Projet :
Optimisation de problèmes pseudo-booléen quadratique (QUBO)

Master Informatique - 1ère année

année 2023-2024

1 Contexte

Les problèmes quadratiques à variables binaires (Quadratic Unconstrained Quadratic Problem - UBQP) sont des problèmes combinatoires génériques NP-difficiles qui modélisent des nombreuses applications [1]. Ces problèmes sont aussi les modèles d'entrée de nombreux algorithmes exécutés sur les nouveaux ordinateurs quantiques. Ainsi, de nombreuses recherches tentent de trouver la meilleure modélisation sous forme QUBO, ou tentent de développer des algorithmes sur machines classiques, sur machines quantiques ou de manière hybride (quantique/classique).

Une instance du problème QUBO de dimension n est défini à partir d'une matrice $A = (a_{ij})_{i,j\in\{1,\dots,n\}}$. La fonction objective f à minimiser est définie de la manière suivante lorsque les variables binaires x_i appartiennent à $\{0, 1\}$:

$$\forall x \in \{0,1\}^n,\ f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j$$

ou de manière équivalente avec les variables binaires $z_i \in \{-1, 1\}$ et une matrice $Q = (q_{ij})_{i,j\in\{1,\dots,n\}}$,

$$\forall z \in \{-1,1\}^n,\ f(z) = \sum_{i=1}^{n} q_{ii}z_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} q_{ij}z_iz_j$$

Le but du projet est de développer des algorithmes d'optimisation performants sur les problèmes QUBO en utilisant des techniques d'optimisation combinatoire et numérique.

2 Méthodologie

Les algorithmes seront testés et comparés sur les instances PUBO_i qui tiennent compte de l'importance des variables [3]. Le dépôt avec 1000 instances du problème et du code python et c++ de la fonction d'évaluation est disponible ici. : `https://gitlab.com/verel/pubo-importance-benchmark/`.

Vous pouvez entrainer vos algorithmes à partir des 990 premières instances (de l'identifiant 0 à 989), et vous devez présenter les résultats (performances des algorithmes, etc.) sur les 10 dernières instances (de l'identifiant 990 à 999). Les algorithmes devront tous avoir le même temps de calcul, *i.e.* 30 secondes de calcul sur la machine "étalon".

Pour connaitre l'équivalent du temps sur votre machine de la machine étalon, compiler le code fournis (`code_prj_2024.zip`) et exécuter le. Le temps d'exécution sur la machine étalon est de 1 seconde. Vous pouvez alors calculer le ratio du temps d'exécution sur votre propre machine. Indiquer dans votre rapport, le temps d'exécution du code étalon sur votre machine. Les algorithmes doivent s'exécuter sur un temps égale à 30 secondes sur la machine étalon.

Les algorithmes qui par nature sont stochastiques, devront être tester par au moins 30 exécutions.

3 Techniques d'optimisation combinatoire

Dans cette section, nous vous proposons de mettre au point des algorithmes d'optimisation combinatoire pour résoudre le problème d'optimisation.

Questions :

2.1 Initialisation :

2.1.a Coder une recherche aléatoire.

2.2.b Tester et commenter les résultats de votre recherche aléatoire.

2.2 Calcul incrémental :

2.2.a En utilisant la définition du problème QUBO, calculer la variation de fitness : $\Delta_i f(x) := f(x \oplus i) - f(x)$ où $x \oplus i$ est la solution x dans laquelle la variable x_i est modifiée.

2.2.b Comparer la complexité de calculer directement $f(x \oplus i)$ et $\Delta_i f(x)$. Conclure.

2.2.c Calculer la variation de la variation : $\Delta^{(2)}_{i,j} f(x) := \Delta_i f(x \oplus j) - \Delta_i f(x)$.

2.2.d En utilisant la formule de 2.2.c, calculer la complexité du calcul de la mise à jour de la variation de fitness pour tous les voisins de x (à une distance de Hamming de 1) lorsqu'une variable x_j est modifiée.

2.3 Recherches locales :

2.3.a Coder un algorithme de type Iterated Local Search (ILS).

2.3.b Coder un algorithme de type Tabu Search (TS).

2.3.c Comparer les performances de vos algorithmes ILS et TS.

2.4 Pour aller plus loin (optionnel) :

2.4.a Proposer et coder votre propre algorithme (algorithme évolutionnaire, opérateurs différents, etc.)

2.4.b Comparer les performances avec les algorithmes ILS et TS.

4 Techniques d'optimisation numérique

Dans cette section, nous vous proposons de mettre au point un algorithme d'optimisation numérique. En effet, il est possible de considérer le même problème avec des variables réelles (flottants) entre $[-1, 1]$ au lieu des variables binaires $\{-1, 1\}$. Cette technique est récemment mise en œuvre pour le problème SAT [2].

Le principe algorithmique utilise une descente de gradient pour minimiser f avec les variables réelles de $[-1, 1]^n$. Si après un pas de gradient, une variable z_i est en dehors de $[-1, 1]$, alors z_i est projeté de nouveau sur $[-1, 1]$ par $z_i = \text{sign}(z_i)$. À la fin de l'algorithme de gradient, les variables sont transformées de la même manière par $z_i = \text{sign}(z_i)$ donnant une solution au problème combinatoire QUBO original. On pourrait imaginer d'autres techniques pour conserver les variables dans l'intervalle $[0, 1]$ (paramétrisation, pénalité, etc.).

Questions :

3.1 Descente de gradient :

3.1.a Coder une descente de gradient pour minimiser f qui projette la solution finale sur $\{-1, 1\}^n$.

3.2.b Comparer les performances de la descente de gradient avec les algorithmes combinatoires.

3.2 Algorithme hybride :

3.2.a Imaginer un algorithme qui combine un algorithme combinatoire avec une descente de gradient.

3.2.b Analyser et comparer les performances de votre proposition.

References

[1] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of combinatorial optimization*, 28:58–81, 2014.

[2] Anastasios Kyrillidis, Anshumali Shrivastava, Moshe Vardi, and Zhiwei Zhang. Fouriersat: A fourier expansion-based algebraic framework for solving hybrid boolean constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1552–1560, 2020.

[3] Sara Tari, Sébastien Verel, and Mahmoud Omidvar. Puboi: A tunable benchmark with variable importance. In *European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar)*, pages 175–190. Springer, 2022.