Load-Balanced Bottleneck Objectives in Process Mapping*

Johannes Langguth† Sebastian Schlag‡ Christian Schulz§

Abstract

We propose a new problem formulation for graph partitioning that is tailored to the needs of time-critical simulations on modern heterogeneous supercomputers.

1 Introduction

Among the many combinatorial problems, graph partitioning (GP) has a central role in the area of parallel high performance computing. Irregular inputs such as graphs, sparse matrices and many meshes typically have to be distributed among the nodes of a supercomputer to allow parallel processing, and the distribution should both balance the workload and minimize the communication. In the last two decades, sophisticated software has been developed for computing high-quality partitionings fast. However, the underlying model, which assumes homogeneous nodes connected by a full-bisection bandwidth network, no longer represents current supercomputers. We therefore propose a new problem formulation which overcomes these drawbacks and discuss its suitability for future challenges in parallel computing.

Irregular inputs can roughly be divided into two main groups. The first consists of problems whose data access pattern resembles sparse-matrix-dense-vector multiplications (SpMV), where the communication is irregular but follows a repeated pattern, such as PageRank [21] and many scientific computations on irregular meshes. The second group represents problems resembling sparse-matrix-sparse-vector multiplications (SpMSpV). Here, only a subset of vertices is active at a time, leading to changing communication patterns. While minimizing cut sizes helps to reduce the communication cost in both cases, in the first case, the overall communication time is determined by the maximum communication per node, assuming the network links of a distributed memory computer operate independently. Thus, minimizing the maximum (i.e., the makespan) is a better partitioning objective, and assuming there are no slowdowns such as concurrent traffic in the network, it is actually optimal in this case. In the second case, it is harder to compare the partitioning objectives. In low-diameter graphs such as social networks, breadth-first search and similar algorithms typically perform a low number of high-volume communication rounds. Consequently, minimizing the maximum communication among the nodes may lead to better results than cut-size optimization. For high diameter graphs this no longer holds, which renders makespan partitioning less attractive in this case.

Finally, current single-criterion partitioning does not allow trade-offs between load balance and cut size. For SPMV type computations, where both communication and computation performance is typically to a certain degree predictable, incorporating this trade-off has the potential to provide better solutions than using a fixed load-balance constraint on the vertices. And while SpMSpV computations are typically less predictable, they are often highly communication-bound, which means that load-balance is less important than communication minimization and might benefit from the trade-off formulation. All this however implies that we must provide a ratio between the cost of communication and computation for the trade-off to function.

2 Related Work

There has been an enormous amount of research on GP, and we refer the reader to Refs. [2, 3, 26] for extensive material and references. The most common objective function is to minimize the total cut size $\sum_{i<j} \omega(E_{ij})$. There are well known software packages based to find partitions minimizing this objective including KaHIP [24], Jostle [29], Metis [15], and Scotch [22]. A related objective is to minimize the *maximum cut size* $\max_{i<j} \omega(E_{ij})$. The first accurate communication volume metrics for sparse-matrix vector multiplication, using hypergraph models, are due to Catalyurek and Aykanat [5]. When *graph* partitioning is used in parallel computing to map the graph nodes to different processors, the *communication volume* is often more representative than the cut [13]. Let V_ℓ be the block containing vertex v. Then, we let $D(v)$ denote the number of blocks in which vertex v has a neigh-

*This work was partially supported by the Austrian Science Foundation (FWF, project P 31763-N31), and it was partially supported by the Research Council of Norway (project RCN 251186).

†Simula Research Laboratory, Oslo, Norway
‡Karlsruhe Institute of Technology, Karlsruhe, Germany
§University of Vienna, Faculty of Computer Science, Vienna, Austria

bor, excluding the block containing v and $c(v)$ be a vertex weight. That is, $D(v) = |\{j \mid \exists u \in V_j \neq V_\ell \text{ s.t. } \{u, v\} \in E\}|$. For a block V_i, the *communication volume* is $cvol(V_i) := \sum_{v \in V_i} c(v)D(v)$. Similar to cut size, one can minimize the *maximum communication volume* $\max_{1 \leq i \leq k} cvol(V_i)$, or the *total communication volume* $\sum_{1 \leq i \leq k} cvol(V_i)$. The choice of either the total or maximum variant of the objective function here is typically dictated by the topology of the interconnection network between the machines [13]. Other objectives are less common, these include the maximum degree in the *quotient graph* (the graph formed by blocks and their connections), expansion and conductance [16], and optimizing the "shape" of partitions [20]. Specialty partitioners tailor-made for applications can give better quality [7]. Furthermore, there are methods for maximizing multiple objective functions simultaneously [25, 28, 6], and finding Pareto-optimal solutions [11].

There is likewise a large literature on process mapping, i.e., decomposition techniques that takes the given connection network into account, often in the context of scientific applications using MPI (Message-Passing Interface). Refs. [30, 23, 4, 12] were among the first to tackle the process mapping problem. Hatazaki [12] and Walshaw [30] proposed graph partitioning to solve the MPI process mapping problem for unweighted process topologies. Mercier and Clet-Ortega and later Jeannot [18, 19] simplify the mapping problem by only considering the topology inside the compute nodes themselves and ignoring the topology of the network. Multiple placement policies are investigated to enhance overall system performance. Yu et al. [31] discuss and implement embedding heuristics for the BlueGene $3d$ torus system. Hoefler and Snir [14] instead optimize the congestion of the mapping and show that this problem is NP-complete. Routing-aware mapping heuristics taking the hierarchy of specific hardware topologies into account were discussed in Ref. [1]. Later, based on a variety of techniques, more sophisticated algorithms to compute mappings for hierarchical systems have been published [27, 10, 8].

The *Lynx* code [17] is an SpMV-type computation for large scale simulations in cardiac electrophysiology on heterogeneous supercomputers. As the problem is relatively communication heavy, high-quality partitioning is crucial for its performance, especially when using nodes equipped with multiple GPUs. Scalability was achieved via hierarchical partitioning. However, due to the lack of actual hierarchical partitioning software, it was emulated by applying conventional partitioning twice. This proved to be highly effective, but difficult to program. As this type of compute node has become common in modern supercomputers, we believe that the need for hierarchical partitioning software will increase in the near future.

3 Problem Specification

We define the basic version of the *graph-constrained makespan partitioning problem* as follows:

Given a tree $C = (B, L)$, an undirected graph $G = (V, E)$, and a communication cost factor F. Find a partitioning $P : V \to B$ which maps each vertex $v \in V$ to a vertex $b \in B$ such that the *makespan* is minimized. We refer to the vertices B of C as *bins*, and the edges L as links. The *makespan* M is defined as:

$$M(P) = \max(\max_{b \in B} comp(b), F \cdot \max_{l \in L} comm(l))$$

where $comp(b) = |\{v \in V \mid P(v) = b\}|$ is the computational load of vertex $b \in B$, $comm(l) = |\{\{u, v\} \in E \mid l \in STP(P(u), P(v))\}|$ is the communication volume along edge $l \in L$, and $STP(P(u), P(v))$ is the shortest path (as a set of edges) between the bins $P(u)$ and $P(v)$ that the endpoints u and v of some edge were assigned to in P. Because C is a tree, the shortest path is unique. Note that C is unweighted, which means that the length of a path is simply the number of edges contained in it. This is based on the intuition that if an edge is cut, communication will be necessary between the compute nodes represented by these bins.

3.1 Generalizations. The most basic generalization is the introduction of a set $R \subset B$ of routers, i.e., bins $r \in R$ that cannot be assigned any work (i.e., $load(r) = 0 \ \forall r \in R$). This is needed to correctly model the networks of most supercomputers. We refer to this variant as the *interconnect graph-constrained makespan partitioning problem*.

To take more complicated interconnects with different routing protocols into account, the protocols become part of the problem formulation. Thus, the *routing graph-constrained partitioning problem* differs from the standard version in that C is no longer required to be a tree. Any algorithm for this problem is given access to an *oracle* (such as a routing table) which for every pair of bins returns a unique path between them. If we allow for multipath routing, the oracle may instead return a set of paths. In that case, assuming k paths, each path $\mathcal{P}$ adds $1/k$ to the load of each edge $l \in \mathcal{P}$.

The hierarchical network structure of interconnect topologies such as fat trees implies that some links need to have much higher capacities than the injection bandwidth of a single node. In order to model this, we change the communication factor F from globally applying to all links to a link-specific factor F_l, thereby

obtaining the *edge-weighted* variant. By the same token, we can define a *vertex-weighted* variant where every $v \in V$ is given a weight $w(v)$. The load of a bin is then computed as the sum of the weights of all vertices assigned to that bin.

3.2 NP Hardness. Unlike the traditional graph partitioning problem, graph constrained makespan partitioning does not have a simple reduction from MINIMUM BISECTION. However, the vertex weighted case can easily be proven to be NP-Hard by reduction from MINIMUM MULTIPROCESSOR SCHEDULING [9].

References

[1] A. H. Abdel-Gawad, M. Thottethodi, and A. Bhatele. RAHTM: routing algorithm aware hierarchical task mapping. In *Intl. Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 325–335, 2014.

[2] C. Bichot and P. Siarry, editors. *Graph Partitioning.* Wiley, 2011.

[3] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz. Recent Advances in Graph Partitioning. In *Algorithm Engineering – Selected Topics, ArXiv:1311.3144*, 2014.

[4] Ü. V Catalyürek and C. Aykanat. Hypergraph model for mapping repeated sparse matrix-vector product computations onto multicomputers. In *Proceedings of International Conference on High Performance Computing (HiPC)*, 1995.

[5] Ü. V. Çatalyürek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10(7):673–693, 1999.

[6] Ü. V. Çatalyürek, M. Deveci, K. Kaya, and B. Uçar. UMPa: A Multi-objective, Multi-level Partitioner for Communication Minimization. In *10th DIMACS Impl. Challenge Workshop: Graph Partitioning and Graph Clustering*. Georgia Institute of Technology, Atlanta, GA, February 13-14 2012.

[7] D. Delling, A. V. Goldberg, I. Razenshteyn, and R. F. Werneck. Graph partitioning with natural cuts. In *2011 IEEE International Parallel Distributed Processing Symposium*, pages 1135–1146, May 2011.

[8] M. Fonseca Faraj, A. van der Grinten, H. Meyerhenke, J. L. Träff, and C. Schulz. High-Quality Hierarchical Process Mapping. *CoRR*, abs/2001.07134, 2020.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition, 1979.

[10] R. Glantz, M. Predari, and H. Meyerhenke. Topology-induced enhancement of mappings. In *Proceedings of the 47th International Conference on Parallel Processing, ICPP 2018*, pages 9:1–9:10. ACM, 2018.

[11] M. Hamann and B. Strasser. Graph Bisection with Pareto-Optimization. In *Proc. of the 18th Algorithm Engineering and Experiments*, pages 90–102. SIAM, 2016.

[12] T. Hatazaki. Rank reordering strategy for MPI topology creation functions. In *5th European PVM/MPI User's Group Meeting*, volume 1497, pages 188–195, 1998.

[13] B. Hendrickson and T. G. Kolda. Graph Partitioning Models for Parallel Computing. *Parallel Computing*, 26(12):1519–1534, 2000.

[14] T. Hoefler and M. Snir. Generic topology mapping strategies for large-scale parallel architectures. In *Proc. 25th Intl. Conf. on Supercomputing (ICS)*, pages 75–84, 2011.

[15] G. Karypis and V. Kumar. Multilevel k-way Partitioning Scheme for Irregular Graphs. *Journal on Parallel and Distributed Compututing*, 48(1):96–129, 1998.

[16] K. Lang and S. Rao. A Flow-Based Method for Improving the Expansion or Conductance of Graph Cuts. In *Proceedings of 10th International Integer Programming and Combinatorial Optimization Conference*, volume 3064 of *LNCS*, pages 383–400. Springer, 2004.

[17] J. Langguth, M. Sourouri, G. T. Lines, S. B. Baden, and X. Cai. Scalable heterogeneous cpu-gpu computations for unstructured tetrahedral meshes. *IEEE Micro*, 35(4):6–15, July 2015.

[18] G. Mercier and J. Clet-Ortega. Towards an efficient process placement policy for MPI applications in multicore environments. In *European Parallel Virtual Machine/Message Passing Interface Users Group Meeting*, pages 104–115. Springer, 2009.

[19] G. Mercier and E. Jeannot. Improving MPI applications performance on multicore clusters with rank reordering. In *18th Eur. MPI Users' Group Meeting*, pages 39–49, 2011.

[20] H. Meyerhenke and T. Sauerwald. Beyond good partition shapes: An analysis of diffusive graph partitioning. *Algorithmica*, 64(3):329–361, Nov 2012.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[22] F. Pellegrini. Scotch Home Page. http://www.labri.fr/perso/pelegrin/scotch/.

[23] P. Sadayappan, F. Erçal, and J. Ramanujam. Cluster partitioning approaches to mapping parallel programs onto a hypercube. *Parallel Computing*, 13(1):1–16, 1990.

[24] P. Sanders and C. Schulz. KaHIP – Karlsruhe High Qualtity Partitioning Homepage. http://algo2.iti.kit.edu/documents/kahip/index.html.

[25] K. Schloegel, G. Karypis, and V. Kumar. A new algorithm for multi-objective graph partitioning. In *Proceedings of the 5th International Euro-Par Conference on Parallel Processing (Euro-Par 1999)*, volume 1685 of *LNCS*, pages 322–331. Springer, 1999.

[26] C. Schulz and D. Strash. Graph partitioning: Formulations and applications to big data. In *Encyclopedia of Big Data Technologies*. Springer, 2019.

[27] C. Schulz and J. L. Träff. Better process mapping and sparse quadratic assignment. In *16th International Symposium on Experimental Algorithms*, volume 75 of *LIPIcs*, pages 4:1–4:15, 2017.

[28] N. Selvakkumaran and G. Karypis. Multi-objective hypergraph partitioning algorithms for cut and maximum subdomain degree minimization. In *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*, pages 726–733, Nov 2003.

[29] C. Walshaw and M. Cross. JOSTLE: Parallel Multilevel Graph-Partitioning Software – An Overview. In *Mesh Partitioning Techniques and Domain Decomposition Techniques*, pages 27–58. 2007.

[30] C. Walshaw, M. Cross, M. G. Everett, S. P. Johnson, and K. McManus. Partitioning & Mapping of Unstructured Meshes to Parallel Machine Topologies. In Afonso Ferreira and José D. P. Rolim, editors, *Proceedings of Parallel Algorithms for Irregularly Structured Problems, Second International Workshop, IRREGULAR '95*, volume 980 of *LNCS*, pages 121–126. Springer, 1995.

[31] H. Yu, I-H. Chung, and J. E. Moreira. Topology mapping for Blue Gene/L supercomputer. In *ACM/IEEE Supercomputing*, page 116, 2006.