Report
A Survey of Graph Processing Kernels

Version 0.1

September 10, 2018

Contents

Chapter 1

Introduction

A **computing kernel** is a common function that may see use as part of larger algorithms for solving problems of interest. This document is an attempt to survey in a relatively standardized way a variety of **graph kernels** that

The rest of this chapter is organized as follows:

- Section **??** provides some basic definitions.
- Section 1.2 provides a change log for this report.

This report was developed as part of an NSF grant CCF-1642280, with drafts of many of the chapters written as part of CSE 60742, a Fall 2018 class in Scalable Graph Processing, at the University of Notre Dame. The names of the contributing students is included at the beginning of each chapter.

1.1 Basic Definitions

Many of the programming systems use different terms to mean the same thing. This report will standardize the notation used in the individual descriptions, with cross-references in each section to relate the terms used by the original sources.

Each of the following sections provides some of these standard definitions.

1.1.1 Relations

- An **object** is some data structure that may be distinguished from another object. All of the following definitions may be used to define objects.
- A **set** is an unordered collection of objects, each of which is unique, with no duplicates. A set with a finite number of elements is written as the list of elements surrounded by "{ }".
- A **bag** is an unordered collection of objects where there may be duplicates.
- A **k-tuple** (or **tuple** for short) is an ordered collection of k objects where the position of objects in the tuple matters, and where the same object may be used more than once (in different positions). Thus two tuples with the same objects but in different orders are different tuples. A k-tuple is written as a list of its components, in order, surrounded by "( )".
- A **pair** is another term for a 2-tuple.

- The **Cartesian product** of k sets $S_1, ..S_k$ is the set of tuples $\{(u_1, ...u_k)\}$ where u_i is an object from S_i. It is typically written as $S_1xS_2x...xS_k$.
- A **k-relation** (or **relation** for short) R on sets S_1, ... S_k is a set of k-tuples where the i'th element of each k-tuple comes from set S_i. Some or all of the S_is may be the same set.

  This is equivalent to saying that R is a subset of the Cartesian product $S_1xS_2x...xS_k$.

  In conventional terminology the term R$(u_1, u_2, ...u_k)$ means that the tuple $(u_1, u_2, ...u_k)$ is in the relation R.
- A **binary relation** R is a set of pairs { (u_i, v_i) } where all the u's come from a set U and all the v's come from a set V. U and V may be the same set, and if so, there is no requirement that if (u, v) is in R then so is (v, u).

  In conventional terminology the term R(u,v) means that the pair (u, v) is in the binary relation R.
- Very often we will say that there is a **relationship** or an **association** between two objects u and v if the pair (u, v) is in some relation.
- A **function** f is a binary relation where for each u in U there is exactly one v from V such that (u, v) is in f. If both (u, v) and (u, w) are in f then v=w.

  In conventional terminology, the object u is termed the argument for f and v is the result.
- A **partition** of a set S is the set of subsets of S that have no overlapping elements and which if unioned together form S.

In all of the above there is nothing preventing the sets from which tuples are created to be sets of tuples themselves.

Binary relations are equivalent to normal predicates in everyday usage. Thus if U = V = the set of integers, then > is equivalent to the binary relation {(u, v)} where numerically $u > v$.

Non-numeric binary relations are also common. For example, if U = V = set of all people, then father(Pete, Tim) is the binary relation where Pete is the father of Tim.

1.1.2 Graphs

Graphs as we use them in this study are closely tied to binary relations.

- A **graph** G is equivalent to a binary relation based on a set of **objects** where there may be some explicit relationships between pairs of objects. Equivalently $G = (V, E)$ where V is the set of objects termed "vertices," and E a set of "edges" representing the relationships.
- A **vertex** is the name of some unique object within a graph.

  The set of vertices for a graph may be partitioned into different classes of vertices.

  In many contexts the term "node" is used interchangeably with vertex, whereas here we will use "node" solely to define a part of a parallel system.
- An **edge** is the name of the expression of a relationship between two vertices in a graph, and is often written as a pair (u, v) where u and v are vertices in the graph.

  Strictly speaking, the set E of edges for a graph may be a bag, meaning that there may be multiple pairs between the same two vertices.

Graphically, if a vertex is drawn as a point, then an edge (u, v) is an arrow drawn from u to v.

- If an edge (u, v) is **directed**, then the relationship implied by that edge only goes from u to v. If an edge (u, v) is **undirected**, then there are two relationships expressed by the edge: (u, v) and (v, u).
- Edges may also be **named**, where the subset of E that has the same "name" represents a set of edges that together express a particular binary relation. There is nothing preventing a graph from including many different types of edges.
- A **property** is a value that may be associated with either a vertex or an edge. An important example may be the name of an edge.
- The **out-degree** of a vertex u is the number of edges (u, v) that leave the vertex.

  There may be variations in this definition based on whether or not duplicate edges are permitted in a graph, and whether or not each such duplicate should be counted. Also, with there are multiple types of edges, the out-degree from a vertex may depend on the edge type.
- The **in-degree** of a vertex v is the number of edges (u, v) that enter the vertex.
- A **directed path** of **path length** n exists from a vertex u to a vertex v if there are edges (u, w_1), (w_1, w_2), ... (w_{n-2}, w_{n-1}), (w_{n-1}, v).
- The **transitive closure** of a graph G_1 is a graph G_2 where there is an edge from u to v in G_2 if there is a directed path (of any length) from u to v in G_1.

1.1.3 Computing Systems

- A **node** is a collection of cores and memory that represent a complete and minimal unit. Typically any thread executing in any core in a node can directly access any memory location in that node.

1.2 Change Log

Version	Date	Changes
0.1	9/10/18	Initial document construction.

Table 1.1: Change History.

Bibliography

Index