Information Processing Letters 109 (2009) 790–794

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Distributed approximation for maximum weight matching on bounded degree bounded integer weight graphs

Satyajit Banerjee *, Atish Datta Chowdhury, Subhas Kumar Ghosh

Honeywell Technology Solutions, 151/1, Doraisanipalya, Bannerghatta Road, Bangalore, India-560076

ARTICLE INFO

Article history:
Received 1 October 2008
Received in revised form 28 January 2009
Accepted 25 March 2009
Available online 27 March 2009
Communicated by K. Iwama

Keywords:
Graph algorithms
Distributed algorithms
Approximation algorithms
Maximum matching

1. Introduction

Consider an undirected graph $G = (V, E, w)$ with vertex set V and edge set E, where $|V| = n$ and the edge weights are given by a function $w : E \to \mathbb{R}^+$. A *matching* M in G is a subset of E such that no two edges in M have common endpoints. A matching is *maximal* if it is not properly contained in any other matching. The *weight* of a matching M is defined as $w(M) = \sum_{e \in M} w(e)$. The *maximum weight matching* (MWM) problem is then to find a matching M^* in G that maximizes $w(M)$. A matching M is a γ-approximation of M^* if $w(M) \geqslant \gamma w(M^*)$.

While there exist efficient algorithms to compute MWM in the sequential model of computation, finding a matching efficiently in the distributed model remains elusive. On the negative side, Kuhn et al. [1] proved a $\Omega(\sqrt{\log n / \log\log n} + \log\Delta / \log\log\Delta)$ lower bound on the time complexity for (possibly randomized) distributed algorithms achieving a constant factor approximation for maximum matching, even with unbounded message size.

While for general graphs distributed approximation algorithms which achieve $(1/2 - \delta)$ factor approximation have poly-logarithmic runtime [2], for some special classes of graphs, constant factor approximation has been achieved in constant or near-constant runtime. In [3] authors presented a randomized distributed 1/4-approximation algorithm on weighted trees that runs in constant time. Hoepman et al. [4] presented a $(1/2 - \delta)$ factor randomized distributed approximation algorithm for weighted matching in trees using constant time, and a deterministic algorithm for weighted trees having $\mathcal{O}(\log^* n)$ runtime and $(1/2 - \delta)$ approximation ratio. Algorithms presented in [4] can also be used to compute maximum unweighted matching on regular and almost regular graphs within a constant factor.

In this paper we show that for deterministic distributed maximum weight matching algorithms on *bounded degree bounded integer weight* graphs, it is possible to improve upon the approximation factor to $(2/3 - \delta)$, for some $0 < \delta < 2/3$, while reducing the round complexity to $\mathcal{O}(\log(\frac{1}{\delta}) + \log^* n)$.

* Corresponding author.
E-mail addresses: satyajit.banerjee@honeywell.com (S. Banerjee), atish.chowdhury@honeywell.com (A.D. Chowdhury), subhas.kumar@honeywell.com (S.K. Ghosh).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2009.03.022

2. Preliminaries

We consider the classical synchronous distributed computation under the congest model [5]. We model our communication network as a bounded degree connected undirected graph $G = (V, E, w)$, where V ($|V| = n$) is the set of computing nodes and E ($|E| = m$) is the set of communication links. We consider a restricted set of graphs such that $w_{\min}, w_{\max} \in \mathcal{Z}^+$ and $w : E \to \{w_{\min}, \ldots, w_{\max}\}$.

Given a matching M on G, a path or cycle is *alternating* if it consists of edges taken from M and $E \setminus M$ alternately. An alternating path or cycle a is said to be an *augmentation* if $M \oplus a$ is also a matching on G, where $X \oplus Y = (X \setminus Y) \cup (Y \setminus X)$. For a set of vertex-disjoint augmentations A, we define $M \oplus A$ as the resultant set obtained by repeatedly augmenting M with each augmentation in A exactly once in any order. An augmentation with at most l non-M edges is called an l-augmentation. A 2-augmentation a with respect to a given matching M is centered at some vertex $v \in V$, if the non-M edge(s) of a is (are) either incident on v or $M(v)$ where $\{v, M(v)\}$ is a matching edge in M. Note that, the center of a 2-augmentation is not unique. We denote a positive gain 2-augmentation a having v as one of its centers as a_v. Gain of an augmentation a, with respect to a matching M, is a measure of the amount by which the weight of M can be increased when augmented with a and it is defined as $g(a) \triangleq w(a \setminus M) - w(a \cap M)$. The definition can be naturally extended for the gain of a set of vertex-disjoint augmentations A. An atom of an augmentation is defined to be either a matched edge or an unmatched vertex. Thus a 2-augmentation can have at most 3 atoms. It is easy to see that a pair of 2-augmentations are vertex-disjoint if and only if they are atom-disjoint. Two augmentations a and a' are said to be *intersecting* if they share vertices; *disjoint* otherwise. In sequel we use the following useful results:

Theorem 2.1. *(See [6,7].) Let M^* be a maximum weight matching and M any matching in G.*

(1) *If M admits no l-augmentation of positive gain, then $w(M) \geqslant \frac{l}{l+1} w(M^*)$.*
(2) *There always exists a collection A of pairwise disjoint l-augmentations such that $w(M \oplus A) \geqslant w(M) + \frac{l+1}{2l+1} \times (\frac{l}{l+1} w(M^*) - w(M))$.*

3. Algorithm

Our algorithm follows an approach similar to [7]. Starting with a valid matching (in particular, $\emptyset$), it improves upon its weight in phases. Let M_i denote the matching at the end of a phase i. Let A be the set of all positive-gain 2-augmentations, at the beginning of phase i, with respect to the matching M_{i-1}. In phase i, the proposed algorithm computes a matching $M_i = M_{i-1} \oplus A'$ s.t. $g(A') \geqslant \alpha g(A^*)$ for some constant fraction α, where A', $A^* \subseteq A$ both are sets of pairwise disjoint 2-augmentations and A^* is the one with maximum gain. Thus using Theorem 2.1, we form the recurrence $w(M_i) \geqslant w(M_{i-1}) + \alpha \frac{3}{5}(\frac{2}{3} w(M^*) - w(M_{i-1}))$ which can be solved to get $w(M_i) \geqslant \frac{2}{3} w(M^*)(1 - (\frac{5-3\alpha}{5})^i)$, assuming $w(M_0) = 0$. In the rest of the document, we will use A, A' and A^* to denote 2-augmentation sets as described above – when the phase in question is clear from the context.

procedure `Distributed_Matching` (input: $G = (V, E)$)
Step 1: Compute $G^5 \triangleq (V', E')$: $V' = V$, $E' = \{\{u, v\}: d_G(u, v) \leqslant 5\}$ and compute χ_v (color of node v) by coloring G^5.
Initialize LV_v using information of 5 neighborhood of v;
for *phase* $= 1, \ldots, \Pi$ do
  Step 2: Compute A_v from LV_v;
  for *block* $= 1, \ldots, \beta$ do
    for *color* $= 1, \ldots, \chi$ do
      if (*color* $= \chi_v$)
        Step 3: If $\exists a_v \in A_v$ satisfying the *augmenting criteria*, command a_v for execution to its 9-hop neighbors;
        Step 4: Do not command till next phase;
      for each command a received (including its own) do
        Step 5: Discard a and all the a' intersecting with a from A_v. Update LV_v appropriately;
end procedure

Fig. 1. Synchronous $(\frac{2}{3} - \delta)$-approximation MWM algorithm for node v.

The detailed description of the algorithm is provided in Algorithm 1. First, we use a specific coloring scheme to color the nodes of the input graph in $\mathcal{O}(\log^* n)$ rounds using a constant number of colors, say χ (step 1). The algorithm then runs for a constant Π number of *phases*, where each phase runs over a constant β number of *blocks* and each block, in turn, runs for χ *color-rounds*. During the algorithm, each node v continuously maintains updated information about the current set of all 2-augmentations centered on all nodes within its 5-hop neighborhood, including itself. This is defined as the *local-view* of v, denoted as LV_v. Let $d_G(u, v)$ be the minimum number of hops between the vertices u and v in G. Then the local-view of v is formally defined as the current set of 2-augmentations centered at nodes u: $d_G(v, u) \leqslant 5$, i.e. in the 5-hop neighborhood of v. At the beginning of a phase, each node v identifies the set of all positive gain 2-augmentations centered at v, from its current local-view. We define this set as A_v (step 2). Clearly, $A = \bigcup_{u \in V} A_u$. A node v withdraws from participating in a phase when A_v becomes null. In each color-round k, all nodes v with color k, apply an augmentation $a \in A_v$ if a satisfies an *augmenting criteria* (vide Definition 4.1) (step 3). Whenever a commanded augmentation a is received at node v (including the one commanded by v), all other augmentations a' that are intersecting with a are immediately *discarded* from A_v (step 5). This changes the local-view at v which is appropriately updated. In each color-round, the applied augmentations are locally broadcast for 9 hops in order to maintain consistency across the local-views of each node (vide Observation 4.2) (step 3). Each phase therefore runs for $9\chi\beta$ rounds.

Our contribution lies in designing an augmenting criteria s.t. each phase runs within *constant* rounds on bounded weight bounded degree graphs, while ensuring that the set of augmentations applied, A', satisfies $g(A') \geqslant \alpha g(A^*)$, as described above. We exploit (i) the bounded integer edge-weights of the graph to arrive at a constant β and (ii) the bounded degree of the input graph to color it with a constant χ number of colors.