Simple Distributed Weighted Matchings*

Jaap-Henk Hoepman
Nijmegen Institute for Computing and Information Sciences (NIII)
Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, the Netherlands
jhh@cs.ru.nl

19th October 2004

Abstract

Wattenhofer *et al.* [WW04] derive a complicated distributed algorithm to compute a weighted matching of an arbitrary weighted graph, that is at most a factor 5 away from the maximum weighted matching of that graph. We show that a variant of the obvious sequential greedy algorithm [Pre99], that computes a weighted matching at most a factor 2 away from the maximum, is easily distributed. This yields the best known distributed approximation algorithm for this problem so far.

1 Introduction

A *matching* $M(G)$ of a graph $G = (V, E)$ is any subgraph of G where no two edges are incident to the same vertex. Let $w(e)$ be the weight of an edge $e \in E$ of G, where $w(e) > 0$. Define the weight $w(G)$ of a graph G to be the sum of the weights of all its edges. Then a *maximum weighted matching* $M^*(G)$ of G is a matching whose weight is the maximum among all matchings of G.

Surprisingly, few distributed algorithms to compute (an approximation of) the maximum weighted matching of the communication graph are known. For unweighted graphs, there are deterministic distributed algorithms computing the *maximal* matching in trees [KS00], and bipartite and general graphs [CHS02]. Randomised algorithms for the general case [II86] also exist.

For weighted graphs, Uehara *et al.* [UC00] present a constant time distributed algorithm that computes a weighted matching that is $O(\Delta)$ away

*Id: weighted-matchings.tex,v 1.3 2004/10/19 08:30:12 jhh Exp

$M(G) = \varnothing$
while $E \neq \varnothing$
do pick locally heaviest edge e from E
    add e to $M(G)$
    remove e and all edges incident to e from E
return $M(G)$

Protocol 2.1: Sequential greedy weighted matching protocol.

from the maximum (where Δ is the maximum degree of the graph). Recently, Wattenhofer *et al.* [WW04] derived a complicated randomised distributed algorithm to compute a weighted matching $M(G)$ with *approximation ratio* 5, i.e., such that $w(M(G)) > \frac{1}{5}w(M^*(G))$.

For sequential algorithms, the problem is much better studied. For unweighted graphs, Micali and Vazirani [MV80] present an $O(\sqrt{|V|}|E|)$ time algorithm that computes a maximal matching. For weighted graphs Gabow [Gab90] gives an $O(|V||E| + |V|^2 \log |V|)$ time algorithm, computing the maximum weighted matching. Both return an exact solution, and not approximations.

Recently, there have improvements in the performance of sequential algorithms to approximate the maximum weighted matching of a graph, that require much less running time than the exact algorithms.

The obvious greedy sequential algorithm (that each time adds the remaining heaviest edge) computes a weighted matching at most a factor 2 away from the maximum, in running time $O(|E| \log |V|)$ [Avi83]. Preis [Pre99] showed that selecting locally heaviest edges instead of globally heavy edges achieves the same approximation, improving the running time to $O(|E|)$. Using a path-growing algorithm, Drake *et al.* [DH03b] achieve the same running time and performance ratio.

Later, Drake *et al.* [DH03a] improved the approximation to $3/2 + \epsilon$, using a slowly converging algorithm using the concept of augmenting paths. Pettie *et al.* [PS04] present both a deterministic and a randomised algorithm achieving the same approximation in running time $O(|E| \log \frac{1}{\epsilon})$.

In this paper, we show that Preis's algorithm is easily distributed deterministically. This gives us an $O(|E|)$ time deterministic distributed algorithm that computes a weighted matching with an approximation ratio 2, the best known so far.

2 A distributed greedy algorithm

We derive a distributed variant from the sequential protocol 2.1 due to Preis [Pre99], who proved that this protocol approximates the maximum

$R := \varnothing$
$N := \Gamma(v)$
$c := \mathit{candidate}(v, N)$
if $c \neq \bot \longrightarrow$ **send** $\langle \mathit{req} \rangle$ **to** c
while $N \neq \varnothing$
do receive m **from** u
  if $m = \langle \mathit{req} \rangle \longrightarrow R := R \cup \{u\}$
  if $m = \langle \mathit{drop} \rangle \longrightarrow N := N \setminus \{u\}$
    if $u = c \longrightarrow c := \mathit{candidate}(v, N)$
      if $c \neq \bot \longrightarrow$ **send** $\langle \mathit{req} \rangle$ **to** c
  if $c \neq \bot \wedge c \in R \longrightarrow$ **forall** $w \in N \setminus \{c\}$ **send** $\langle \mathit{drop} \rangle$ **to** w
    $N := \varnothing$
{ if $c \neq \bot$ then $(v, c) \in M$ }

Protocol 2.2: Distributed greedy weighted matching protocol (node v).

matching by a factor 2.

Lemma 2.1 (Preis) *Protocol 2.1 returns for any graph G a matching $M(G)$ such that $w(M(G)) \geq \frac{1}{2} w(M^*(G))$.*

We assume an asynchronous distributed system where nodes in V can send messages to their neighbours over the communication links E. We set $G = (V, E)$. Message passing is asynchronous but reliable.

Let $\Gamma(v)$ be the set of neighbours of v in G. Define

$$\mathit{candidate}(u, N) = v \in N \text{ s.t. } \big(\forall v' \in N :: w(u, v) \geq w(u, v')\big)$$

to be the node in the set of remaining neighbours reached by the locally heaviest edge as seen from u.

In the distributed version of the greedy protocol (see protocol 2.2), each node u start with a set N equal to all its neighbours in the graph. A node sends a *request* to its current *candidate* neighbour connected to it over the locally heaviest edge (from u's point of view). This request is either granted (because the neighbour replies with a request to u as well, meaning that both see this as the locally heaviest edge), or the edge is eventually dropped (if the target node added a different edge to the matching, dropping all remaining edges from the graph). The set N maintains the set of neighbours that are still reachable by non-dropped edges. The set R contains all nodes from which requests have been received. If an edge over which a request was sent is dropped, u sends a new request to a new candidate in N.

2.1 Proof of correctness

In the proof of the protocol we assume all edge weights are unique. If they aren't, node identities can be added to break symmetry. For node v we write N_v and c_v for its local variables.

The main idea of the proof is to show that protocol 2.2 essentially simulates protocol 2.1. Define for a run of protocol 2.2 the event that two nodes u, v *match* when u receives from v a $\langle req \rangle$ message while u sent a $\langle req \rangle$ message to v before (i.e., $c_u = v$ and $c_v = u$).

Consider a run of protocol 2.2 on input G. Consider all matching events x_i in that run (as defined above), and order them in order of occurrence (x_1 being the first, x_0 is the wake up event of the algorithm). Let matching event x_i match the pair (u_i, v_i) (which adds edge $e_i = (u_i, v_i)$ to the matching). Define for event x_i the set of remaining edges E_i inductively as follows. Set $E_0 = E$, and set

$$E_i = E_{i-1} \setminus \{\text{all edges incident to } u_i \text{ and } v_i\} \ .$$

Proposition 2.2 *In protocol 2.2, each node sends at most one message over each incident edge.*

Proof: A node only sends a $\langle req \rangle$ message after it removed the previous candidate from the N. It sends a $\langle drop \rangle$ message to all remaining nodes in N (to which it didn't send a $\langle req \rangle$ yet), except for the current candidate, and then terminates by setting $N = \varnothing$. ◁

Proposition 2.3 *After x_i, and before x_{i+1} (if it occurs), if $(u, v) \in E_i$ then $u \in N_v \ \wedge \ v \in N_u$.*

Proof: The proposition holds initially. Consider the moment when a node u is removed from a set N_v. This either happens when v receives a matching $\langle req \rangle$ by some node $w = c_v$, or a $\langle drop \rangle$ from u. In the first case, a match event x_i occurs and all edges incident to v are removed from E_{i-1} to construct E_i, including (u, v). In the second case, if u sent a $\langle drop \rangle$ message, it was because of another match event x_j equal or before x_i in which all edges incident to u were removed (similar to the first case). As $E_j \supseteq E_i$, the proposition follows. ◁

Proposition 2.4 *For all i, we have $e_i \in E_{i-1}$.*

Proof: Suppose not. If $e_i = (u, v)$ is removed from E_{j-1} (to construct E_j) for some $j < i$, then a matching event (u, w) (or (v, w)) occurred removing all edges incident to u. But then $c_u = w$ remains forever, contradicting that u is involved in matching event x_i (even if $w = v$). ◁

Proposition 2.5 *Protocol 2.2 terminates for every node in the graph, with $E_t = \varnothing$ for some t.*

Proof: By proposition 2.2, a node can receive at most one $\langle req \rangle$ from each neighbour. After all those are received, each iteration of the loop removes elements from N_v. Hence, eventually $N_v = \varnothing$ and v terminates, unless v waits for receipt of a message forever in the first line of the loop. But then $N_v \neq \varnothing$ and hence $c_v = u \neq \perp$ for some u. This means a $\langle req \rangle$ message was sent to u. Then either $v \in N_u$, or a $\langle drop \rangle$ message is in transit to v (contradicting that v waits forever for a new message). But if $v \in N_u$ it will either become a candidate for u (in which case u sends $\langle req \rangle$ to v), or u finds another candidate, sending a $\langle drop \rangle$ to all remaining nodes in N_u including v.

To show that for some t we have $E_t = \varnothing$, consider the moment all nodes have terminated. Then for all v we have $N_v = \varnothing$. By propostion 2.3 the proposition follows. ◁

Proposition 2.6 *Matching edge e_i is a locally heaviest edge in E_{i-1}.*

Proof: Let $e_i = (u, v)$. By proposition 2.4 $e_i \in E_{i-1}$. To see that this is also the locally heaviest edge in E_{i-1}, suppose an edge $(u, w) \in E_{i-1}$ is heavier. Then $w \in N_u$ by proposition 2.3, but then $c_u = w$ instead. ◁

Theorem 2.7 *Protocol 2.2 computes for any graph $G = (V, E)$ a matching $M(G)$ such that $w(M(G)) \leq \frac{1}{2} w(M^*(G))$ in time $O(|E|)$.*

Proof: We first show that if protocol 2.2 computes a matching $M(G)$, then there is a run of protocol 2.1 that returns the same matching. Consider a run of protocol 2.2 on input G. Let x_i be the ordered sequence of matching events in that run as defined above.

Now consider the sequential algorithm 2.1. Define $E'_0 = E$, and let E'_i be the set of remaining edges in the graph after adding the i-th edge e'_i to the matching and removing the incident edges. Clearly $E'_0 = E_0$. A simple inductive argument shows that $E'_i = E_i$ for all i, if we let protocol 2.1 select edge $e'_i = e_i$ (by proposition 2.6 and the induction hypothesis this is a locally heaviest edge and therefore a possible selection).

We conclude that the sequential algorithm adds the same edges to the matching as the distributed algorithm in this run. According to proposition 2.5, for some t we have $E_t = \varnothing$. Then also $E'_t = \varnothing$ so the sequential algorithm doesn't add any more edges. The bound on the approximation follows from Lemma 2.1. The time complexity follows from proposition 2.2. ◁

3 Conclusions

We have described a distributed algorithm that computes a $\frac{1}{2}$ approximation of the maximum matching of a weighted graph in $O(|E|)$ time, based on a sequential algorithm achieving the same approximation.

Other sequential algorithms, that improve the approximation to $3/2 + \epsilon$ are known [DH03a, PS04]. It is an open question whether these algorithms can also be distributed, and if so, at which cost in terms of running time.

References

[Avi83] Avis, D. A survey of heuristics for the weighted matching problem. *Networks* **13** (1983), 475–493.

[CHS02] Chattopadhyay, S., Higham, L., and Seyffarth, K. Dynamic and self-stabilizing distributed matching. In *21st PODC* (Monterey, CA, USA, 2002), ACM Press, pp. 290–297.

[DH03a] Drake, D., and Hougardy, S. Improved linear time approximation algrotihms for weighted matchings. In *2764 7th Int. Workshop on Randomization and Approximation Techniques in Computer Science (APPROX)* (2003), no. 2764 in LNCS, pp. 14–23.

[DH03b] Drake, D., and Hougardy, S. A simple approximation algorithm for the weighted matching problem. *Inf. Proc. Letters* **85** (2003), 211–213.

[Gab90] Gabow, H. Data structures for weighted matching and nearest common ancestors with linking. In *1th SODA* (San Fransisco, Ca., USA, 1990), ACM, pp. 434–443.

[II86] Israeli, A., and Itai, A. A fast and simple randomized parallel algorithm for maximal matching. *Inf. Proc. Letters* **22** (1986), 77–80.

[KS00] Karaata, M., and Saleh, K. A distributed self-stabilizing algorithm for finding maximal matching. *Computer Systems Science and Engineering* **3** (2000), 175–180.

[MV80] Micali, S., and Vazirani, V. An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs. In *21nd FOCS* (??, 1980), IEEE Comp. Soc. Press, pp. 17–27.

[PS04] Pettie, S., and Sanders, P. A simple linear time $2/3-\epsilon$ approximation for maximum weight matching. Tech. Rep. MPI-I-2004-1-002, Max-Planck-Institut für Informatik, Saarbrücken, germany, 2004.

[Pre99] Preis, R. Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs. In *16th STACS* (Trier, Germany, 1999), C. Meinel and S. Tison (Eds.), LNCS 1563, Springer, pp. 259–269.

[UC00] Uehara, R., and Chen, Z. Parallel approximation algorithms for maximum weighted matching in general graphs. *Inf. Proc. Letters* **76** (2000), 13–17.

[WW04] Wattenhofer, M., and Wattenhofer, R. Distributed weighted matching. In *18th DISC* (Amsterdam, the Netherlands, 2004), R. Guerraoui (Ed.), LNCS 3274, Springer, pp. 335–348.