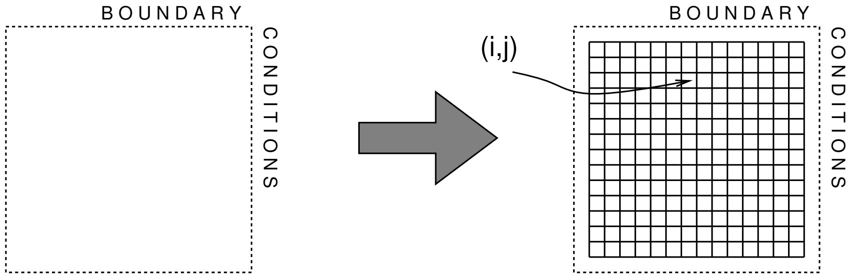
1 Jacobi Relaxation

Jacobi relaxation is an iterative algorithm which, given a set of boundary conditions, finds (discretized) solutions to differential equations of the form $\nabla^2\mathcal{A} + \mathcal{B} = 0$. We begin by choosing the grid which will form the basis of our discretization:

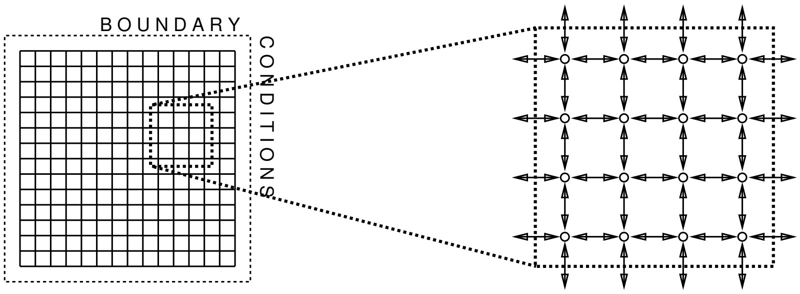


To find a solution on a grid, we repeatedly apply the following iterative step until we converge on a solution.

$$A^{k+1}_{\mathrm{i,j}} = \frac{A^k_{\mathrm{i+1,j}} + A^k_{\mathrm{i-1,j}} + A^k_{\mathrm{i,j+1}} + A^k_{\mathrm{i,j-1}}}{4} + b_{\mathrm{i,j}} \tag{1}$$

Jacobi differs from other iterative relaxation algorithms in that the update of each point (at iteration step $k + 1$) requires the *previous* values of the neighboring points (from iteration step k).

We use a simple graphical representation to capture those features we are interested in and abstract away excess detail. In this representation, graph nodes represent fixed amounts of computation; graph edges represent communication between computation nodes. In such a representation, a single iteration of Jacobi relaxation looks as follows:



In this graph, each node represents the computation required to compute a new value for a single grid point. To compute a new value for a grid point, Jacobi relaxation dictates that we average the previous values of each of the neighboring grid points – thus each node is connected to its four neighboring nodes with an edge that represents the communication of two grid values (one in each direction).

The following figure shows one step of a relaxation on a four by four grid. The A^k matrix contains the values of the grid at step k, and the A^{k+1} matrix has the values for the next step. The boundary conditions are shown on the edges of the A^k and A^{k+1} matrices. The B matrix, which is constant, contains the values that are added to the grid points on each step according to Equation 1.

A^k

	0	*0*	*0*	*0*	
4	6	4	5	0	*0*
4	8	5	8	3	*0*
4	1	8	7	8	*0*
4	4	3	4	2	*0*
	0	*0*	*0*	*0*	

B

4	0	1	2
0	–3	–1	0
2	0	–3	1
2	0	–3	5

A^{k+1}

	0	*0*	*0*	*0*	
4	8	4	4	4	*0*
4	4	4	4	4	*0*
4	8	4	4	4	*0*
4	4	4	0	8	*0*
	0	*0*	*0*	*0*	

You might want to think about the values in the A^{k+2} matrix after the next relaxation step.

There are a number of ways to partition the data in this grid to a bunch of processors. Figure 1 shows two possible ways to partition the four by four grid to four processors. One partition allocates one row of the matrix to each processor, and the other allocates a two by two submesh of the matrix to each processor. For every step in the relaxation, *each processor will calculate the new values for its own matrix cells.* To accomplish this calculation, the processors need to get cell values from neighboring processors. The values on the arrows between the nodes represent the data that must be communicated between processors.

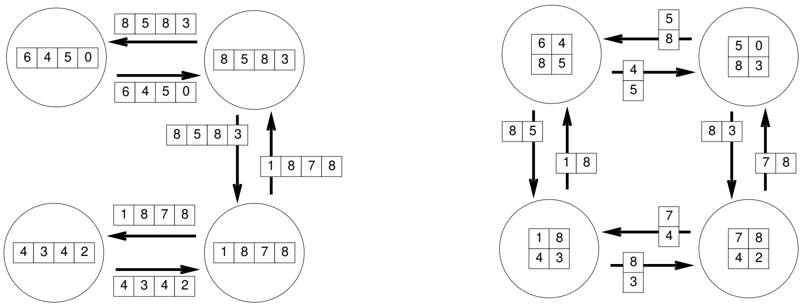


Figure 1: Partitioning by rows and by square tiles.