Calcul parallèle par tâches de graphes de Reeb augmentés avec les tas de Fibonacci

Task-based Augmented Reeb Graphs with Fibonacci Heaps

Charles Gueunet, Pierre Fortin, Julien Jomier, Julien Tierny

English Abstract—This talk presents the first algorithm for the parallel computation of the augmented Reeb graph on triangulations of arbitrary dimention. Such an augmentation of the Reeb graph is required to enable the full extent of its applications, such as data segmentation or level set seeding. Our approach completely revisits the sequential algorithm with the best (optimal) complexity and expresses every part using tasks. The resulting approach is conducive to parallelism and benefits from the dynamic load balancing induced by the tasks. We present performance results on triangulated surfaces and tetrahedral meshes and show that our implementation results in superior time performance in practice compared to related work, both in sequential and in parallel. All this work is available in the open source library TTK.

1 Introduction

L'analyse topologique est un domaine fournissant des outils très utilisés en visualisation scientifique tels que le graphe de Reeb [14], le complexe de Morse-Smalle [5] ou encore l'arbre de contour [1], [4]. Ces structures jouent un rôle fondamentale dans l'analyse topologique de données, permettant par exemple l'extraction et la simplification rapide d'ensembles de niveaux [2], [15], la compression de données [12] ou encore le suivi de zones d'intérêt [11].

Ce travail s'intéresse plus particulièrement au cas du graphe de Reeb. Les algorithmes de calcul de cette structure [7], [8], [14] reposent généralement sur une vue globale des données et un traitement séquentiel qui les empêchent d'exploiter pleinement la puissance des architectures parallèles modernes. Nous présentons ici un nouvel algorithme exprimable en tâches, pour le calcul de cette abstraction sur maillages triangulaires et tétraédriques. L'implémentation correspondante utilise VTK [9] et OpenMP [6] et est disponible au sein de la plateforme *open-source: the Topology ToolKit [13].*

2 Prérequis

2.1 Les données

L'entrée de notre algorithme est un champ scalaire linéaire par morceaux: $f : \mathcal{M} \rightarrow \mathbb{R}$ défini sur une variété (également linéaire par morceaux). Nous considèrerons ici sans perte de généralité que nous sommes en dimension 3 (maillage tétraédrique).

Le champ scalaire f peut être étendu à l'ensemble de la géométrie via une interpolation linéaire barycentrique. On définit alors un ensemble de niveau comme étant la pré-image d'une valeur scalaire $i \in \mathbb{R}$ dans $\mathcal{M}$ par f : $f^{-1}(i) = \{p \in \mathcal{M} \mid f(p) = i\}$. Un ensemble de niveau peut être constitué de plusieurs composantes connexes qui sont appelées contours*. La figure 1 (b) nous montre deux ensembles de niveaux constitués de deux contours chacun.*

2.2 Le graphe de Reeb

Prenons $f^{-1}(f(p))_p$ le contour contenant p. Le graphe de Reeb *tel qu'illustré Figure 1 (b) est un complexe simplicial de dimension 1 défini comme l'espace quotient $\mathcal{R}(f) = \mathcal{M}/\sim$ par la relation d'équivalence $p_1 \sim p_2$:*

$$\begin{cases} f(p_1) = f(p_2) \\ p_2 \in f^{-1}(f(p_1))_{p_1} \end{cases}$$

Si chaque arc du graphe contient la liste des sommets de $\mathcal{M}$ se projetant sur lui par la relation d'équivalence $\sim$, on peut obtenir une segmentation de la géométrie en zones où le nombre de contour est de 1. Le graphe de Reeb est alors dit augmenté *(voir les zones de couleurs sur la Figure 1 (b)).*

3 Algorithme

Notre approche se base sur l'algorithme possédant la meilleur complexité en temps pour le calcul du graphe de Reeb augmenté, établi par S. Parsa [7] en 2012. Cet algorithme construit le graphe de Reeb durant une traversée globale des données par valeur scalaire. Une structure de graphe dynamique [10] est utilisée pour traquer l'évolution des composantes connexes d'ensembles de niveaux et ainsi construire le graphe de sortie à la volée. Notre travail a été de revisiter cette approche intrinsèquement séquentielle en utilisant des propagations indépendantes, basées sur

- *Gueunet Charles: Kitware*
  E-mail: charles.gueunet@kitware.com.
- *Fortin Pierre: Sorbonne Université, LIP6*
  E-mail: pierre.fortin@lip6.fr.
- *Julien Jomier: Kitware*
  E-mail: julien.jomier@kitware.com.
- *Tierny Julien: Sorbonne Université, LIP6*
  E-mail: julien.tierny@lip6.fr.

les tas de Fibonacci [3]. Ces propagations sont initiées depuis les minima (correspondant à des feuilles du graphe, comme illustré Figure 2 (a-b)) et visitent les sous-ensembles de niveaux du jeu de données en se rencontrant sur des points critiques de jointure, auxquels elles fusionnent puis continuent récursivement (cf. Figure 2 (c-d)). L'utilisation du graphe dynamique doit également être adapté à ce mode de propagation.

Afin de tirer partir des architectures multi-cœurs, notre approche peut être exprimée en utilisant le parallélisme de tâche. Chaque propagation est alors gérée par une tâche, et peut être exécutée indépendamment des autres propagations (jusqu'à ce qu'elle rencontre un point critique de jointure). Dans le but d'augmenter le nombre de tâches disponibles, nous présentons également une double traversée *illustrée Figure 3, dans laquelle les tâches sont démarrées simultanément des minima et des maxima, et s'arrêtent quand elles se rencontrent (au milieu du jeu de données). Le nombre de tâches ainsi créées est donc plus grand, correspond au nombre de feuilles du graphe au lieu du nombre de minima.*

4 Résultats

Les résultats présentés ici ont été obtenus en utilisant l'implémentation disponible dans TTK [13], sur une machine possédant deux processeurs Intel Xeon CPU E5–2630 v3 (2.4 GHz, 8 cœurs physiques chacun) et 64 GB de RAM.

Afin d'illustrer les performances de notre méthode, nous l'avons comparé à l'implémentation de l'algorithme de référence [7] généreusement fournie par l'auteur, sur des jeux de données en 2 et 3 dimensions. Notre approche est en moyenne 1,56 fois plus rapide lors des exécutions séquentielles et 11,14 fois plus rapide en utilisant les 16 cœurs. De plus, sur ces jeux de données nous observons des passages à l'échelle variant entre 2,57 et 14,29 sur notre machine de travail.

5 Limitations

Le nombre de tâches crées pour la construction du graphe de Reeb avec cette approche est initialement égale au nombre de feuilles dans le graphe. Le nombre de tâches actives diminue au fur et à mesure du calcul, quand les propagations fusionnent aux points critiques. Une partie du calcul est alors exécutée avec un nombre de tâches actives inférieur au nombre de cœurs, réduisant l'efficacité parallèle de l'approche. En pratique, cette zone parallèle peut être plus ou moins importante selon le jeu de données, cependant elle est inhérente à notre approche et des modifications profondes de l'algorithme seraient requises pour éviter cet effet.

6 Conclusion

L'approche présentée dans ce papier est la première pour le calcul parallèle du graphe de Reeb augmenté sur maillages triangulaires et tétraédriques. Cette approche se base sur l'algorithme séquentiel ayant la meilleur complexité pour le calcul de cette abstraction et améliore significativement les temps d'exécutions, à la fois en séquentiel et en parallèle.

References

[1] *H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In* Symposium on Discrete Algorithms*, 2000.*

[2] *H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In* Proc. of IEEE VIS*, pages 497–504, 2004.*

[3] *M. Fredman and R. Tarjan. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms.* Journal of the ACM*, 1987.*

[4] *C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based augmented contour trees with fibonacci heaps.* Transactions on Parallel and Distributed Systems*, abs/1902.04805, 2019.*

[5] *A. Gyulassy, P.-T. Bremer, B. Hamann, and P. Pascucci. A practical approach to Morse-Smale complex computation: scalabity and generality.* IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, pages 1619–1626, 2008.*

[6] *OpenMP Architecture Review Board. OpenMP Application Program Interface, V 4.5, 2015.*

[7] *S. Parsa. A deterministic $o(m \log m)$ Time Algorithm for the Reeb graph.* Discrete Comput. Geom.*, 49(4):864–878, June 2013.*

[8] *V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. In* Acm transactions on graphics (tog)*, volume 26, page 58. ACM, 2007.*

[9] *W. J. Schroeder, K. Martin, L. S. Avila, and C. C. Law.* The VTK user's guide*. Kitware, 2001.*

[10] *D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees.* Journal of Computer and System Sciences*, 26(3):362 – 391, 1983.*

[11] *B. S. Sohn and C. L. Bajaj. Time varying contour topology.* IEEE Transactions on Visualization and Computer Graphics*, 2006.*

[12] *M. Soler, M. Plainchault, B. Conche, and J. Tierny. Topologically controlled lossy compression. In* Proc. of PacificVis*, 2018.*

[13] *J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. Technical report, UPMC, https://topology-tool-kit.github.io/stuff/ttk.pdf.*

[14] *J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees.* IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 15:1177–1184, 2009.*

[15] *M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pasucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In* Proc. of ACM Symposium on Computational Geometry*, 1997.*

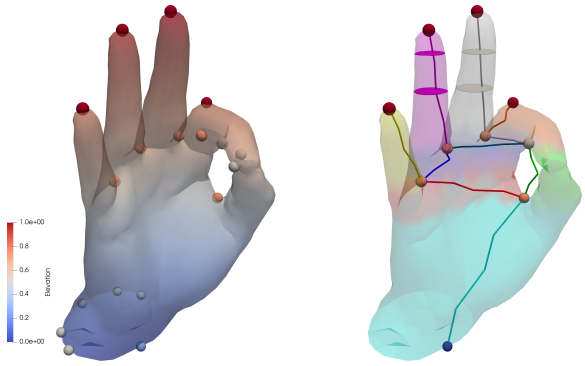

Fig. 1. Jeu de données d'une main, le champ scalaire étant l'élévation (du bleu vers le rouge). (a) Visualisation du champ scalaire et des points critiques. (b) Le graphe de Reeb correspondant avec la segmentation induite ainsi que deux ensembles de niveaux illustrant le mécanisme de contraction.

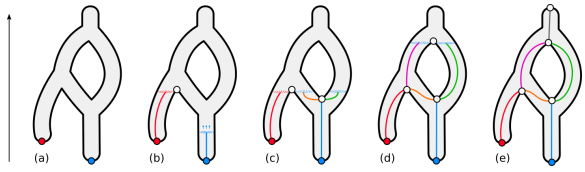

Fig. 2. Aperçus du calcul de graphe de Reeb augmenté avec notre approche, sur un jeu de données simple dont le champ scalaire est l'élévation. (a) Les minima (correspondant à des feuilles du graphe) sont extraits. (b) Les arcs de chaque minimum sont construits par des propagations indépendantes. (c) La propagation bleue rencontre un point critique de séparation et continue donc la traversée en gérant deux arcs. Elle s'arrête quand elle rencontre un point selle de jointure. (d) Le voisinage inférieur du point selle de jointure ayant été entièrement visité, les propagations correspondantes peuvent fusionner et la propagation continue. (e) Le maximum global est atteint, le graphe de Reeb est complet.

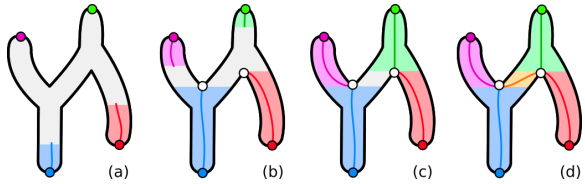

Fig. 3. Aperçus du calcul du graphe de Reeb augmenté en utilisant notre *double traversée* sur un jeu de données simple dont le champ scalaire est l'élévation. (a) Tous les extrema sont extraits afin d'initier les propagations simultanément des minima et des maxima. (b) Les propagations issues des maxima s'exécutent en parallèle grâce au parallélisme de tâche. (c) Les propagations arrêtent de traiter un arc quand celui-ci rencontre une zone déjà visitée par un arc issu d'une propagation opposée. (d) Le graphe de Reeb augmenté final.