Distributing the Heat Equation

Tom Cornebize Yassine Hamoudi

Sunday, December 7th 2014

1 Cellular automata

Question 1

Lemma 1. *N^2 applications of function δ are necessary to compute X^t from X^{t-1}.*

Proof. Each cell $X^t_{i,j}$ needs one application of δ to be computed from $X^{t-1}_{i,j}$. There are N^2 cells, so N^2 applications of δ are needed. □

Property 2. *tN^2 applications of function δ are necessary to compute X^t on $[\![0, N-1]\!]^2$.*

Proof. X^t is obtained after t applications of $\delta^\dagger$ on X^0. Each application needs N^2 calls to δ according to lemma 1. The whole computation needs tN^2 applications of δ. □

Question 2

Let p^2 be the number of processors.

For the sake of simplicity, we will suppose that p divides N. Take $n = \frac{N}{p}$.

We divide the grid into square zones of size n. Each of this zones is given to one processor, which stores the data in its own memory and performs the computation of δ for all its cells. See figure 1 for an example.

At each step of computation, each processor updates its sub-matrix cells using a temporary sub-matrix that replaces the old one once the computation step is finished. Indeed, if we update the cells "in place", we overwrite values that are still necessary to compute other cells.

The computation of δ for the cells at the edges of the zones requires communication to retrieve the current states of their neighbours in other zones.

Question 3

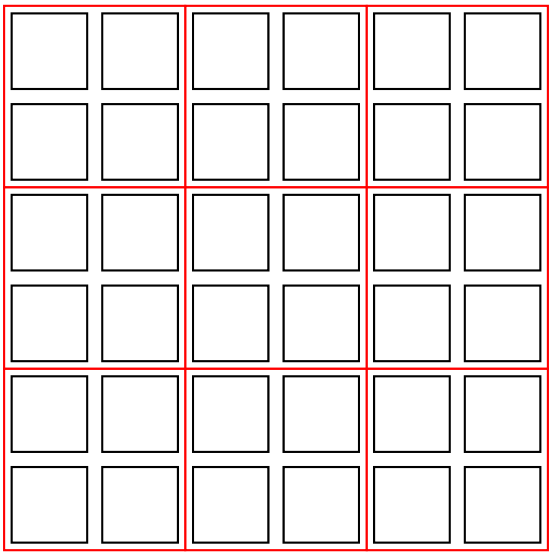
We assume that X^t is given as an array $PREV$ of size $(n+2) \times (n+2)$, where $X^t_{i,j}$ is written in $PREV[i][j]$ and where $PREV[i][j]$ are dummy values for $i \in \{0, n+1\}$ or $j \in \{0, n+1\}$.

We also assume that the order of messages is preserved.

Let

$$\delta(PREV, i, j) = \delta\left(\begin{array}{|c|c|c|}\hline PREV[i-1, j-1] & PREV[i-1, j] & PREV[i-1, j+1] \\ \hline PREV[i, j-1] & PREV[i, j] & PREV[i+1, j+1] \\ \hline PREV[i+1, j-1] & PREV[i+1, j] & PREV[i+1, j+1] \\ \hline \end{array}\right)$$

Figure 1: Graphical representation of the topology for $N = 6$ and $p^2 = 9$.



We consider functions `Send_X` (resp. `Receive_X`) for `X = Up, Down, Left, Right` which sends (resp. receives) to (resp. from) the corresponding processor. We suppose that this function has a time cost of 1 and a communication cost of $L + b$ where L is the latency and b the bandwidth.

We also consider functions to send an entire row (resp. column) as one single message, to decrease the overall latency. For instance, `Send_Down_Row(n,PREV)` will send to the down processor the nth row, whereas `Send_Up_Row(0,PREV)` will receive from the up processor a row, which will be stored as the 0^{th} row. We suppose that this function has a time cost of n and a communication cost of $L + nb$ where L is the latency, b the bandwidth, and n the size of the row/column.

Algorithm 1: Stencil algorithm on a toric 2D grid

```
Input: PREV: array[0..n+1,0..n+1] of real
Output: NEXT: array[0..n+1,0..n+1] of real
/* Columns and row                                        */
Send_Left_Column(1,PREV)
Send_Right_Column(n,PREV)
Send_Up_Row(1,PREV)
Send_Down_Row(n,PREV)
Receive_Left_Column(0,PREV)
Receive_Right_Column(n+1,PREV)
Receive_Up_Row(0,PREV)
Receive_Down_Row(n+1,PREV)
/* Corners                                                */
Send_Up(PREV[1][0])
Send_Up(PREV[1][n+1])
Send_Down(PREV[n][0])
Send_Down(PREV[n][n+1])
Receive_Up(PREV[0][0])
Receive_Up(PREV[0][n+1])
Receive_Down(PREV[n+1][0])
Receive_Down(PREV[n+1][n+1])
/* Computation of δ                                       */
for i=1 to n do
    for j=1 to n do
        NEXT[i][j] = δ(PREV, i, j)
```

Time complexity: $8(n+1)+n^2\text{cost}(\delta) = 8(\frac{N}{p}+1)+\left(\frac{N}{p}\right)^2\text{cost}(\delta) = \mathcal{O}\left(\left(\frac{N}{p}\right)^2\right)$ if $\text{cost}(\delta)=\mathcal{O}(1)$.

Communication complexity (one processor): $8(L+nb+L+1) = 8(2L+\frac{N}{p}b+1)$.

Communication complexity (all processors): $8p^2(2L+\frac{N}{p}b+1) = 8(2p^2L+Npb+p^2)$.

Non-toric grid? Ring topology? Don't know a nice way to do this...

2 Average automata

Question 4

See the implementation in file `average.c`.

Question 5

Property 3. *In the case of a p-average automaton, $\delta^\dagger$ is linear.*

Proof. Let $\delta^\dagger$ be the global transition function of a *p-average automaton*. To prove that $\delta^\dagger$ is linear, it suffices to prove that the local transition function δ is linear:

$$\delta\left(\begin{array}{|c|c|c|}\hline a & b & c\\\hline d & e & f\\\hline g & h & i\\\hline\end{array}\right) = (1-p)\cdot e + p\cdot\frac{b+d+f+h}{4}$$

Let consider a real $k \in \mathbb{R}$ and two local configurations $\begin{array}{|c|c|c|}\hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline\end{array}$ and $\begin{array}{|c|c|c|}\hline a' & b' & c' \\ \hline d' & e' & f' \\ \hline g' & h' & i' \\ \hline\end{array}$. We have:

$$\begin{aligned}
\delta\left(k \cdot \begin{array}{|c|c|c|}\hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline\end{array} + \begin{array}{|c|c|c|}\hline a' & b' & c' \\ \hline d' & e' & f' \\ \hline g' & h' & i' \\ \hline\end{array}\right) &= (1-p)\cdot(k\cdot e+e') + p\cdot\frac{(k\cdot b+b')+(k\cdot d+d')+(k\cdot f+f')+(k\cdot h+h')}{4} \\
&= k\cdot\left((1-p)\cdot e+p\cdot\frac{b+d+f+h}{4}\right)+(1-p)\cdot e'+p\cdot\frac{b'+d'+f'+h'}{4} \\
&= k\cdot\delta\left(\begin{array}{|c|c|c|}\hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline\end{array}\right)+\delta\left(\begin{array}{|c|c|c|}\hline a' & b' & c' \\ \hline d' & e' & f' \\ \hline g' & h' & i' \\ \hline\end{array}\right)
\end{aligned}$$

Thus δ is linear, and $\delta^{\dagger}$ too. □

Definition 1. For $0 \leqslant i, j \leqslant N-1$ we define the matrix $E^{i,j}$ such that $E^{i,j}_{i,j} = 1$ and $E^{i,j}_{k,l} = 0$ otherwise.

Lemma 4. *For all $0 \leqslant i, j, k, l \leqslant N-1$, $\delta^{\dagger^t}(E^{i,j})_{k,l} = \delta^{\dagger^t}(E^{0,0})_{k-i,l-j}$ (indices are taken modulo N). Thus, knowing $\delta^{\dagger^t}(E^{0,0})$ we obtain $\delta^{\dagger^t}(E^{i,j})$ in constant time.*

Now let's consider a configuration X. We have : $X = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{i,j}\cdot E^{i,j}$. Since $\delta^{\dagger}$ is linear, for all t :

$$\delta^{\dagger^t}(X) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{i,j}\cdot\delta^{\dagger^t}(E^{i,j})$$

Moreover :

$$\begin{aligned}
\delta^{\dagger^{2t}}(X) &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{i,j}\cdot\delta^{\dagger^{2t}}(E^{i,j}) \\
&= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{i,j}\cdot\delta^{\dagger^t}(\delta^{\dagger^t}(E^{i,j})) \\
&= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{i,j}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\delta^{\dagger^t}(E^{i,j})_{k,l}\cdot\delta^{\dagger^t}(E^{i,j}) \\
&= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{i,j}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\delta^{\dagger^t}(E^{0,0})_{k-i,l-i}\cdot\delta^{\dagger^t}(E^{i,j})
\end{aligned} \tag{1}$$

Thus, for all $0 \leqslant m, n \leqslant N-1$:

$$\begin{aligned}
\delta^{\dagger^{2t}}(X)_{m,n} &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{i,j}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\delta^{\dagger^t}(E^{0,0})_{k-i,l-i}\cdot\delta^{\dagger^t}(E^{i,j})_{m,n} \\
&= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{i,j}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\delta^{\dagger^t}(E^{0,0})_{k-i,l-i}\cdot\delta^{\dagger^t}(E^{0,0})_{m-i,n-j}
\end{aligned} \tag{2}$$

Especially, for $X = E^{0,0}$:

$$\delta^{\dagger^{2t}}(E^{0,0})_{m,n} = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\delta^{\dagger^t}(E^{0,0})_{k,l}\cdot\delta^{\dagger^t}(E^{0,0})_{m,n} \tag{3}$$

Property 5. *Equations 2 and 3 enable us to compute $\delta^{\dagger^t}(X)$ in time $O(log(t))$ for a fixed N.*

Proof. For computing $\delta^{\dagger^{2t}}(X)$ we need to compute $\delta^{\dagger^t}(E^{0,0})$ using equation 3 and then apply equation 2 to get $\delta^{\dagger^{2t}}(X)$. The algorithm is:

Algorithm 2: Fast iteration on average automaton

Input: t, X
Output: $\delta^{\dagger^t}(X)$
$R \leftarrow E^{0,0}$;
for *i=1* ***to*** $\log(t) - 1$ **do**
  Compute $R \leftarrow \delta^{\dagger^{2^i}}(E^{0,0})$ using $\delta^{\dagger^{2^{i-1}}}(E^{0,0})$ (previous value of R) and equation 3
Compute and return $\delta^{\dagger^t}(X)$ using $R = \delta^{\dagger^{t/2}}(E^{0,0})$ and equation 2

Let $T'(t)$ be the time needed to compute $\delta^{\dagger^t}(E^{0,0})$. According to equation 3 we have:

$$\begin{aligned} T'(t) &= T'(t/2) + O(1) \\ &= O(\log(t)) \end{aligned}$$

Now let $T(t)$ be the time needed to compute $\delta^{\dagger^t}(X)$. According to equation 2 and previous remarks, we have:

$$\begin{aligned} T(t) &= T'(t/2) + O(1) \\ &= O(\log(t)) \end{aligned}$$

Thus, applying the operations described in equations 2 and 3, we can compute $\delta^{\dagger^t}(X)$ in time $O(\log(t))$. □

Property 6. *The time complexity $T_r(t,N)$ in terms of both t and N is $T_r(t,N) = O(N^4 \cdot \log(t) + N^6)$.*

Proof. Let $T_e(t,N)$ be the time needed to compute $\delta^{\dagger^t}(E^{0,0})$. According to equation 3 we have:

$$\begin{aligned} T_e(t) &= T_e(t/2) + N^2 \cdot N^2 \text{ (there are } N^2 \text{ cells and each one needs to perform } N^2 \text{ sums)} \\ &= O(N^4 \cdot \log(t)) \end{aligned}$$

According to equation 2 and the previous algorithm, knowing $\delta^{\dagger^{t/2}}(E^{0,0})$ it remains to perform for each cell the four sums described in equation 2 (it takes time $O(N^4)$ per cell). Finally, we have:

$$T_r(t,N) = N^2 \cdot N^4 + T_e(t/2, N)$$

Thus: $T_r(t,N) = O(N^4 \cdot \log(t) + N^6)$. □

Property 7. *The space complexity $T_s(t,N)$ in terms of both t and N is $T_s(t,N) = O(N^2)$.*

Proof. We need N^2 space to store the initial matrix.

When we compute $\delta^{\dagger^{2^i}}(E^{0,0})$ using $\delta^{\dagger^{2^{i-1}}}(E^{0,0})$, we need to use N^2 space to store $\delta^{\dagger^{2^i}}$. However, the space used by $\delta^{\dagger^{2^{i-1}}}$ can be re-use to compute $\delta^{\dagger^{2^{i+1}}}$. Thus, all the computation can be done using $O(N^2)$ space. □

Comparer au cas général

Question 6

Question 7

Question 8

See the implementation in file `sparse.c`.

Question 9

3 Thermal reservoirs

Question 10

Example. The following configuration X^0 is a fixed point with one constant (in blue):

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Example. The following configuration X^0 is a fixed point with two constants (in blue):

1	2	0	-2	-1
2	5	0	-5	-2
4	16	0	-16	-4
2	5	0	-5	-2
1	2	0	-2	-1

Definition 2. We denote X^∞ the limit of (X^t) if it exists (i.e. (X^t) converges).

Lemma 8. *For each sequence (X^t), if X^∞ exists then it is a fixed point.*

Proof. Let us assume that X^∞ exists. By definition, for all position $X_{i,j}$ that is not a constant:

$$\begin{aligned}
\lim_{t\to\infty} \delta^{+^t}(X)_{i,j} &= X^\infty_{i,j} \\
&= \lim_{t\to\infty} \left((1-p)\cdot \delta^{+^{t-1}}(X)_{i,j} + p \cdot \frac{\delta^{+^{t-1}}(X)_{i,j+1} + \delta^{+^{t-1}}(X)_{i+1,j} + \delta^{+^{t-1}}(X)_{i-1,j} + \delta^{+^{t-1}}(X)_{i,j-1}}{4} \right) \\
&= (1-p)\cdot (\lim_{t\to\infty} \delta^{+^{t-1}}(X)_{i,j}) + p \cdot \frac{(\lim_{t\to\infty} \delta^{+^{t-1}}(X)_{i,j+1}) + (\lim_{t\to\infty} \delta^{+^{t-1}}(X)_{i+1,j})}{4} \\
&\quad + p \cdot \frac{(\lim_{t\to\infty} \delta^{+^{t-1}}(X)_{i-1,j}) + (\lim_{t\to\infty} \delta^{+^{t-1}}(X)_{i,j-1})}{4} \\
&= (1-p)\cdot X^\infty_{i,j} + p \cdot \frac{X^\infty_{i+1,j} + X^\infty_{i,j+1} + X^\infty_{i-1,j} + X^\infty_{i,j-1}}{4}
\end{aligned}$$

Since $X^\infty_{i,j} = (1-p)\cdot X^\infty_{i,j} + p \cdot \frac{X^\infty_{i+1,j}+X^\infty_{i,j+1}+X^\infty_{i-1,j}+X^\infty_{i,j-1}}{4}$, X^∞ is a fixed point.

□

Lemma 9. *If X is a fixed point then for each position $X_{i,j}$ that is not a constant we have :*

$$X_{i,j} = \frac{X_{i,j+1} + X_{i+1,j} + X_{i-1,j} + X_{i,j-1}}{4}$$

Proof. If X is a fixed point then for each position $X_{i,j}$ that is not a constant:

$$X_{i,j} = (1-p) \cdot X_{i,j} + p \cdot \frac{X_{i,j+1} + X_{i+1,j} + X_{i-1,j} + X_{i,j-1}}{4}$$

If we remove $X_{i,j}$ from both side of this equality and then we simplify it by p ($p \neq 0$), we obtain the lemma. □

Lemma 10. *If X is configuration without any constant then for all t the sum of all values in $\delta^{\dagger^{t+1}}$ is equal to the sum of all values in $\delta^{\dagger^{t}}$.*

Proof. We directly obtain that:

$$\begin{aligned}\sum_{i,j} \delta^{\dagger^{t+1}}(X)_{i,j} &= \sum_{i,j} (1-p) \cdot \delta^{\dagger^{t}}(X)_{i,j} + p \cdot \frac{\delta^{\dagger^{t}}(X)_{i+1,j} + \delta^{\dagger^{t}}(X)_{i,j+1} + \delta^{\dagger^{t}}(X)_{i-1,j} + \delta^{\dagger^{t}}(X)_{i,j-1}}{4} \\ &= \sum_{i,j} \delta^{\dagger^{t}}(X)_{i,j}\end{aligned}$$

□

Property 11. *Let X^0 be a configuration without any constant. If it exists, the limit of (X^t) is $\overline{X^0}$ (the average value of all cells of X^0).*

Proof. We assume the limit X^∞ of (X_t) exists.

First of all, we prove that all the values of X^∞ are equal.

Let us consider a value d contained into X^∞ and assume not all the cells of X^∞ are equal to d.

First case There exists a cell in X^∞ that contains a value strictly lower than d. We consider a cell C of X^∞ containing the lowest value e of X^∞ and such that one of its 4 neighbours is different from e (at least one cell contains $d > e$, so C exists). According to lemmas 8 and 9, the value of C must be the average of its 4 neighbours. Since e is the lowest value of X^∞ and one of the neighbours of C contains a value different from e (so strictly greater than e), C cannot contains the average of its 4 neighbours. It is absurd.

Second case There exists a cell in X^∞ that contains a value strictly greater than d. As previously, we prove it is absurd.

Finally, it is absurd that one of the cells of X^∞ contains a value different from d. All the cells of X^∞ must be equal.

According to lemma 10 the sum of all values does not change at any step of the computation. Thus, all the cells of x^∞ must be equal to the average value of all the initial cells in X^0. □

Property 12. *Let X^0 be a configuration with only one constant c. If it exists, the limit of (X^t) is the configuration with all its cells equal to c.*

Proof. We assume the limit X^∞ of (X_t) exists.

As in property 11, we prove by contradiction that all the cells of X^∞ must be equal to c, in the corresponding proof we choose c for the value d and the corresponding cell C (that is supposed to contain the lowest or greatest value of the configuration, different from $c = d$) cannot be a constant cell. □

Question 11

Property 13. *For a p-average automaton with constants, $\delta^{\dagger}$ can be non-linear.*

Proof. Let's consider the following local configuration:

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

We assume that 1 is a constant, but 0 not. We take $p = 0.5$.

We have:

$$\begin{aligned} \delta\left(\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}\right) &= \delta\left(\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}\right) \\ &= 0.5 \cdot 2 = 1 \end{aligned}$$

However:

$$\delta\left(\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}\right) + \delta\left(\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}\right) = 1 + 1 = 2$$

δ is not linear, and so $\delta^{\dagger}$ too. This result proves that $\delta^{\dagger}$ can be non-linear for a *p-average automaton* with constants. ☐

Question 12

Question 13

See the implementation in file `constants.c`.