ID2222 Data Mining
Niklas Barth and Yu Gao

Lab 5 Report

Abstract

In this lab, we are to study, implement distributed graph partitioning using gossip-based peer-to-peer techniques, such as JaBeJa described in the paper "F. Rahimian, et al., JA-BE-JA: A Distributed Algorithm for Balanced Graph Partitioning, SASO2013" by Fatemeh Rahimian, Amir H. Payberah, Sarunas Girdzijauskas, Mark Jelasity, Seif Haridi.

We are to analyze 3elt, add20, and Facebook/Twitter graphs using our implementation of the JaBeJa algorithm. Each node iteratively selects another node among either its neighbors or a random sample, and investigates the pairwise benefit of a color exchange. This algorithm provides a cost function which minimizes the energy locally.

Code Work

sampleAndSwap Method

```
  private void sampleAndSwap(int nodeId) {
    Node partner = null;
    Node nodep = entireGraph.get(nodeId);

    // line 3 Algorithm 1 in the paper
    if (config.getNodeSelectionPolicy() == NodeSelectionPolicy.HYBRID
            || config.getNodeSelectionPolicy() == NodeSelectionPolicy.LOCAL) {
      // swap with random neighbors

      partner = findPartner(nodeId, getNeighbors(nodep));

    }

    // line 5 Algorithm 1 in the paper
    if (config.getNodeSelectionPolicy() == NodeSelectionPolicy.HYBRID
            || config.getNodeSelectionPolicy() == NodeSelectionPolicy.RANDOM) {
      // if local policy fails then randomly sample the entire graph
      if (partner != null){
        //No Operation because in this case we already have a local partner found
      }
      else{
        partner = findPartner(nodeId, getSample(nodeId));
```

```
            }
        }

        // line 7: swap the colors
        if(partner != null){
            int colorP = nodep.getColor();
            nodep.setColor(partner.getColor());
            partner.setColor(colorP);
        }
    }
```

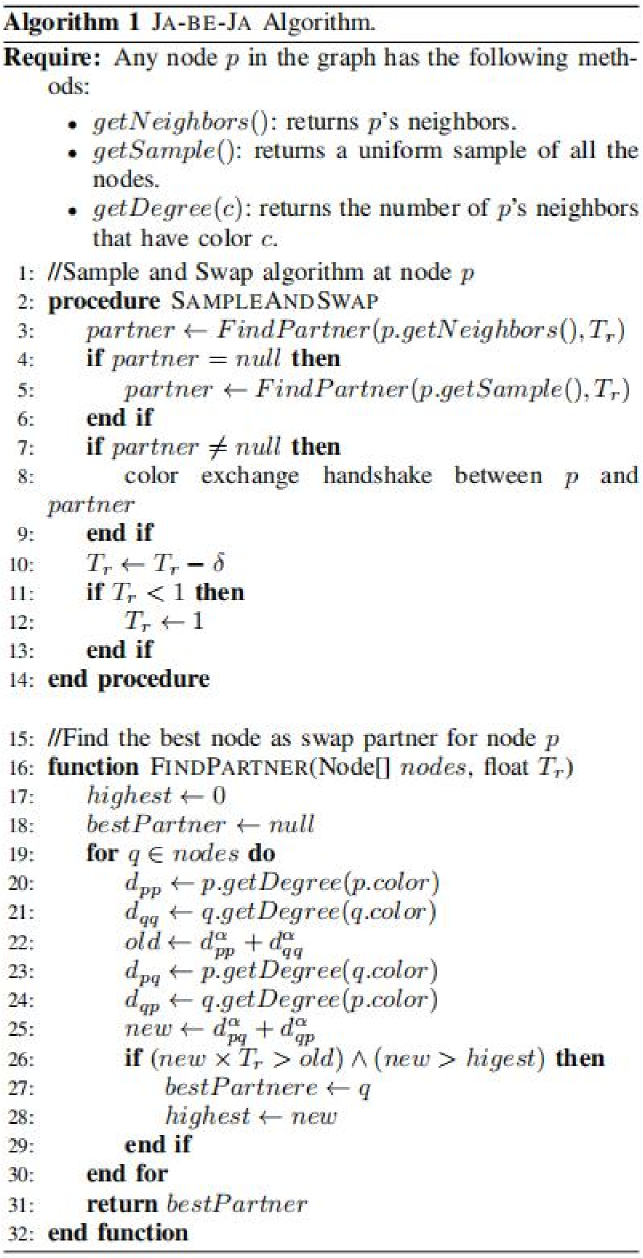
findPartner Method

```
    public Node findPartner(int nodeId, Integer[] nodes){

        Node nodep = entireGraph.get(nodeId);

        Node bestPartner = null; //line 18
        double highestBenefit = 0; //line 17

        double alpha = this.config.getAlpha();
        for(Integer id_q : nodes){
            Node q = entireGraph.get(id_q);
            double d_pp = getDegree(nodep, nodep.getColor()); //line 20
            double d_qq = getDegree(q, q.getColor()); //line21
            double _old = Math.pow(d_pp, alpha) + Math.pow(d_qq, alpha); //line22
            double d_pq = getDegree(nodep, q.getColor()); //line23
            double d_qp = getDegree(q, nodep.getColor()); //line24
            double _new = Math.pow(d_pq, alpha) + Math.pow(d_qp, alpha); //line25
            if(_new * this.config.getTemperature() > _old || _new > highestBenefit){ //equation 6
                bestPartner = q;
                highestBenefit = _new;
            }
        }
        return bestPartner;
    }
```



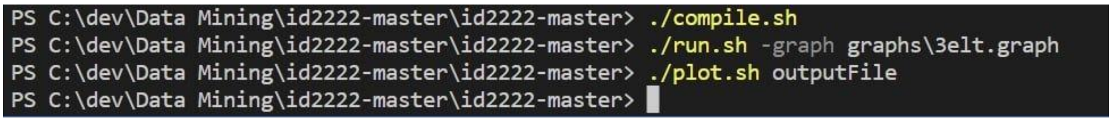
Algorithm 1 JA-BE-JA Algorithm.

```
Require: Any node $p$ in the graph has the following methods:
    • $getNeighbors()$: returns $p$'s neighbors.
    • $getSample()$: returns a uniform sample of all the nodes.
    • $getDegree(c)$: returns the number of $p$'s neighbors that have color $c$.
//Sample and Swap algorithm at node $p$
procedure SAMPLEANDSWAP
    $partner \leftarrow FindPartner(p.getNeighbors(), T_r)$
    if $partner = null$ then
        $partner \leftarrow FindPartner(p.getSample(), T_r)$
    end if
    if $partner \neq null$ then
        color exchange handshake between $p$ and $partner$
    end if
    $T_r \leftarrow T_r - \delta$
    if $T_r < 1$ then
        $T_r \leftarrow 1$
    end if
end procedure

//Find the best node as swap partner for node $p$
function FINDPARTNER(Node[] $nodes$, float $T_r$)
    $highest \leftarrow 0$
    $bestPartner \leftarrow null$
    for $q \in nodes$ do
        $d_{pp} \leftarrow p.getDegree(p.color)$
        $d_{qq} \leftarrow q.getDegree(q.color)$
        $old \leftarrow d_{pp}^{\alpha} + d_{qq}^{\alpha}$
        $d_{pq} \leftarrow p.getDegree(q.color)$
        $d_{qp} \leftarrow q.getDegree(p.color)$
        $new \leftarrow d_{pq}^{\alpha} + d_{qp}^{\alpha}$
        if $(new \times T_r > old) \wedge (new > higest)$ then
            $bestPartnere \leftarrow q$
            $highest \leftarrow new$
        end if
    end for
    return $bestPartner$
end function
```

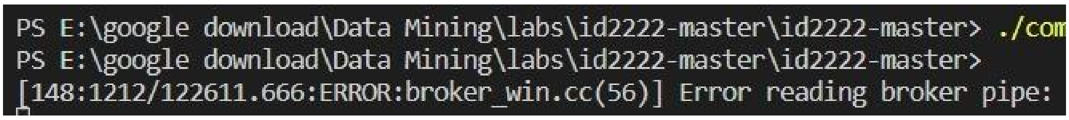
Result

In this lab, both of us have problems with activating shell script that given. Niklas got nothing as response and Yu got an error of pipe when we tried to run the project by typing "./compile.sh" in the terminal.



```
PS C:\dev\Data Mining\id2222-master\id2222-master> ./compile.sh
PS C:\dev\Data Mining\id2222-master\id2222-master> ./run.sh -graph graphs\3elt.graph
PS C:\dev\Data Mining\id2222-master\id2222-master> ./plot.sh outputFile
PS C:\dev\Data Mining\id2222-master\id2222-master> █
```

Figure 1: Niklas got nothing as response



```
PS E:\google download\Data Mining\labs\id2222-master\id2222-master> ./com
PS E:\google download\Data Mining\labs\id2222-master\id2222-master>
[148:1212/122611.666:ERROR:broker_win.cc(56)] Error reading broker pipe:
```

Figure 2: Yu got an error as response

We both tried to find solutions to run the script, but there are not so many tutorials about our problem. So we focus on the algorithm itself and make efforts on explaining how it works, i.e. to push the configuration towards lower energy states .