Graph partitioning for High Performance Scientific Simulations

Kirk Schloegel, George Karypis and Vipin Kumar

Andi Škrgat, 12.12.2021

1 Summary

Scientific simulations are frequently done on graphs and to parallelize such a computation, some kind of partitioning algorithm is needed. More precisely these algorithms operate on meshes so one can use them to perform computation on mesh nodes or on mesh elements. In latter case creating graph is trivial and we can construct it by having vertex for each mesh node and an edge for each edge between two nodes. In former case one can create graph by taking dual of the node graph.
Goal of partitioning is to split the vertices of graph G(V,E) into k disjoint subdomains such that every computer node has its own subdomain with approximately equal amount of work to do(in order to ensure load balance) while minimizing necessary communication between neighboring subdomains. **Min-cut** is reasonable metric that one can use for minimizing the **inter-processor communication** because of an assumption that vertices have approximately same number of edges. More precise measures would include number of processors to which one processor needs to sent data and amount of data that needs to be sent. Because of that, one can conclude that min-cut is just an approximation for **total communication volume**. For message passing architectures **minimizing number of startups** between processors is probably better option than minimizing total communication volume.
Due to the fact that this problem is NP complete, there are a lot of algorithms that use different assumptions in order to obtain best possible solutions. **Geometric techniques(mesh partitioning)** use coordinate information of the mesh nodes and try to minimize metric related to edge-cut e.g. number of mesh elements that are adjacent to non-local elements. However, information about local connectivity is not used which is why solutions can be typically produced very fast but are of low quality. **Coordinate nested dissection** tries to recursively minimize the boundary between the subdomains by splitting the mesh in half normal to its longest dimension. This kind of scheme is computationally efficient, it is easily parallelizable but generally produces low quality solutions that very often contain disconnected subdomains. **Recursive inertial bisection** is improvement of CND scheme that also takes angles into account when orienting bisection. Problem with last 2 approaches is that they are based only on single dimension because mesh elements' order is created only according to coordinate or inertial exes(angle). **Space-filling curve** techniques order the

mesh elements according to the positions of their centers-of-mass along curve that fills up higher dimensional spaces. By following this approach we usually get orderings of the mesh elements where closeness in space is copied to closeness in order which is desirable property. These methods are computationally efficient and work especially well when computational nodes depend on their spatial proximity. **Sphere-cutting approach** uses overlap graphs and because of their good theoretical foundations, guarantees quality of the partitioning made by using such a strategy.
Combinatorial techniques are opposite approach than geometric techniques. They try to put in the same subdomain vertices that are highly connected no matter whether they are near to each other in space. In general, these techniques produce same or solutions of better quality than geometric but are slower because of difficult parallelization. One possible way of implementing this is by using **levelized nested dissection** which guarantees at least one subdomain will be connected.
Instead of trying to partition the graph optimally from the beginning, another possible approach would be to create some computationally inexpensive solution which will then be improved by using one of the **refinement** algorithms. Examples of such algorithms is **Kernighan-Lin(KL)** and **Fiduccia-Mattheyses(FM)** which both work in same manner: given an initial bisection of a graph, try to find two subsets X and Y that are from subdomain A and subdomain B, such that placing X into B and Y into A will reduce edge-cut metric but no **load imbalance** will occur. Difference between them is that FM algorithm improves complexity by smartly using appropriate data structures. These algorithms are proven to work better when the average degree of graph is large and when load balance constraints are not strict. For example, this would mean that when partitioning for 2 processors, one shouldn't ask for exactly 50-50 perfectly balanced situation but rather allow 10-15% asymmetry. If load balance constraints are however strict, algorithms can get stuck in local minimum and then significantly depend on initial heuristic or random solution.
Spectral methods typically produce higher quality partitionings than geometric methods but are much more computationally expensive. They use approach in which they formulate graph partitioning problem as optimization of continuous quadratic function. **Multilevel paradigm** partitions graph in 3 steps. In first step graph coarsening is done in several iterations, where in each iteration new graph is constructed with smaller number of edges by merging vertices from its parents. After this, smallest graph is partitioned and multilevel refinement with **KL** or **FM** algorithm is applied on each of the coarse graphs. This kind of scheme works because by applying graph coarsening, new graph has smaller number of edges and partitioning is then easier to do. Secondly, refinement algorithms are in this case much more powerful because by moving one vertex in one coarse graph they actually move many vertices in original vector but in significantly reduced computational complexity. Such a paradigm usually creates much better solutions than spectral methods in terms of both speed and quality. Karypis and Kumar showed that by using heavy-edge matching, which tries to merge edges with larger weights, initial partitionings of graphs are of good quality and with that in mind one can use relaxed versions of KL/FM refinement algorithms that are faster and easier to parallelize. **Multilevel k-way partitioning** was presented again by Karypis and Kumar and it is a scheme that generalizes simplified versions of KL/FM bisection refinement algorithm.

They are extremely fast and very often they partition graph in less time than is needed to read it from memory.
To consider what kind of scheme works best, one could consider many characteristics, e.g quality of solutions, its run time, degree of parallelism, stability in producing quality solutions... Schemes that are faster like geometric, on average produce solutions of lower quality than computationally expensive methods like multilevel partitioning or spectral methods. Approach that is often used in this area is to combine advantages of different schemes to hide their disadvantages. Example of such a combination of techniques is algorithm in which initial partitioning is done by fast geometric method on which is then applied KL/FM refinement algorithm. Second possible way of such an utilization is to calculate coordinate information for vertices and then forward this information to geometric scheme for partitioning.