Généralisation des techniques d'énumération pour les méthodes variationnelles quantiques

Lorenzo Gaggini

Université de Montpellier, LIRMM

Plan de la présentation

- Introduction, motivation et objectifs
- Méthodes variationnelles et algorithmes d'énumération
- Représentations opératoires, partitions et encodage minimal
- Implémentation, résultats et perspectives
- Conclusion

Le voyageur de commerce

- Le **problème du voyageur de commerce** (TSP) consiste à trouver le plus court chemin passant par un ensemble de villes, en visitant chaque ville une seule fois et en revenant au point de départ.
- Le record confirmé pour une instance résolue de façon exacte sur ordinateur classique est de 85 900 villes, atteint en 2006.(Applegate et al. (2006))

Introduction, motivation et objectifs

Figure 1: Résultat méthode quantique (16 villes, simulation) (Meghazi and Bourreau 2025)

À ce jour, les méthodes quantiques permettent de traiter des instances allant typiquement de 5 à 20 villes.

Introduction, motivation et objectifs

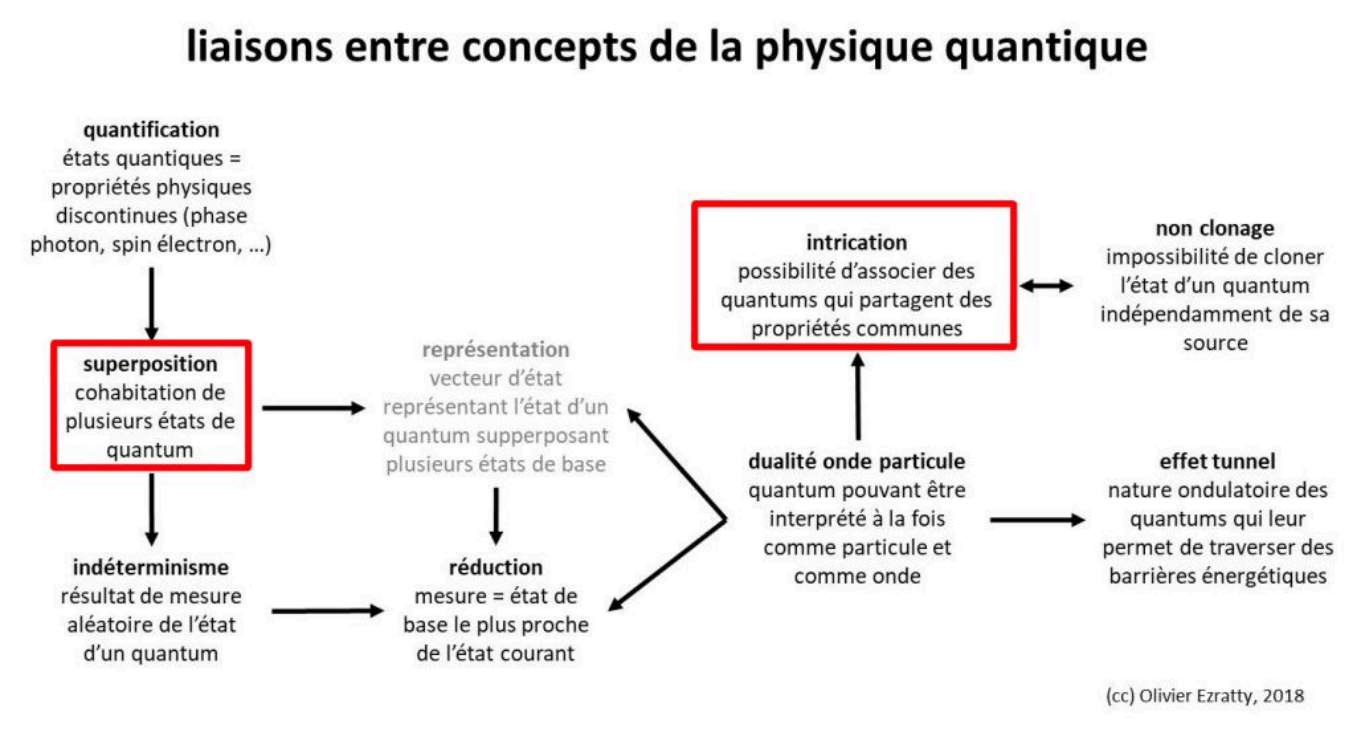


Figure 2: Schéma issu de Ezratty (2018)

- Pourquoi l'optimisation combinatoire en quantique ?

Introduction, motivation et objectifs

- Limitation des architectures NISQ actuelles : le nombre de qubits utilisables reste faible (~30)

Introduction, motivation et objectifs

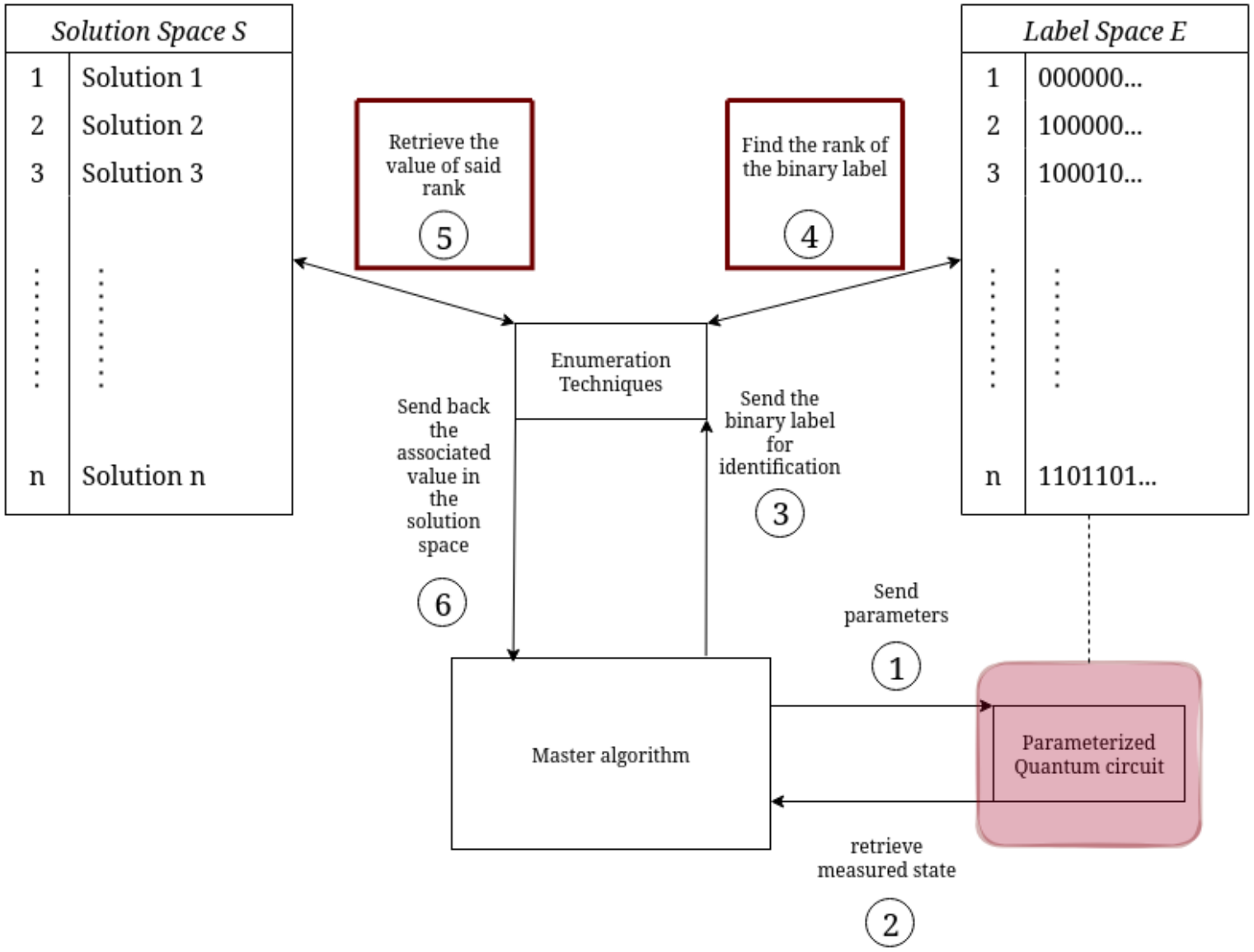


Figure 3: Schéma du protocole algorithmique.

Introduction, motivation et objectifs

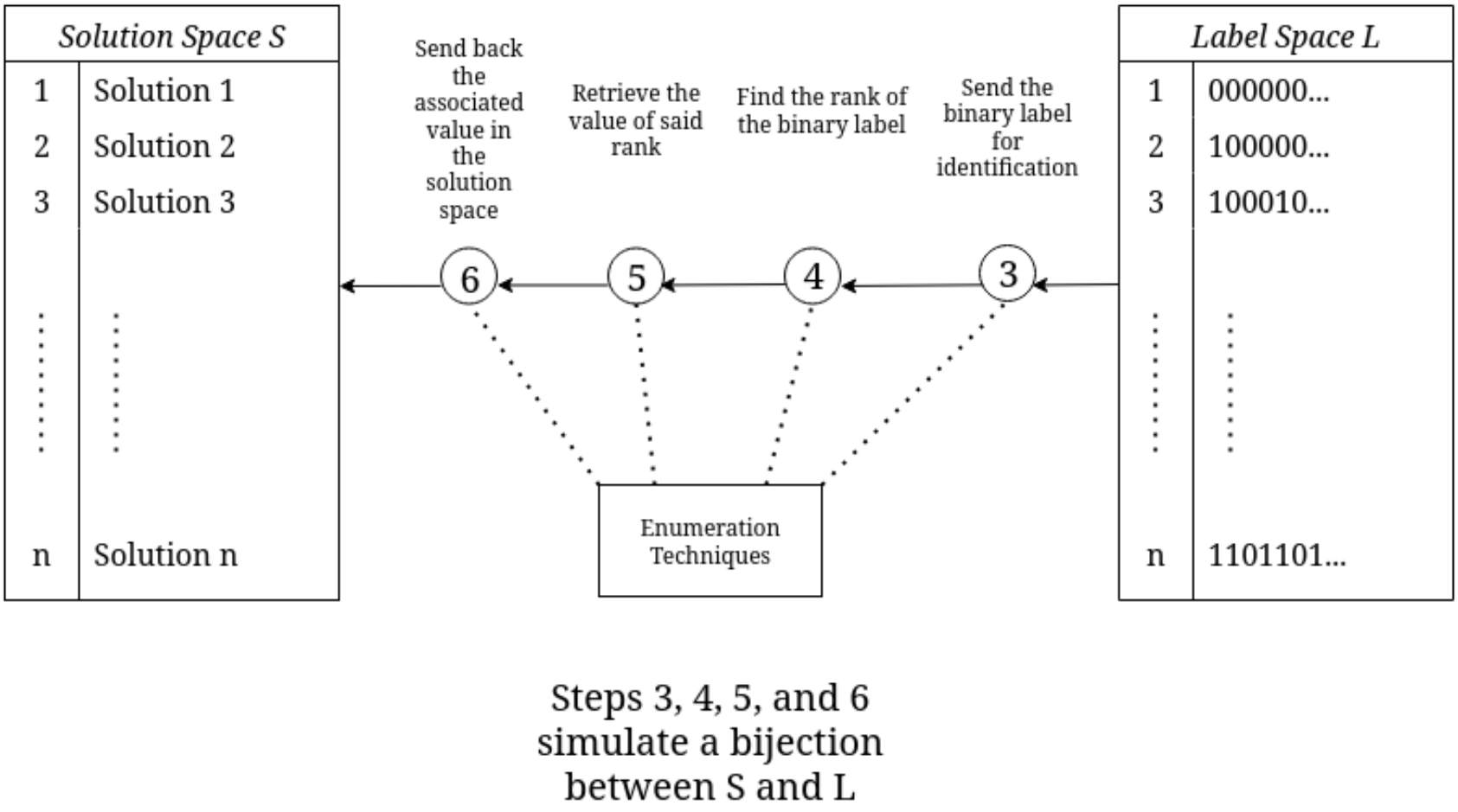


Figure 4: Proposer une bijection explicite entre espace des solutions et représentation binaire

Introduction, motivation et objectifs

Problématiques :

- 1 : Comment encoder efficacement un espace de solutions combinatoires sur circuit quantique ?
- 2 : Comment garantir la validité et l'implémentabilité de la bijection ?
- 3 : Quels sont les impacts sur la performance des algorithmes hybrides ?

Les principaux encodages pour le TSP (Schnaus et al. (2024))

- TSP via `QUBO` : Chaque permutation est représentée par une matrice binaire X de taille $n \times n$, où $X_{i,j} = 1$ si la ville i est visitée en position j, 0 sinon.

  Coût : n^2 qubits.

- TSP via `HOBO` : Chaque permutation est représentée par une séquence de labels binaires, où chaque ville i est associée à un mot binaire unique de longueur $\log_2(n)$.

  Coût : $n \log_2(n)$ qubits.

- TSP via `Permutation Encoding` : Représentation directe des permutations sous forme d'indices. Chaque permutation est associée à un entier unique compris entre 0 et $n! - 1$.

  Coût : $\log_2(n!)$ qubits.

Avec une limite de 30 qubits disponibles :

- TSP via `QUBO` : $\boxed{n = 5}$ (car $n^2 = 25$ qubits)
- TSP via `HOBO` : $\boxed{n = 9}$ (car $n \log_2(n) = 9 \times \log_2(9) \approx 29$ qubits)
- TSP via `Permutation Encoding` : $\boxed{n = 12}$ (car $\log_2(12!) \approx 29$ qubits)

Algorithmes d'énumération pour le `Permutation Encoding`

- Code de Gray : énumération des mots binaires par inversion de bit
- Trotter-Johnson : énumération des permutations par transpositions adjacentes

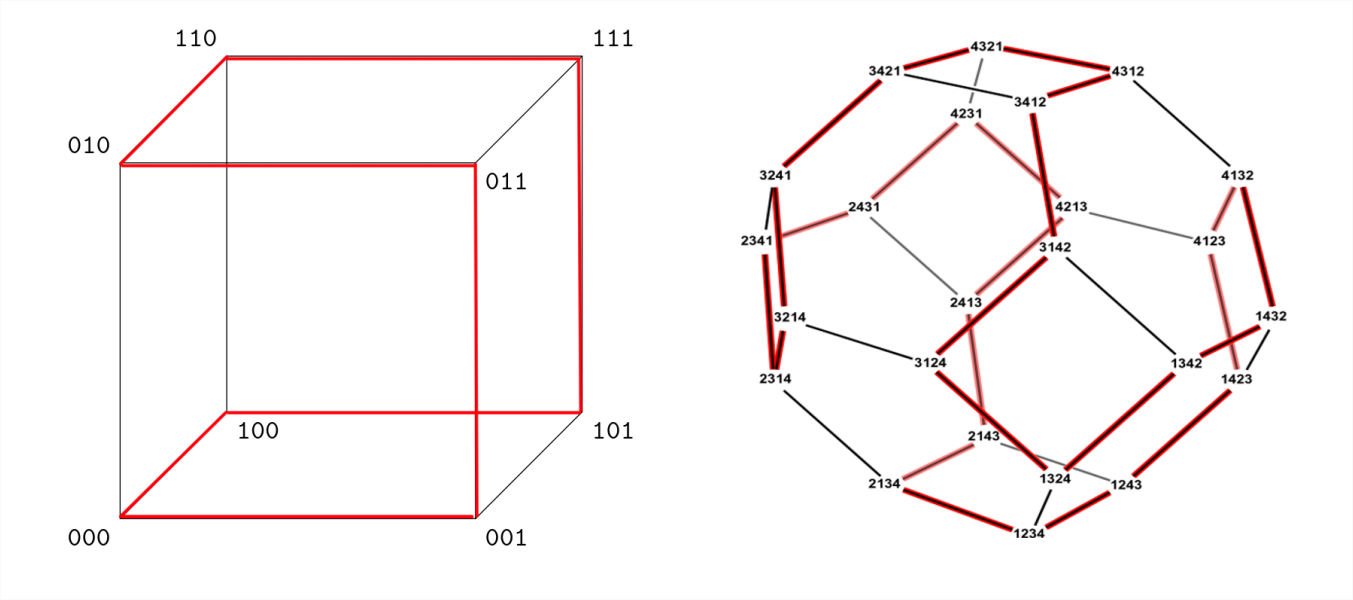


Figure 5: Algorithmes d'énumération et parcours hamiltoniens

Définitions

Soit E un ensemble fini (appelé **espace de symboles** ou **espace d'états**).

On appelle **représentation opératoire** de G sur E la donnée d'une **action opératoire** de ces générateurs sur E, qui est un morphisme de monoïdes :

$$\pi : \Sigma^* \to \mathrm{Part}(E), \tag{1}$$

tel que :

- Σ^* est le monoïde engendré par Σ (les mots sur Σ par concaténation),
- $\mathrm{Part}(E)$ désigne l'ensemble des applications partielles $f : E \rightharpoonup E$.

Représentations opératoires, partitions et encodage

- Chaque générateur $\sigma \in \Sigma$ est envoyé sur une transformation partielle $\pi(\sigma)$
- $\pi(w)$ est défini pour tout mot $w = \sigma_{i_1} \cdots \sigma_{i_k} \in \Sigma^*$ par :

$$\pi(w) := \pi(\sigma_{i_1}) \circ \cdots \circ \pi(\sigma_{i_k}), \tag{2}$$

lue de gauche à droite, où $\circ$ est la composition usuelle de fonctions.

- Objectif : modélisation des algorithmes d'énumération comme suites d'instructions sur un espace d'état.

Représentations opératoires, partitions et encodage

Soient E et E' deux espaces d'états de même cardinalité. Soient (G, Σ, E, π) et (G', Σ', E', π') deux représentations opératoires.

On dit qu'il existe un **isomorphisme de représentations opératoires** entre les deux systèmes s'il existe :

- une bijection $\varphi : \Sigma \to \Sigma'$,
- une bijection $\theta : E \to E'$, telles que pour tout mot $w = \sigma_{i_1} \cdots \sigma_{i_k} \in \Sigma^*$, on ait :

$$\theta\big(\pi(w)(\cdot)\big) = \pi'\big(\varphi(w)\big)\big(\theta(\cdot)\big) \tag{3}$$

dans le sens des applications partielles (définies là où les deux côtés le sont), **où $\varphi(w)$ est un raccourci de notation pour désigner le mot $\varphi(\sigma_{i_1}) \cdots \varphi(\sigma_{i_k}) \in \Sigma'^*$.**

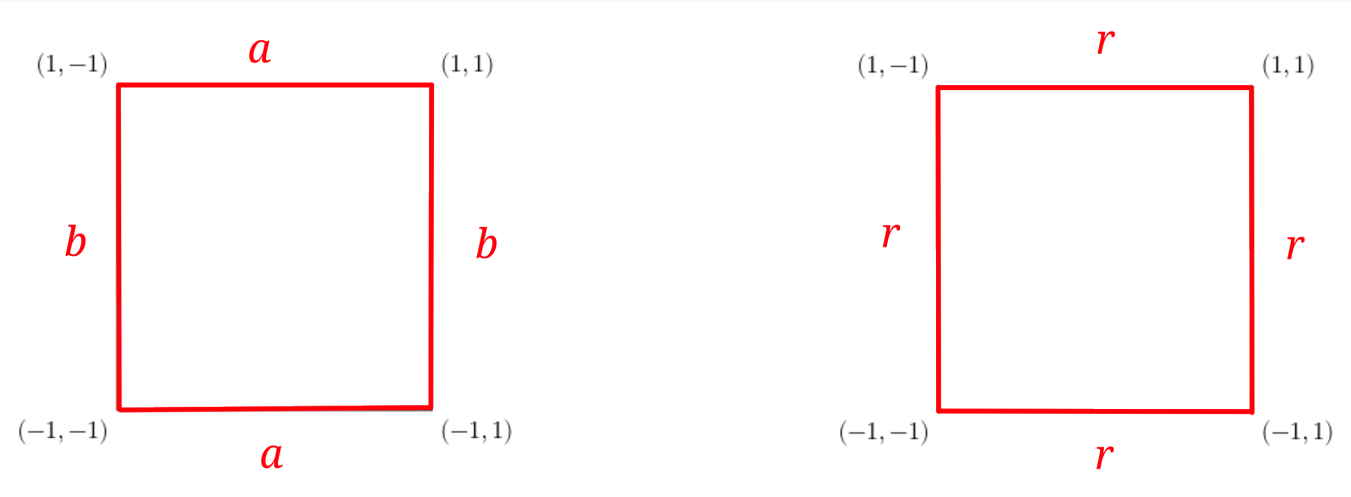
Exemple : le groupe de Klein V_4 et le groupe cyclique C_4

- Espace d'états : $E = \{(1,1),\ (1,-1),\ (-1,-1),\ (-1,1)\}$
- Représentation opératoire (V_4, Σ, E, π) :
  - $\Sigma = \{a, b\}$, où a est la symétrie verticale et b la symétrie horizontale.
  - Pour tout mot $w \in \Sigma^*$, $\pi(w)$ agit sur E par produit matriciel.
  - Procédure : $w = abab$
- Représentation opératoire (C_4, Σ', E, π') :
  - $\Sigma' = \{r\}$, où r est la rotation d'un quart de tour.
  - Pour tout mot $w' \in {\Sigma'}^*$, $\pi'(w')$ agit sur E par produit matriciel.
  - Procédure : $w' = rrrr$

Représentations opératoires, partitions et encodage

- Isomorphisme de représentations opératoires :
  - Bijection sur les générateurs : $\varphi : \Sigma \to \Sigma'$, définie par $a \mapsto r$, $b \mapsto r$, tel que $\varphi(abab) = rrrr$.
  - Bijection sur l'espace d'états : $\theta = \mathrm{id}_E$ (identité sur E).

Représentations opératoires, partitions et encodage



- Générateurs :

$$\pi(a) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\pi(b) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\pi'(r) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

- Trajectoire depuis $(1,1)$:
  $(1,1) \to (1,-1) \to (-1,-1) \to (-1,1) \to (1,1)$

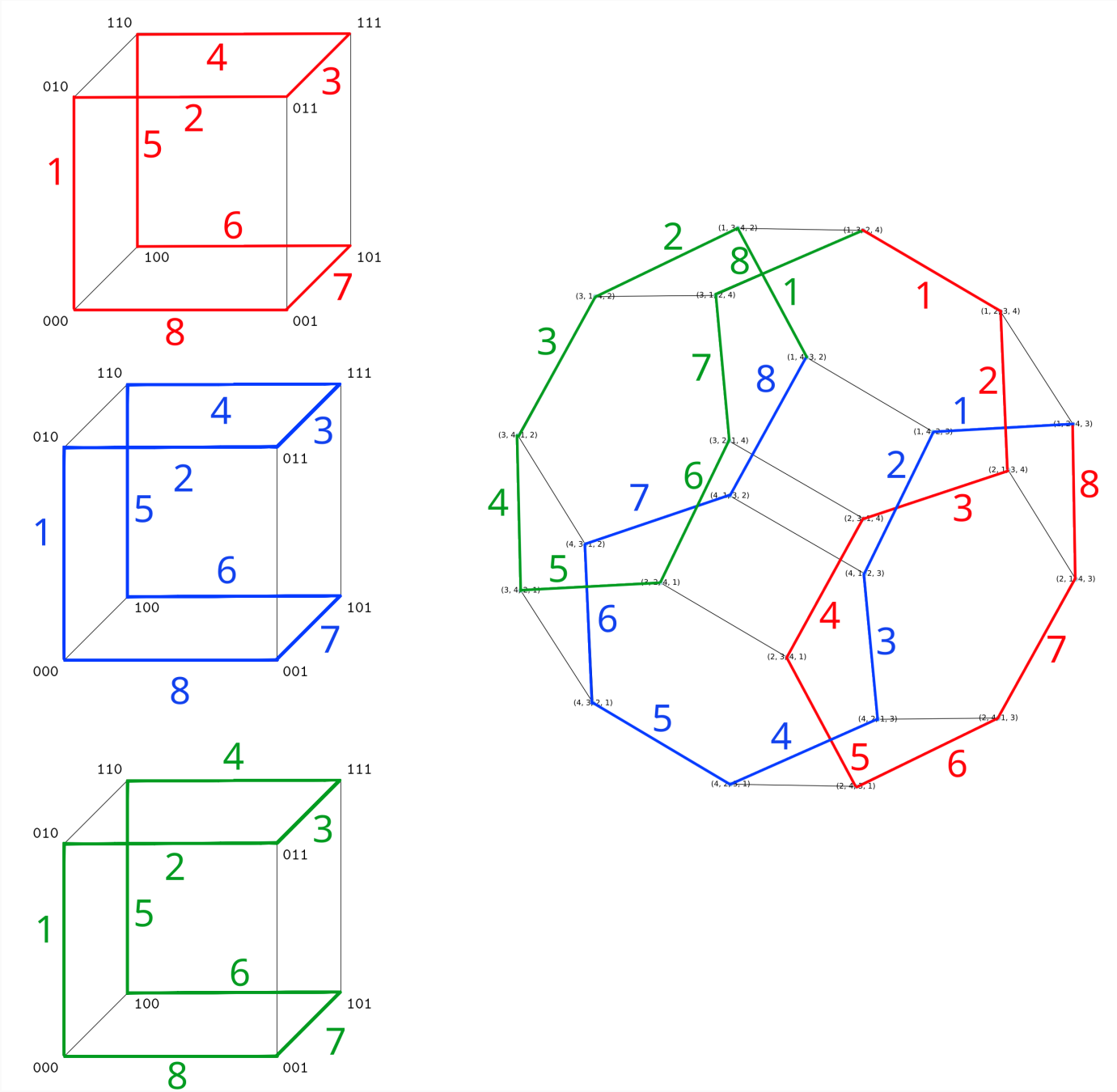


Figure 6: Transfert entre un code de Gray et de Trotter-Johnson

Exemple : Partitions de S_4.

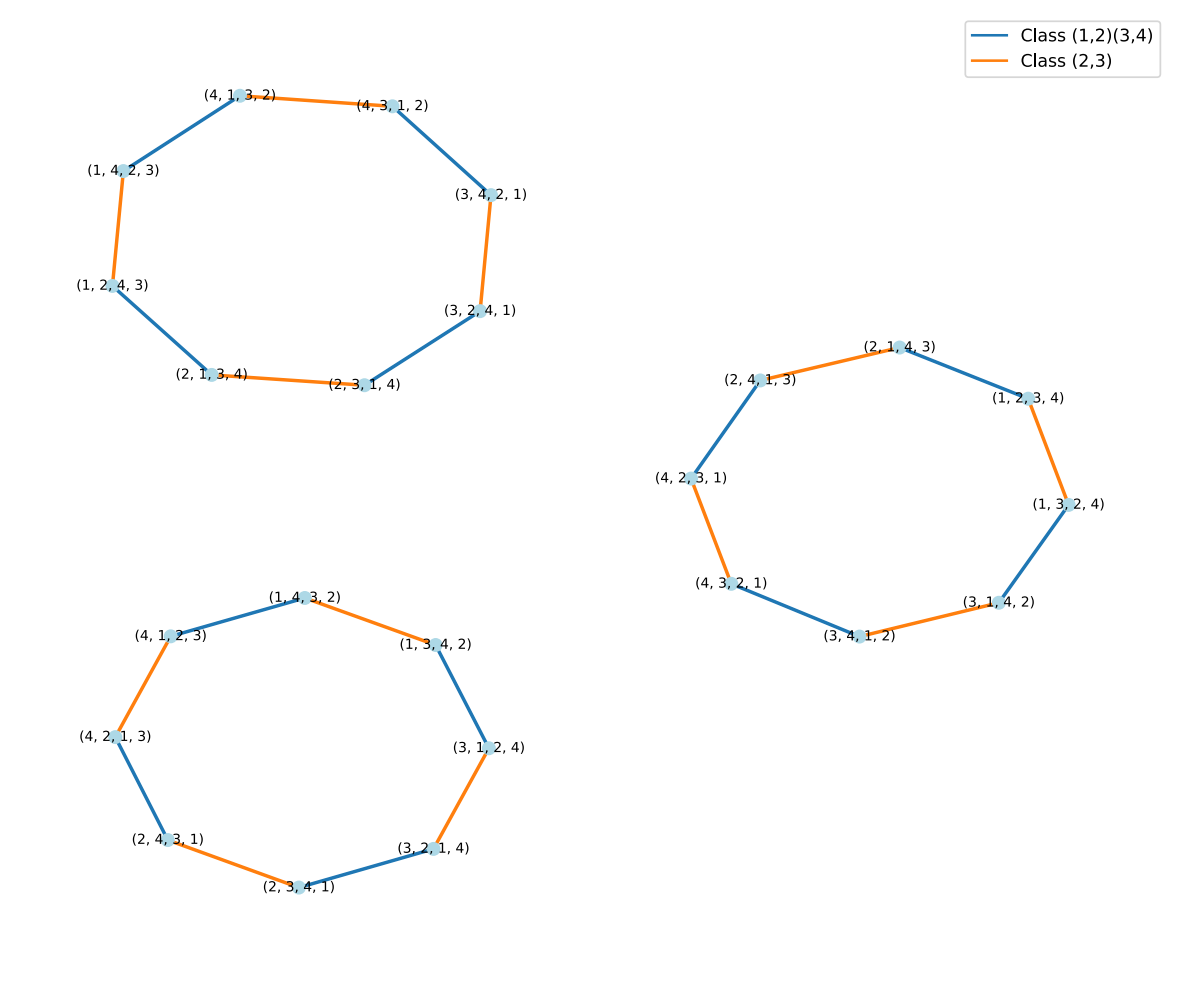
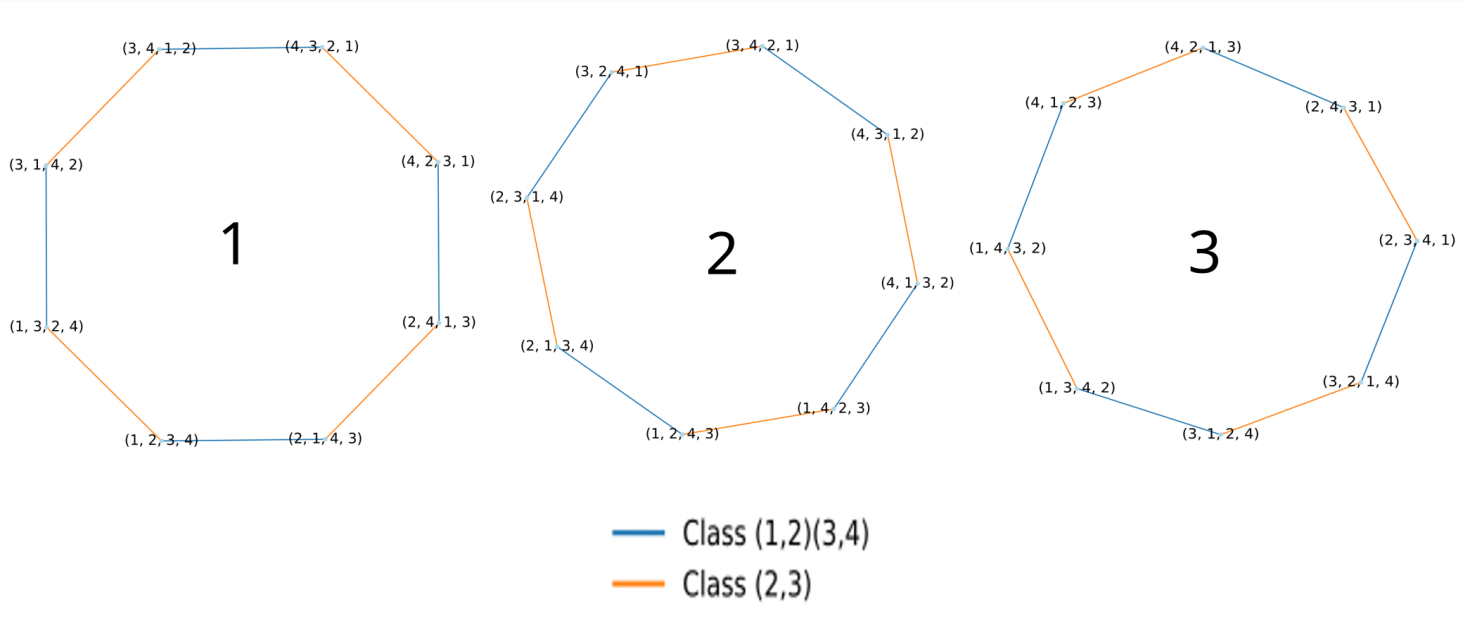


Figure 7: Illustration des orbites générées par le sous-groupe d'ordre 8, engendré par (1,2)(3,4) et (2,3), dans S_4.



Parcour/Index	"1"	"2"	"3"
(2,3)	[1, 2, 3, 4]	[2, 1, 3, 4]	[1, 4, 3, 2]
(1,2)(3,4)	[1, 3, 2, 4]	[2, 3, 1, 4]	[1, 3, 4, 2]
(2,3)	[3, 1, 4, 2]	[3, 2, 4, 1]	[3, 1, 2, 4]
(1,2)(3,4)	[3, 4, 1, 2]	[3, 4, 2, 1]	[3, 2, 1, 4]
(2,3)	[4, 3, 2, 1]	[4, 3, 1, 2]	[2, 3, 4, 1]
(1,2)(3,4)	[4, 2, 3, 1]	[4, 1, 3, 2]	[2, 4, 3, 1]
(2,3)	[2, 4, 1, 3]	[1, 4, 2, 3]	[4, 2, 1, 3]
(1,2)(3,4)	[2, 1, 4, 3]	[1, 2, 4, 3]	[4, 1, 2, 3]

Représentations opératoires, partitions et encodage

Binaire	**"11"**	**"10"**	**"01"**
"000"	[1, 2, 3, 4]	[2, 1, 3, 4]	[1, 4, 3, 2]
"001"	[1, 3, 2, 4]	[2, 3, 1, 4]	[1, 3, 4, 2]
"101"	[3, 1, 4, 2]	[3, 2, 4, 1]	[3, 1, 2, 4]
"100"	[3, 4, 1, 2]	[3, 4, 2, 1]	[3, 2, 1, 4]
"110"	[4, 3, 2, 1]	[4, 3, 1, 2]	[2, 3, 4, 1]
"111"	[4, 2, 3, 1]	[4, 1, 3, 2]	[2, 4, 3, 1]
"011"	[2, 4, 1, 3]	[1, 4, 2, 3]	[4, 2, 1, 3]
"010"	[2, 1, 4, 3]	[1, 2, 4, 3]	[4, 1, 2, 3]

Exemple : voisins binaires de “11” “000” = [1, 2, 3, 4]

Binaire	Permutation	Voisin par transposition directe
”10” ”000”	[2, 1, 3, 4]	Oui ($1 \leftrightarrow 2$)
”01” ”000”	[1, 4, 3, 2]	Oui ($2 \leftrightarrow 4$)
”11” ”100”	[3, 4, 1, 2]	Non
”11” ”010”	[2, 1, 4, 3]	Non
”11” ”001”	[1, 3, 2, 4]	Oui ($2 \leftrightarrow 3$)

Exemple : voisins binaires de “10” “000” = [2, 1, 3, 4]

Binaire	Permutation	Voisin par transposition directe
”11” ”000”	[1, 2, 3, 4]	Oui ($2 \leftrightarrow 1$)
”01” ”000”	[1, 4, 3, 2]	Non
”10” ”100”	[3, 4, 2, 1]	Non
”10” ”010”	[1, 2, 4, 3]	Non
”10” ”001”	[2, 3, 1, 4]	Oui ($3 \leftrightarrow 1$)

Partition optimale

La procédure de ranking sur l'espace binaire partitionné est :

1. On découpe le mot binaire s en blocs selon la partition optimale I.
2. Pour chaque bloc, on calcule son rang local.
3. Le rang global R est obtenu en combinant les rangs locaux :

$$R = \sum_{j=1}^{k} r_j \cdot \left(\prod_{l=1}^{j-1} M_l \right) \tag{4}$$

Chaque bloc compte, et on additionne les rangs en tenant compte de la taille des blocs précédents.

Implémentation, résultats et perspectives

- Algorithmes maîtres : F-VQE (descente de gradient), COBYLA (sans gradient)

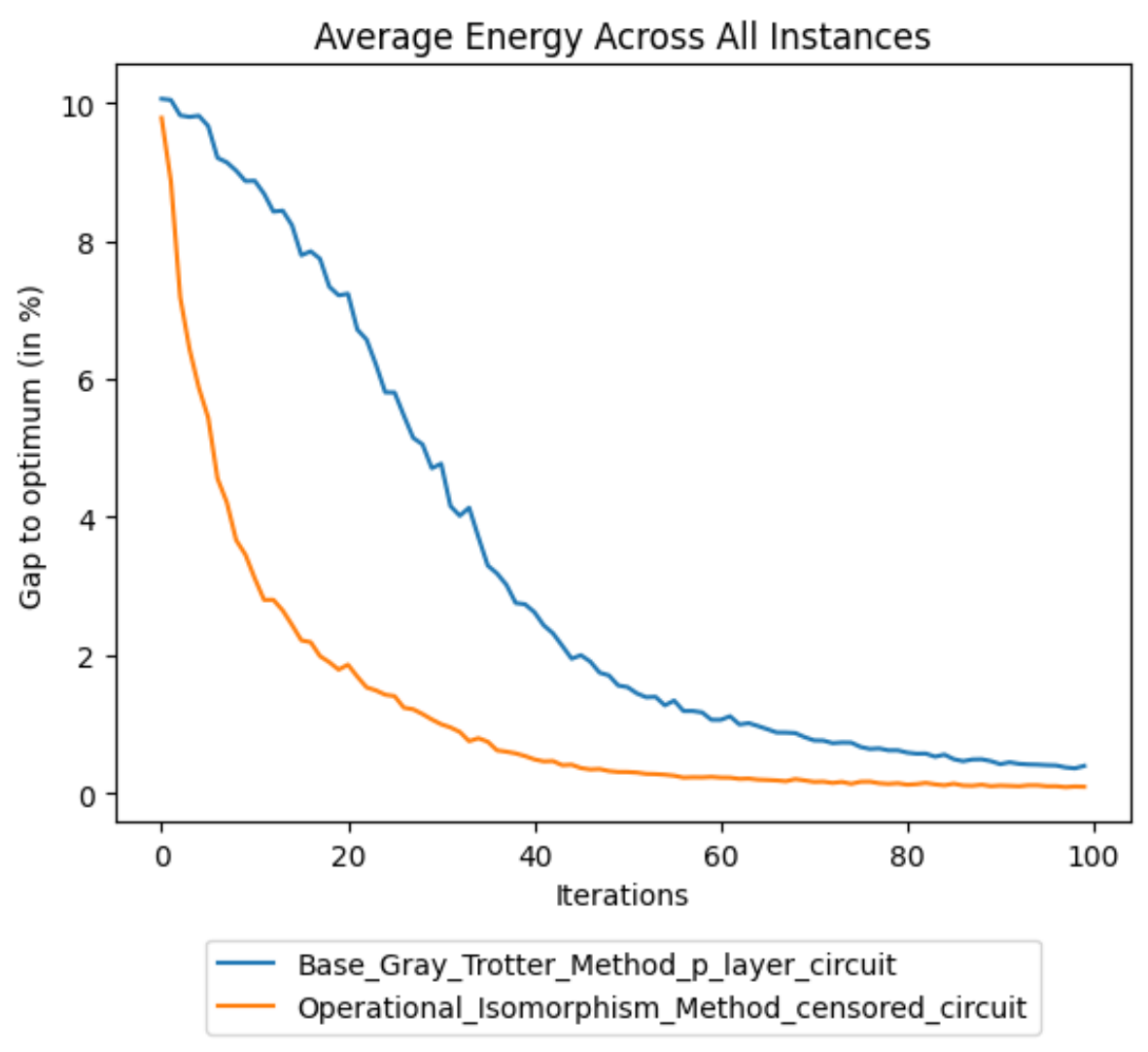


Figure 8: (F-VQE, S_4, 100 instances)

Exemple : Partitions de S_8 en orbite sous l'action d'un sous-groupe d'ordre 384

Parcours/Index	"1"	"2"	...	"105"
"1"	[1,2,3,4,5,6,7,8]	[1,2,3,4,5,7,6,8]	$\cdots$	[4,5,2,7,6,3,8,1]
"2"	[4,3,2,1,5,6,7,8]			
$\vdots$				
"384"	[1,2,3,4,6,5,7,8]			

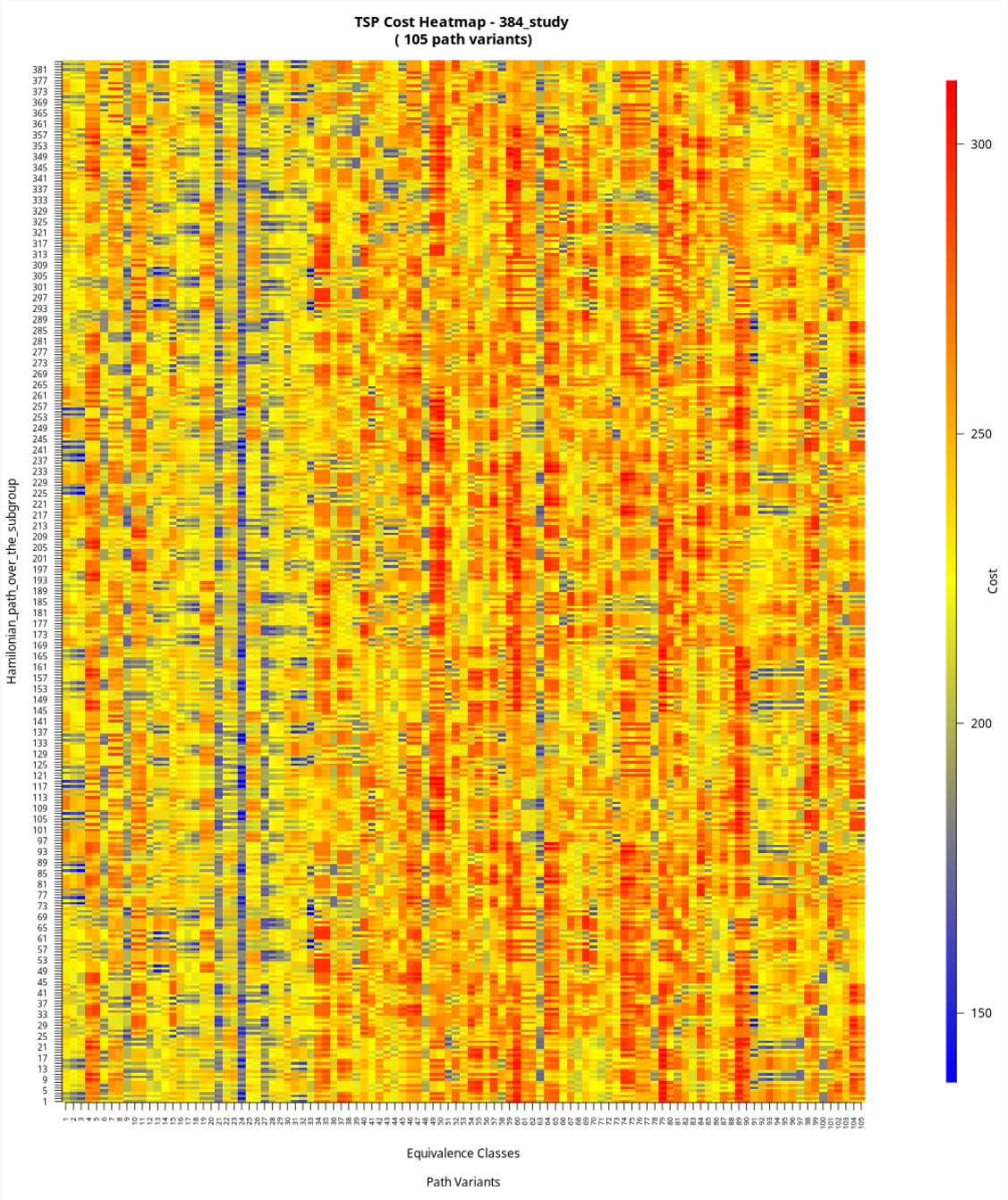


Figure 9: Organisation des coûts issue de la représentation opératoire.

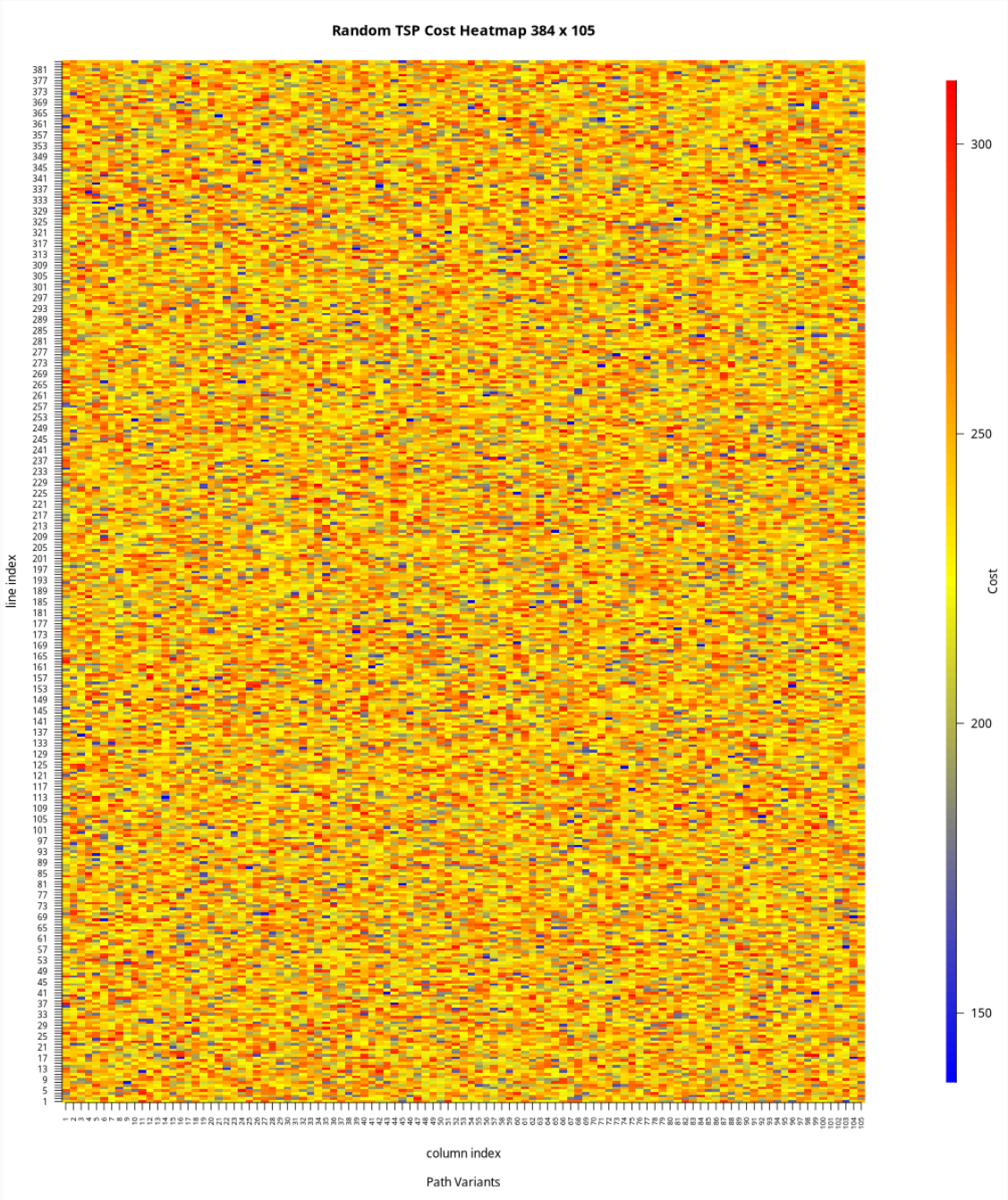


Figure 10: Répartition aléatoire des coûts, pour comparaison.

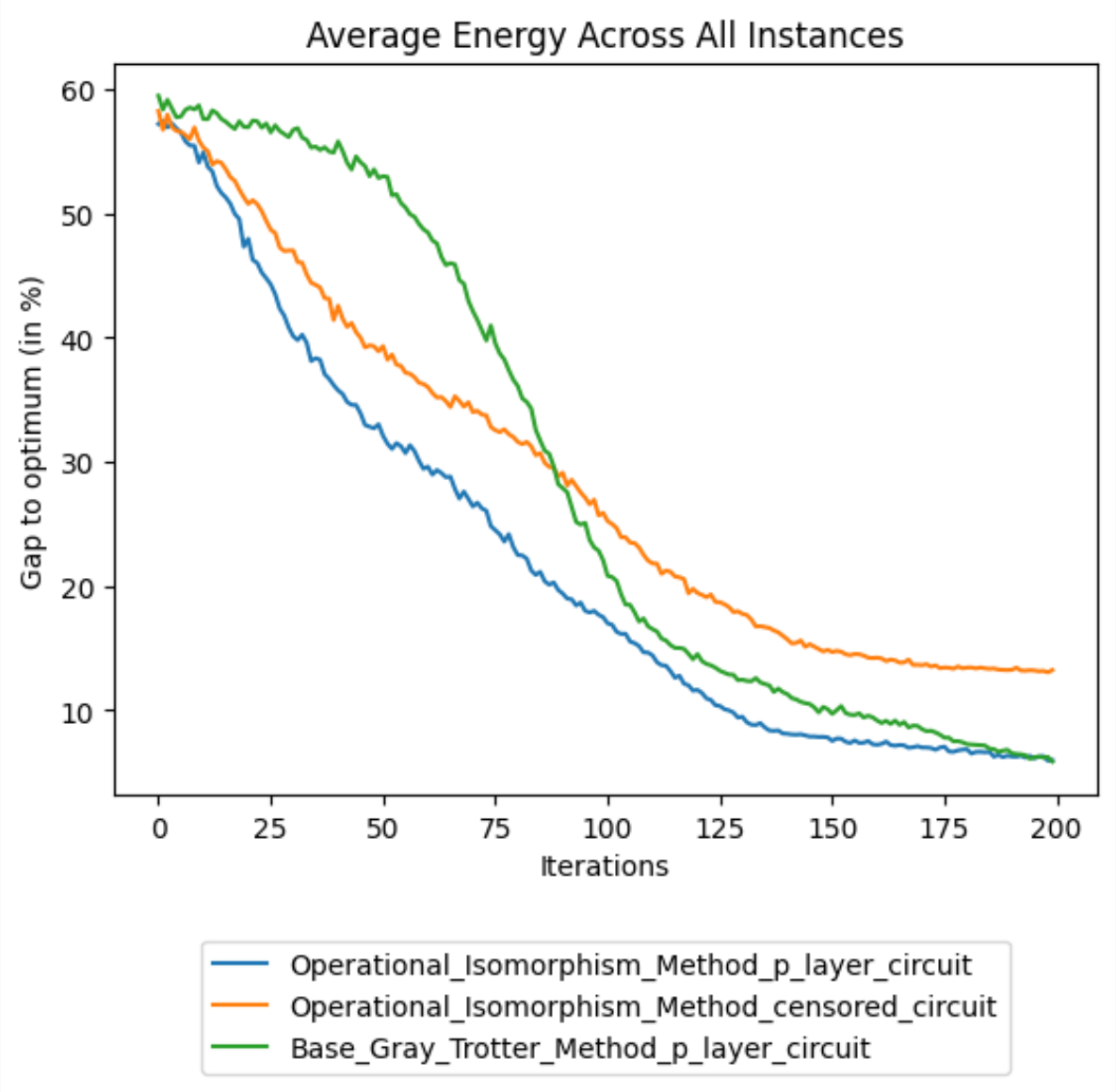


Figure 11: F-VQE, S_8, 40 instances

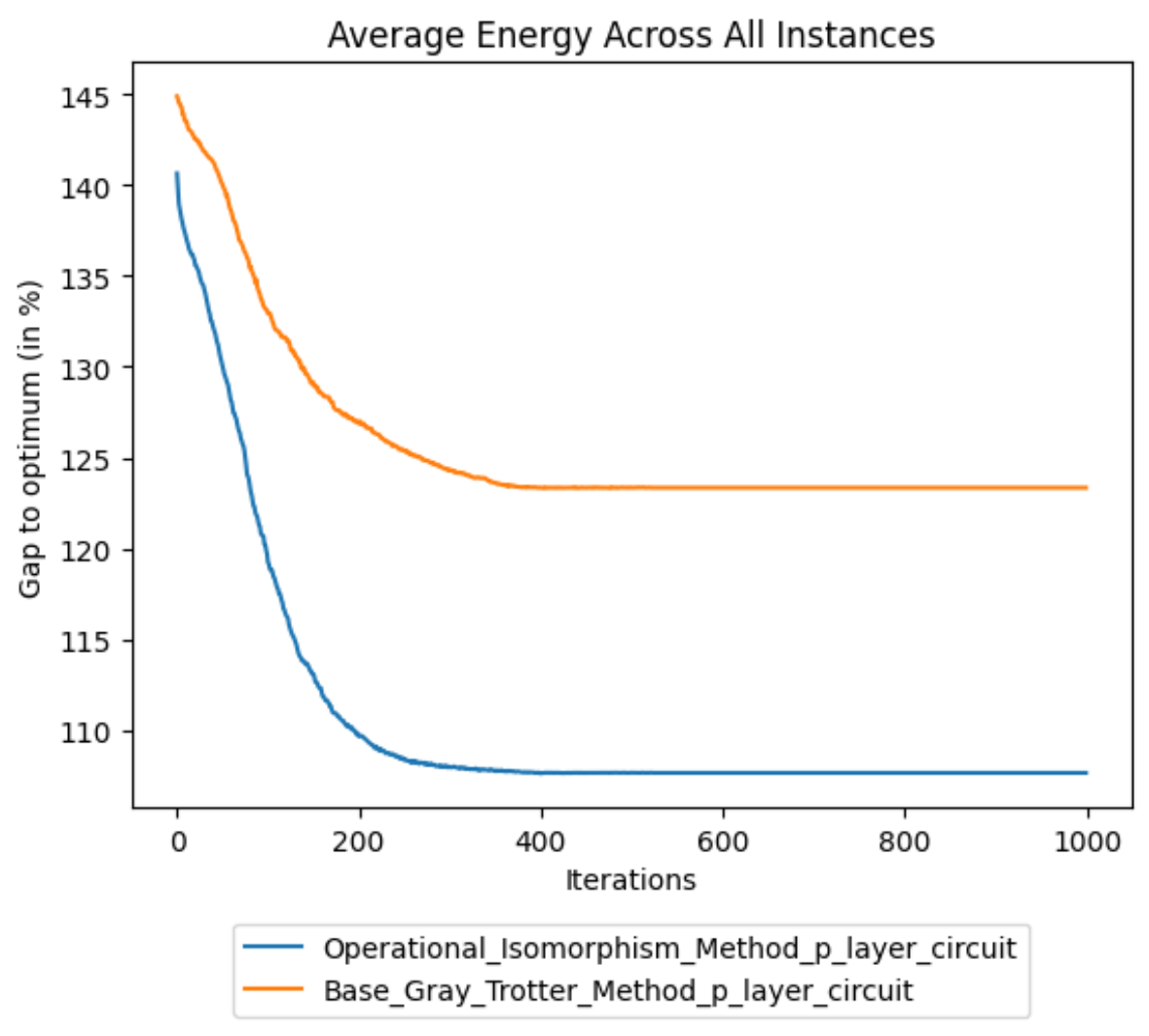


Figure 12: COBYLA, S_{16}, 30 instances

Difficulté

- Scalabilité : Construire des procédures opératoires couvrantes pour de grands groupes en toute généralité.

Perspectives

- Généralisation à d'autres problèmes combinatoires
- Optimisation du design des algorithmes maîtres
- Généralisation à d'autres architectures que les méthodes variationnelles

Conclusion

- Problématique 1 : Comment encoder efficacement un espace de solutions combinatoires sur circuit quantique ?
  - Réponse : Par isomorphisme de représentations opératoires.
- Problématique 2 : Comment garantir la validité et l'implémentabilité de la bijection ?
  - Réponse : Par décomposition en blocs et ranking local.
- Problématique 3 : Quels sont les impacts sur la performance des algorithmes hybrides ?
  - Réponse : Les simulations ont montré une accélération de la convergence par rapport à l'implémentation originale.

Supplément

Partition optimale

Table 1: Décompositions en sous-group et partitions binaires pour S_8

Taille	Décomposition	Partition binaire	Facteurs premiers utilisés
64	$8! = 64 \times 630$	$(6) \mid (1,2,3,4)$	$64 = 2^6, 630 = 2 \times 3 \times 5 \times 7$
128	$8! = 128 \times 315$	$(7) \mid (2,3,4)$	$128 = 2^7, 315 = 3^2 \times 5 \times 7$
192	$8! = 192 \times 210$	$(6,2) \mid (1,3,4)$	$192 = 2^6 \times 3, 210 = 2 \times 3 \times 5 \times$
384	$8! = 384 \times 105$	$(7,2) \mid (3,4)$	$384 = 2^7 \times 3, 105 = 3 \times 5 \times 7$
720	$8! = 720 \times 56$	$(4,4,2) \mid (3,3)$	$720 = 2^4 \times 3^2 \times 5, 56 = 2^3 \times 7$

Gap

Le coût moyen C pour l'itération courante est alors défini par :

$$C = \frac{\sum_i c_i \cdot n_i}{N_{\text{shots}}} \tag{5}$$

Pour évaluer la performance de l'algorithme, on calcule à chaque itération l'écart relatif à la solution optimale connue C_{opt} :

$$\text{gap} = \left(\frac{C}{C_{\text{opt}}} \times 100 \right) - 100 \tag{6}$$

Applegate, David L., Robert E. Bixby, Vasek Chvátal, and William J. Cook. 2006. *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ: Princeton University Press.

Ezratty, Olivier. 2018. "Opinions Libres." https://www.oezratty.net.

Meghazi, Imran, and Eric Bourreau. 2025. "Encodage Indirect Amélioré Pour Résolution Du Problème Du Voyageur de Commerce Avec Algorithmes Variationnels Quantiques."

Schnaus, Manuel, Lilly Palackal, Benedikt Poggel, Xiomara Runge, Hans Ehm, Jeanette Miriam Lorenz, and Christian B. Mendl. 2024. "Efficient Encodings of the Travelling Salesperson Problem for Variational Quantum Algorithms." In *2024 IEEE International Conference on Quantum Software (QSW)*, 81–87. IEEE. https://doi.org/10.1109/qsw62656.2024.00022.