Communication Complexity of Minimum Spanning Tree Problem

Mirek Kutyłowski

joint work with

M.Adler, W. Dittrich, B. Juurlink, I. Rieping

Computational model

- the system consists of many processors
- input data is distributed between processors
- the processors are communicating via point-to-point messages
- no assumptions on
  - computational power
  - synchronization
  - obliviousness
  - ...

Point-to-point messages is not a serious restriction, since

- shared memory cell simulated by a processor
- read operation = read request + message with the value

Problems with defining communication volume

- naive: the number of message bits sent
  for some models ok, but:
- there are tricky ways of transmitting information without transmitting message bits

Example 1

Saving bits while writing

- PRAM, processor P knows $k \in \{1, \ldots n\}$
- memory cells $M_1, \ldots, M_n$ are initially empty
- P writes a 1 into M_k

Communication volume= ???

- k may be determined from the contents of memory
- only 1 message bit sent !!!

Example 2

Saving bits while reading

- M_k stores 1, the rest is empty
- for $i \leq n$, processor P_i reads from M_i
- processor P_k gets the full information on k, the rest only a partial information
- the total number of message bits retrieved $= 1$

Example 3

Bits saved using synchronous clock

- P_1 and P_2 have a common clock
- P_1 has to sent P_2 a message $m \leq n$
- P_1 waits until a moment t such that $t = m \bmod n$ and sends a single bit to P_2
- P_2 receives the 1 before the change on the clock
- the total number of message bits transmitted $= 1$!

Definition - encoding communication

- let A be an algorithm on a multiprocessor system
- Definition:

  string s **encodes** communication between processor P and the rest of the world on input x

  iff

  the whole communication from the point of view of P may be computed from s and a description of A
- information retrieved from s describes everything: timing, addresses, messages, ...
- input x has **not** been used!

Definition - communication volume

communication volume at processor P during execution of algorithm A is at least k

if

$\forall$ encoding scheme for the communication between P and the rest of the world,
$\exists$ an input x such that the code describing the communication at processor P on x has length $\geq k$

Vector minimum problem

Task:

- given n sets $X_1, \ldots, X_n$
- each X_i consists of p keys, $x_{i1}, \ldots x_{ip}$
- goal: compute $\min(X_1), \min(X_2), \ldots, \min(X_n)$

Allocation of input data:

- there are p processors $P_1, \ldots, P_p$
- $\forall j$: P_j holds the numbers $x_{1j}, \ldots, x_{nj}$ (one number from each X_i)

Main technical theorem

- **Input:** vector minimum problem, where each of n sets X_i consists of k-bit numbers and the number of processors is p
- **Claim 1:**

  for any deterministic algorithm solving this problem,
  $\exists$ an input that requires total communication volume $\Omega(npk)$.
- **Claim 2:**

  for any randomized algorithm that always answers correctly,
  $\exists$ an input where the expected communication volume is $\Omega(npk)$.

Comments on theorem

- it is against intuition:
  - compare the most significant bits and many candidates of minimum are gone!
  - tricks with computations within groups
  - ...
- well, it usually works, but not always!
- Theorem says:
  no randomization can help!!!

Proof headlines

- analysis of deterministic algorithms running on an input drawn from a special probability distribution β
- we show that

  expected communication volume is large (expectation for distribution β)
- it follows: there is a bad input for any deterministic algorithm
- the claim for randomized algorithms follows by Yao's Theorem

Distribution β

An input is chosen as follows

- take n seeds $s_1, \ldots, s_n$,
  each chosen independently and uniformly at random from the set of all $k-1$ bit binary strings
- For $1 \leq i \leq n$ and $1 \leq j \leq p$:
  $x_{ij} = s_i$
- With probability ε we choose one processor P_u uniformly at random, and replace each seed held by processor P_u with a completely random $k-1$ bit string
- $\forall\ P_j$ we append a 1 to each x_{ij}, $1 \leq i \leq p$, except for strings x_{hj}, such that $h = j \bmod p$, to which we append a 0.

Comments:

- $\varepsilon > 0$ is an arbitrary constant)
- 3rd step called *scrambling* processor P_u
- the bit appended is the least significant bit

Correct output

- nothing scrambled: $\min X_i = s_i 0 \quad (s_i 0 \text{ is at } P_{i \bmod p})$
- scrambled: ... who knows?

Key technical lemma

Lemma
For any processor P_i, when the input is chosen according to distribution β, then the expected communication volume at P_i is $\Omega(kn)$.

The theorem follows by linearity of expectation:

$$E(\sum_{i \leq p} V_i) = \sum_{i \leq p} E(V_i) = p \cdot \Omega(kn)$$

(V_i = communication volume at P_i)

Standard and scrambled inputs

- assumption: no other processor than P_i can be scrambled
  (this event has probability $> 1 - \varepsilon$,
  $\Rightarrow$ it changes the expectations by a constant factor at most)
- notation:
  - S from seeds, no scrambling — input called *standard string*,
    notation $(S, -)$
  - S– standard string,
    P_j scrambled, and S replaced at P_j by S',
    notation (S, S')

Viewpoint of P_j

- (S, S') can be distinguished from $(S', -)$ only through communication with the rest
- do they have to distinguish??
  we answer **YES, they MUST distinguish**

distinguished pairs:

- $(S, -)$ and $(S', -)$ that differ on some seed s_i where $i \neq j \bmod p$

Distinguishing distinguished pairs

Claim

Communication between P_j and the rest is different on $(S, -)$ and $(S', -)$, if it is a distinguished pair.

Corollary

so the communication must encode these values ...

Proof of the claim - fooling

- assumptions: the communication is the same
  the seeds satisfy $s'_i < s_i$
- consider (S, S'),

  - who outputs $\min(X_i)$?
  - if this is P_j, then P_j does the same for $(S', -)$
    but for $(S', -)$: $\quad \min(X_i) = s'_i 0$
    for (S, S'): $\quad \min(X_i) = s'_i 1$
  - if this is somebody else, then the same value is outputted for $(S, -)$
    but for $(S, -)$: $\quad \min(X_i) = s_i 0$
    for (S, S'): $\quad \min(X_i) = s'_i 1$

□ Claim

Counting

- number of distinguished pairs:
  for given S, we may alter $k-1$ bits for each seed s_i except for $i = j \bmod p$.
- $2^{(k-1)n(p-1)/p}$ possibilities
- for encoding this number of different communications
  at least $(k-1)n(p-1)/p$ bits on average

Application to MST – case $p \leq m/n$

Corollary

MST-problem for graphs with n vertices, m edges, p processors and $p \leq m/n$, requires communication volume $\Omega(npk)$ (for some bad input, or expected volume for some bad input)

Reduction – case $p \le m/n$

nodes:

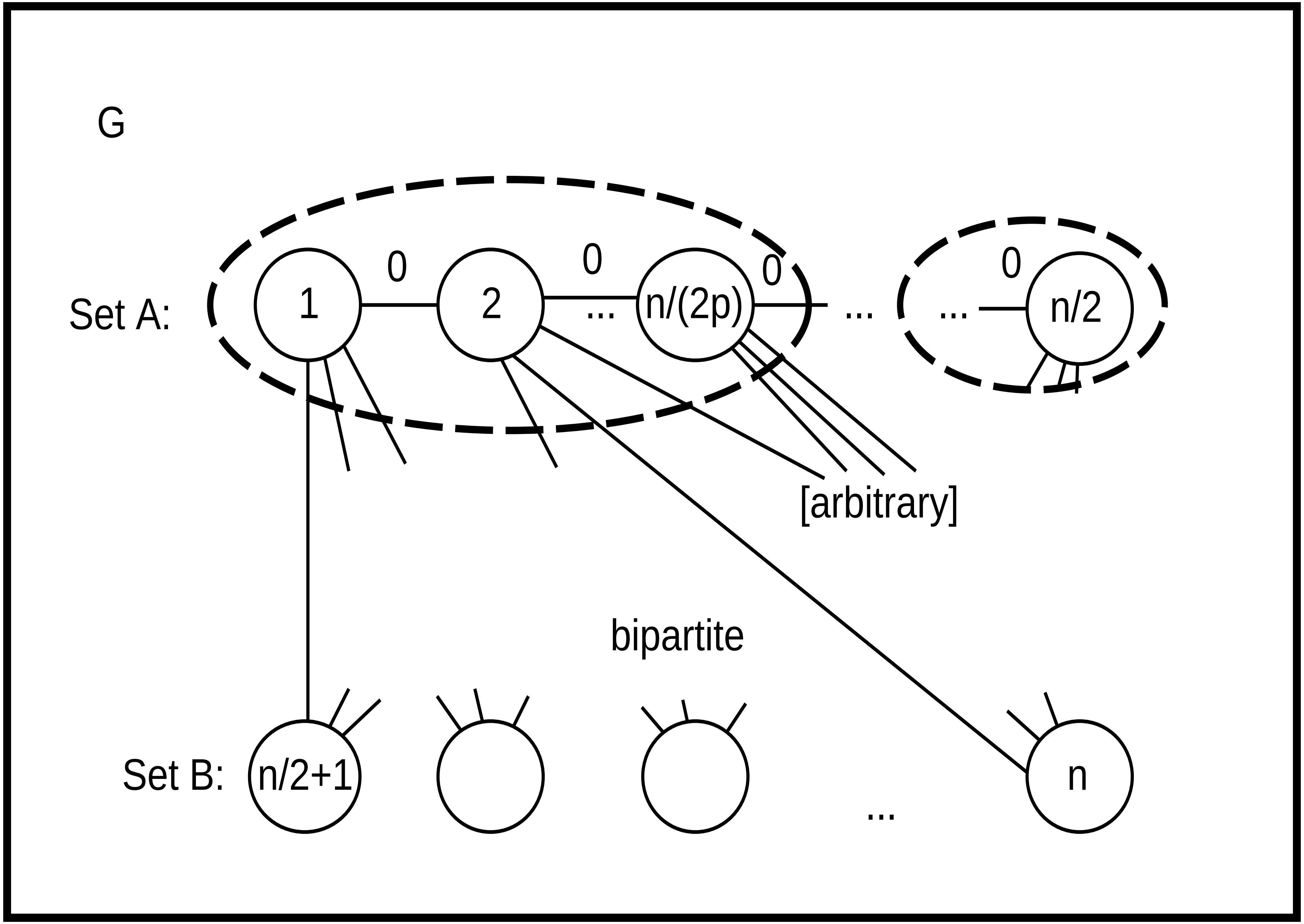
- disjoint sets A and B of $n/2$ nodes each
- A partitioned into $A_1, \ldots, A_p$
  each of size $n/(2p)$

Reduction – case $p \leq m/n$

The edges of G:

- $n/2-1$ edges of weight 0 connecting the nodes of A.
- Edges connecting the nodes of A with the nodes of B.
  Each node of A is connected with p nodes of B,
  the nodes of each A_i, are connected with all nodes of B.
  The weights of these edges are arbitrary odd numbers in $[1, 2^{k-1} - 1]$.
- Completing the number of edges to m:
  $m - (n/2-1) - pn/2$ edges connecting arbitrary nodes of weight $2^k - 1$.

Reduction –case $p \leq m/n$


G
Set A:
1
0
2
0
...
n/(2p)
0
...
...
0
n/2
[arbitrary]
bipartite
Set B:
n/2+1
...
n

Application to MST – case $p \geq m/n$

Corollary

MST-problem for graphs with n vertices, m edges, p processors and $p \geq m/n$, $m \geq 2n$, requires communication volume $\Omega(npk)$
(for some bad input, or expected volume for some bad input)

Reduction– case $p \geq m/n$

nodes:

- A – $n/2$
- B – $m/(2p)$
- C – filling up to totally n nodes

groups:

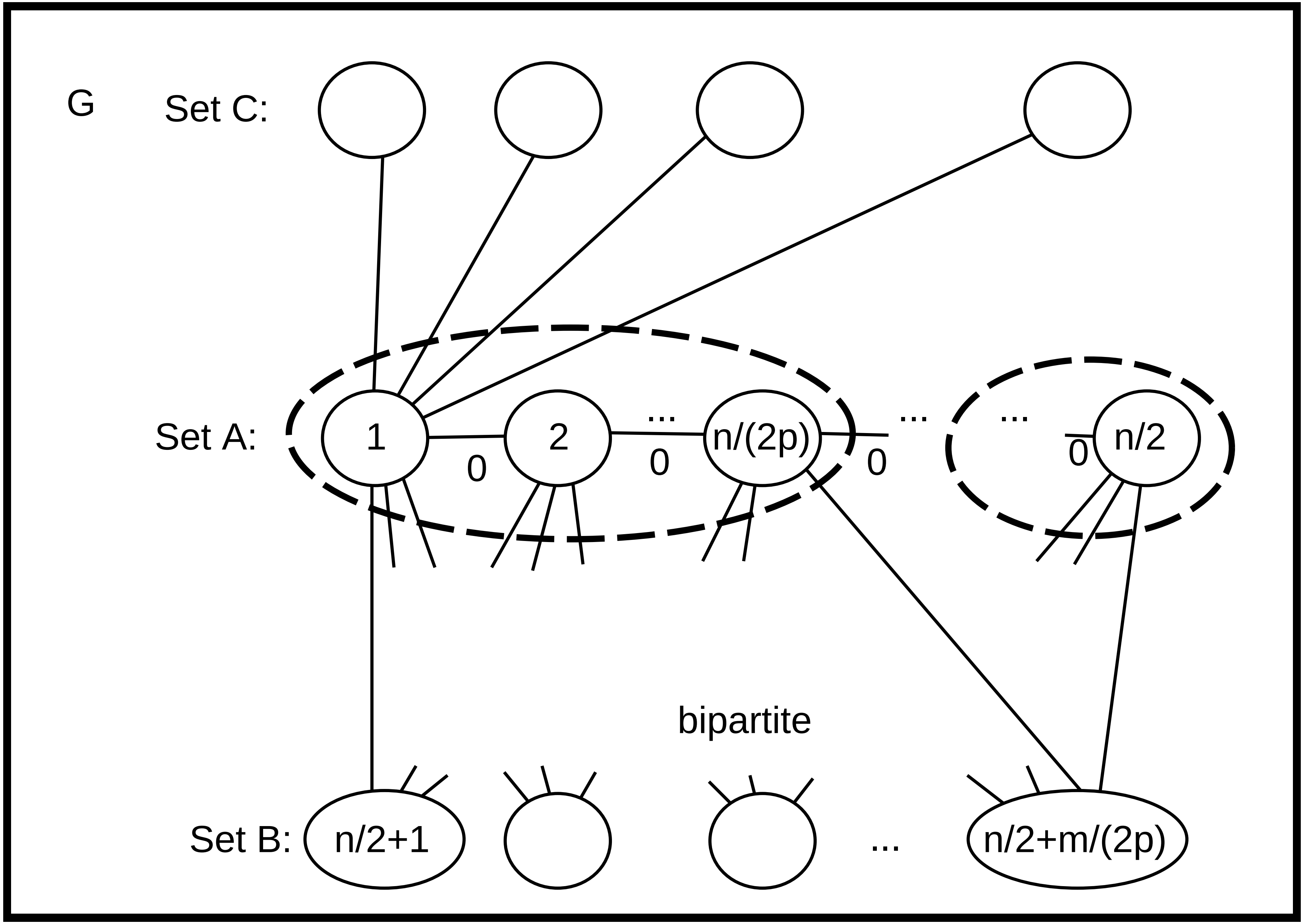
- A partitioned into $A_1, \ldots, A_p$
  each of size $n/(2p)$

Reduction– case $p \geq m/n$

edges:

- The nodes of A are connected by $n/2 - 1$ edges of weight 0.
- Each node of A is connected with m/n nodes of B,
  the nodes of a single group A_i are connected with all $m/(2p) = m/n \cdot n/(2p)$ nodes of B.
  The weights are arbitrary odd numbers from $[1, 2^{k-1} - 1]$.
- The nodes of C are connected to the first node of A by $|C|$ edges of weight 0.
- Some arbitrary edges of weight $2^k - 1$ to make the total number of edges equal to m.

Reduction


G
Set C:
Set A:
1
2
n/(2p)
n/2
0
0
0
0
...
...
...
bipartite
Set B:
n/2+1
...
n/2+m/(2p)