A pseudo-polylog average time parallel maxflow algorithm

J.F. Sibeyn

RUU-CS-90-17
April 1990

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

A pseudo-polylog average time parallel maxflow algorithm

J.F. Sibeyn

Technical Report RUU-CS-90-17
April 1990

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN:0924-3275

A Pseudo-Polylog Average Time Parallel Maxflow Algorithm

Jop F. Sibeyn *
Department of Computer Science, University of Utrecht
P.O. Box 80.089, 3508 TB Utrecht , the Netherlands

April 19, 1990

Abstract

We present a novel algorithm for the integer maxflow problem. On a PRAM CREW with n^2 processing units we find over a number of common models for random graphs an average calculation time $T_{av} = \mathcal{O}((\log C + \log^4 n) \cdot \log n / \log(m/n))$. Here C is the average edge capacity and m the (expected) number of edges. If for a random graph $m \geq n \cdot \log^3 n$ and that the maximal capacity M satisfies $M \leq (m/n)^{1/5} \cdot C$. We will show that the capacity condition does not really limit the applicability of the algorithm.

1 Introduction

In this paper we will present an algorithm for finding the maxflow of a directed graph with integer capacities. We start with an introduction in which the problem is introduced and where a sketch of the algorithm is given. After this the algorithm is worked out in a top-down way. At first the results will be proved as average over a suitable class of graphs. This will be generalized in section 3. At the end of the text we give some appendices containing back-ground theorems.

The parallel machine we will use is an PRAM CREW consisting of n^2 Processing-Units (c.f. [10]). We will give an algorithm which solves the (integer) maxflow problem (MFP). Mostly we will use small Greek letters for constants; small italic letters for variables; boldface type-style for matrices; and calligraphic letters for classes. The graphs we consider are directed and have vertices numbered from $1 = s$ to $n = t$. Graphs are identified with their adjacency matrix.

1.1 The maxflow problem

For a good introduction to the MFP and its terminology we refer to Papadimitriou [8].

A preflow $\mathbf{f}$ in a graph $\mathbf{c}$ is an assignment of values (edge-flows) to the edges of $\mathbf{c}$ such that $0 \leq f_{ij} \leq c_{ij}\, \forall i,j$ (capacity restriction). A flow is a preflow which satisfies $\sum_{j=0}^{n}(f_{ji} - f_{ij}) = 0 \forall 1 < i < n$ (flow conservation). A flow $\mathbf{f}$ in $\mathbf{c}$ is a maxflow of $\mathbf{c}$ if $\mathbf{f}$ cannot be augmented without violating the capacity restriction. An s-t cut (V_s, V_t) is a partition of the set of vertices into two subsets $V_s \ni s, V_t \ni t$. The capacity of an s-t cut (V_s, V_t) for a preflow $\mathbf{f}$ is given by $F_{\mathbf{f}}(V_s, V_t) = \sum_{\{i \in V_s, j \in V_t\}} f_{ij}$. The capacity of the minimum cut of $\mathbf{f}$ will be denoted by $F(\mathbf{f})$. The maxflow-mincut theorem (See [8, th 6.2 p 119]) states: $maxflow(\mathbf{c}) = F(\mathbf{c})$. This gives

Theorem 1 *A flow $\mathbf{f}$ in $\mathbf{c}$ is a maxflow iff $F_{\mathbf{f}}(V_s, V_t) = F_{\mathbf{c}}(V_s, V_t)$ for some cut (V_s, V_t).*

The importance of the MFP is, apart from its practical value, given by the fact that it is $\mathcal{P}$-complete [5]. Given the presumed impossibility to solve MFP in parallel in polylog time order, it comes about naturally to try to design a parallel polylog average time algorithm. We come rather far in achieving this.

*The work of the author was financially supported by the Foundation for Computer Science (SION) of the Netherlands Organization for Scientific Research (NWO). This research was partially supported be the ESPRIT II Basic Research Actions Project of the EC under contract No. 3075 (Project ALCOM).

1.2 Approach and results

Basic observations underlying the construction of the algorithm are

- Almost always the mincut of a graph is situated right behind the source s, or just before the terminal t (theorem 2).
- Almost always the preflow **c** is almost balanced (6).

With $\mathit{outflow}_i = \sum_j f_{ij}$ the flow out of i and $\mathit{inflow}_j = \sum_i f_{ij}$ the flow into j, this suggests the following simple maxflow algorithm:

1. Start by saturating all edges of the graph. This gives no flow in general but we will not be far away from it.
2. Determine $\min\{\mathit{outflow}_s, \mathit{inflow}_t\}$, generally this is the value of the maxflow of **c**.
3. If $\min\{\mathit{outflow}_s, \mathit{inflow}_t\} = \mathit{outflow}_s$ then reduce inflow_t to $\mathit{outflow}_s$ else analogously.
4. Try to balance **f** by reducing flows on edges not adjacent to s or t. If this is successful and the mincut is situated at s or t then we have found a maxflow.
5. If the balancing of **f** has failed then find the maxflow of **c** with a polynomial worst-case time algorithm.

The first step seems to be original. The concept of a preflow is used in a number of algorithms (cf. [4, 7, 11]) but only in the restricted sense that in each vertex the outgoing flow is at most equal to the incoming flow, i.e. a vertex may have a flow excess but never a flow deficit. In our algorithm we use preflows in which vertices may have flow deficits. Another important point is that step 4 parallellizes extremely well. Once we have proved that step 4 requires pseudo-polylog time and is successful almost always, the average case time will follow easily.

Definition 1 *$A(n)$ happens almost always (aa) if for some function ω, $\lim_{n\to\infty} \omega(n) = \infty$,*

$$P(\neg A(n)) < n^{-\omega(n)} \tag{1}$$

A pair (X, p), where X is a positive distribution with mean value $C(X)$ is called acceptable if

$$X < (p \cdot n)^{1/5} \cdot C(X) \ aa;\ p \cdot n \geq \log^3 n \tag{2}$$

An algorithm is pseudo-polylog for a problem A if it solves any instance I of A in time bounded by a polynomial in $\log |I|$ and $\log \mathit{number}(I)$.

Pseudo-polylog is defined analogously to pseudo-polynomial [8]. $|I|$ is the size of the instance and $\mathit{number}(I)$ the largest integer appearing in I. In section 2 we will prove: If (X, p) is an acceptable pair then $T_{\mathrm{av}}(n, p, C) = \mathcal{O}((\log C + \log^4 n) \cdot \log n / \log(n \cdot p))$. Because of the term "$\log C$" this is only pseudo-polylog. The acceptability of (X, p) implies that the graphs should not be too sparse (actually, the diameter should not be too large and there should be a reasonable number of edges at every vertex) and that it is extremely rare that an edge has a capacity considerably larger than the average. We will show in section B.1 that N_C, the normal distribution with variance $\pi \cdot C^2/2$ is among the many distributions giving an acceptable pair as soon as (2) is satisfied. From the results we have put together in the following table, we conclude that if our algorithm is applicable then for polynomial C it is best. m is the (expected) number of edges.

algorithm	number of PUs	time order	type
Karzanov	1	n^3	worst case
Sleator	1	$n \cdot m \cdot \log n$	worst case
Shiloach & Vishkin	n	$n^2 \cdot \log n$	worst case
Johnson	n^4	$\log^3 n$	worst case, planar graphs
Gabow	1	$n \cdot m \cdot \log C$	worst case
Sibeyn	n^2	$\log n / \log(m/n) \cdot (\log C + \log^4 n)$	average case

1.3 Graphs and properties

As notation we use: $\tilde{n} = (n-1)\cdot p, N = n\cdot(n-1), \tilde{N} = N\cdot p$ and $indeg_i$, $outdeg_i$ for the in and out-degree of i. We work with the class $\mathcal{G}(n,p,X)$ of directed graphs. An arbitrary element $\mathbf{c} \in \mathcal{G}(n,p,X)$ has n vertices and edges $(1 \le i < n, 1 < j \ne i \le n)$ which occur with probability p. The capacities of the edges are chosen according to some positive discrete probability distribution X. If ξ is the density function of X then the expectation $C(X)$ of X is $C(X) = \sum_1^\infty x\cdot\xi(x)$. In section 3.1 we show that the results hold for other graph classes as well. The probability that a graph $\mathbf{c}$ with m edges occurs in $\mathcal{G}(n,p,X)$ is $P_{\mathcal{G}}(\mathbf{c}) = \prod_{k=1}^m \xi(c_{ij(k)})\cdot p^m\cdot(1-p)^{N-m}$. Here ij is the function which enumerates the non-zero edges. Occasionally a preflow $\mathbf{f}$ will be viewed as the graph with edge capacities f_{ij}. This identification is implicit when we write $\mathbf{f} \in \mathcal{H}$, where $\mathcal{H}$ is a subclass of $\mathcal{G}(n,p,X)$ with probability distribution $P_{\mathcal{H}}$ induced by $P_{\mathcal{G}}$. Thus $\mathbf{f} \in \mathcal{H}$ also means that the probability on $\mathbf{f}$ is $P_{\mathcal{H}}(\mathbf{f})$. (27) and theorem 8 give

Lemma 1 *If $\tilde{n} > \log^3 n$ then for all $\epsilon > 0$ aa for all i*

$$|indeg_i - \tilde{n}| < \epsilon\cdot\tilde{n}^{7/10},\ |outdeg_i - \tilde{n}| < \epsilon\cdot\tilde{n}^{7/10} \tag{3}$$

From now on (X,p) is assumed to be an acceptable pair. As notation we use: $b_i = inflow_i - outflow_i$, the imbalance of $\mathbf{f}$ in i; $b_{\max} = \max_i\{|b_i|\}$, the maximal imbalance; $B = \sum_i |b_i|$, the total imbalance; and $G = \sum_{i,j} f_{ij}$, the total flow. A vertex i is called "balanced" if $b_i(\mathbf{f}) = 0$. With (29) we find

Lemma 2 *For all ϵ aa for all i (where this applies)*

$$|inflow_i - \tilde{n}\cdot C| < \epsilon\cdot C\cdot\tilde{n}^{9/10},\ |outflow_i - \tilde{n}\cdot C| < \epsilon\cdot C\cdot\tilde{n}^{9/10} \tag{4}$$

$$b_{\max} < \epsilon\cdot C\cdot\tilde{n}^{9/10},\ B < \epsilon\cdot C\cdot n\cdot\tilde{n}^{9/10} \tag{5}$$

$$|G - \tilde{N}\cdot C| < \epsilon\cdot C\cdot\tilde{N}^{9/10}$$

We see that $B(\mathbf{c})$ is of smaller order than $G(\mathbf{c})$:

$$B(\mathbf{c})/G(\mathbf{c}) < \frac{n\cdot\tilde{n}^{9/10}\cdot C}{\tilde{N}\cdot C - \tilde{N}^{9/10}\cdot C} < 2/\tilde{n}^{1/10} < 2/\log^{3/10} n \text{ aa} \tag{6}$$

Theorem 2 *$F(\mathbf{c}) = \min\{outflow_s, inflow_t\}$ aa*

Proof: We show that in the minimal s-t cut, (V_s, V_t), $|V_s| = 1$ or $n-1$. If $|V_s| = 2$ then $V_s = \{s,v\}$ for some v and we find: $F(V_s,V_t) \ge outflow_s + outflow_v - |c_{sv}| - |c_{vs}| > 2\cdot\tilde{n}\cdot C - 2\cdot C\cdot\tilde{n}^{9/10} - 2\cdot M > \tilde{n}\cdot C + C\cdot\tilde{n}^{9/10} > outflow_s$ aa. If $3 \le |V_s| \le n/2$ then we get using (29) with $n' = |V_t|$: $F(V_s,V_t) \ge |V_s|\cdot(p\cdot n'\cdot C - (p\cdot n')^{9/10}\cdot C) \ge 3/2\cdot(\tilde{n}\cdot C - \tilde{n}^{9/10}\cdot C) > \tilde{n}\cdot C + C\cdot\tilde{n}^{9/10} > outflow_s$ aa. The cases $n/2 \le |V_s| \le n-2$ are analogous. ∎

From Bollobás [1] we find for $diam(\mathbf{c})$, the diameter of a graph $\mathbf{c}$

Theorem 3 *If $\mathbf{c} \in \mathcal{G}(n,p)$ then $diam(\mathbf{c}) \le \lceil \log n/\log\tilde{n}\rceil$ aa*

2 The maxflow algorithm

The sketch of the algorithm given in section 1.2 is worked out in figure 1. The algorithm of Balance is given in section 2.1. The internal vertices are the vertices $i \notin \{s,t\}$. Internal edges, (edge) flows, etc, are incident on internal vertices only. With the trivial observation that internal reductions do not reduce $F(s, V-\{s\})$ and using theorem 1 we conclude

Corollary 1 *If the mincut of $\mathbf{c}$ is situated right behind s and $\mathbf{c}$ can be reduced by internal reductions to a flow $\mathbf{f}$ then $\mathbf{f}$ is a maxflow of $\mathbf{c}$.*

Proc MaxFlow$(n, p, C, \mathbf{c}, \mathbf{f})$;
Proc InitFlow$(n, \mathbf{c}, \mathbf{f})$;
($\mathbf{f} := \mathbf{c}$;
determine $outflow_s(\mathbf{f})$, $inflow_t(\mathbf{f})$;
if $inflow_t(\mathbf{f}) > outflow_s(\mathbf{f})$ **then**
for all i **do** $f_{in} := \lceil f_{in} \cdot outflow_s(\mathbf{f}) / inflow_t(\mathbf{f}) \rceil$ **od** ;
determine $inflow_t(\mathbf{f})$;
decrease by 1 the flow on $inflow_t(\mathbf{f}) - outflow_s(\mathbf{f})$ in-edges of n
else operate analogously **fi**) ;
Proc Balance$(\mathbf{f}, n, p, C)$;
(Under condition $\mathbf{f} \in \mathcal{G}(n, p, X)$ with $outflow_s(\mathbf{f}) = inflow_t(\mathbf{f})$ $\mathbf{f}$ is balanced aa
in $\mathcal{O}(\log n / \log \tilde{n} \cdot (\log C + \log^4 n))$ time, by internal reductions only.) ;
Proc StandardMaxFlow$(n, \mathbf{c}, \mathbf{f})$;
(Finds the maxflow of $\mathbf{c}$ in $\mathbf{f}$ in polynomial time.) ;
(InitFlow$(n, \mathbf{c}, \mathbf{f})$ { Now $F(\mathbf{f}) = F(\mathbf{c})$ and $outflow_s = inflow_t$ } ;
Balance$(\mathbf{f}, n, p, C)$ { Now $B(\mathbf{f}) = 0$ aa } ;
if $B(\mathbf{f}) > 0$ **then** StandardMaxFlow$(n, \mathbf{c}, \mathbf{f})$ **fi** { Now $\mathbf{f}$ is a maxflow. }) ;

Figure 1: The maxflow algorithm.

Lemma 3 *Proc InitFlow needs* $\mathcal{O}(\log n)$ *time to make* $\mathbf{f}$ *a preflow satisfying*

$$F(\mathbf{f}) = F(\mathbf{c}) \tag{7}$$
$$outflow_s(\mathbf{f}) = inflow_t(\mathbf{f}) \tag{8}$$
$$f_{ij} = c_{ij} \text{ for all internal edges} \tag{9}$$

(8, 9) make that after InitFlow $\mathbf{f} \in \mathcal{H} = \{\mathbf{g} \in \mathcal{G}(n, p, X) | outflow_s(\mathbf{g}) = inflow_t(\mathbf{g})\}$ in the sense it was introduced in section 1.3. Informally we will say: "$\mathbf{f} \in \mathcal{G}(n, p, X)$ satisfying (8)". (The expected $outflow_s(\mathbf{f})$ is slightly too small for having $\mathbf{f} \in \mathcal{H}$. However, this only increases the probability that $\mathbf{f}$ can be balanced). Observe that $\mathbf{f} \in \mathcal{H}$ only holds when excess flow out of s (resp. a flow deficit into t) is eliminated by reducing the flow in edges starting in s (resp. ending in t) proportionally to their capacities.

Lemma 4 *If* $\mathbf{c} \in \mathcal{G}(n, p, X)$ *and* $\mathbf{f}$ *is a preflow satisfying (7, 8, 9) then aa* $\mathbf{f}$ *is transformed by Proc Balance to a maxflow of* $\mathbf{c}$.

Proof: According to theorem 2 the mincut of $\mathbf{f}$ is situated right behind s aa. The specification of Balance gives that $\mathbf{f}$ will be balanced aa and then corollary 1 gives that this is a maxflow of $\mathbf{f}$ and thus of $\mathbf{c}$. ∎
Combining these lemmas with theorem 9 we obtain our main theorem:

Theorem 4 *If* (X, p) *is an acceptable pair then MaxFlow as given in figure 1 finds the maxflow of* $\mathbf{c} \in \mathcal{G}(n, p, X)$ *in pseudo-polylog time aa. Furthermore* $T_{\mathcal{G}(n,p,X)} = \mathcal{O}(n^2 \cdot \log n)$ *and*

$$T_{\text{av}, \mathcal{G}(n,p,X)} = \mathcal{O}(\log n / \log \tilde{n} \cdot (\log C + \log^4 n)) \tag{10}$$

2.1 Balance

We have seen that the only really important part in the algorithm is the balancing of a preflow satisfying (8) by internal reductions in pseudo-polylog time aa. We observe two guidelines for the reductions:To assure good parallellism reductions are always directed by a vertex; to make small the probability that due to the reductions the flow in an edge is reduced to zero, reductions are made proportional to the flow on the edge as far as possible. Define $A^- = \#\{i | b_i < 0\}, A^+ =$

$\#\{i|b_i > 0\}, A = A^- + A^+$. The algorithm consists of calls of four subprocedures which reduce $B(\mathbf{f})$ from its initial value given by (5) to 0 aa:

1. call PreBal($\mathbf{f}, n$) $\lfloor 11 \cdot \log C / \log \tilde{n} \rfloor$ times { $b_{\max}(\mathbf{f})$ was given by (5), now $b_{\max}(\mathbf{f}) < \tilde{n}^{7/10}$ aa. }
2. call InterBal($\mathbf{f}, n$) $\lfloor 7 \cdot \log\log \tilde{n} \rfloor$ times { After these calls $b_{\max}(\mathbf{f}) < \log^{4/3} n$ aa. }
3. call NearBal($t, \mathbf{f}, n$) for t increasing from 1 as long as $\log^{4/3} n < \min\{A^-(\mathbf{f}), A^+(\mathbf{f})\} < n \cdot (2/\tilde{n})^{t-1}$ { This ends with $\min\{A^-(\mathbf{f}), A^+(\mathbf{f})\} \leq \log^{4/3} n$ aa, and thus (b_i only decreases) $B < 2 \cdot \log^{8/3} n$. }
4. If $B(\mathbf{f}) < 2 \cdot \log^{8/3} n$ then call CompleteBal($\mathbf{f}, n, p$) $B(\mathbf{f})/2$ times { Aa this balances $\mathbf{f}$ if the condition is satisfied. }

PreBal and InterBal make large steps and reduce B very fast. However, they are too coarse to finish the task of eliminating all imbalance. Therefore, we proceed with NearBal and CompleteBal. For the time being we do not pay attention to dependencies between the calls within a subprocedure or between the subprocedures. This will be justified in section 2.5.

We derive properties of the preflow $\mathbf{f}$ after the calls of the subprocedures, they will imply the correctness of the specification of Balance in figure 1. Without further notice we will give results that only hold asymptotically. Furthermore we assume that $n = o(C)$. Application of the subprocedures imposes conditions on in and out-flows, in and out-degrees and the diameter. We should prove they are satisfied by induction. To keep the line of reasoning clear, this is done separately in section 2.2.

If a recurrence relation x_t is given by $x_0 = x; x_{t+1} = x_t/\zeta + \eta/2$ and $\zeta > 3$ then $x_t < \eta \; \forall t \geq \lceil (\log x + 2 - \log \eta)/\log \zeta \rceil$. Using (5) and lemma 10 this gives

Lemma 5 *After $\lfloor 11 \cdot \log C / \log \tilde{n} \rfloor$ calls of PreBal*

$$b_{\max}(\mathbf{f}) < \tilde{n}^{7/10} \; aa \tag{11}$$

The total time consumption during this first reduction phase amounts to $\mathcal{O}(\log n \cdot \log C / \log \tilde{n})$

If a recurrence relation x_t is given by $x_0 = x; x_{t+1} = x_t^\zeta$ and $0 < \zeta < 1$, then : $x_t \leq 2 \; \forall t \geq \lceil \log\log x / \log(1/\zeta) \rceil$. Using this, lemma 5 and lemma 11 we get

Lemma 6 *A preflow satisfying (11) is transformed in $\lfloor 7 \cdot \log\log \tilde{n} \rfloor$ calls of InterBal to a preflow $\mathbf{f}$ for which*

$$b_{\max}(\mathbf{f}) < \log^{4/3} n \; aa \tag{12}$$

The total cost of these calls is $\mathcal{O}(\log n \cdot \log\log \tilde{n})$ time.

This lemma makes that after the calls of InterBal $\mathbf{f}$ satisfies the conditions for application of NearBal(1) (apart from the condition on the degrees) aa. We proceed by induction. Lemma 13 gives that if $\mathbf{f}$ satisfies the conditions for application of NearBal(d) then it satisfies afterwards the conditions for NearBal($d+1$) aa. So aa we come to the point that we have (12) and

$$\min\{A^-(\mathbf{f}), A^+(\mathbf{f})\} \leq \log^{4/3} n \tag{13}$$

Lemma 7 *A preflow satisfying (12) is transformed by calling NearBal(t) for increasing $t \in \{1, \ldots, \lfloor 2 \cdot \log n / \log \tilde{n} \rfloor\}$ into a preflow $\mathbf{f}$ for which*

$$B(\mathbf{f}) < 2 \cdot \log^{8/3} n \tag{14}$$

These calls cost $\mathcal{O}(\log^{13/3} n / \log \tilde{n})$ time in total.

Proof: It is easy to check that with $d = \lfloor 2 \cdot \log n / \log \tilde{n} \rfloor$ we have $n \cdot (2/\tilde{n})^d < 1 < \log^{4/3} n$. Hence, after calls of NearBal(t) for $t = 1, 2, \ldots, d_0$ for some $d_0 \leq d$, $\min\{A^-, A^+\}$ must have come below $\log^{4/3} n$. (14) follows from (13, 12) and $B \leq 2 \cdot \min\{A^-, A^+\} \cdot b_{\max}$ (this relation follows from $B^- = B^+$). The time consumption follows from lemma 13. ∎

Now lemma 14 gives

Lemma 8 *A preflow satisfying (14) can be transformed in at most* $\lfloor \log^{8/3} n \rfloor$ *calls of CompleteBal to a flow* $\mathbf{f}$ *aa. The time consumption of these calls is* $\mathcal{O}(\log^{14/3} n / \log \tilde{n})$ *all together.*

From the lemmas on the subprocedures we get the important result on Balance:

Theorem 5 *If* (X, p) *is an acceptable pair and* $\mathbf{f} \in \mathcal{G}(\mathbf{n}, \mathbf{p}, \mathbf{X})$ *satisfying (8) then* $\mathbf{f}$ *will be balanced by Balance aa. Balance operates on internal edges only. For the time consumption we find* $T_{\text{Balance}} = \mathcal{O}(\log n / \log \tilde{n} \cdot (\log C + \log^{10/3} n))$.

2.2 Flow reductions while executing Balance

Our balancing procedures might behave unproperly when the amount of flow coming into (or going out of) a vertex becomes too small in some sense. This might also happen when too few edges contribute to the incoming (or outgoing) flow of a vertex. In this section we will prove that this undesirable situation occurs only rarely. A similar result holds for the diameter of the graph (of edges with a non-zero flow): its diameter should not become too large.

Lemma 9 *At all times during Balance for all internal* i

$$\mathit{inflow}_i(\mathbf{f}) > 2/3 \cdot \tilde{n} \cdot C, \ \mathit{outflow}_i(\mathbf{f}) > 2/3 \cdot \tilde{n} \cdot C \ \mathit{aa} \tag{15}$$

Proof: The reductions of in and out-flow due to a call of PreBal, ..., CompleteBal are given by lemma 10, lemma 11, lemma 13 and lemma 14. The reduction of $b_{\max}$ is so fast that the reduction of in and out-flow due to all calls of PreBal and InterBal can easily be estimated on twice the reduction during the first call. If, as we assume, $n = o(C)$ then the reductions due to the calls of NearBal and CompleteBal are of smaller order. Combining (5, 20) we find that the reduction during the first call of PreBal is bounded by $2 \cdot C \cdot \tilde{n}^{9/10}$. Using (4) we can finish the proof. ∎

Corollary 2 *At all times during Balance for all internal* i

$$\mathit{indeg}_i(\mathbf{f}) > \tilde{n}/2, \ \mathit{outdeg}_i(\mathbf{f}) > \tilde{n}/2 \ \mathit{aa} \tag{16}$$

Proof: We only check the first part. Let $\mathit{flowdegratio} = \frac{\mathit{inflow}_i}{\mathit{indeg}_i}$. Then we have for $\mathbf{c}$ $\mathit{flowdegratio}_i(\mathbf{c}) < \frac{C \cdot (\tilde{n} + \tilde{n}^{9/10}/3)}{\tilde{n} - \tilde{n}^{7/10}} < C + C/(2 \cdot \tilde{n}^{1/10})$. If by the action of Balance the reduction of flows on edges would be exactly proportional to those flows then this ratio would decrease. Actually this is not quite true but, as we have seen, the great bunch of flow reduction is performed by PreBal and PreBal reduces flows on edges approximately proportional to those flows. So the ratio decreases or eventually increases slightly. In any case we will have $\mathit{flowdegratio}_i(\mathbf{f}) < C + C/\tilde{n}^{1/10}$ where $\mathbf{f}$ is the preflow we get after application of Balance. With (15) this gives $\mathit{indeg}_i(\mathbf{f}) = \frac{\mathit{inflow}_i(\mathbf{f})}{\mathit{flowdegratio}_i(\mathbf{f})} > \frac{2/3 \cdot \tilde{n} \cdot C}{(C + C/\tilde{n}^{1/10})} > \tilde{n}/2$. ∎
Not taking into account dependency we find from (16) and theorem 3

Corollary 3 *At all times during Balance* $\mathit{diam}(\mathbf{f}) \leq \lceil \log n / (\log \tilde{n} - 1) \rceil$ *aa.*

2.3 PreBal and InterBal

In this section the effect of PreBal and InterBal on $b_{\max}$ for a preflow $\mathbf{f}$ is analysed. PreBal reduces $b_{\max}(\mathbf{f})$ considerably if $\mathbf{f}$ satisfies (15) and $b_{\max}(\mathbf{f}) > \tilde{n}^{7/10}$. InterBal requires (16) and

$$\tilde{n}/2 > b_{\max}(\mathbf{f}) > \log^{4/3} n \tag{17}$$

First we treat PreBal. The algorithm returns excess in-flow over the in-edges in proportion to the flow over these edges and for excess out-flow analogously. In the proof of corollary 2 we have seen the use of this proportional reduction. We use help variables $\delta_{i;ij}$, the change i wants to make to (i, j) and $\delta_{j;ij}$ the change j wants to make to (j, i), to make the actions of the PUs completely independent. The algorithm is given by

Proc PreBal($\mathbf{f}, n$);
for all $1 < i < n$ **do**
if $b_i(\mathbf{f}) \geq 0$
then for all $1 < j < n$ **do** $\delta_{i;ij} := 0; \delta_{i;ji} := round(+b_i(\mathbf{f}) \cdot f_{ji}/inflow_{\text{int},\,i})$ **od**
else for all $1 < j < n$ **do** $\delta_{i;ji} := 0; \delta_{i;ij} := round(-b_i(\mathbf{f}) \cdot f_{ji}/outflow_{\text{int},\,i})$ **od fi od** ;
for all $1 < i, j < n$ **do**
$f_{ij} := \max\{f_{ij} - \delta_{i;ij} - \delta_{j;ij}, 0\}$ **od**) .

Call the preflow we obtain after application of PreBal $\mathbf{f}'$. We will express $b_{\max}(\mathbf{f}')$ in terms of $b_{\max}(\mathbf{f})$. We will do this by giving an upper bound on $b_i(\mathbf{f}')$ for an i for which both $b_i(\mathbf{f})$ and $b_i(\mathbf{f}')$ are positive, the other cases can be treated in the same way. There are three sources for $b_i(\mathbf{f}')$, they will be denoted by $b_i^{(1)}, b_i^{(2)}, b_i^{(3)}$. $b_i^{(1)}$ is the difference between the effect of the rounding down and the rounding up of the $\delta_{i;ji}$. Using (30) with $M = 0.5$ we find $b_i^{(1)} < \epsilon \cdot \tilde{n}^{7/10}$ for all $\epsilon > 0$. $b_i^{(2)}$ is due to the difference in reductions made on the in and out-edges by the neighbors. If for some j $b_j > 0$ then we can bound $\delta_{j;ij}$, the contribution made by j to the balance of i by

$$\delta_{j;ij} = b_j \cdot f_{ij}/inflow_{\text{int},\,j} < 2 \cdot b_{\max} \cdot M/(\tilde{n} \cdot C) < 2 \cdot b_{\max}/\tilde{n}^{4/5} \tag{18}$$

Here (15) and the relation between M and C are essential. The same bound holds for the contributions $\delta_{j;ji}$. Application of (30) with $M = 2 \cdot b_{\max}/\tilde{n}^{4/5}$ gives $b_i^{(2)} < \epsilon \cdot b_{\max}/\tilde{n}^{1/10}$ for all $\epsilon > 0$. The factor "$b_{\max}(\mathbf{f})$" occurring here is the origin of the term "$\log C$" in the time order of Balance and MaxFlow. $b_i^{(3)}$ stems from a possible insufficiency of f_{ij}. It can be only non-zero when at least one of the adjacent vertices i, j wants to reduce more than half of its in or out-flow. The first case occurs only when $inflow_i > 2 \cdot outflow_{\text{int},\,i}$. This will almost never happen, and thus $b_i^{(3)} = 0$ aa.

Lemma 10 $\mathbf{f} \in \mathcal{G}(n, p, X)$ *satisfying (15) is transformed by PreBal in* $\mathcal{O}(\log n)$ *time to an* $\mathbf{f}'$ *for which*

$$b_{\max}(\mathbf{f}') < \epsilon \cdot (\tilde{n}^{7/10} + b_{\max}(\mathbf{f})/\tilde{n}^{1/10}) \text{ for all } \epsilon > 0 \text{ aa} \tag{19}$$

$$inflow_i(\mathbf{f}') - inflow_i(\mathbf{f}') < 2 \cdot b_{\max}(\mathbf{f}),\ outflow_i(\mathbf{f}') - outflow_i(\mathbf{f}') < 2 \cdot b_{\max}(\mathbf{f}) \text{ aa} \tag{20}$$

Proof: The reduction of $inflow_i(\mathbf{f})$ consists of $inred_i^{(1)}$ the reduction i performs on its own in-flow and $inred_i^{(2)}$ the reduction caused by the vertices adjacent to i. Depending on the sign of b_i $inred_i^{(1)}$ is 0 or bounded by $b_{\max}$. $inred_i^{(2)} = \sum_j \delta_{j;ji}$. We use (29) to give an estimate of this. For the value of p in (29) we substitute $p/2$. From $inred_{\text{av}}^{(2)} = \sum_i inred_i^{(2)}/n = B^+/n \leq b_{\max}/2$ we conclude that we can use $b_{\max}/\tilde{n}$ for the average contribution C of (29). (18) gives us the value for M. So we get $inred_i^{(2)} < b_{\max}/2 + b_{\max}/\tilde{n}^{1/10} < b_{\max}$. ∎

Now we will focus on InterBal. The algorithm and its analysis are analogous to those for PreBal. The algorithm is expressed by

1. if $b_i > 0$ then select from the in-edges a subset S_i of size b_i else select from the out-edges a subset of size $|b_i|$ { This can be done if (16) and (17) hold. Take for S_i the first $|b_i|$ vertices with indices larger (cyclical) than i satisfying the condition. }
2. reduce the flow on the selected edges by 1 { Exactly b_i is returned or fetched back in this way. }

There are other good choices for S_i e.g. random. Any choice is good as long as no vertex occurs on the average more often as endpoint of a selected edge than any other. Let $\mathbf{f}'$ be the preflow obtained after application of InterBal to a preflow $\mathbf{f}$. The contribution $b_i^{(1)}$ to $b_i(\mathbf{f}')$ is 0 for InterBal. So aa we will have $b_i(\mathbf{f}') = b_i^{(2)}$. We use (31) (observe that we need $b_i(\mathbf{f}) \geq \log^{4/3} n$) with $p = b_{\max}/(2 \cdot n), M = 1$ to bound $b_i^{(2)}$. This gives

Lemma 11 $\mathbf{f} \in \mathcal{G}(n, p, X)$ *satisfying (16, 17) is transformed by InterBal in* $\mathcal{O}(\log n)$ *time to an* $\mathbf{f}'$ *for which (20) holds and for which we have* $b_{\max}(\mathbf{f}') < \max\{\epsilon \cdot b_{\max}^{9/10}(\mathbf{f}), \log^{4/3} n\}$ *for all* $\epsilon > 0$ *aa.*

2.4 NearBal and CompleteBal

First we study the effect of NearBal(d) on $\min\{A^-, A^+\}$ for some preflow $\mathbf{f} \in \mathcal{G}(n,p,X)$. Suppose $\mathbf{f}$ satisfies (16) and (12) together with

$$\min\{A^-(\mathbf{f}), A^+(\mathbf{f})\} \leq n \cdot (2/\tilde{n})^{d-1} \tag{21}$$

The algorithm for NearBal(d) is just a shell for calling repeatedly the subprocedures NBMinus and NBPlus:

while $indeg_{\min}, outdeg_{\min} > \tilde{n}/2$ **and** $A^-, A^+ \geq \max\{\log^{4/3} n, n \cdot (2/\tilde{n})^d\}$ **do**
  if $A^-(\mathbf{f}) \leq A^+(\mathbf{f})$
  then NBMinus(d)
  else NBPlus(d) **fi** **od** ;

The main part of this section will be spent on the analysis of the effect of NBMinus(d) on $\min\{A^-, A^+\}$. In this case $A^-(\mathbf{f}) \leq A^+(\mathbf{f})$. Of course the results for NBPlus(d) are the same. The algorithm of NBMinus(d) proceeds as follows:

1. perform a breadth-first search with branching degree $\tilde{n}/2$ for vertices j with $b_j > 0$ at a depth d in parallel from all vertices i with $b_i < 0$ { With at most $n \cdot (2/\tilde{n})^{d-1}$ vertices with negative balance, 21), and n^2 PUs we can assign $n^2/A^- \geq n \cdot (\tilde{n}/2)^{d-1}$ PUs to each of the $A^-(\mathbf{f})$ parallel calls of NBMinus. This is sufficient to perform this search in $\mathcal{O}(d \cdot \log n)$. }
2. make the paths begin and end-vertex disjunct { This gives that b_i is non-increasing for all i. }
3. make the paths vertex disjunct { This is done to prevent write conflicts and to guarantee that the flow in an edge is not reduced to less than 0. }
4. reduce the flow on all edges that lie in a path selected in step 2 and 3 by 1

Lemma 12 *If the number of selected paths in NBMinus(d) is Δ_d then B is reduced by $2 \cdot \Delta_d$. The in and out-flows are reduced by at most 1. NBMinus(d) uses $\mathcal{O}(d \cdot \log n)$ time.*

If more than one path comes together in a vertex, one of them can be selected in the same way as in InterBal. Lemma 12 shows that we have to give an estimate on Δ_d. We may assume that

$$\max\{\log^{4/3} n, n \cdot (2/\tilde{n})^d\} \leq \min\{A^-, A^+\} \tag{22}$$

because otherwise we are already done for this d. Using (22) we find for the probability δ that a path ends in a vertex with positive balance $\delta = A^+/n > (2/\tilde{n})^d$. The number of paths of length d from a certain vertex i, $nr_i^{(d)}$, is given by $nr_i^{(d)} = (\tilde{n}/2)^d$ for all i aa. Thus $nr_i^{(d)} \cdot \delta > 1$. From this it follows that δ_d, the probability that at least one of the paths from a vertex ends in a vertex with positive balance, satisfies $\delta_d > 1/2$. Using corollary 5 we find for Δ_d'', the number of vertices which find a j with $b_j > 0$ at distance d, $\Delta_d'' > A^-/2 - (A^-/2)^{9/10} > A^-/3$. After step 2 of the algorithm only an end-point disjunct subset of these paths remain. Application of corollary 7 gives for Δ_d', the number of these paths, $\Delta_d' > \Delta_d''/3 > A^-/9$. We still should calculate Δ_d the number of paths among the Δ_d' which are vertex disjunct. Assume that $\Delta_d' - 1$ paths are disjunct, then (use (21) and $\Delta_d' < A^-$) $P(path_{\Delta_d'}$ lies disjunct$) = (1 - (d+1) \cdot (\Delta_d' - 1)/n)^{d-1}$. For $d = 1$ this probability equals 1 as it should be: The start and end-vertex were already made disjunct from the rest. For $d \geq 2$ we find $P_{d \geq 2}(path_{\Delta_d'}$ lies disjunct$) > (1 - (d+1) \cdot (2/\tilde{n})^{d-1})^{d-1} > 1 - d^2 \cdot (2/\tilde{n})^{d-1} > 1 - 8/\tilde{n}$. The probability that an arbitrary path can be added disjunctly is greater than the probability that the last path can be added disjunctly. Therefore we may apply (28) with $p = 1 - 8/\tilde{n}$: $\Delta_d > \Delta_d' \cdot (1 - 8/\tilde{n}) - (\Delta_d' \cdot (1 - 8/\tilde{n}))^{9/10} > A^-/10$. It is this application of DeMoivre which brings along the requirement $\min\{A^-, A^+\} > \log^{4/3} n$. Substituting Δ_d in lemma 12 and $\mathbf{f}'$ is the preflow we get after application of NBMinus(d) then $B(\mathbf{f}') < B(\mathbf{f}) - 2 \cdot \min\{A^-(\mathbf{f}), A^+(\mathbf{f})\}/10$ aa.

The slowest decrease rate of B is obtained if the ratio $B/\min\{A^-(\mathbf{f}), A^+(\mathbf{f})\}$ remains stable at its maximum, this means: We can substitute $B = \min\{A^-(\mathbf{f}), A^+(\mathbf{f})\} \cdot b_{\max}$. With (12) this gives

$$\min\{A^-(\mathbf{f}'), A^+(\mathbf{f}')\}/\min\{A^-(\mathbf{f}), A^+(\mathbf{f})\} < (1 - 1/(10 \cdot \log^{4/3} n)) \text{ aa} \qquad (23)$$

We return to the analysis of NearBal(d). NBMinus and NBPlus are called as long as the conditions are satisfied with a maximum of $10 \cdot \log^{4/3} n \cdot \log \tilde{n}$ times. Let $\mathbf{f}''$ be the preflow we would have after this number of calls, then (23) gives $\min\{A^-(\mathbf{f}''), A^+(\mathbf{f}'')\}/\min\{A^-(\mathbf{f}), A^+(\mathbf{f})\} < (1-1/(10 \cdot \log^{4/3} n))^{10 \cdot \log^{4/3} n \cdot \log \tilde{n}} \simeq 1/e^{\log \tilde{n}} < 2/\tilde{n}$. With the trivial observation that $\min\{A^-(\mathbf{f}), A^+(\mathbf{f})\} \leq n/2$, this gives the desired result on NearBal(d):

Lemma 13 *$\mathbf{f} \in \mathcal{G}(n,p,X)$ satisfying (16, 12, 21 and 22) is transformed by calling NearBal(d) in $\mathcal{O}(d \cdot \log^{7/3} n \cdot \log \tilde{n})$ time to an $\mathbf{f}''$ for which $\min\{A^-(\mathbf{f}''), A^+(\mathbf{f}'')\} \leq n \cdot (2/\tilde{n})^d$ aa, $inflow_i(\mathbf{f}) - inflow_i(\mathbf{f}'') < 10 \cdot d \cdot \log^{4/3} n \cdot \log \tilde{n}$, $outflow_i(\mathbf{f}) - outflow_i(\mathbf{f}'') < 10 \cdot d \cdot \log^{4/3} n \cdot \log \tilde{n}$.*

After the calls of NearBal CompleteBal reduces the imbalance of $\mathbf{f}$ ($B(\mathbf{f}) > 0$) further. It proceeds as follows:

1. select i, j such that $b_i < 0$, $b_j > 0$ { If (8) holds then there is at least one i with $b_i < 0$ and one j with $b_j > 0$ as long as $B > 0$. }
2. find a path from i to j of maximal length $2 \cdot \log n / \log \tilde{n}$ { Such a path exists aa and assigning n PUs to every vertex it is easy to give an algorithm which finds it in $\mathcal{O}(\log^2 n / \log \tilde{n})$ time. }
3. decrease the flow along the path from i to j by 1 { Along the path the balances remain unchanged, for both i, j it has improved by 1. Thus we achieved a decrease of B by 2. }

Lemma 14 *If $B(\mathbf{f}) > 0$ and $diam(\mathbf{f}) \leq \lfloor 2 \cdot \log n / \log \tilde{n} \rfloor$ then CompleteBal reduces B to $B - 2$. The reduction of $inflow_i$ and $outflow_i$ is at most 1. The time consumption is $\mathcal{O}(\log^2 n / \log \tilde{n})$.*

2.5 Dependency

Many times we used conditions of the form "$\mathbf{f} \in \mathcal{H} \subset \mathcal{G}(n,p,X)$". This means that we assumed that the possible $\mathbf{f}$ were distributed as in $\mathcal{H}$. For the $\mathbf{f}$ obtained after a number of steps of the algorithm this is not correct. In this section we analyse what a better analysis would give.

Assume that after a certain number of calls of PreBal (InterBal) we have obtained a preflow $\mathbf{f}$ for which for some i, j: $b_i(\mathbf{f}) < 0$ and $f_{ji} > 0$, then i fetches $\delta_{i;ij}(\mathbf{f}) = f_{ij} \cdot b_i(\mathbf{f}) / outflow_{\text{int},\,i}(\mathbf{f})$ back from j in the next call of PreBal (InterBal)(not taking into account the rounding). That is $b_j(\mathbf{f}')$ depends on $b_i(\mathbf{f})$: it has become smaller than we expected a priori. This may influence the value of $b_i(\mathbf{f}'')$ where $\mathbf{f}''$ is the preflow we get after another call of PreBal (InterBal). The nature of this dependency depends on the sign of $b_j(\mathbf{f}')$: If $b_j(\mathbf{f}') > 0$ then due to the sending of $\delta_{i;ij}(\mathbf{f})$ $b_i(\mathbf{f}'')$ is smaller than expected by an amount of $\delta_{j;ij,\,\text{dep}} = -f'_{ij} \cdot \delta_{i;ij}(\mathbf{f}) / inflow_{\text{int},\,j}(\mathbf{f}')$. This situation will occur about $\tilde{n}/2$ times. If $b_j(\mathbf{f}') < 0$ then $b_i(\mathbf{f}'')$ depends only on $b_i(\mathbf{f})$ if $f'_{ji} > 0$. In this case $\delta_{j;ji,\,\text{dep}} = -f'_{ji} \cdot \delta_{i;ij}(\mathbf{f}) / outflow_{\text{int},\,j}(\mathbf{f}')$. $\delta_{j;ji,\,\text{dep}} > 0$ will happen about $p \cdot \tilde{n}/2$ times (the situation is illustrated in figure 2). This gives for $\Delta_{i,\,\text{dep}}$, the expected total contribution to the imbalance of $b_i(\mathbf{f}'')$ due to dependencies of this kind $\Delta_{i,\,\text{dep}} \simeq b_i(\mathbf{f}) \cdot (1/(2 \cdot \tilde{n}) + 1/(2 \cdot n)) \leq b_{\max}(\mathbf{f})/\tilde{n}$. Comparing this with the upper bound on $b_i(\mathbf{f}'')$ we find from (19): $\epsilon \cdot (\epsilon \cdot (b_{\max}(\mathbf{f})/\tilde{n}^{1/10} + \tilde{n}^{7/10})/\tilde{n}^{1/10} + \tilde{n}^{7/10}) > \epsilon^2 \cdot b_{\max}(\mathbf{f})/\tilde{n}^{2/10}$, we conclude that the effect of dependency can be neglected (for InterBal $b_{\max}(\mathbf{f}'')$ is of smaller order but the conclusion remains the same).

So far we calculated the first order contribution of the dependency. The contributions resulting from imbalance which returns after a longer path to i are of even smaller order.

Reductions in NearBal and CompleteBal cannot induce new imbalances and thereby make necessary new reductions as was the case with PreBal and InterBal. Therefore, the kind of reductions we find during PreBal and CompleteBal do not occur. The dependency we find here arises from the fact that near to an unbalanced vertex edge flows are reduced more than elsewhere. E.g. in figure 3

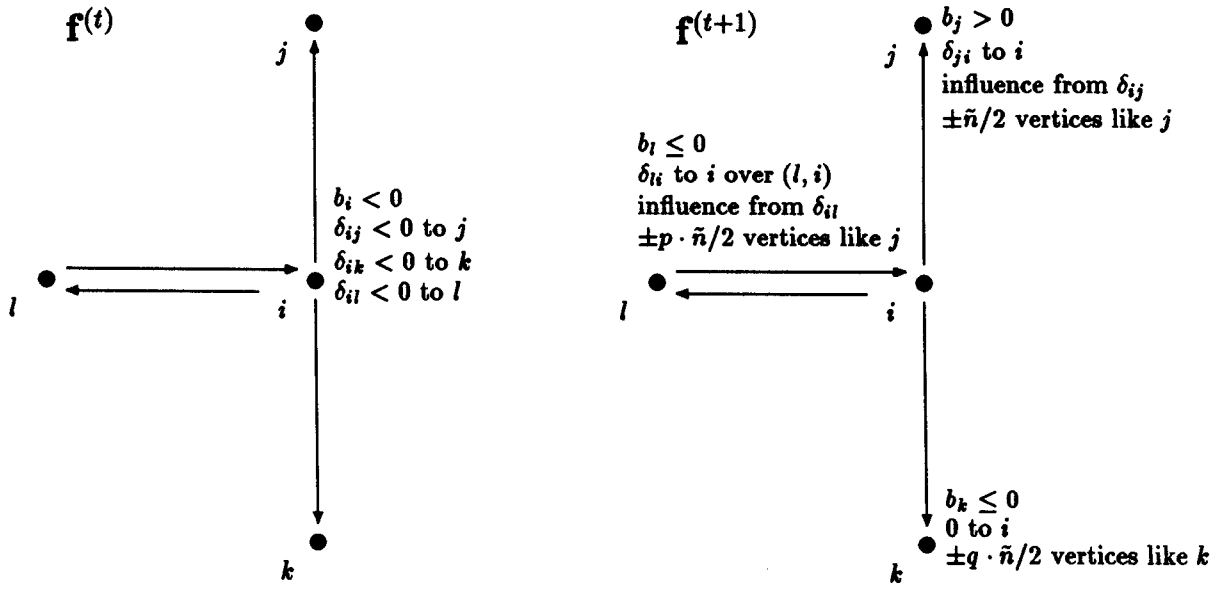

Figure 2: Dependency on the imbalance at vertex i with PreBal (InterBal).

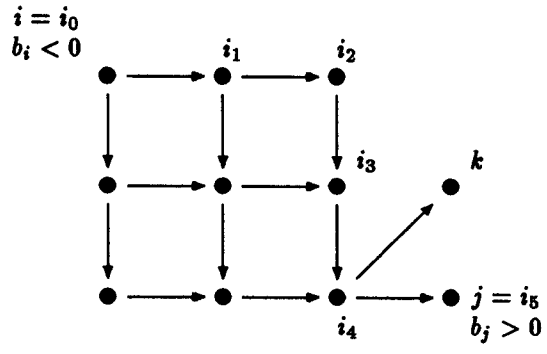

Figure 3: Flow reduction during NearBal takes place along the paths from i to j.

the probability that one of the edges in the path $(i_0, \ldots, i_5)$ gets emptied is non-zero. For (i_4, j) this is even more probable while it may lie in more then one path. On the contrary (i_4, k) remains as it was. These specific reductions might deform the graph such that corollary 2 and corollary 3 no longer hold. These corollaries were based on (15). Without dependency the reduction of inflow_j for a $j \in S^+$ was at most $4 \cdot C \cdot \tilde{n}^{9/10}$. The largest effect we have is the extra emptying of edges adjacent to an j because they were last edge in a path from an $i \in S^-$. This happens less than $b_j < \log^{4/3} n$ times. This extra reduction of inflow_j is no obstruction in obtaining (15).

3 Extensions

In this section subjects are studied which make the range of applicability of the developed algorithm larger: We prove that the results hold for other classes of graphs than $\mathcal{G}(n, p, X)$; we give conditions which make it probable that the algorithm will be successful and we give suggestions how the algorithm can be used in practical situations.

3.1 Other classes of graphs

In a practical situation one usually knows the number of edges, m, or it can be determined easily. Therefore, is it desirable to have results for $\mathcal{H}(n, m, X)$, the class of directed graphs with m non-parallel edges weighted by X, and $\mathcal{MH}(n, m, X)$, the class of directed graphs weighted by X with m possibly parallel edges. We can define classes $\mathcal{UG}(n, p, X), \mathcal{UH}(n, m, X), \mathcal{MUH}(n, m, X)$ of undirected graphs analogously.

3.1.1 The class $\mathcal{H}(n, m, X)$

In this section we will prove that the results for the average performance of MaxFlow over $\mathcal{G}(n, p, X)$ hold just as well over $\mathcal{H}(n, m, X)$. For the proof we need a theorem and some notions from the probability theory on graphs. They can be found in Bollobás [1, p 32 ...35].

The probability that a graph $\mathbf{c}$ with m edges occurs in $\mathcal{H}(n, m, X)$ is $P_{\mathcal{H}}(\mathbf{c}) = \prod_{k=1}^{m} \xi(c_{ij(k)}) \,/ \binom{N}{m}$. Here ij is the function which enumerates the non-zero edges. Let $\mathcal{G}^n(X) = \mathcal{G}(n, 1/2, X)$ the class of all graphs.

Definition 2 *Q is a property of graphs if Q is a subset of $\mathcal{G}^n(X)$ which contains full isomorphism classes.*

We are specially interested in the property R defined by

$$R = \{\mathbf{c} \in \mathcal{G}^n(X) | \text{ MaxFlow needs polylogarithmic time when applied to } \mathbf{c}\} \tag{24}$$

$P_{\mathcal{G}^n(X)}(Q)$, the probability on a property Q is defined as the probability on the subset Q. For a subclass $\mathcal{C}$ of $\mathcal{G}^n$ the probability on Q is $P_{\mathcal{C}(X)}(Q) = P_{\mathcal{C}(X)}(Q \cap \mathcal{C}(X))$. For $\mathcal{G}$ and $\mathcal{H}$ this amounts to

$$P_{\mathcal{G}(n,p,X)}(Q) = \sum_{\mathbf{c} \in Q} P_{\mathcal{G}(n,p,X)}(\mathbf{c}) = \sum_{\mathbf{c} \in Q} p^{E(\mathbf{c})} \cdot q^{N - E(\mathbf{c})} \cdot \prod_{k=1}^{E(\mathbf{c})} \xi(c_{ij(k)}) \tag{25}$$

$$P_{\mathcal{H}(n,m,X)}(Q) = \sum_{\mathbf{c} \in Q \cap \mathcal{H}(n,m,X)} P_{\mathcal{H}(n,m,X)}(\mathbf{c}) = \sum_{\{\mathbf{c} \in Q | E(\mathbf{c}) = m\}} \prod_{k=1}^{m} \xi(c_{ij(k)}) / \binom{N}{m} \tag{26}$$

Example 1 *A property of graphs from $\mathcal{G}^n(X)$ is $Q = \{\mathbf{c} \in \mathcal{G}^n(X) | E(\mathbf{c})$ is even$\}$. For this property $P_{\mathcal{G}(n,p,X)}(Q) = 1/2$; $P_{\mathcal{H}(n,m,X)}(Q) = 0$ if m odd.*

This shows that properties holding for $\mathcal{G}$ not necessarily hold for $\mathcal{H}$. On the other hand it is clear that properties holding for $\mathcal{H}$ hold at least sometimes for $\mathcal{G}$:

Lemma 15 *If $p > 0, Q \subset \mathcal{G}^n(X)$, then we have $P_{\mathcal{H}(n,m,X)}(Q) \leq 3 \cdot m^{1/2} \cdot P_{\mathcal{G}(n,p,X)}(Q)$.*

Proof: From (25 and 26) it is immediate that $P_{\mathcal{G}(n,p,X)}(Q) = \sum_{\mu=0}^{N} \binom{N}{\mu} \cdot p^{\mu} \cdot q^{N-\mu} \cdot P_{\mathcal{H}(n,\mu,X)}(Q)$ and thus $P_{\mathcal{G}(n,p,X)}(Q) > \binom{N}{m} \cdot p^m \cdot q^{N-m} \cdot P_{\mathcal{H}(n,m,X)}(Q) > P_{\mathcal{H}(n,m,X)}(Q)/((2 \cdot \pi \cdot p \cdot q \cdot N)^{1/2} \cdot e^{m/6})$. To obtain this last inequality we used [1, (5) p 4]. ■
By the example it seems difficult to derive results for $\mathcal{H}$ from those of $\mathcal{G}$. Fortunately we have

Corollary 4 *If $p > 0$ and Q is a property which holds over $\mathcal{G}(n,p,X)$ aa then Q holds over $\mathcal{H}(n,m,X)$ aa.*

Proof: From the definition of aa (definition 1) and lemma 15 we find that there is an $\omega, \lim_{n\to\infty} \omega(n) = \infty$ such, that $P_{\mathcal{H}(n,m,X)}(\neg Q) \leq 3 \cdot m^{1/2} \cdot P_{\mathcal{G}(n,p,X)}(\neg Q) < 3 \cdot n \cdot n^{-\omega(n)}$. ■
This corollary and theorem 4 give

Theorem 6 *If $(X, m/N)$ form an acceptable pair and $\mathbf{c} \in \mathcal{H}(n,m,X)$ then $\mathbf{c} \in R$ aa and $T_{\text{av},\,\mathcal{H}(n,m,X)} = \mathcal{O}((\log C + \log^4 n) \cdot \log n / \log \tilde{n})$.*

3.1.2 The class $\mathcal{MH}(n,m,X)$

In $\mathcal{G}(n,p,X)$ and $\mathcal{H}(n,m,X)$ we assumed that there was at most one edge (i,j). In this section we will show that if $m < N^{1-\epsilon}$ for some $\epsilon > 0$ the results also hold for the class $\mathcal{MH}(n,m,X)$ where the m edges are distributed completely arbitrary.

Let $\mathbf{c} \in \mathcal{G}^n(X)$ a graph and $Q \subset \mathcal{G}^n(X)$ a property, then we define $P_{\mathcal{MH}}(\mathbf{c}) = \prod_{k=1}^{m} \xi(c_{i(k)j(k)}) / N^m$, $DePar(\mathbf{c}) = \mathbf{c}$ with all parallel edges replaced by one edge with summed capacity, $Q_m = \{\mathbf{c} \in Q | E(DePar(\mathbf{c})) = m\}$, $MaxPar(\mathbf{c}) = \max_{0 \leq i,j \leq n}\{k | k =$ number of directed edges (i,j) in $\mathbf{c}\}$ and the skeleton of a graph $\mathbf{c} \in \mathcal{H}(n,m,X)$ is the graph $Skelet(\mathbf{c}) \in \mathcal{H}(n,m,1)$ obtained by replacing all edge capacities by 1. Of course the maxflow is invariant under $DePar$:

$$F(DePar(\mathbf{c})) = F(\mathbf{c})$$

There is no X' such, that $DePar : \mathcal{MH}_{m'}(n,m,X) \to \mathcal{DH}(n,m',X')$ is a probability preserving mapping. Nevertheless, $DePar$ has some nice properties. It is easy to check that

Lemma 16 *If $\mathbf{c} \in \mathcal{MH}_{m'}(n,m,X)$ then $P(Skelet(DePar(\mathbf{c})) = sk_1) = P(Skelet(DePar(\mathbf{c})) = sk_2)$ for all $sk_1, sk_2 \in \mathcal{DH}(n,m',1)$.*

Lemma 17 *If $m \leq N^{1-\epsilon}, \epsilon > 0, \mathbf{c} \in \mathcal{MH}(n,m,X)$ and $(X, m/N)$ is acceptable then*

$$\max_{0 \leq i,j \leq n} \{DePar(\mathbf{c})_{ij}\} \leq C(X) \cdot \tilde{n}^{9/40} \text{ aa}$$

Proof: $\max_{0 \leq i,j \leq n}\{c_{ij}\} \leq M(X)$ aa. Substitution of $r = m, n = N$ in corollary 8 gives $MaxPar(\mathbf{c}) \leq \log\log n$ aa. Using $\max_{0 \leq i,j \leq n}\{DePar(\mathbf{c})_{ij}\} \leq \max_{ij}\{c_{ij}\} \cdot MaxPar(\mathbf{c})$, the definition of acceptable and $\log\log n < \log^{3/40} n \leq \tilde{n}^{1/40}$ we get the result. ■
Let R be the maxflow property (24) generalized to $\mathcal{MG}^n(X)$, then we can use these two lemmas for the central lemma of this section:

Lemma 18 *Under the conditions of lemma 17 there is an $\omega_{m'}$ with $\lim_{n\to\infty} \omega_{m'}(n) = \infty$ such, that*

$$P_{\mathcal{MH}_{m'}(n,m,X)}(\neg R) < n^{-\omega_{m'}(n)} \text{ for all } m' \geq m/3$$

Proof: We know $P_{\mathcal{MH}(n,m',X')}(\neg R) < n^{-\omega_{m'}(n)}$ for any acceptable pair (X', m'). If we pass over the proofs again we find that we do not need X' to be used independently for every edge. It is sufficient that the capacities and edges are regularly distributed aa. This amounts to

- All graph skeletons with m' edges are equi-probable.

- (3, 4) hold aa.
- (2) holds.

Set $\tilde{n}' = m'/(n-1), C' = C \cdot m/m'$, then $\tilde{n}' \geq \log^{29/10} n, X \leq \tilde{n}'^{9/40} \cdot C'$ aa. This gives alternative values for η, ζ in section B.2. If we leave θ, θ' as they were we find (3, 4) anew and the only consequence of taking the modified (2) is that the number of calls of PreBal should be increased slightly (disappearing in the estimates). ■
Use corollary 7 with $r = m, n = N$ to prove

Lemma 19 *If $m \leq N$ then there is an χ with $\lim_{n\to\infty} \chi(n) = \infty$ such, that*

$$P_{\mathcal{MH}(n,m,X)}(\mathcal{MH}_{m'<m/3}(n,m,X)) < n^{-\chi(n)}$$

This gives the main theorem for $\mathcal{MH}(n,m,X)$:

Theorem 7 *If $m \leq N^{1-\epsilon}, \epsilon > 0, (X, m/N)$ is acceptable and $\mathbf{c} \in \mathcal{MH}(n,m,X)$ then $\mathbf{c} \in R$ aa, $T_{\text{av, } \mathcal{MH}(n,m,X)} = \mathcal{O}((\log C + \log^4 n) \cdot \log n / \log \tilde{n})$.*

Proof: Define $\omega(n) = \min_{m' \geq m/3}\{\omega'_m(n)\}$, then $P_{\mathcal{MH}(n,m,X)}(R) = \sum_{m'=1}^{m} P_{\mathcal{MH}_{m'}(n,m,X)}(R) \cdot P_{\mathcal{MH}(n,m,X)}(\mathcal{MH}_{m'}(n,m,X)) > (1 - n^{-\omega(n)}) \cdot \sum_{m'=m/3}^{m} P_{\mathcal{MH}(n,m,X)}(\mathcal{MH}_{m'}(n,m,X)) \geq (1 - n^{-\omega(n)}) \cdot (1 - n^{-\chi(n)})$. ■

3.1.3 Undirected graphs

Sometimes the maxflow in an undirected network has to be found. We solve this by replacing undirected edges by a directed edge in both directions. The maxflow we find in this network can be transformed in a solution of the problem by taking the net-flow through every undirected edge (i, j).

We show that under certain conditions on p, X the maxflow for $\mathbf{c} \in \mathcal{UG}(\mathbf{n}, \mathbf{p}, \mathbf{X})$ can be found in pseudo-polylogarithmic time aa. From this the same result follows for the classes $\mathcal{UH}(n,m,X)$, $\mathcal{MUH}(n,m,X)$ in the same way as the result for their directed counterparts followed from that on $\mathcal{G}(n,p,X)$.

The proof for $\mathcal{UG}(n,p,X)$ is analogous to that for $\mathcal{G}(n,p,X)$. Only the situation of dependency is slightly different. In figure 2 vertices like k no longer exist. Now about half of the adjacent vertices are like j and the other half like l. The effect of dependencies is thereby at most doubled in comparison with the dependencies occurring for graphs of $\mathcal{G}(n,p,X)$. Thus they remain negligible.

3.2 Requirements on a graph, choice of constants

In this subsection some properties are given which a graph should have to make it probable that its maxflow can be found in polylog time. Also we give expressions for the values of p, C that should be used in the call of MaxFlow.

3.2.1 Requirements

If we pass over the proofs given for the results of theorem 4 and for the results of PreBal and NearBal we find that we need something like

$$\begin{aligned}
b_{\max}(\mathbf{c}) &< C \cdot \tilde{n}^{9/10} \\
\tilde{n} &\geq \log^3 n \\
M &\leq C \cdot \tilde{n}^{1/5} \\
\mathit{indeg}_{\min}, \mathit{outdeg}_{\min} &\geq \tilde{n}/2 \\
\mathit{inflow}_{\min}, \mathit{outflow}_{\min} &\geq \tilde{n} \cdot C/2 \\
\mathit{diam}(\mathbf{c}) &\leq \lceil 2 \cdot \log n / \log \tilde{n} \rceil
\end{aligned}$$

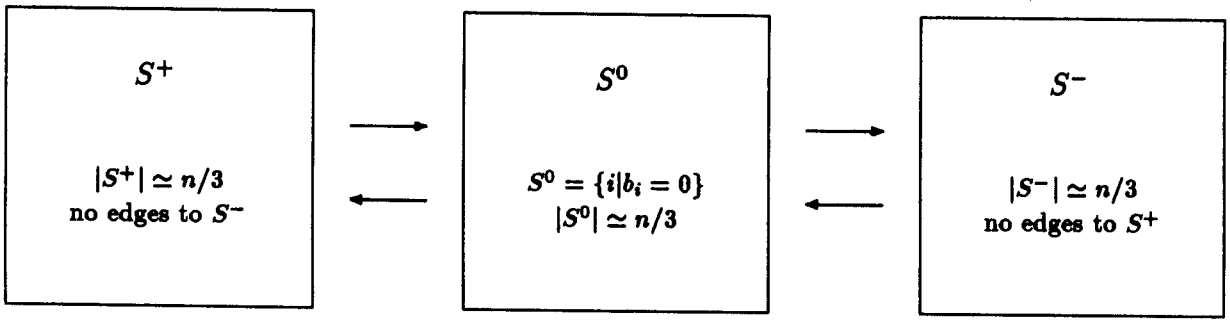

Figure 4: An exceptional distribution of imbalances. Here A cannot be reduced by NearBal(1)

The maxflow of a graph satisfying these requirements can be found pseudo-polylogarithmically with the algorithm of figure 1 aa.

However, there are two different origins of problems.

- It is well possible that for a graph c, although all requirements are satisfied, the mincut is not situated behind s or in front of t. So we should add the requirement

$$F(\mathbf{c}) = \min\{outflow_s(\mathbf{c}), inflow_t(\mathbf{c})\}$$

- The second problem is due to the algorithm and can occur with any graph (showing that for this algorithm there is no complete requirement set). During PreBal and InterBal the distribution of imbalances (apart from dependencies) is completely remodeled at every call. So it is the very last call of InterBal which determines the distribution of imbalances that is handed over to NearBal for further reduction. It may happen that the situation is now as depicted in figure 4. In this situation NearBal(1) will not change anything at all and therefore the algorithm will break off at the start of NearBal(2) because then $\min\{A^-, A^+\} = n/3 \geq 2 \cdot n/\tilde{n}$. This problem can be overcome by replacing Balance by a randomized version of it: If (the last call of) InterBal selects the edges in which the flow will be reduced randomly, then rerunning the algorithm will give a different distribution of imbalances which aa can be further reduced again.

We call the suggested alternative algorithm AltMaxFlow. Let $\mathcal{SG}_n(X)$ be the subclass of $\mathcal{G}^n(X)$ satisfying the requirements given above inclusively the one on the mincut. It is tempting to state

Hypothesis 1 *Using AltMaxFlow:* $T_{\text{expected}}(\mathbf{c}) = \mathcal{O}((\log C + \log^4 n) \cdot \log n / \log \tilde{n})$ *for all* $\mathbf{c} \in \mathcal{SG}_n(X)$.

It is easy to show, in fact most of it was done in this paper, that the requirements are satisfied by aa graphs of $\mathcal{G}^n(X)$. If the hypothesis is correct, this implies (10). On the other hand the proof of the hypothesis requires careful analysis of the reduction procedures.

3.2.2 Practical situation

For practical applications it is undesirable that the parameters p and C occur in the heading of MaxFlow. Usually one does not know p and the distribution X from which C follows. We can try to solve this problem in the following way:

1. $\mathbf{d} := DePar(\mathbf{c})$;
2. $p := E(\mathbf{d})/N$;
3. $C :=$ average of all edge capacities; $M :=$ maximal edge capacity;
4. MaxFlow$(n, p, \max\{C, M/\tilde{n}^{1/5}\}, \mathbf{c}, \mathbf{f})$.

In step 4 the value for C may be too small to assure enough calls of PreBal, therefore chose the maximum with $M/\tilde{n}^{1/5}$.

4 Conclusion

We presented an algorithm for the MFP. The algorithm started by saturating all edges and then aa balanced this preflow by repeatedly calling a number of balancing procedures. If the balancing was successful the remaining flow was maximal. We proved a pseudo-polylog average case time over a number of classes of random graphs. The problem to single out a class of graphs which has a pseudo-polylog time for all its elements remains to be solved. Other subject of future research can be the implementation of this algorithm on other computer models.

Acknowledgement

I would like to thank Marinus Veldhorst for his helpful comments, his careful reading and his judgement of the matter.

A Notation

n		number of vertices
N	$n \cdot (n-1)$	number of pairs (i,j), $i \neq j$
p		probability that an edge exists
m		number of edges for the classes of graphs where this is fixed
$\tilde{n}$	$p \cdot n$	the expected number of edges at a vertex
$\tilde{N}$	$p \cdot N$	the expected total number of edges
q	$1-p$	probability that an edge does not exist
X		distribution for edge capacities
ξ		density function of X
$S_{n,p}$		binomial distribution with parameters n, p
$N(0,\sigma)$		normal distribution with mean 0 and variance σ^2
$\nu(0,\sigma,t)$		density function for $N(0,\sigma)$
N_C	$N(0,\sqrt{\pi/2}\cdot C)$	normal distribution with variance $\pi \cdot C/2$
$\mathcal{C}$		a class of graphs
$\mathcal{G}(n,p,X)$		class of graphs we are working with
$P_{\mathcal{C}}$		probability distribution on $\mathcal{C}$
P		probability distribution on $\mathcal{G}(n,p,X)$
$C(X)$	$\sum_{x=1}^{\infty} x \cdot \xi x$	expectation of X
$M(X)$	$X \leq M(X)$ aa	maximal edge capacity
$\mathbf{c}$		graph (given by its capacity matrix)
$\mathbf{f}$		(pre-) flow in a graph
i,j		indices of vertices
$indeg_i(\mathbf{f})$	$\#\{j \mid f_{ji} > 0\}$	indegree of vertex i
$outdeg_i(\mathbf{f})$	$\#\{j \mid f_{ij} > 0\}$	outdegree of vertex i
$b_i(\mathbf{f})$	$\sum_j f_{ji} - f_{ij}$	balance of a preflow $\mathbf{f}$ at i
$inflow_i(\mathbf{f})$	$\sum_j f_{ji}$	flow into vertex i
$outflow_i(\mathbf{f})$	$\sum_j f_{ij}$	flow out of vertex i
$S^+(\mathbf{f})$	$\{i \mid b_i(\mathbf{f}) > 0\}$	set of vertices with positive balance
$S^-(\mathbf{f})$	$\{i \mid b_i(\mathbf{f}) < 0\}$	set of vertices with negative balance
$A(\mathbf{f})$	$\#S^+(\mathbf{f}) \cap S^-(\mathbf{f})$	number of vertices for which $b_i \neq 0$
$A^+(\mathbf{f})$	$\#S^+(\mathbf{f})$	number of vertices for which $b_i > 0$
$A^-(\mathbf{f})$	$\#S^-(\mathbf{f})$	number of vertices for which $b_i < 0$
$B(\mathbf{f})$	$\sum_i \lvert b_i(\mathbf{f})\rvert$	the total imbalance of $\mathbf{f}$
$E(\mathbf{f})$	$\#\{(i,j) \mid f_{ij} > 0\}$	number of non-zero edges of $\mathbf{f}$
$F(\mathbf{f})$	$\min\{F_{\mathbf{f}}(V_s,V_t) \mid$ (V_s,V_t) is an s-t cut$\}$	value of mincut of $\mathbf{f}$
$F_{\mathbf{f}}(V_s,V_t)$	$\sum_{\{i \in V_s, j \in V_t\}} f_{ij}$	the value of (V_s,V_t) for $\mathbf{f}$
$G(\mathbf{f})$	$\sum_{i,j} f_{ij}$	the total flow of $\mathbf{f}$
$G_{\text{int}}(\mathbf{f})$	$\sum_{1<i,j<n} f_{ij}$	the internal part of the total flow of $\mathbf{f}$
$T_{\mathcal{C}}$		worst case time on $\mathcal{C}$
$T_{\text{av},\mathcal{C}}$		average case time on $\mathcal{C}$ with probability distribution $P(\mathcal{C})$

B Probability theory

In this section we give a number of results from probability theory that are used over and over in the proofs. We will use P as our probability measure.

Theorem 8 *If $A_1, \ldots, A_i$ are all independent, satisfy (1) with a common $\omega(n)$ and $i = \mathcal{O}(n^\delta)$ for some fixed δ, then aa $\bigwedge_{j=1}^{i} A_j$.*

Theorem 9 *A polynomial time algorithm which needs aa $\mathcal{O}(f(n))$ time on a class $C(n)$ satisfies $T_{av}(C(n)) = \mathcal{O}(f(n))$.*

Let $S_{n,p}$ be the binomial probability distribution with parameters p and n and let $\simeq$ mean: "is converging to". Derived from a result by DeMoivre and LaPlace (c.f. [1, p 13]) is

Theorem 10 (DeMoivre) *If $q(n) = 1 - p(n), \lim_{n\to\infty} p(n) \cdot q(n) \cdot n = \infty, x(n) = o((p \cdot q \cdot n)^{1/6}), \lim_{n\to\infty} x(n) = \infty, h(n) = x \cdot (p \cdot q \cdot n)^{1/2}$ then $P(|S_{n,p} - p \cdot n| \geq h) \sim \frac{2}{x \cdot \sqrt{2 \cdot \pi}} \cdot e^{-x^2/2}$.*

Corollary 5 *For all $\epsilon > 0$*

$$|S_{n,p} - p \cdot n| < \epsilon \cdot (p \cdot n)^{7/10} \quad aa, \text{ if } n \cdot p(n) \geq \log^3 n \tag{27}$$

$$|S_{n,p} - p \cdot n| < \epsilon \cdot (p \cdot n)^{9/10} \quad aa, \text{ if } n \cdot p(n) \geq \log^{4/3} n \tag{28}$$

Actually the exponents $3, 7/10, 4/3, 9/10$ etc. can be replaced by others but the given values are convenient (c.f. section B.2). A "continuous" version of DeMoivre is (cf. [6])

Theorem 11 (Hoefdings inequality) *If X_i are independent positive random variable with mean C_i and maximal value M_i then $P(|\sum_{i=1}^n X_i - \sum_{i=1}^n C_i| \geq h) \leq 2 \cdot e^{-2 \cdot h^2 / \sum_{i=1}^n M_i^2}$.*

Given a probability distribution X we define $X(p)$ by $X(p) = X$ with probability p, $X(p) = 0$ with probability $1 - p$. It is by independent $X_i(p)$ that the graphs of $\mathcal{G}(n, p, X)$ are formed. Now we find by combining theorem 11 and corollary 5

Corollary 6 *If X is a positive distribution with mean C and $X_i(p)$ are independently formed from X for $1 \leq i \leq n$, $X < M$ aa and $n \cdot p \geq \log^3 n$, then for all $\epsilon > 0$*

$$|\sum_{i=1}^{n} X_i(p) - n \cdot p \cdot C| \leq \epsilon \cdot (n \cdot p)^{7/10} \cdot M \text{ aa } \leq \epsilon \cdot (n \cdot p)^{9/10} \cdot C \text{ if } M < \sqrt[5]{n \cdot p} \cdot C \tag{29}$$

If X is a symmetric distribution with positive mean $C = \sum_{-\infty}^{\infty} |i| \cdot \xi(i)$ and $X_i(p)$ are independently formed from X for $1 \leq i \leq n$, $X < M$ aa, then for all $\epsilon > 0$

$$|\sum_{i=1}^{n} X_i(p)| \leq \epsilon \cdot (n \cdot p)^{7/10} \cdot M \quad aa \text{ if } n \cdot p \geq \log^3 n \tag{30}$$

$$|\sum_{i=1}^{n} X_i(p)| \leq \epsilon \cdot (n \cdot p)^{9/10} \cdot M \quad aa \text{ if } n \cdot p \geq \log^{4/3} n \tag{31}$$

B.1 Normal distribution

The normal distribution with variance σ^2 and mean 0, $N(0, \sigma)$ has density function $\nu(0, \sigma, t) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-t^2/(2 \cdot \sigma^2)}$. Define $\phi(t), \Phi(x)$ to be the density function and distribution for $N(0, 1)$. Standard probability theory gives the following fact (see e.g. [2, p 175]): If $x \to \infty$ then

$$1 - \Phi(x) \sim \frac{1}{x} \cdot \phi(x) = \frac{1}{x \cdot \sqrt{2 \cdot \pi}} \cdot e^{-x^2/2} \tag{32}$$

Let $N_C = N(0, \sqrt{\pi/2} \cdot C)$, then with (32) it is easy to derive

Lemma 20 *The expectation of $|N_C|$ is C. $1 - N_C(x) \sim \frac{C}{2 \cdot x} \cdot e^{-x^2/(\pi \cdot C^2)}$, if $x \to \infty$.*

Theorem 12 *(N_C, p) is acceptable if $n \cdot p > \log^3 n$.*

Proof: From lemma 20 we get $P(N_C(x) > \sqrt[5]{n \cdot p} \cdot C) \leq \frac{1}{2 \cdot \sqrt[5]{n \cdot p}} \cdot e^{-(n \cdot p)^{2/5}/\pi} < e^{-\log^{6/5} n/\pi}$ ∎

This tells us that acceptability does not impose too much restrictions on the distribution.