CONCEPTION D'UN PROGRAMME DE THREADING PAR DOUBLE PROGRAMMATION DYNAMIQUE

Nadezhda ZHUKOVA

Université Paris Cité, Master 2 Bioinformatique

<nadezhda.zhukova@etu.u-paris.fr>

Septembre 2024

RÉSUMÉ

Ce projet porte sur la prédiction de la structure des protéines en utilisant la méthode de **threading**, implémentée à l'aide de la double programmation dynamique sur Python. Le threading est une approche informatique qui aligne une séquence protéique avec des structures modèles connues afin de prédire sa structure tertiaire, même lorsque la similarité des séquences est faible. Cette approche identifie des modèles candidats en tenant compte des similitudes structurelles, telles que les structures secondaires prédites et l'accessibilité aux solvants. En alignant la séquence sur des modèles structurellement similaires, le threading permet de prédire les repliements des protéines, ce qui en fait un outil important en bioinformatique, ainsi qu'en médecine et en biotechnologie.

GitHub

L'implémentation de cet algorithme est disponible sur GitHub : <https://github.com/zhukovanadezhda/protein-threading>

1. INTRODUCTION

La prédiction des structures protéiques à partir des séquences d'acides aminés constitue un enjeu majeur en bioinformatique, car la conformation tridimensionnelle d'une protéine est déterminante pour sa fonction biologique. Selon le modèle conceptuel le plus simple, la structure native d'une protéine correspond au minimum global de son énergie libre, transformant ainsi le processus de repliement en un problème de minimisation de fonction d'énergie.

Si la recherche de ce minimum se faisait en explorant aléatoirement toutes les conformations possibles, le temps nécessaire serait astronomique en raison de la complexité de l'espace conformationnel, comme l'illustre le paradoxe de Levinthal [1] *(LEVINTHAL, 1968)*. Ainsi, il est irréaliste que les protéines recherchent leur minimum global d'énergie de manière aléatoire. Il est donc plus probable que les voies de repliement soient encodées directement dans la séquence, permettant à des fragments plus courts de trouver leurs minima locaux et de guider la structure finale, qui correspondrait souvent à un minimum local d'énergie plutôt qu'au minimum global. Cette observation, combinée à de nombreuses preuves indiquant l'existence d'un nombre limité de replis protéiques naturels *(CHAN ; DILL, 1990; DOOLITTLE, 1992; GREGORET ; COHEN, 1990)*, suggère qu'il devrait être possible de retrouver la structure d'une protéine en s'appuyant sur des fragments issus de protéines dont les structures sont déjà connues.

Il existe actuellement plusieurs approches pour aborder le problème de la prédiction de la structure tertiaire des protéines. La modélisation comparative par homologie repose sur la similarité de la séquence avec des protéines dont la structure 3D est connue. Cette méthode s'appuie sur le principe que des séquences similaires tendent à partager des structures similaires. En revanche, la modélisation *de novo*, ou *ab initio*, tente de prédire la structure sans aucune donnée expérimentale préalable. Cela implique des simulations basées sur des approches thermodynamiques, telles que Monte Carlo *(ZHANG ; CHOU, 1992)*, la dynamique moléculaire, ou des algorithmes utilisant des potentiels d'énergie (comme le potentiel DOPE, utilisé par des logiciels tels que Modeller *(ŠALI ; BLUNDELL, 1993)*).

Enfin, le threading moléculaire est utilisé lorsque la similarité de séquence avec des structures connues est faible. Cette méthode permet de prédire la structure tertiaire en "enfilant" une séquence d'acides aminés sur une structure protéique connue,

1. Une protéine possède un nombre astronomique de conformations possibles, de l'ordre de 3^{198} ou 10^{47}.

puis en évaluant l'adéquation entre la séquence et la structure à l'aide d'un potentiel énergétique. En appliquant cette méthode à une banque de structures, il est possible d'identifier celle qui correspond le mieux à une séquence donnée, ce qui fait du threading une approche utilisée en modélisation moléculaire, surtout lorsque les autres méthodes échouent à trouver des correspondances.

Dans le cadre de ce projet, nous avons développé un programme de threading basé sur la double programmation dynamique, une approche inspirée du programme THREADER *(JONES, 1996)*. La programmation dynamique, qui divise un problème en sous-problèmes résolus individuellement pour obtenir une solution optimale, est couramment utilisée pour l'alignement de séquences biologiques, comme le montrent les algorithmes de Needleman & Wunsch *(NEEDLEMAN; WUNSCH, 1970)* ou de Smith & Waterman *(SMITH; WATERMAN et al., 1981)*. Toutefois, cette méthode n'est pas adapté pour l'alignement des structures non linéaires, telles que celles des protéines.

Pour surmonter ces limitations, nous avons recours à la double programmation dynamique. Contrairement à la programmation dynamique classique, cette méthode prend en compte les différences structurelles non linéaires en réalisant un alignement structurel. Elle utilise une matrice de haut niveau et des matrices de bas niveau pour intégrer les scores d'adéquation, permettant ainsi une évaluation énergétique finale plus précise entre la séquence et la structure.

2. MATÉRIELS ET MÉTHODES

2.1. L'environnement

Le script a été développé en Python, version 3.12.5. Les bibliothèques utilisées sont celles de base de Python, ainsi que les packages suivants : numpy 2.0.0, pandas 2.2.2, joblib 1.4.2, scipy 1.14.1, Bio 1.84. La gestion de l'environnement a été effectuée à l'aide de conda, version 24.7.1.

2.2. Les données

Les données pour les tests de l'algorithme sont divisées en deux examples.

Pour le premiere exemple, des protéines de petite taille (10-50 acides aminés) ont été sélectionnées à partir de la base de données Protein Data Bank (PDB) : 1CRN (hélice alpha et feuillet bêta), 1L2Y (hélice alpha), 1VII (hélice alpha), 5AWL (boucle), 1LE0 et 1LE1 (feuillet bêta).

Pour le deuxieme exemple, des protéines de plus grande taille (70-120 acides aminés) ont également été extraites de PDB. Les protéines 1E68 (hélice alpha) et 3E8V (feuillet bêta) ont été utilisées comme séquences pour lesquelles la structure a été prédite. Les protéines 1E68 (hélice alpha), 1UBQ (hélice alpha et feuillet bêta), 3ZOW (hélice alpha), 3E8V (feuillet bêta), 1TIT (feuillet bêta), 1TDV (hélice alpha et feuillet bêta), et 3ZBV (hélice alpha et feuillet bêta) ont été utilisées comme structures connues pour les comparisons.

2.3. L'algorithme

L'algorithme commence par la préparation et l'analyse des données d'entrée. Les utilisateurs spécifient une liste de séquences protéiques et de modèles structuraux, ainsi que divers paramètres tels que le score de gap, le nombre de processus parallèles, et des options de configuration telles que le mode "dry run" (pour visualiser les actions prévues sans effectuer l'analyse), "verbose" (pour activer une sortie détaillée pour le débogage), et "print_alignments" (pour afficher les alignements entre les séquences et les modèles). Le nom du fichier de sortie est également défini pour enregistrer les résultats. Une fois ces paramètres établis, l'algorithme configure le niveau des logs pour produire des messages en fonction des besoins de l'utilisateur.

La première étape de l'algorithme consiste à charger les données, incluant les séquences, les structures, et les scores DOPE, essentiels pour évaluer la qualité des alignements. Ces scores, qui quantifient l'énergie de l'alignement entre une séquence et un modèle, sont utilisés pour guider le processus d'alignement. Les séquences et les modèles sont chargés à partir des fichiers fournis ou des répertoires par défaut si aucune liste spécifique n'est fournie.

Lorsque l'outil est configuré pour une exécution réelle (et non en mode "dry run"), il procède à l'alignement des séquences avec les modèles. Pour chaque modèle, l'outil extrait les coordonnées des résidus à partir des fichiers PDB et calcule une matrice des distances entre ces résidus. En utilisant cette matrice, l'algorithme remplit des matrices de bas niveau, où chaque matrice correspond à un acide aminé fixé à une position précise de la structure, intégrant les scores DOPE et les scores de gap. Ces matrices de bas niveau, qui représentent les coûts

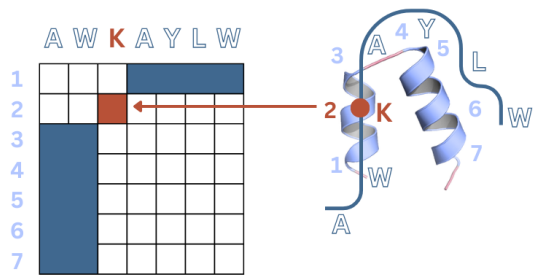

FIGURE 1. Illustration du principe de construction d'une matrice à bas niveau

Type	Temps
Pas de parallélisation	5.36 s ± 0.17 s
Parallélisation des templates	3.81 s ± 0.05 s
Parallélisation des matrices du bas niveau	1.85 s ± 0.10 s
Parallélisation des templates et des matrices du bas niveau	4.41 s ± 0.35 s

TABLE 1. Temps d'exécution pour différentes configurations de parallélisation

d'alignement local, sont ensuite combinées pour produire une matrice de haut niveau qui évalue l'alignement global entre la séquence et chaque modèle.

Enfin, les scores d'énergie calculés pour chaque paire séquence-modèle sont enregistrés dans un fichier CSV. Ce fichier permet de consulter et d'analyser les résultats de manière structurée.

3. RÉSULTATS

3.1. Le temps d'exécution en fonction des types de parallélisation

L'algorithme implémenté, dont la complexité est de $\theta(n^2m^2)$, où n représente la taille de la séquence et m celle de la structure, exige un temps de calcul considérable. Compte tenu de cette complexité, des optimisations ont été nécessaires avant d'appliquer l'algorithme à des protéines de grande taille. Plusieurs stratégies de parallélisation ont été étudiées, en utilisant les bibliothèques Python *joblib* et *timeit*. Chaque test a été répété 100 fois sur 3 répétitions pour garantir la robustesse des résultats en calculant la moyenne et l'écart type, afin d'évaluer l'impact de ces optimisations sur le temps de calcul.

Nous avons testé trois approches de parallélisation : la parallélisation des alignements structurels, où les alignements sont effectués simultanément ; la parallélisation des calculs des matrices de bas niveau, où les matrices $n \times m$ sont calculées en parallèle ; et une combinaison des deux approches, parallélisant à la fois les alignements et les calculs de matrices.

Les trois approches de parallélisation ont été comparées à l'approche native sans parallélisation pour la protéine 5AWL (10 acides aminés) en la comparant à sa propre structure et à la structure 1LE0 (12 acides aminés). Les résultats sont résumés dans le tableau 1.

Le temps d'exécution sans parallélisation, fixé à 5,36 secondes, sert de base pour évaluer les gains de performance liés à la parallélisation. La parallélisation des templates uniquement réduit le temps à 3,81 s (réduction de 29 %). La parallélisation des matrices de bas niveau uniquement permet de réduire considérablement le temps à 1,85 s (réduction de 65 %).

Toutefois, la combinaison de la parallélisation des templates et des matrices du bas niveau entraîne une augmentation inattendue du temps d'exécution, atteignant 4,41 secondes. Ce résultat suggère des problèmes liés à la gestion des ressources due à la parallélisation simultanée de deux niveaux. De plus, l'écart-type plus important (± 0,35 s) indique une instabilité accrue, probablement causée par des conflits dans la gestion des threads ou une surcharge du système.

3.2. Les résultats pour les petites protéines

L'algorithme testé sur le jeu de données de petites protéines avec une pénalité de gap de 0,2 a montré qu'il parvient à retrouver sa propre structure et à lui attribuer le score énergétique minimal (tableau 2).

	1CRN	1L2Y	1VII	5AWL
1crn	**-283.99**	-14.77	-157.37	6.73
1l2y	-35.37	**-67.4**	-24.16	-19.18
1le0	-10.5	-19.18	7.61	-28.12
1le1	-13.05	-20.23	7.23	-28.32
1vii	-162.04	-41.71	**-169.07**	3.62
5awl	-9.02	-17.53	6.95	**-32.68**

TABLE 2. Résumé des résultats du premier exemple. Les structures sont en minuscules et les séquences en majuscules.

3.3. Les performances en fonction de la pénalité des gaps

En modifiant la valeur de la pénalité de gap, différents résultats ont été observés. Plusieurs valeurs ont été testées afin d'explorer ce paramètre et de déterminer une valeur optimale. Pour évaluer les performances du modèle, un score spécifique a été défini. Si l'algorithme parvient à prédire correctement la structure de la séquence, cela rapporte 2 points. Si, en plus, une structure similaire (hélice alpha ou feuillet bêta) figure parmi les deux premières prédictions, 1 point supplémentaire est attribué. Ainsi, pour chaque séquence, un score maximal de 3 peut être obtenu. Comme quatre séquences ont été comparées, le score maximal total est de 12. Le score final est divisé par cette valeur maximale afin de normaliser la performance. Les résultats sont présentés dans le tableau 3.

Gap	Performance	Correcte	Similaire
0.0	0.50	1	4
0.1	0.58	2	3
0.2	**0.92**	**4**	**3**
0.3	**0.92**	**4**	**3**
0.5	0.83	4	2
0.7	0.83	4	2
0.9	0.75	4	1
1.0	0.75	4	1
2.0	0.75	4	1
5.0	0.75	4	1
10.0	0.75	4	1

TABLE 3. Performances en fonction de la pénalité de gap

On constate qu'avec l'augmentation de la pénalité de gap, le nombre de correspondances correctes augmente, tandis que le nombre de structures similaires correctement prédites diminue. Cela peut s'expliquer par le fait qu'une pénalité de gap plus élevée favorise des alignements plus conservateurs, limitant ainsi les insertions ou délétions. Cela conduit à des correspondances plus exactes entre la séquence et les structures modèles, mais réduit la flexibilité de l'algorithme pour capturer des structures similaires, comme les hélices alpha ou les feuillets bêta.

La pénalité de gap de 0.2 ou 0.3 semble être la plus pertinente, car elle permet de prédire correctement toutes les structures et presque toutes les structures similaires. Les légères baisses de performance dans ces cas peuvent s'expliquer par la présence de deux structures très similaires dans le jeu de données ainsi que par des boucles sans structure secondaire définie. Ainsi, la pénalité de 0.2 a été retenue pour les tests finaux sur ce jeu de données, car elle offre le meilleur compromis entre précision des prédictions et flexibilité pour les structures similaires.

3.4. L'évaluation statistique des résultats

Afin de déterminer si les résultats obtenus sont significativement différents de ceux attendus par hasard, une approche statistique basée sur le calcul des z-scores a été employée. Les scores d'énergie observés pour chaque alignement ont été comparés à une distribution générée par 100 réordonnancements aléatoires des séquences. Le z-score quantifie l'écart par rapport à la moyenne des scores aléatoires : un z-score inférieur à -1.96 indique, avec une confiance de 95%, une correspondance structurale significative, tandis qu'un z-score supérieur à 1.96 signale un alignement significativement moins bon que le hasard.Un z-score proche de zéro suggère l'absence d'écart significatif.

	1crn	1l2y	1le0	1le1	1vii	5awl
1CRN	**-2.28**	-0.25	-0.66	-1.32	0.83	-0.58
1L2Y	1.57	**-2.11**	2.58	2.48	1.59	3.06
1VII	0.02	0.12	-0.89	-1.72	-1.53	**-2.08**
5AWL	-1.03	-0.29	-0.99	-0.54	-0.82	**-2.16**

TABLE 4. z-scores pour les alignements entre différentes structures

Par manque de temps, une distribution générée à partir de 10 et 20 réordonnancements aléatoires des séquences a été utilisée pour les séquences 1VII (36 acides aminés) et 1CRN (46 acides aminés). Les z-scores obtenus pour ces deux protéines doivent donc être interprétés avec prudence, car le théorème central limite n'est pas entièrement applicable avec un nombre aussi limité de réordonnancements. Néanmoins, le test de Shapiro effectué sur ces distributions a montré que les scores suivent une distribution normale, ce qui justifie l'utilisation des z-scores malgré le nombre restreint de réordonnancements.

Il ressort que, pour presque toutes les protéines, seul le score de leur propre structure est significativement plus bas que la moyenne des scores aléatoires. Cela indique que l'algorithme est capable de reconnaître sa propre structure, ce qui est déjà un résultat positif. Toutefois, il n'a pas réussi à identifier de manière robuste des structures similaires parmi les réordonnancements.

Pour la structure 1VII, l'algorithme a effectivement reconnu sa propre structure, mais le z-score associé est supérieur à -1,96. En revanche, le z-score pour l'alignement de cette séquence avec la structure 5AWL est significatif. Ce résultat peut s'expliquer par le fait que 5AWL est une boucle, une caractéristique qui pourrait ne pas être bien traitée par l'algorithme. Ce phénomène pourrait également résulter du nombre trop limité de réordonnancements effectués.

3.5. Les résultats pour les grandes protéines

Pour les deux grandes protéines testées, l'algorithme a réussi à attribuer le score le plus bas à la structure correcte (voir tableau 5). Cependant, les deuxièmes scores les plus bas ne correspondent pas nécessairement à des structures similaires. En effet, pour la protéine 1E68, qui contient majoritairement des hélices alpha, le deuxième score le plus bas est associé à 3E8V, qui contient principalement des feuillets bêta. Inversement, pour 3E8V, le deuxième score est associé à 3ZOW, qui contient majoritairement des hélices alpha.

	1E68	**3E8V**
1e68	**-490.53**	-426.15
1ubq	-466.0	-479.47
3zow	-472.0	-527.57
3e8v	-480.22	**-558.23**
1tit	-412.13	-493.82
1tdv	-316.45	-415.08
3zbv	-312.76	-407.47

TABLE 5. Résumé des résultats du deuxième exemple. Les structures sont en minuscules et les séquences en majuscules.

4. DISCUSSION

La réimplémentation de l'algorithme nécessite généralement une validation par comparaison avec un outil de référence. Cependant, l'adresse de l'outil publié dans l'article de référence (JONES, 1996) n'est plus valide, et nos recherches sur le site du laboratoire n'ont pas permis de retrouver l'outil THREADER. Par conséquent, nous avons opté pour une validation qualitative et statistique des résultats, ainsi qu'une analyse des performances en termes de temps d'exécution. Les conclusions suivantes sont donc basées non pas sur une comparaison directe avec un outil de référence, mais sur la cohérence biologique des résultats obtenus et les applications testées.

L'algorithme réimplémenté parvient de manière robuste à reconnaître la structure correcte de la séquence, ce qui est un signe positif indiquant une bonne qualité de l'implémentation. Toutefois, l'intérêt principal de cet outil réside dans sa capacité à identifier des structures similaires à la structure donnée. On observe que, soit les exemples utilisés ne contiennent pas de protéines présentant un niveau de similarité suffisamment élevé, soit les simplifications apportées ne permettent pas de détecter ces structures de manière plus sensible.

Pour surmonter ces limitations, il serait envisageable d'implémenter une pénalité de gap dynamique, différenciant l'ouverture et l'extension des gaps. Il faudrait aussi exclure les résidus trop proches, dont le score ne devrait pas être calculé, car leurs potentiels énergétiques sont influencés par les liaisons peptidiques, conduisant à des résultats imprécis. De plus, ajuster la pénalité de gap en fonction de la structure secondaire pourrait être bénéfique. Étant donné la conservation élevée des structures secondaires, il serait pertinent d'appliquer une pénalité plus élevée dans les régions secondaires et une pénalité moins élevée dans les régions flexibles comme les boucles.

Pour aller plus loin, un alignement local au sein des matrices de bas niveau pourrait être envisagé pour se concentrer sur les fragments structurels, plutôt que sur la structure entière. De plus, l'exclusion des régions flexibles lors du prétraitement des structures complexes pourrait être bénéfique. L'optimisation des temps de calcul via le multithreading et le multiprocessing pourrait également être envisageable, bien que cela n'affecterait pas directement la qualité des résultats.

5. REMERCIEMENTS

Je tiens à exprimer mes remerciements à Monsieur Jean-Christophe Gelly pour son temps, sa disponibilité constante, ainsi que pour ses conseils et ses explications détaillées, qui ont été d'une grande aide tout au long de ce projet.

Références

CHAN, H. ; DILL, K. The principles of protein folding. *Proc. Natl. Acad. Sci. USA*, v. 87, p. 6388–6392, 1990.

DOOLITTLE, R. Protein evolution and phylogenetic trees. *Prot. Sci.*, v. 1, p. 191–200, 1992.

GREGORET, L. ; COHEN, F. Protein folding mechanisms and models. *J. Mol. Biol.*, v. 211, p. 959–974, 1990.

JONES, D. Threader : protein sequence threading by double dynamic programming. In : BERNARDI, G. (Ed.). *Computational Methods in Molecular Biology*. [S.l.] : Elsevier, 1996. v. 32, cap. 13, p. 312–338.

LEVINTHAL, C. Chimie et physique des protéines. *Chimie et Physique*, v. 65, p. 44–45, 1968.

NEEDLEMAN, S. B. ; WUNSCH, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, Elsevier, v. 48, n. 3, p. 443–453, 1970.

ŠALI, A. ; BLUNDELL, T. L. Comparative protein modelling by satisfaction of spatial restraints. *Journal of molecular biology*, Elsevier, v. 234, n. 3, p. 779–815, 1993.

SMITH, T. F. ; WATERMAN, M. S. et al. Identification of common molecular subsequences. *Journal of molecular biology*, Elsevier Science, v. 147, n. 1, p. 195–197, 1981.

ZHANG, C.-T. ; CHOU, K.-C. Monte carlo simulation studies on the prediction of protein folding types from amino acid composition. *Biophysical journal*, Elsevier, v. 63, n. 6, p. 1523–1529, 1992.