3 : Temps d'exécution médian en fonction du nombre de noeuds

CNN domine nettement en temps de calcul, mais CR rattrape son retard quand la taille explose, ce qui laisse entrevoir un éventuel croisement des courbes pour de très grands graphes ; on le voit à la forme des tracés : la courbe CNN monte régulièrement (presque droite en log-log), tandis que celle de CR reste plate puis franchit un palier brutal avant de repartir, rapprochant ainsi progressivement les deux algorithmes.

C. Bilan par famille : précision vs temps de calcul

Pour compléter l'analyse globale, nous agrégeons maintenant les résultats par famille de graphes ; les deux diagrammes suivants opposent CR et CNN sur le ratio compétitif moyen et le temps médian d'exécution.



(i) ratio compétitif moyen par famille (ii) temps médian d'exécution par famille

Document 4 : CR vs CNN par famille de graphes

Le premier histogramme comparant, famille par famille, le ratio compétitif moyen obtenu par CR et par CN, nous montre :

- B, C, H – Uniforme, clusters et hub-and-spoke : les deux barres sont collées à $\rho = 1$; CR et CNN sont ici virtuellement optimaux.
- E – Adversarial : l'écart se creuse nettement ; CR plafonne à ~1.6 quand CNN dépasse 2.3, confirmant l'avantage du routage cyclique quand la tournée est sévèrement fragmentée.
- D – Grille avec diagonales : la barre orange (CR) est absente car toutes les exécutions CR sur cette famille ont échoué (timeout) ; seul CNN fournit donc un ratio, très proche de 1. Sur la famille D, plusieurs diagonales « bon marché » sont bloquées ; la règle de saut de CR teste alors de nombreuses arêtes successives et son temps explose, dépassant le délai fixé, d'où l'absence de barre CR dans cette colonne.

Le second histogramme, qui compare le temps moyen d'exécution par famille, montre que CNN reste systématiquement le plus rapide ; l'écart est modeste sur la famille adversariale E, mais nettement plus marqué sur les familles B, C et H. L'absence de barre CR pour D reflète les mêmes timeouts : CNN boucle en ≈ 45 ms, tandis que CR n'a pas terminé dans la limite fixée.

En bref : hors scénario adversarial, CNN combine généralement la même qualité de parcours que CR avec un temps de calcul très inférieur ; la seule exception marquante reste la famille E, où CR accepte un surcoût modéré mais constant tandis que CNN peut plus que doubler la distance.

IV. Discussion

Les essais font ressortir deux comportements bien distincts. Le routage cyclique (CR) garantit toujours un détour limité ; même lorsque des arêtes sont bloquées, la longueur du trajet reste sous contrôle. Ce confort se paie toutefois en temps de calcul : dès que le graphe est dense ou qu'il contient une grille truffée de diagonales – comme dans la famille D – la tournée de Christofides suit justement ces diagonales bon marché. Si plusieurs sont bloquées, la règle de « saut » de CR doit tester, l'une après l'autre, de nombreux raccourcis avant d'en trouver un libre ; cette cascade d'essais ralentit fortement l'algorithme et finit par provoquer les time-outs observés.

La stratégie CNN, à l'inverse, termine très vite sur les petites et moyennes instances et reste la plus rapide tant que les blocages restent dispersés. Mais son avance peut s'effondrer lorsque les arêtes interdites forment une longue portion continue du tour – scénario de la famille E : progressant sommet par sommet, CNN bute alors successivement sur chaque arête bloquée et accumule des détours, tandis que CR franchit l'obstacle d'un seul saut. Ainsi, la topologie particulière de D pénalise d'abord CR, alors que la construction adversariale de E frappe surtout CNN.

V. Conclusion

Sur l'ensemble de nos tests, CNN est l'option par défaut : il fournit la même qualité de parcours que CR dans la plupart des cas, tout en terminant beaucoup plus vite. CR ne devient incontournable que dans un contexte précis : présence de blocs d'arêtes critiques, comme dans l'instance adversariale, ou besoin d'une performance garantie quels que soient les blocages. En résumé, CNN pour la réactivité au quotidien ; CR en secours lorsque la robustesse prime.