Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 35 (2008) 2694–2708

computers & operations research

www.elsevier.com/locate/cor

Efficient preflow push algorithms

R. Cerulli[a,*], M. Gentili[a], A. Iossa[b]

[a]*Dipartimento di Matematica ed Informatica, Università di Salerno, Via Ponte Don Melillo, 84084 Fisciano (SA), Italy*
[b]*Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università degli Studi di Roma "La Sapienza", Piazzale A. Moro 5, 00185 Roma, Italy*

Available online 22 December 2006

Abstract

Algorithms for the maximum flow problem can be grouped into two categories: *augmenting path* algorithms [Ford LR, Fulkerson DR. Flows in networks. Princeton University Press: Princeton, NJ: 1962], and *preflow push* algorithms [Goldberg AV, Tarjan RE. A new approach to the maximum flow problem. In: Proceedings of the 18th annual ACM symposium on theory of computing, 1986; p. 136–46]. Preflow push algorithms are characterized by a drawback known as *ping pong effect*. In this paper we present a technique that allows to avoid such an effect and can be considered as an approach combining the augmenting path and preflow push methods. An extended experimentation shows the effectiveness of the proposed approach.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Maximum flow; Ping pong effect; Preflow push

1. Introduction

The maximum flow problem is one of the most studied problems in the literature and there exist many contributions proposing different efficient algorithms to solve the problem. Algorithms for the maximum flow problem can be grouped into two main categories: *augmenting path* algorithms [1], and *preflow push* algorithms [2]. There are also contributions in the literature presenting a generic resolution approach that combines both methods [3]. These methods capture the computational efficiency of both these classical approaches, however, they could not overcome the main drawback of the preflow push algorithms known as the *ping pong effect*. Many efforts have been devoted to overcome such an effect (the gap property introduced by [4] is an example), however, they are not always effective, as it will be shown in the sequel of the paper.

We present a new generic technique, namely the *budget algorithm*, that can be considered as an approach combining the two classical augmenting path and preflow push methods. Therefore, on the one hand, we maintain the computational efficiency of the classical approaches, and, on the other hand, we overcome the ping pong effect greatly improving the computation times.

The rest of the paper is organized into three parts as follows. In the first part (Section 2), we report some well known results on network flows, we briefly describe the preflow push algorithms, and we show a bad example for the preflow

* Corresponding author. Tel.: +39 89963326; fax: +39 89963303.
E-mail addresses: raffaele@unisa.it (R. Cerulli), mgentili@unisa.it (M. Gentili), aiossa@unisa.it (A. Iossa).

0305-0548/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2006.12.023

push algorithm that generates the ping pong effect. The second part (Section 3), describes our budget algorithm and shows how it overcomes the ping pong effect. In the third part of the paper (Section 4), we present the results coming from a wide experimentation to evaluate our algorithm's performance.

2. The maximum flow problem and existing approaches

In this section we recall the main definitions and properties about admissible flows and admissible preflows, for additional details we refer the reader to [5].

A network is a directed graph $G = (N, A, s, t, u)$ where N is a set of n nodes and A is a set of m arcs; s is the source node and t is the sink node; u is a capacity function that assigns to an arc $(i, j) \in A$ a nonnegative real value u_{ij}.

An *admissible flow* is a real function $X : A \longrightarrow R$ satisfying the flow conservation (1) and capacity (2) constraints:

$$\sum_{(j,i)\in\delta_G^-(i)} x_{ji} - \sum_{(i,j)\in\delta_G^+(i)} x_{ij} = 0 \quad \forall i \in N \backslash \{s, t\}, \tag{1}$$

$$0 \leqslant x_{ij} \leqslant u_{ij} \quad \forall (i, j) \in A, \tag{2}$$

where x_{ij} are the classical flow variables denoting the flow on each arc $(i, j) \in A$, $\delta_G^+(i)$ is the set of arcs outgoing from i in G, that is $\delta_G^+(i) = \{(i, j) \in A | j \in N\}$ and $\delta_G^-(i)$ is the set of arcs coming into i in G, that is $\delta_G^-(i) = \{(j, i) \in A | j \in N\}$. The value $|f|$ of a flow X is the net flow entering t, that is $|f| = \sum_{(i,t)\in\delta_G^-(t)} x_{it}$. In the *maximum flow problem*, we want to determine an admissible flow of maximum value.

A *preflow* is a real function $X : A \longrightarrow R$ that satisfies the capacity constraints (2) and the following relaxation (3) of the flow conservation constraints (1):

$$\sum_{(j,i)\in\delta_G^-(i)} x_{ji} - \sum_{(i,j)\in\delta_G^+(i)} x_{ij} \geqslant 0 \quad \forall i \in N \backslash \{s, t\}. \tag{3}$$

In the *maximum preflow problem* we want to determine a preflow of maximum value, that is, a preflow where the net flow entering the sink t is maximum.

All the algorithms solving the maximum flow problem work on the residual network $G(X)$, obtained from G once an initial admissible flow and the residual capacity of each arc is computed. Given a flow (preflow) X, the *residual capacity* r_{ij} of an arc $(i, j) \in A$ is the maximum additional flow that can be sent from node i to node j through the arcs (i, j) and (j, i). The residual capacity has two components:

- u_{ij}—x_{ij}, that represents the not used capacity of arc (i, j);
- the current flow x_{ji} on the arc (j, i) that can be reduced in order to increase the flow from node i to node j.

Therefore, we can compute the residual capacity as $r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$. Given a network flow $G = (N, A, s, t, u)$ and a flow (preflow) X, the *residual network* induced by X is the graph $G(X) = (N, A')$ where $A' = \{(i, j) \in N \times N | r_{ij} > 0\}$. Given a pair of vertices i, j of G, an augmenting path from i to j in G is a directed path P from i to j in $G(X)$. The residual capacity of an augmenting path P is the value $r(P) = \min\{r_{ij} | (i, j) \in P\}$. An augmenting path from i to j can be used to move at most $r(P)$ units of flow from i to j.

Any algorithm based on the augmenting path approach, (i) finds an initial admissible flow X, (ii) looks for an augmenting path P on the residual graph $G(X)$ if it exists, and (iii) determines a new admissible flow X' pushing $r(P)$ units on the arcs of the path. When it is not possible to find any augmenting path then the actual flow is optimum and the algorithm stops.

On the other hand, any algorithm based on the preflow push approach, uses the relation between the Max Flow and the Max Preflow problems. Indeed, we can see the maximum flow problem as a particular case of the maximum preflow problem, as explained next.

Given a preflow X, we define the excess of a node i, namely $e(i)$, as the value $\sum_{(j,i)\in\delta_{G(X)}^-(i)} x_{ji} - \sum_{(i,j)\in\delta_{G(X)}^+(i)} x_{ij}$. That is, the net flow of node i in the residual graph $G(X)$. A node is *active* if it has a strictly positive excess. Thus, we

Download English Version:

https://daneshyari.com/en/article/474061

Download Persian Version:

https://daneshyari.com/article/474061

Daneshyari.com