Layered Label Propagation: A Coordinate-free Node Ordering Method for Graph Compression

Gonçalo Gaspar (58803)   João Alves (79155)
Rodrigo Bernardo (78942)

Group 10

Abstract

We present a summary of how Layered Label Propagation (LLP), a highly scalable, coordinate-free, graph reordering algorithm, uses community finding techniques to permute very large immutable graphs, with applications to graph compression.

Introduction

Real-world networks are rich with information that can be gathered through graph mining techniques. Cases of study are friendship relations or community finding in social networks, cooperation networks in scientific co-authorships or protein-protein interactions.

Obtaining this kind of informations becomes a real implementation problem when the subjacent graph has millions or billions of nodes, since most of the standard graph mining algorithms assume that the graph is stored in main memory, which may not be the case. This can be witnessed, for example, in the web graph which describes the directed links between pages of the addressable World Wide Web, or when regarding the most used social networks, like Facebook or Twitter.

Therefore, techniques for efficient storage and fast access/traversal of large graphs are needed in order to become feasible to analyze not only web graphs, but large graphs of any kind.

While several approaches have been taken regarding graph compression, we chose to analyze how LLP uses community finding techniques to reorder the nodes of a graph in order to exploit its inner structure and obtain, in combination with the Boldi and Vigna (BV) compression method (which we do not explore here), superior compression products than current alternatives.

LLP was first mentioned in [BRSV10]. The objective was to find effective general techniques to store and access graphs. Moreover, the resulting compressed data structure must provide fast amortised random access to an edge. The authors address this problem by applying *intrinsic* heuristics (i.e., ones that depend only on the inner structure of the network), in contrast to *extrinsic* heuristics (which are features of each specific kind of network). For instance, in a web graph one can find an extrinsic permutation of the nodes in the URL-based lexicographic order which, in practice, can be shown to produce impressive compression ratios. However, this ordering isn't applicable to all sorts of networks.

Nonetheless, the ordering of the nodes becomes important once we consider that, in order to achieve good compression performances, compression algorithms usually exploit two properties: (1) *similarity*: nodes tend to have resembling sets of neighbours if they're close to each other in the ordering; (2) *locality*: most of the edges are shared between nodes close to each other in the ordering. Furthermore, most current algorithms are sensitive to the initial ordering of the graphs, generating different compression ratios depending on how the dataset is originally presented. LLP, on the other hand, is *coordinate-free*, i.e., attains similar results independently of the original ordering.

Problem Definition

Given an input graph, devise an ordering $\pi : V_G \to |V_G|$ that minimizes the number of bits per edge needed to store the graph, while providing fast amortised random access to its edges.

Obviously, the algorithm must be parameterised by a compression method. The authors of LLP chose to combine it with the BV compression algorithm, as they regard it as the "*de facto* standard for handling large web-like graphs". The procedure relies heavily on similarity and locality to manage its good results, which are exactly the properties what LLP tries to maximise.

It is also worth noting that this problem is NP-hard. Therefore, it is appropriate to try to craft heuristics that work well in most practical cases. In this case we are only interested in intrinsic heuristics.

Label Propagation

As said before, LLP exploits the inner structure of the network to devise intrinsic orderings of its nodes, so it may be appropriate to approach the issue of graph reordering as a community finding problem. However, the size of the graphs we are handling demand the usage of highly efficient algorithms.

Label propagation algorithms seem fit to address this problem because, not only no *a priori* information is needed regarding the structure of the network, they are also efficient, requiring only a few passes through the network and are linear in the number of edges.

What the authors of LLP call the (*standard*) label propagation algorithm is described in [RAK07] and works as follows: at the beginning each node is assigned a unique label. At each iteration, in random order each node takes up the label that most of its neighbours have, with ties resolved uniformly randomly. As the labels propagate through the graph, densely connected groups of nodes form an agreement over their labels. This iterative process goes on until every node in the graph is assigned a label equal to most of its neightbours. At the end, nodes with the same labels are grouped together as communities.

However, the authors found that the algorithm just described, tends to produce one giant cluster containing the bulk of the nodes when applied to social networks, which is due to the very nature of the topology of this kind of networks.

One interesting variant of label propagation is the Absolute Pott Model (APM) [RN10]. This algorithm addresses the resolution limit problem in community detection by introducing nonlocal weight discount parameter when considering weight preferences for assigning node labels. In standard label propagation, the label λ_i assigned to a node x, was the one that maximized k_i, being k_i, the number of neighbours having the given label λ_i. In APM, the value to maximize becomes $k_i - \gamma(v_i - k_i)$, where v_i is the number of nodes in the network with label λ_i. Note that, if γ is fixed to zero, we get the standard label propagation method.

However, the APM algorithms suffers from some shortcomings. First, there is no known way to predetermine an “optimal” value for γ: every value of this parameter accounts for a different resolution of the analyzed graph. Second, the sizes of the produced clusters lean towards a heavy-tailed decreasing distribution, yielding an enormous amount of huge-sized clusters.

Layered Label Propagation

For low values of the resolution parameter γ of the APM algorithm, outer and further nodes tend to have more weight at the time of updating the labels, highlighting fewer, larger and sparser clusters, while higher values of γ result in denser and smaller clusters, reveiling a finer structure of the network. This idea motivates the definition of *Layered Label Propagation*.

The algorithm starts with any initial ordering of the nodes. The APM algorithm is then iterated through different values of γ and at each iteration the produced labelling is converted into an ordering that keeps nodes with the same label close to each other; nodes within the same community are left in the same order they had before. Regarding nodes from different communities, their relative order is determined by the labels themselves.

LLP becomes parameter-free once the initial ordering and the sequence of resolution terms are defined. For that matter, the initial ordering is defined to be the original ordering of the input graph (any ordering will suffice because LLP is coordinate-free). Regarding the choice of resolution parameters, they are picked uniformly randomly from the set $\{0\} \cup \{2^{-i}, i = 0, ..., K\}$.

It's worth noting that the algorithm as described lends itself naturally to a parallel implementation. It is shown to be possible to obtain performance improvements linear in the number of cores.

Analysis

In [BRSV10] the authors presented two measures to analyze empiricaly why the Layered Label Propagation approach to compression works.

The authors argue that locality is the most important characteristic of an ordering in order to achieve good compression performances in the case of web graphs, i.e., we want an ordering that keeps nodes with the same host close to each other. With this in mind, we would like to know how much a given ordering π respects the partition induced by the hosts, $\mathcal{H}$. The first metric proposed is the probability to have a *host transition* (HT), which is simply the fraction of nodes that are followed, in the order, by another node with a different host:

$$HT(\mathcal{H}, \pi) = \frac{\sum_{i=1}^{|V_G|-1} \delta(\mathcal{H}[\pi^{-1}(i)], \mathcal{H}[\pi^{-1}(i-1)])}{|V_G| - 1}$$

where the δ in the formula is Kronecker's delta and $[x]$ is the set of nodes with the same host as x.

We argue that this formula is wrong and that the probability to have a host transition is actually its complement. The reason is simple: the sum in the numerator is only "incremented" when two contiguous nodes in the ordering have the same host, but the opposite was to be expected. We argue that the correct formula is then

$$HT(\mathcal{H}, \pi) = 1 - \frac{\sum_{i=1}^{|V_G|-1} \delta(\mathcal{H}[\pi^{-1}(i)], \mathcal{H}[\pi^{-1}(i-1)])}{|V_G| - 1}$$

We expect a good ordering to minimize this metric, in order to maximize locality.

Let $\mathcal{H}_{|\pi}$ be the partition of $\mathcal{H}$ whose sets are those of nodes contiguous in the order π and that the belong to the same host. If, for example, we have $\mathcal{H} = \{\{0\}, \{1, 2, 3\}, \{4, 6\}, \{5\}\}$ and $\pi =< 0, 1, 4, 3, 2, 5, 4, 6 >$, then the induced partition becomes $\mathcal{H}_{|\pi} = \{\{0\}, \{1\}, \{2, 3\}, \{4, 6\}, \{5\}\}$.

With $\mathcal{H}_{|\pi}$ defined we can now expose the second metric, Variation of Information (VI), which, in our case, can be show to be equal to:

$$VI(\mathcal{H}, \mathcal{H}_{|\pi}) = H(\mathcal{H}_{|\pi}) - H(\mathcal{H})$$

where H, called entropy, is defined as:

$$H(\mathcal{U}) = -\sum_{i=0}^{R} P(i)\, log(P(i)), \quad P(i) = \frac{|\mathcal{U}_i|}{|\mathcal{V}_i|}$$

since $\mathcal{H}_{|\pi}$ is always a refinement of $\mathcal{H}$.

Again, we want this metric to be as low as possible. VI is always greater or equal to zero so we always have $H(\mathcal{H}_\pi) \geqslant H(\mathcal{H})$. Intuitively, this means that $\mathcal{H}_\pi$ holds "needless information" in order to recover the community structure present in $\mathcal{H}$.

The authors of LLP go on to show experimentally that there is a correlation between the minimization of these metrics and better compression ratios.

References

[BRSV10] P. Boldi, M. Rosa, M. Santini and S. Vigna. Layered Label Propagation: A MultiResolution Coordinate-Free Ordering for Compressing Social Networks. arXiv:1011.5425v2.

[RAK07] U. Raghavan, R. Albert and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. arXiv:0709.2938v1.

[RN10] P. Ronhovde and Z. Nussinov. Local resolution-limit-free Potts model for community detection. arXiv:0803.2548v4.