SEQUENTIAL AND PARALLEL ALGORITHM TO FIND MAXIMUM FLOW ON EXTENDED MIXED NETWORKS BY REVISED POSTFLOW-PULL METHODS

Lau Nguyen Dinh and Tran Quoc Chien

University of Da Nang, Danang City, Vietnam
launhi@gmail.com
dhsp@dng.vnn.vn

ABSTRACT

The problem of finding maximum flow in network graph is extremely interesting and practically applicable in many fields in our daily life, especially in transportation. Therefore, a lot of researchers have been studying this problem in various methods. Especially in 2013, we has developed a new algorithm namely, postflow-pull algorithm to find the maximum flow on traditional networks. In this paper, we revised postflow-push methods to solve this problem of finding maximum flow on extended mixed network. In addition, to take more advantage of multi-core architecture of the parallel computing system, we build this parallel algorithm. This is a completely new method not being announced in the world. The results of this paper are basically systematized and proven. The idea of this algorithm is using multi processors to work in parallel by postflow_push algorithm. Among these processors, there is one main processor managing data, sending data to the sub processors, receiving data from the sub-processors. The sub-processors simultaneously execute their work and send their data to the main processor until the job is finished, the main processor will show the results of the problem.

KEYWORDS

Processor, alogrithm, maximum flow, extended mixed network, parallel.

1. INTRODUCTION

The maximum flow problem on the network is one of the optimization problems on graphs that is widely applicable in practice as well as in combinatorial theory. The problem was proposed and solved by two American mathematicians Ford and Fulkerson in the early 1950 [2] and more and more scientists are interested in research. Edmonds and Karp gave method with complexity $O(|V|.|E|^2)$ [3]. In 1986, A. Goldberg and R.E. Tarjan [4] have developed pre-flow push method with complexity $O(|V|^2.|E|)$ and a lot of paper concerning parallel algorithm are written by many interested researchers [6], [7], [8], [9]. Especially in 2013 we has developed a new algorithm namely, postflow-pull algorithm to

Jan Zizka et al. (Eds) : ICAITA, SAI, CDKP, Signal, NCO - 2015
pp. 29–41, 2015. © CS & IT-CSCP 2015 DOI : 10.5121/csit.2015.51504

find the maximum flow on traditional networks [10]. The work of Naveen Garg and Jochen Konemann in 2007 and the above works just concentrate on traditional traffic networks without any specific steps and correct proof. In fact, it is necessary to build extended mixed networks. The problem of finding maximum flow in network mixed network is extremely interesting and practically applicable in many fields in our daily life, especially in transportation. Therefore, a lot of researchers have been studying this problem in various methods. In an ordinary graph the weights of edges and vertexes are considered independently where the length of a path is the sum of weights of the edges and the vertexes on this path. However, in many practical problems, weights at a vertex are not the same for all paths passing this vertex, but depend on coming and leaving edges. The paper develops a model of extended mixed network that can be applied to modelling many practical problems more exactly and effectively. Currently, parallel processing method is a promising and effective solution for the deadlock problems that sequential method encounters such as: program execution time, processing speed, the ability of memory storage, the advantage of multi-core architecture, large-scale data processing. The main contribution of this paper is the revised postflow-push [10] algorithm finding maximal flow on extended mixed network and we build parallel algorithms on multi processors. This is a completely new approach aiming to take advantage of multi-core architecture, to reduce computation time and to solve the problem with large-scale data [10].

2. EXTENDED MIXED NETWORK

Given a graph network G (V, E) with a set of vertices V and a set of edges E, where edges can be directed or undirected, with edge capacity ce:E$\rightarrow R^*$, so that $ce(e)$ is adge capacity $e \in$ E and vertices capacity cv:V$\rightarrow R^*$, so that $cv(u)$ is vertices capacity $u \in$ V. [12], [16].

With edge cost be be:E$\rightarrow R^*$, $be(e)$: cost must be return to transfer an unit transport on edge e.

With each $v \in$ V, Set E_v are set edge of vertice v.

Vertice cost bv:V$\times E_v \times E_v \rightarrow R^*$, $bv(u,e,e')$: cost must be return to transfer an unit transport from edge e to vertice u to edge e'.

A set (V, E, *ce, cv, be, bv*) is called extended mixed network.

3. FLOW EDGE ON EXTENDED MIXED NETWORK

Given an extended mixed network G = (V, E, *ce, cv, be, bv*). where s is source vertex, t is sink vertex. A set of flows on the edges f = $\{f(x,y) \mid (x,y) \in E\}$ is called flow edge on extended mixed network. So that

(i) $0 \le f(x,y) \le ce(x,y)\ \forall (x,y) \in E$

(ii) For any vertex k is not a source or sink

$$\sum_{(v,k)\in E} f(v,k) = \sum_{(k,v)\in E} f(k,v)$$

(iii) For any vertex k is not a source or sink

$$\sum_{(v,k)\in E} f(v,k) \le cv(\mathrm{k})$$

- **Theorem 3.1** Given $f = \{f(x,y) \mid (x,y) \in \mathrm{E}\}$ is flow edge on extended mixed network G, where s is source vertex, t is sink vertex, that is

$$\sum_{(s,v)\in E} f(s,v) - \sum_{(v,s)\in E} f(v,s) = \sum_{(v,t)\in E} f(v,t) - \sum_{(t,v)\in E} f(t,v)$$

Namly total flow go from source vertex equal to total flow going to sink vertex

Proof. $\forall\, x,y \in \mathrm{V} \mid \nexists\, (\mathrm{x,y}) \in \mathrm{E}$, then assign $f(x,y) = 0$, where

$$\sum_{u\in V}\sum_{v\in V} f(u,v) = \sum_{v\in V}\sum_{u\in V} f(v,u) \Leftrightarrow \sum_{v\in V}\left(\sum_{u\in V} f(u,v) - \sum_{u\in V} f(v,u)\right) = 0$$

$$\Leftrightarrow \sum_{v\in V\setminus\{s,t\}}\left(\sum_{u\in V} f(u,v) - \sum_{u\in V} f(v,u)\right) + \left(\sum_{(u,s)\in E} f(u,s) - \sum_{(s,u)\in E} f(s,u)\right) + \left(\sum_{(u,t)\in E} f(u,t) - \sum_{(t,u)\in E} f(t,u)\right) = 0$$

From (ii) in section 3, the first term equal to zero, so

$$\left(\sum_{(u,s)\in E} f(u,s) - \sum_{(s,u)\in E} f(s,u)\right) + \left(\sum_{(u,t)\in E} f(u,t) - \sum_{(t,u)\in E} f(t,u)\right) = 0$$

$$\Leftrightarrow \sum_{(s,u)\in E} f(s,u) - \sum_{(u,s)\in E} f(u,s) = \sum_{(u,t)\in E} f(u,t) - \sum_{(t,u)\in E} f(t,u)$$

The value of flow:

$$val(f) = \sum_{(s,u)\in E} f(s,u) - \sum_{(u,s)\in E} f(u,s)$$ is called value of flow f.

The maximum problem:

Given an extended mixed network G(V, E, *ce, cv, be, bv*), where s is source vertex, t is sink vertex. The task required by the problem is finding the flow which has a maximum value. The flow value is limited by the total amount of the circulation possibility on the roads starting from source vertex. As a result of this, there could be a confirmation on the following theorem.

- ***Theorem 3.2.*** Given an extended mixed network G(V, E, *ce, cv, be, bv*), where s is source vertex, t is sink vertex , then exist is the maximal flow.

4. MAXIMUM FLOW AND THE MINIMUM CUT

Given an extended mixed network G(V, E, ce, cv, be, bv), where s is source vertex, t is sink vertex. For any set S, T ⊂V, symbol (S, T) is a set of all edges reached and an unreached going from S input T, (S,T) = {(x, y) ∈ E |x∈ S &y∈ T}.

If S, T ⊂ V| S∪T = V & S∩T = ∅ and s∈ S, t∈T, then (S, T) is called cut (source-sink) of G.

Given f = {$f(x,y)$ | (x,y)∈E} is flow edge on extended mixed network G. Symbols

$$f(\mathrm{S},\mathrm{T}) = \sum_{(x,y)\in(S,T)} f(x,y)$$

• **Theorem 4.1.** Given an extended mixed network G(V, E, ce, cv, be, bv), where s is source vertex, t is sink vertex.

Given f = {$f(x,y)$ | (x,y)∈E} is flow edge on extended mixed network G and (S, T) is cut of G. Where, $val(f) = f(\mathrm{S},\mathrm{T})–f(\mathrm{T},\mathrm{S})$

Proof. ∀ x,y∈ V|∄ (x,y) ∈E, then assign $f(x,y)$= 0, we have

$$val(f) = \sum_{(s,u)\in E} f(s,u) - \sum_{(u,s)\in E} f(u,s) = \sum_{u\in V} f(s,u) - \sum_{u\in V} f(u,s) = \sum_{v\in S}\sum_{u\in V} f(v,u) - \sum_{v\in S}\sum_{u\in V} f(u,v)$$

$$(\text{as } \forall v\in \mathrm{S}\backslash\{\mathrm{s}\}, \sum_{u\in V} f(v,u) - \sum_{u\in V} f(u,v) = 0)$$

$$= \sum_{v\in S}\sum_{u\in S} f(v,u) + \sum_{v\in S}\sum_{u\in T} f(v,u) - \left(\sum_{v\in S}\sum_{u\in S} f(u,v) + \sum_{v\in S}\sum_{u\in T} f(u,v)\right)$$

$$= \sum_{v\in S}\sum_{u\in S} f(v,u) - \sum_{v\in S}\sum_{u\in S} f(u,v) + \sum_{v\in S}\sum_{u\in T} f(v,u) - \sum_{v\in S}\sum_{u\in T} f(u,v)$$

$$= \sum_{v\in S}\sum_{u\in T} f(v,u) - \sum_{v\in S}\sum_{u\in T} f(u,v) = f(\mathrm{S},\mathrm{T}) - f(\mathrm{T},\mathrm{S}).$$

Given (S,T) is cut. Symbol S(T) = {u∈ S| ∃v∈ T, (u,v)∈ (S,T)}

• **Theorem 4.2.** Given an extended mixed network G(V, E, ce, cv, be, bv), where s is source vertex, t is sink vertex.

Given f = {$f(x,y)$ | (x,y)∈E} is flow edge on extended mixed network G and (S, T) is cut of G. Where, ∀S'⊂S(T) we have

$$f(\mathrm{S},\mathrm{T}) \le \sum_{v\in S'} c_V(v) + \sum_{(x,y)\in(S,T)\backslash(S',T)} c_E(x,y)$$

Proof. we have

$$f(\mathrm{S},\mathrm{T}) = \sum_{(x,y)\in(S,T)} f(x,y) = \sum_{(x,y)\in(S',T)} f(x,y) + \sum_{(x,y)\in(S,T)\backslash(S',T)} f(x,y) = \sum_{x\in S'}\sum_{(x,y)\in(\{x\},T)} f(x,y) + \sum_{(x,y)\in(S,T)\backslash(S',T)} f(x,y)$$

$$\le \sum_{v\in S'} c_V(v) + \sum_{(x,y)\in(S,T)\backslash(S',T)} c_E(x,y)$$

The capacity of slice cut

Given (S, T) is slice cut of G. Symbol cap(S, T) is capacity of (S, T) slice cut. We have

$$cap(\mathrm{S,T}) = \min\{\sum_{v\in S'} cv(v) + \sum_{(x,y)\in(S,T)\setminus(S',T)} ce(x,y) \mid \mathrm{S'}\subset \mathrm{S(T)}\}$$

From **Theorem 4.1** and **Theorem 4.2** infered that **Theorem 4.3**

• **Theorem 4.3.** Given $f = \{f(x,y) \mid (x,y)\in \mathrm{E}\}$ is flow edge on extended mixed network G and (S, T) is cut of G. Where $val(f) \le cap(\mathrm{S,T})$.

5. POSTFLOW-PULL METHODS

5.1. Some basic concept

5.1.1. Residual extended mixed network G_f

For flow f on G = (V, E, *ce, cv, be, bv*), where s is source vertex, t is sink vertex. Residual extended network, denoted G_f is defined as the extended mixed network with a set of vertices V and a set of edge E_f with the edge capacity is ce_f and vertices capacity is cv_f as follows:

- For any edge (u, v) ∈ E, if f(u, v)> 0, then (v, u) ∈ E_f with edge capacity is ce_f (v,u)=f(u, v)

- For any edge (u,v) ∈ E, if c(u,v) -f(u, v)> 0, then (u, v) ∈ E_f with edge capacity is ce_f(u,v) = ce(u,v) - f(u,v)

- For any vertices v∈ V then $cv_f(v) = cv(v) - \sum_{(x,v)\in E} f(x,v)$.

5.1.2. Preflow

For extended mixed network G = (V, E, *ce, cv, be, bv*). *Preflow* is a set of flows on the edges f = {f(x, y) | (x, y)∈ G} So that

(i) $0 \le f(x, y) \le ce(x, y)\ \forall(x, y) \in E$

(ii) for any vertex k is not a source or sink, inflow is not smaller than outflow, that is

$$\sum_{(v,k)\in E} f(v,k) \ge \sum_{(k,v)\in E} f(k,v)$$

(iii) for any vertex k is not a source or sink

$$\sum_{(v,k)\in E} f(v,k) \le cv(\mathrm{k})$$

5.1.3. Postflow

For extended mixed network G = (V, E, *ce, cv, be, bv*). *Postflow* is a set of flows on the edges f = {f(x, y) | (x, y)∈ G} So that

(i) $0 \le f(x, y) \le ce(x, y)\ \forall(x, y) \in E$

(ii) for any vertex k is not a source or sink, outflow is not smaller than inflow, that is

$$\sum_{(v,k)\in E} f(v,k) \le \sum_{(k,v)\in E} f(k,v)$$

(iii) for any vertex k is not a source or sink

$$\sum_{(v,k)\in E} f(v,k) \leq cv(\mathrm{k})$$

Each vertex whose outflow is larger than its inflow is called the unbalanced vertex. The difference between a vertex's inflow and outflow is called excess. The concept of residual extended mixed network G_f is similarly defined as flow.

The idea of this methods is balancing inflow and outflow at the balanced vertices by pushing along an outgoing edge and pushing against an incoming edge. Process of balancing is repeated until no more the unbalanced vertex then we get maximum flow. We store the unbalanced vertices on a generalized queue. A tool called a depth function is used to help select the edge available in residual network to eliminate the unbalanced vertices. Now we assume that a set of the network is denoted as V={0,1,...,|V|-1}.

5.1.4. Depth function

Depth function of the *Postflow* in the extended mixed network G = (V, E, *ce, cv, be, bv*), is a set of non-negative vertex weights d(0), ..., d(|V| −1) such that d(s) = 0(s is source vertex) and d(u)+1 ≥ d(v) for every edge (u,v) in the residual extended mixed network for the flow. An eligible edge is an edge (u,v) in the residual extended mixed network with d(u)+1=d(v).

A trivial depth function is d(0) = d(1) = ... = d(|V| − 1) = 0. Then if we set d(u)= 1, any positive edge to u is the priority edge.

We define a more interesting depth function by assigning to each vertex the latter's shortest–path distance to the sink (its distance to the root in any BFS tree of the network rooted at s. This depth function is valid because d(s)= 0, and for any pair of vertices u and v connected by an edge (u,v) in residual mixed network G_f, then d(u)+1≥ d(v), because the path from a to v with edge (u,v) (d(u)+1 must be not shorter than the shortest path from s to v i.e d(v)).

Property 5.1. For any flow f in extended mixed network G and associated depth function d. a vertex's depth d(v) is not larger than the length of the shortest path from vertex s to vertex v in residual extended mixed network G_f .

Proof: For any given vertex v, assume l be the shortest-path length from s to v in the residual extended mixed network G_f. And let ($s=v_1, v_2, ..., v_l=v$) from s to v. then

$$\begin{aligned} d(v) = d(v_1) &\leq d(v_{l-1}) + 1 \\ &\leq d(v_{l-2}) + 2 \\ &\vdots \\ &\leq d(v_1) + l = d(s) + l = l \text{ (because d(s)=0)} \end{aligned}$$

The intuition behind depth function is the following: when an unbalanced node's depth is less then the depth of the sink, it is possible that there is some way to push flow from that node down to the source; else, if an unbalanced node's depth exceeds the depth of the sink, we know that node's flow needs to be pushed back to the sink.

Corollary: if a vertex's depth is greater then |V|, then there is no path from the source to that vertex in the residual extended mixed network G_f.

5.2. General Postflow-pull methods

General Postflow-push methods is briefly described as follows:

Step1:

Initialize: the only Postflow is in the edges leaving for the sink vertices is the following:

$f(v,t)=min\{ce(v,t), cv(v)\}$

The other flows are 0.

Select any available depth function d in the extended mixed network G.

Step 2:

Condition to terminate : If there are no available unbalanced vertices, then postflow f becomes max flow.

Step 3: (pull flow)

Choose unbalanced vertex v.

If exists priority edge $(u, v) \in E_f$ then

If $f(v,u)>0$, then pull along the edge (u,v) a flow with value $min\{-delta, ce_f(u,v)\}$(where $delta<0$ is the *excess* of the vertex v).

If $(u,v)\in E$ and $cv_f(u)>0$, then pull along the edge (u,v) a flow with value $min\{-delta, ce_f(u,v), cv_f(u)\}$(where $delta<0$ is the *excess* of the vertex v).

If it does not exists the priority edge from v, then increased the depth of the vertex v as follows:

$$d(v) := 1 + \min \{d(u) \mid (u, v) \in E_f\}$$

Back to step 2.

◊ Note. In the general Postflow-pull methods, we do not give the detailed steps how to select the initial depth function, how to choose the unbalanced vertices as well as how to choose the priority edges. Performing these detailed steps for many algorithms belongs to the general Postflow-pull methods.

Property 5.2. Postflow-pull methods always preserve the validity of the depth function.

Proof:

(i) Where it exists priority edge $(u,v) \in E_f$: We have $d(u)+1 = d(v)$. After pulling along edge (u,v) a flow, we still have $d(v) +1= d(u) +2 \geq d(u)$.

(ii) if it does not exist priority edges to v: we have $\forall u$: $(u,v) \in E_f \Rightarrow d(u)+1 > d(v)$. After incrementing $d(v):d(v):=1+\min\{d(u)|(u,v)\in E_f\}$ then d(v) still satisfied $\forall u$, $(u,v) \in E_f$: $d(u)+1 \geq d(v)$.

*Property 5.3***.** While Postflow-pull algorithm is in execution, there always exists a directed path from sink vertex to the unbalanced vertex in the residual extended mixed network, and there are no directed paths from source vertex to sink vertex in the residual extended mixed network.

Proof. (by induction)

Initially, the only Postflow is in the edges leaving for the sink vertices is the following: $f(v,t)=min\{ce(v,t),cv(v)\}$ and other flows are 0. Then the first vertices of those edges directed to the sink are unbalanced. With any unbalanced vertex u, we have $(t, u) \in E_f$ and $(u,t) \notin E_f$, inferred there exists paths from t to u, and there are no directed paths from source vertex a to sink vertex in the residual extended mixed network G_f. So the properity is true with the initial flow.

Next, the new unbalanced vertex u only appears when a flow is pushed to the old unbalanced vertex v on the priority edge (u,v). Then the residual extended mixed network will have more edge (v,u). Due to exist of the path from residual extended mixed network from t to v based on inductive hypothesis, there exists a path from t to u in the residual extended network.

To prove that there are no paths from source vertex s to sink t in the residual extended mixed network. It can be argued as follows.

First, vertices u adcajent to sink vertices t, $(u, t) \in E$, since the initial flow on the edge (u,t) is $f(u,t)=min\{ce(u,t),cv(u)\}$, if $(u,t) \in G_f$, then the flow pushed back along t to u. Where (t, u) is the priority edge, d(t)+1 = d(u) > t(t). Thus each vertex a can reach to t in the residual extended mixed network must have the depth which is greater than the depth of t.

For any u to t in the residual extended mixed network. There exists paths from u to t in the residual extended mixed network: $(u \rightarrow u_1 \rightarrow u_2 \rightarrow \ ... \ u_k \rightarrow t)$. Similarly argued as above we have $d(u) > d(u_1) > ... > d(u_{k-1}) > d(u_k) > d(t)$

Thus each vertex to t must have a depth which is greater than t. Besides, the depth of the source vertex is 0, so it's impossible to reach to t. So there are no directed paths from source vertex to sink vertex in the residual extended network.

• *Corollary*. Vertex's depth is always less than 2.|V|.

Proof. We need to consider only unbalanced vertices, the depth of each unbalanced vertex is either the same as or 1 greater than it was the last time that the vertex was balanced. By the same argument as in the proof of Property 5.1, the path from s source vertex to a given unbalanced vertex in the residual extended mixed network G_f implies that unbalanced vertex's depth is not greater than the sink vertex's depth plus |V| -2 (the source vertex can not be on the path). Since the depth of the sink never changes, and it is initially not greater than |V|, the given unbalanced vertex's depth is not greater than 2.|V| - 2, and no vertex has depth 2|V| or greater.

• Theorem 5.4 General Postflow-pull methods is true.

Proof. First we prove the general Postflow-pull method that terminates after perforing some finite steps. We confirm that after implementing these finite steps there is not any unbalanced vertex. Proof by contradiction method is used. Assume that the set of vertices are infinite, there will exist vertex u that appears infinite times in that set. Since the number of vertices in the network is finite so there exists vertex $v \neq u$ so that the flow is pulled on along (u,v) and (v,u) in infinite times. Since edge (u,v) and edge(v,u) are the priority ones in infinite residual network and d(u)+1= d(v) and d(v)+1=d(u), then the depth of u and v will increment indefinitely, and this conflicts with the above corollary.

When this method terminates, we receive the flow. Based on *property 5.3*, it does not exist a path from the source to the sink in the residual network. According to augmenting-path algorithm, it is max flow.

The complexity of the following method is $O(|V|^2|E|)$ [10].

5.3. Postflow-pull algorithm

This is a particular algorithm in Postflow-pull method. Here the unbalanced vertices are pushed into the queue. With each vertex from the queue, we will pull the flow in the priority edge until the flow becomes either balanced or does not have any priority edge. If it does not exist priority edge but there are unbalanced vertices, then we increase the depth and push it into the queue.

Now we can describe the Postflow-pull algorithm as follows:

Inputs: Extended mixed network G with source s, sink t,

Output: Maximum flow

$$F = (f_{ij}), (i, j) \in E$$

Step 1: Initialized:

Initialize: the only postflow is in the edges for the source vertices is the following:

$f(v,t)=min\{ce(v,t), cv(v)\}$

The other flows are 0.

Choose depth function d(v) which is the length of the shortest path from source s to vertex v.

Push all unbalanced vertices into the queue Q.

Step 2: Condition to terminate: If Q = ∅, then postflow f becomes maximum flow, end.

Step 3:

Get unbalanced vertex v from the queue Q.

Browsing the priority edge $(u, v) \in E_f$

- If f(v,u)>0, then pull along the edge (u,v) a flow with value $min\{-delta, ce_f(u,v)\}$(where delta<0 is the *excess* of the vertex v).

- If $(u,v) \in E$ and $cv_f(u)>0$, then pull along the edge (u,v) a flow with value $min\{-delta, ce_f(u,v), cv_f(u)\}$ (where delta<0 is the *excess* of the vertex v).

- If vertex u is the new unbalanced vertex, then push this vertex u into queue Q.

- If vertex v is still unbalanced, then increased the depth of the vertex v as follows:

$$d(v) := 1 + \min \{d(u) \mid (u, v) \in E_f\}$$

Back to step 2.

6. POSTFLOW-PULL PARALLEL ALGORITHM TO FIND THE MAXIMUM FLOW

6.1. The idea of the algorithm

Based on the parallel algorithm [10], we build parallel algorithms on m processors. In m processors, there will be a main processor to manage data, divide the set of vertex V of the graph into m-1 sub-processors, and send data to the sub-processors as well as receive data from the sub-processors sending to [6],[7],[8],9], [10].

Sub-processors receive the values from the main processor, then proceed to pull and replace label (pull_relabel) and transfer the results to the main processor.

The main processor after receiving the results from the sub-processors will perform replacement label (Relabel) until finding the maximum flow

6.2. Building the parallel algorithm

Inputs: Extended mixed network G with source s, sink t m processors (P_0, P_1,…, P_{m-1}), where P_0 is the main processor

Output: Maximum flow

$$F = (f_{ij}), (i, j) \in E$$

Step 1: The main processor P_0 performs

(1.1). initialize: e, d, f, c_f, Q: set of unbalanced vertices (excluding the vertices s and t) are the vertices with positive excess.

(1.2). divide set of vertices V into sub-processors:

Let P_i be the i^{th} sub-processor (i = 1,2, ..., m-1)

P_i will receive the set of vertices V_i so that

$$(V_i \cap V_j = \phi \text{ if } i \neq j, \text{ and } \cup_i \{V_i\} = V)$$

(1.3). The main processor sends e, c_f to sub-processors

Step 2: The Condition to terminate: If Q = ∅, then postflow f becomes maximum flow, end. Else, go to step 3.

Step 3: The main processor sends d to sub-processors

Step 4: m-1 sub-processors (P_1, P_2, …,P_{m-1}) implement

(4.1) Receive e, c_f, d and the set of vertices from the main processor

(4.2) Handling unbalanced vertice v (pull and replace label). Get unbalanced vertexs v from Q and $v \in V_i$ (i= 1,2, ..., m-1). Browsing the priority edge $(u, v) \in E_f$

- If f(v,u)>0, then pull along the edge (u,v) a flow with value $min\{-delta, ce_f(u,v)\}$(where delta<0 is the *excess* of the vertex v).

- If $(u,v) \in E$ and $cv_f(u)>0$, then pull along the edge (u,v) a flow with value $min\{-delta, ce_f(u,v), cv_f(u)\}$. (where delta<0 is the *excess* of the vertex v).

If vertex v is still unbalanced, then increased the depth of the vertex v as follows:

$$d(v) := 1 + \min \{d(u) \mid (u, v) \in E_f\}$$

(4.3) Send e, c_f, d to the main processor

Step 5: The main processor implements

(5.1) Receive e, c_f, d from step 4.3

(5.2) This step is distinctive from the sequential algorithms to synchronize our data, after receiving the data in (5.1), the main processor checks if all the edges $(u,v) \in E$ that have d(v)> d(u)+1, the main processor will relabel for vertices u, v as follows:

- e(u):= e(u)-ce_f(u,v), e(v):= e(v)+ce_f(u,v)

- If f(v,u)>0, then f(u,v):= *min*{-*delta*,$ce_f(u,v)$} (where delta<0 is the *excess* of the vertex v).

- If $(u,v) \in$ E and $cv_f(u)>0$, then f(u,v):= *min*{-*delta*, $ce_f(u,v)$, $cv_f(u)$} (where delta<0 is the *excess* of the vertex v). Put the new unbalanced vertex into set Q

(5.3) If $\forall u \in V$ e(u)=0, eliminate u from active set Q. Back to step 2.

Theorem 3.1. Postflow-pull parallel algorithm is true and has complexity O(|V|^2 |E|).

Proof: Similar to [10]. postflow-pull parallel algorithm is build in accordance with other parallel computing system such as: PRAM, Cluster system, CUDA, RMI, threads,… Push and replace label using *atomic*, due to support of *atomic 'read-modify-write'* instructions, are executed atomically by the architecture. Other than the two execution characteristics provided by the architecture, we do not impose any order in which executions from multiple sub-processors can or should be interleaved, as it will be left for the sequential consistency property of the architecture to decide.

The outcome of the execution reduces to only a few simplified scenarios. By analyzing these scenarios, we can show that function f is maintained as a valid depth function. A valid d guarantees that there does not exist any paths from s to t throughout the execution of the algorithm, and hence guarantees the optimality of the final solution if the algorithm terminates. The termination of the algorithm is also guaranteed by the validatity of d, as it bounds the number of pull and relabel operations to O(|V |^2|E|). □

Parallel algorithm for finding maximum flow in the extended mixed network is built on m processors. The program written in Java with database administration system MySQL We experimentally sampled nodes as follows: The extended mixed graph corresponds to 18000 nodes and 25000 edge. The simulation result is shown in figure 1. This result demonstrates that the runtime of parallel algorithms is better than sequential algorithm.

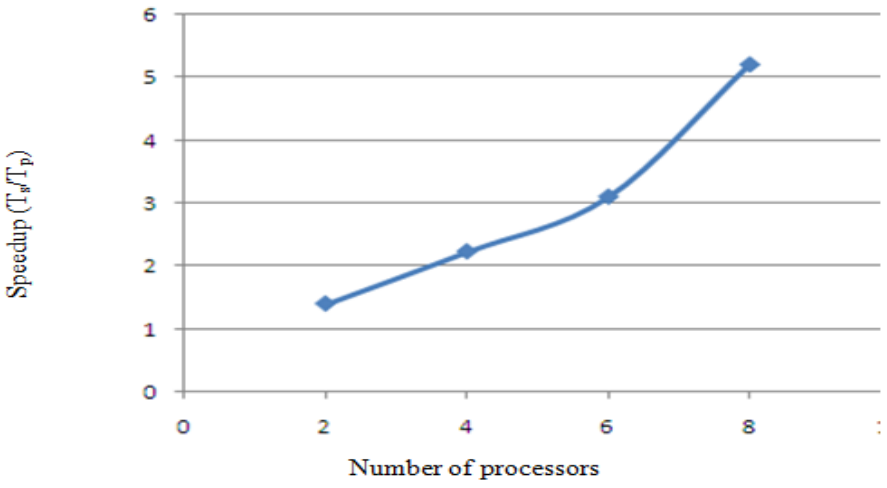

Figure 1. Chart performs the speedup of extended Mixed graph having 18000 nodes and 25000 edge

7. CONCLUSION

The detail result of this paper is building sequential and parallel algorithm by postflow-pull methods to find maximum flow in extended mixed network. In addition, to take more advantage of multi-core architecture of the parallel computing system and reduce the computing time of this algorithm, we build this algorithm on multiple processors. This is a completely new method not being announced in Vietnam and in the world. The results of this paper are basically systematized and proven.

REFERENCES

[1] Chien Tran Quoc, 2010 "Postflow-pull methods to find maximal flow", Journal of science and technology - University of DaNang, 5(40), pp 31-38.

[2] L. R. Ford and D. R. Fulkerson, 1962 Flows in Networks Princeton University Press.

[3] J. Edmonds and R. M. Karp, 1972 "Theoretical improvements in algorithmic efficiency for network flow problems," J. ACM, vol. 19, no. 2, pp. 248–264.

[4] A. V. Goldberg and R. E. Tarjan, 1986 "A new approach to the maximum flow problem," in STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing. New York, NY, USA: ACM, pp. 136-146.

[5] Robert Sedgewick, 2011 Algorithms in C, Part 5: Graph Algorithms. Addison-Wesley.

[6] R. J. Anderson and a. C. S. Jo, 1992 "On the parallel implementation of goldberg's maximum flow algorithm" in SPAA '92: Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures. New York, NY, USA: ACM, pp. 168–177.

[7] D. Bader and V. Sachdeva, 2005 "A cache-aware parallel implementation of the push- relabel network flow algorithm and experimental evaluation of the gap relabeling heuristic" in PDCS '05: Proceedings of the 18th ISCA International Conference on Parallel and Distributed Computing Systems.

[8] B. Hong, 2008 "A lock-free multi-threaded algorithm for the maximum flow problem" in IEEE International Parallel and Distributed Processing Symposium, Aprail.

[9] Zhengyu He, Bo Hong, 2010 "Dynamically Tuned Push-Relabel Algorithm for the Maximum Flow Problem on CPU-GPU-Hybrid Platforms", School of Electrical and Computer Engineering-Georgia Institute of Technology.

[10] Chien Tran Quoc, Lau Nguyen Dinh, Trinh Nguyen Thi Tu, 2013 "Sequential and Parallel Algorithm by Postflow-Pull Methods to Find Maximum Flow", Proceedings 2013 13th International Conference on Computational Science and Its Applications, ISBN:978-0-7695-5045-9/13 $26.00 © 2013 IEEE, DOI 10.1109/ICCSA.2013.36, published by IEEE- CPS pp 178-181.

[11] Lau Nguyen Dinh, Thanh Le Manh, Chien Tran Quoc, 2013 "Sequential and Parallel Algorithm by Pre-Push Methods to Find Maximum Flow", Vietnam Academy of Science and Technology AND Posts & Telecommunications Institute of Technology, special issue works Electic, Tel, IT; 51(4A) ISSN: 0866 708X, pp 109-125.

[12] Lau Nguyen Dinh, Chien Tran Quoc and Manh Le Thanh, 2014 "Parallel algorithm to divide optimal linear flow on extended traffic network", Research, Development and Application on Information & Communication Technology, Ministry of Information & Communication of Vietnam, No 3, V-1.

[13] Naveen Garg, Jochen Könemann, 2007 "Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems", SIAM J. Comput, Canada, 37(2), pp. 630-652.

[14] Lau Nguyen Dinh, Chien Tran Quoc, Thanh Le Manh, 2014 "Improved Computing Performance for Algorithm Finding the Shortest Path in Extended Graph", proceedings of the 2014 international conference on foundations of computer science (FCS'14), July 21-24, 2014 Las Vegas Nevada, USA, Copyright © 2014 CSREA Press, ISBN: 1-60132-270-4, Printed in the United States of America, pp 14-20.

[15] Chien Tran Quoc, Thanh Le Manh, Lau Nguyen Dinh, 2013 "Sequential and parallel algorithm by combined the push and pull methods to find maximum flow", Proceeding of national Conference on

Fundamental and Applied Infromation Technology Research (FAIR), Hue, Vietnam, 20-21/6/2013.ISBN: 978-604-913-165-3, pp 538-549.

[16] Chien Tran Quoc, Thanh Le Manh, Lau Nguyen Dinh, 2013 “Parallel algorithm to find maximum flowcostlimits on extended traffic network”, Proceeding national Conference XVI "Some selected issues of Information Technology and Communications” Danang 14-15/11/2013, ISBN: 978-604-67-0251-1, pp 314-321.

[17] Lau Nguyen Dinh, Tran Ngoc Viet, 2012 “Parallelizing algorithm finding the shortest paths of all vertices on computer cluster system”, Proceedings national Conference XVth "Some selected issues of Ìnormation Technology and Communications” Ha Noi, 03-04-2012, pp 403-409.

[18] Lau Nguyen Dinh, Tran Ngoc Viet, 2012 “A parallel algorithm finding the shortest paths of multiple pairs of source and destination vertices in a graph”, Journal of science and technology - University of DaNang 9 (58),2012, pp 30-34.

[19] Lau Nguyen Dinh, Tran Ngoc Viet, 2012 “Parallelizing algorithm dijkstra’s finding the shortest paths from a vertex to all vertices”, Journal of science, University of Hue, 74B, 5, pp 81-92.

AUTHORS

1.Dr. LAU NGUYEN DINH

Born in 1978 in Dien Ban, Quang Nam, Vietnam. He graduated from Maths_IT faculty of Hue university of science in 2000. He got master of science (IT) at Danang university of technology and hold Ph.D Degree in 2015 at Danang university of technology. His main major: Applicable mathematics in transport, parallel and distributed process, discrete mathemetics, graph theory, grid Computing and distributed programming.

2.Ass. Prof. DrSc. CHIEN TRAN QUOC

Born in 1953 in Dien Ban, Quang Nam, Vietnam. He graduated from Maths_IT faculty. He got Ph.D Degree of maths in 1985 in Charles university of Prague, Czech Republic and hold Doctor of Science in Charles university of Prague, Czech Republic in 1991. He received the tittle of Ass. Pro in 1992. He work for university of Danang, Vietnam. His main major: Maths and computing, applicable mathematics in transport, maximum flow, parallel and distributed process, discrete mathemetics, graph theory, grid Computing, distributed programming.