Parallel Sorting on a Spatial Computer

21 October 2012
RACES workshop at SPLASH'12

Max Orhai and Andrew P. Black
Portland State University

Maseeh College of Engineering and Computer Science
Undergraduate Research and Mentoring Program

Supported by a gift from **IBM Research**

what is a *spatial computer*?

(definition based on 2012 Spatial Computing Workshop)

a networked system of computing devices distributed through a physical space, in which:

the 'problem'

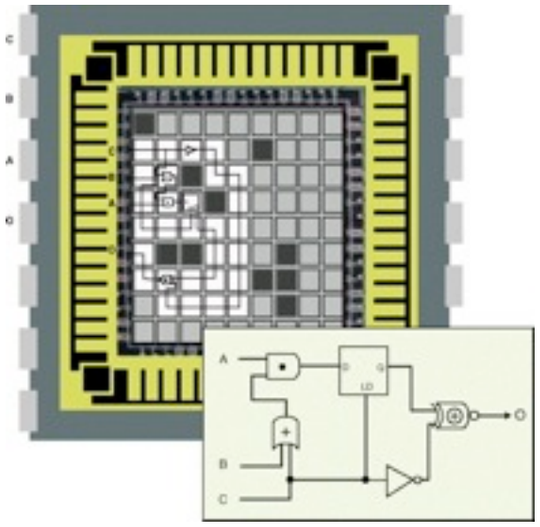
1. the *difficulty* of moving information between any two devices depends on the physical *distance* between them, and

the 'solution'

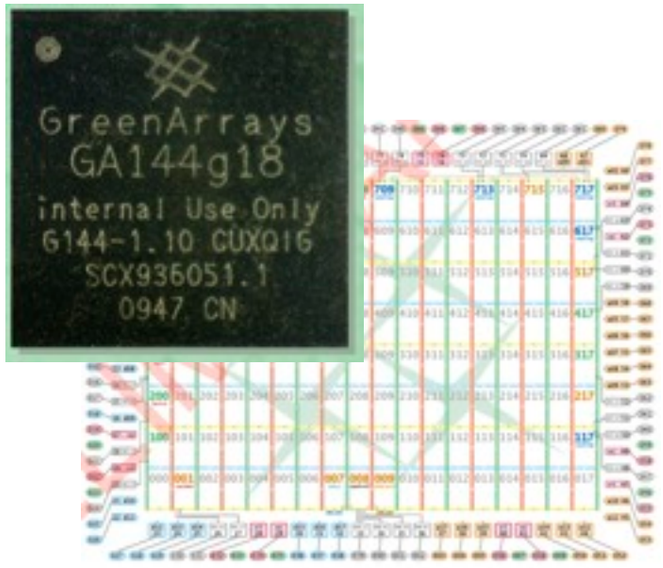
2. the *functional goals* of the system are defined in terms of the system's *spatial structure*.

are these spatial computers?

fine-grained → coarse-grained



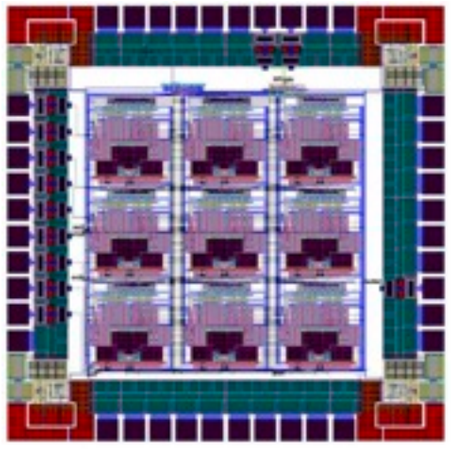
cell matrix tile
(Macias & Durbeck 2009)



GreenArrays *GA144*
asynchronous stack
machine array
(2010)

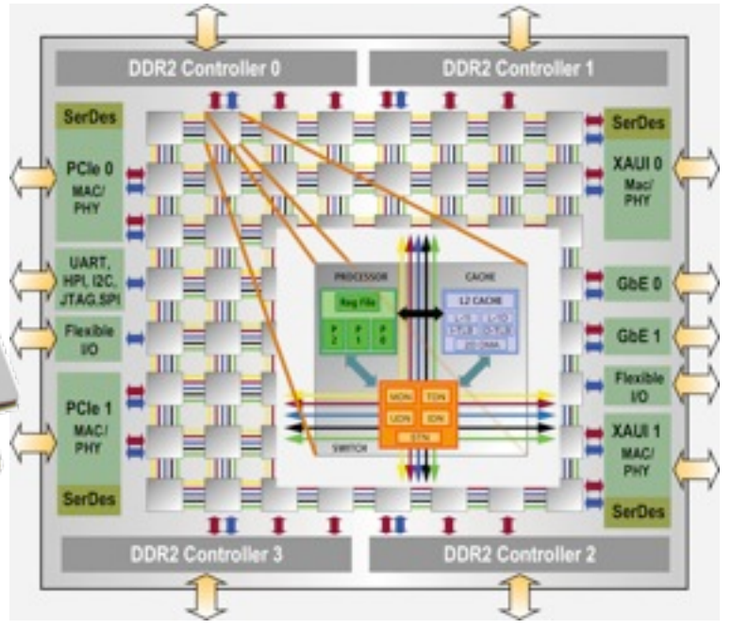
Inmos *transputer* array
(early 1980s)

XMOS *XMP-64*
dev kit (2010)



asynchronous logic automata
cell (Chen 2009)

Tilera *Tile64* (2008)



a *spatial computer* is

a networked system of computing devices distributed through a physical space, in which:

the 'problem'

1. the *difficulty* of moving information between any two devices depends on the physical *distance* between them, and

the 'solution'

2. the *functional goals* of the system are defined in terms of the system's *spatial structure*.

message of this talk

spatial computing offers real insights into

1. the *costs* and *constraints* of communication in large parallel computer arrays
2. how to design *algorithms* that respect these costs and constraints

match the *communication structure* of the program to the *physical structure* of the computer network

our example: collision sort

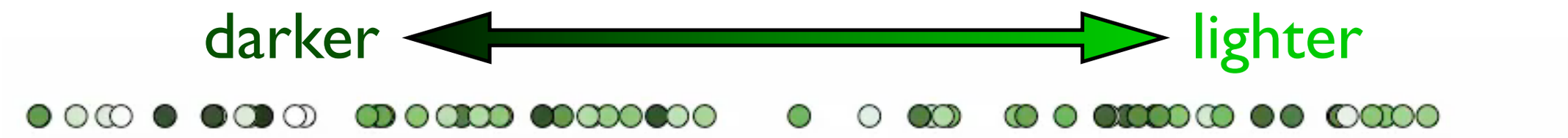
approach:
an entropy-reducing particle system

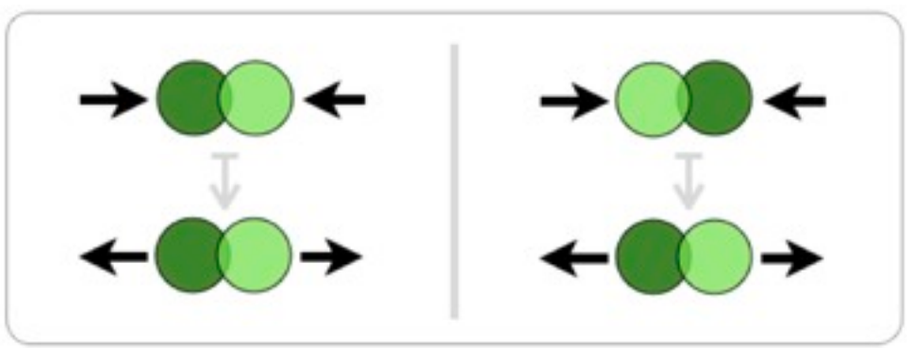
goal:
arrange particles by color

simulation physics ~ *computing machinery*

space ~ order
particles ~ data
collisions ~ comparisons

distributed computation of a *global* solution using only *local* information and *minimal* communication

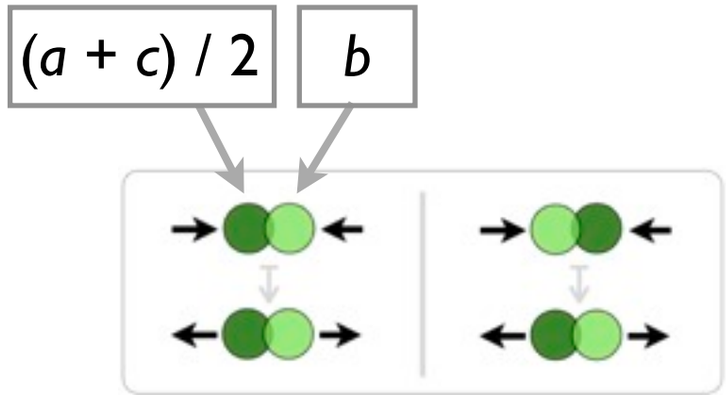




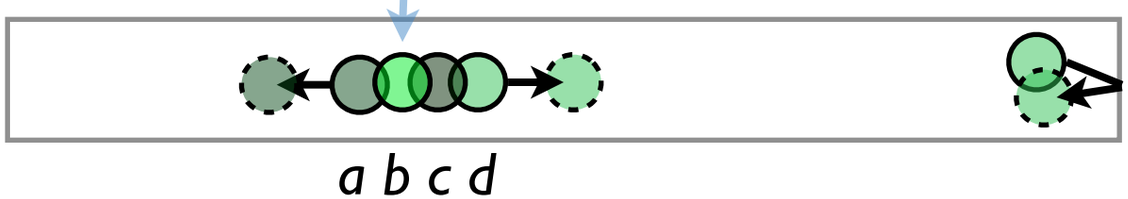
collision sort: mechanism

physics		*computation*
space	~	order
particles	~	data
collisions	~	comparisons

in a *single* step, the color of particle *b* is compared with the *average color* of overlapping particles *a* and *c*



multi-way collisions quantize time



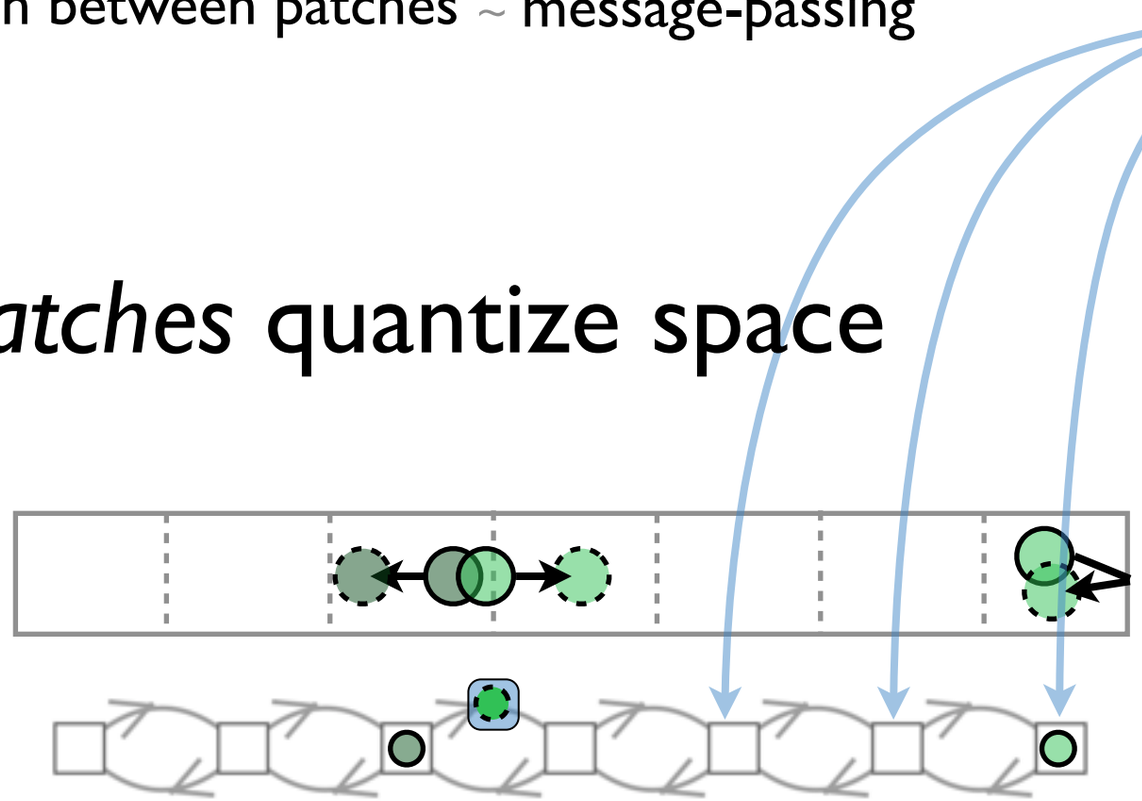
‘boundary collisions’ always cause rebound

collision sort: mechanism

physics		*computation*
space	~	order
particles	~	data
collisions	~	comparisons
patches of space	~	processors
motion between patches	~	message-passing

collisions *across* patches are *not* detected!

patches quantize space

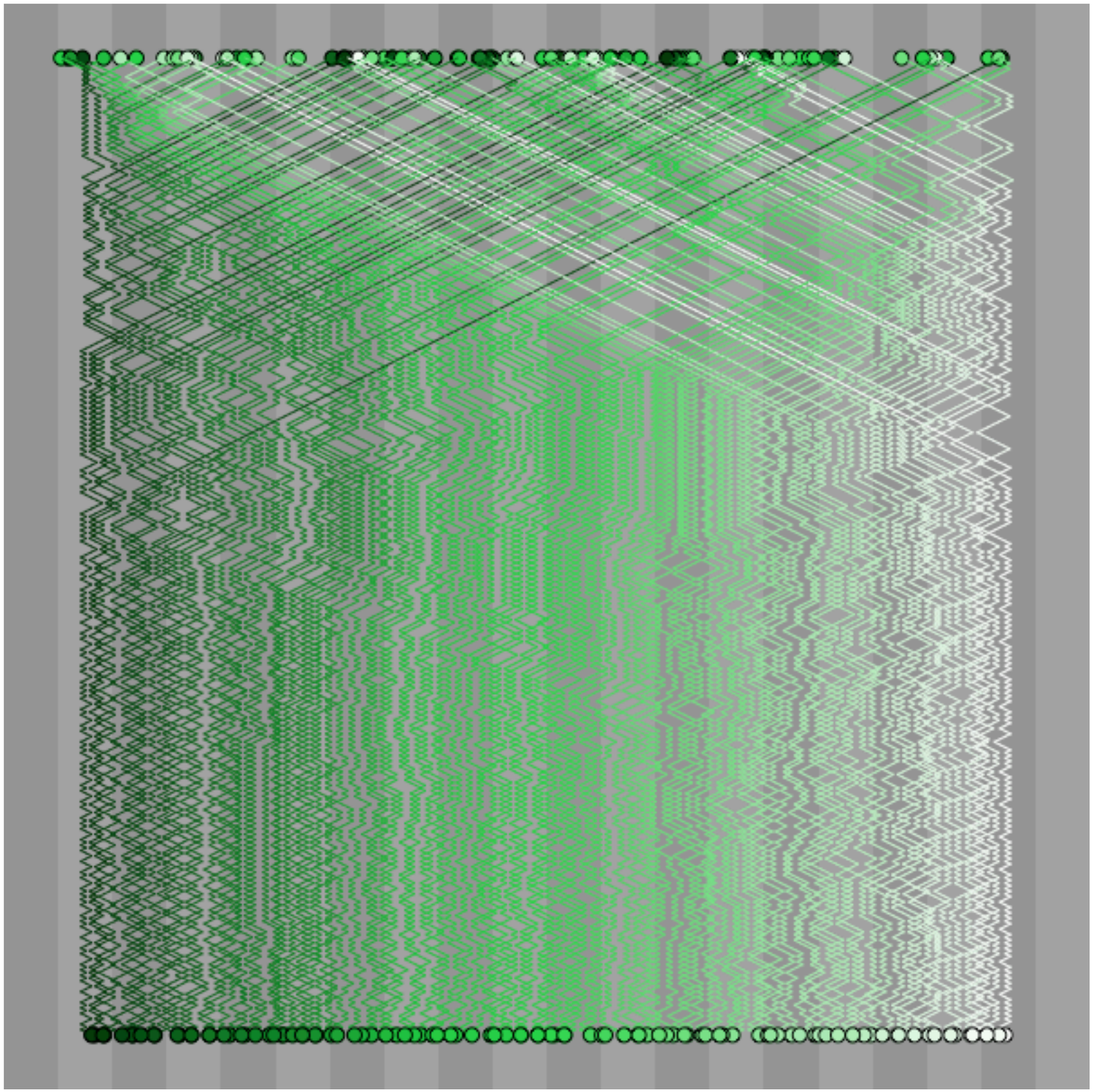


loop (one *step*):

- accept incoming particles from neighbors
- detect collisions and alter velocities accordingly
- increment positions, sending outgoing particles to neighbors

collision sort: low speed

100
particles



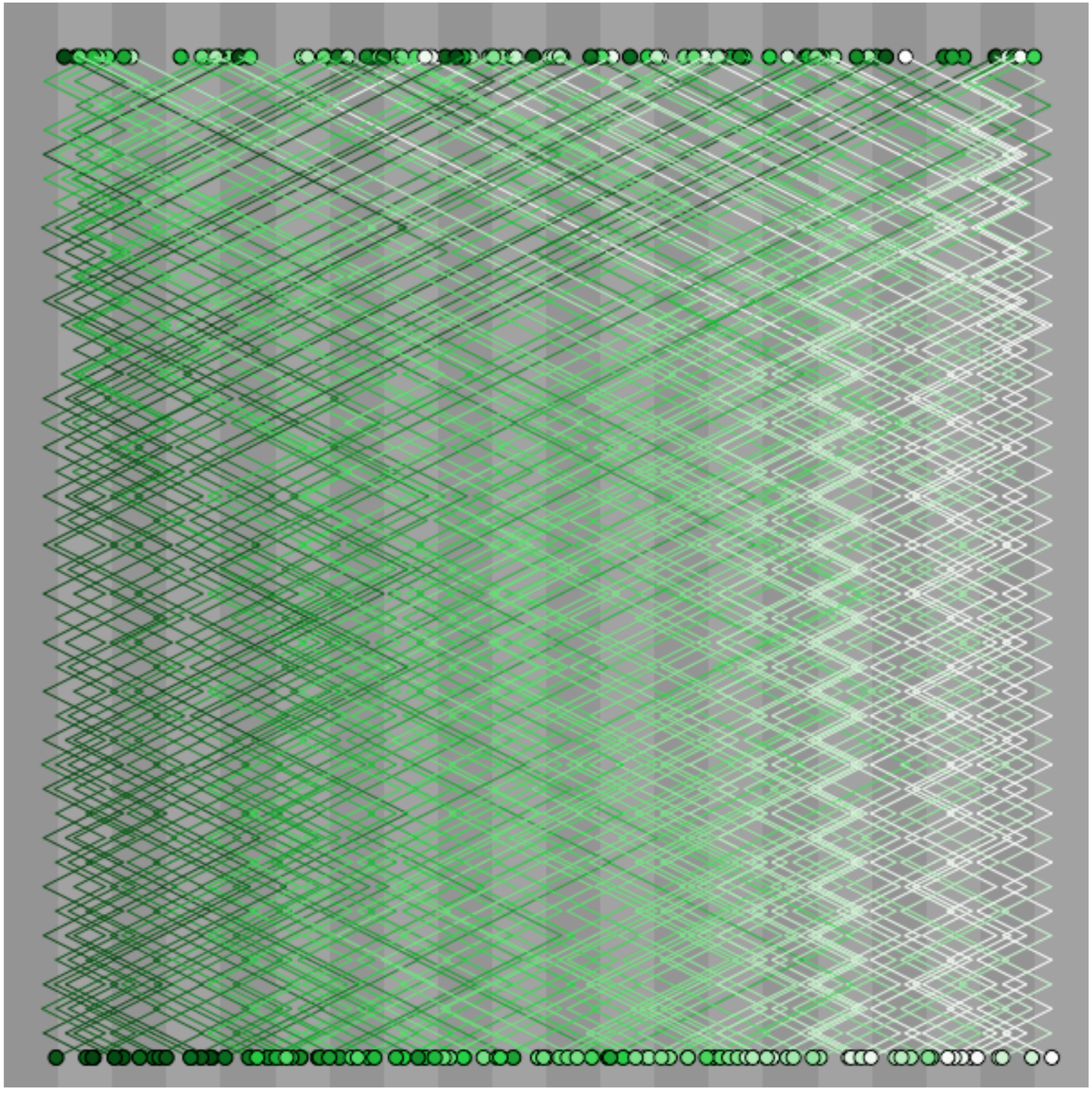
constant
speed:

0.1

patch-widths
per step

collision sort: high speed

100
particles



constant
speed:
0.9

patch-widths
per step

collision sort: variable speed

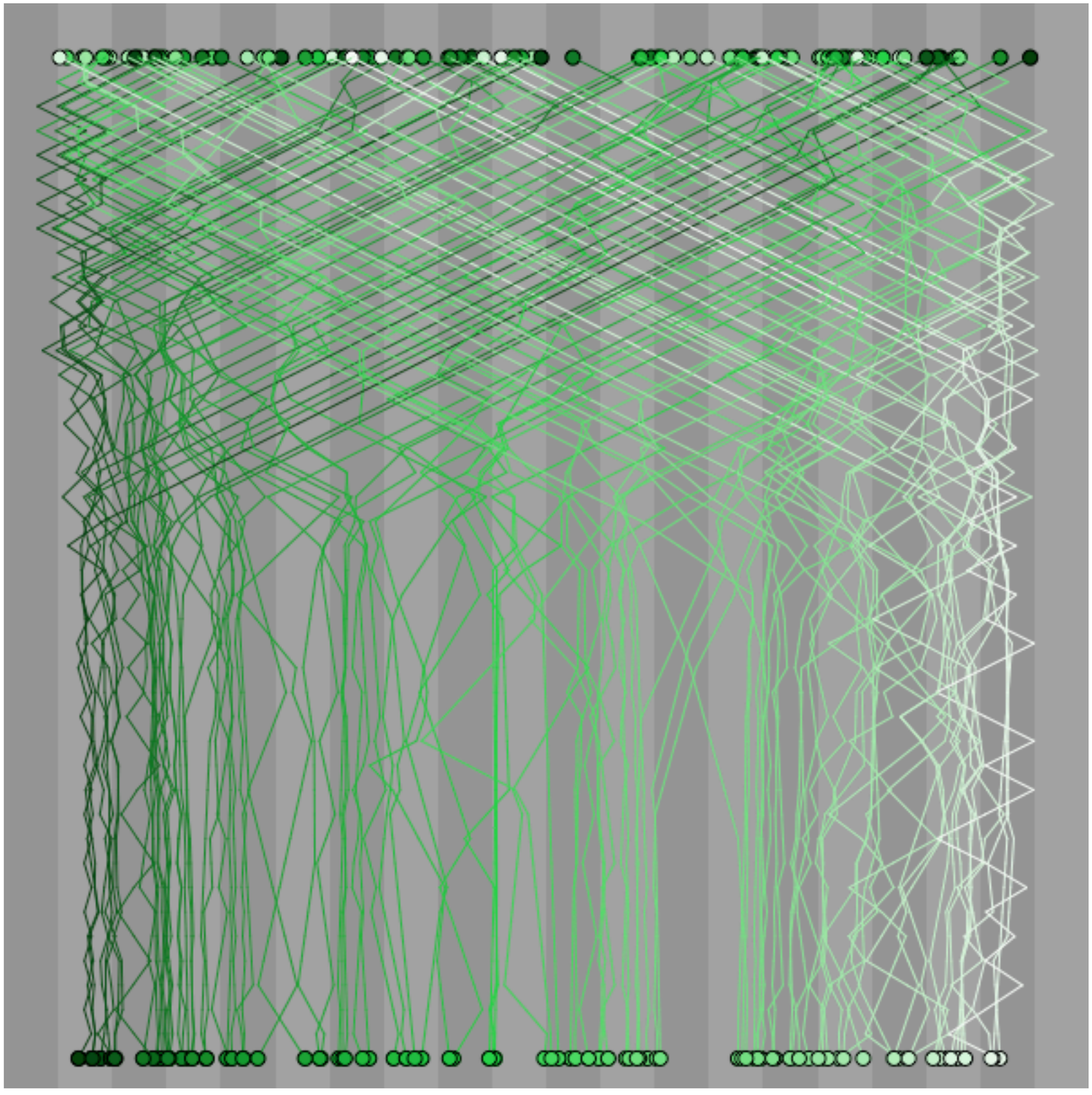
100 particles

speeds proportional to color differences

maximum speed:

0.9

patch-widths per step



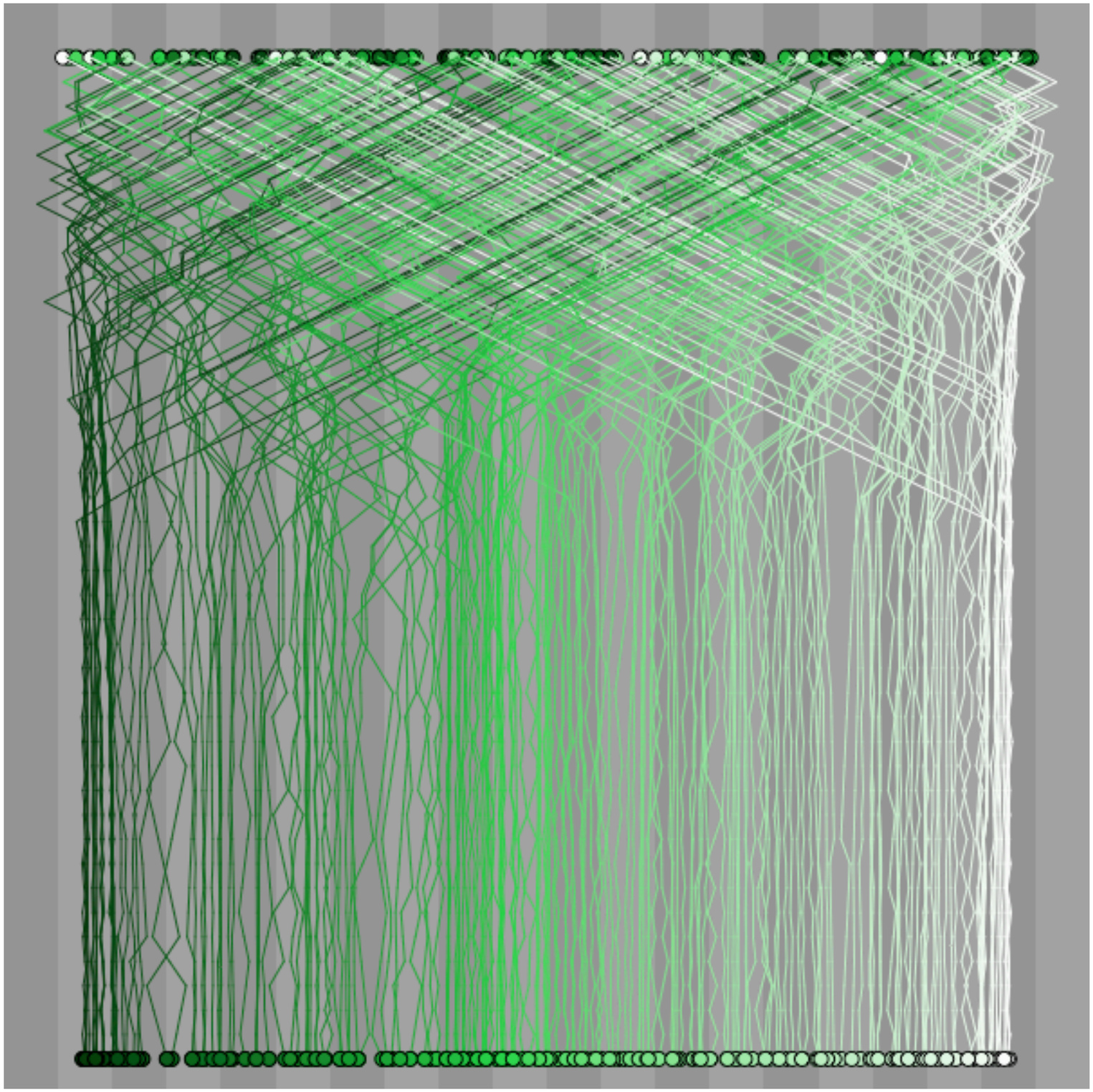
collision sort: more data

200
particles

speeds
proportional
to color
differences

maximum
speed:
0.9

patch-widths
per step



0

time

40
steps

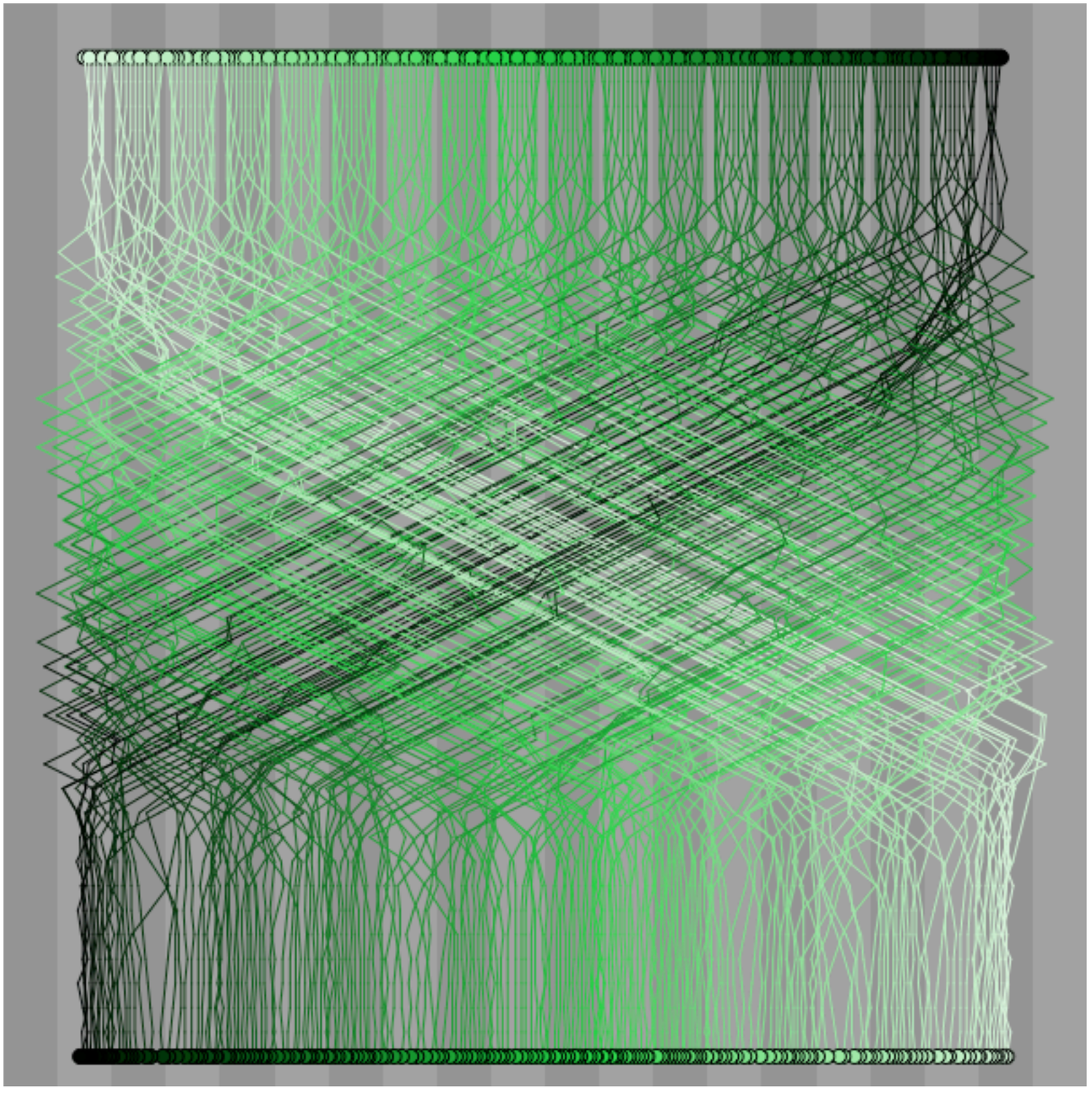
collision sort: worst case?

200 particles in reverse order

proportional speeds

maximum speed: 0.9

patch-widths per step



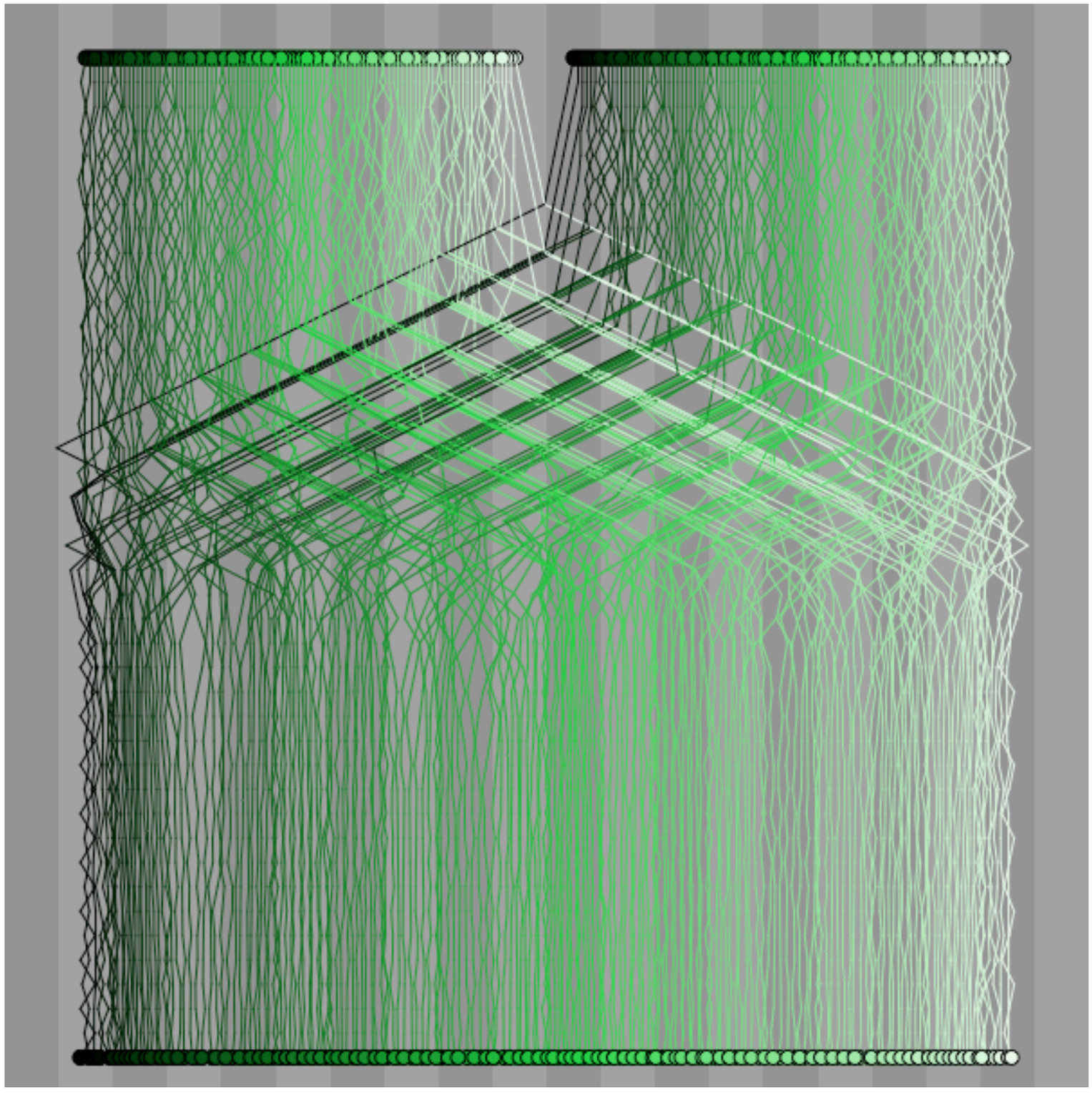
collision sort: worst case?

200
particles
in two sorted
subsequences

proportional
speeds

maximum
speed:
0.9

patch-widths
per step



collision sort: in 2D

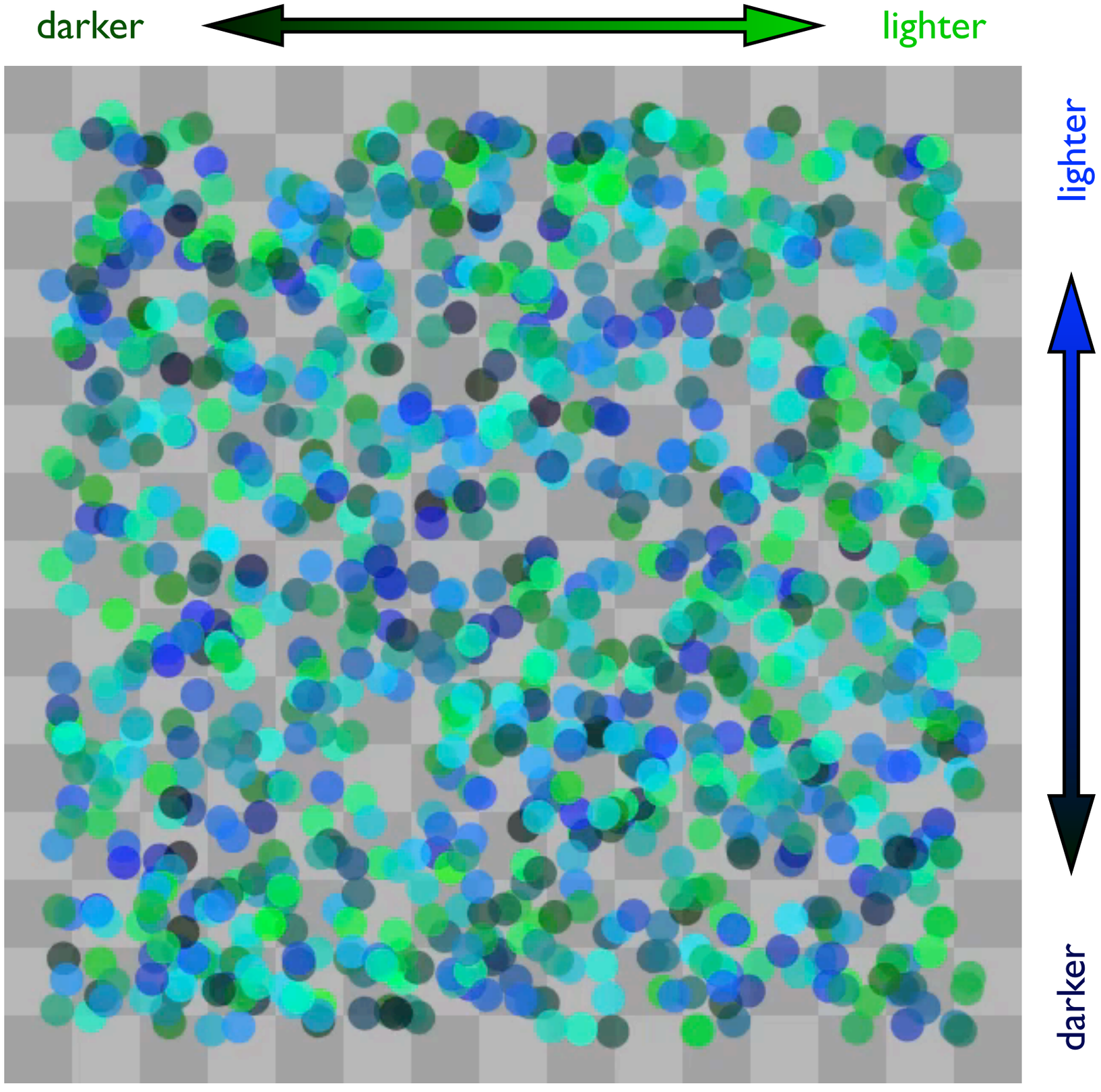
1000 particles each with *both* blue *and* green values

proportional speeds

maximum speed:

0.9

patch-widths per step per axis



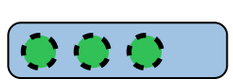
collision sort: performance

the total cost of any parallel
algorithm is a combination of
local computation costs and
communication costs

for simplicity, we assume that
local computation is cheap,
and focus on the
cost of *communication*
between system components

collision sort: performance

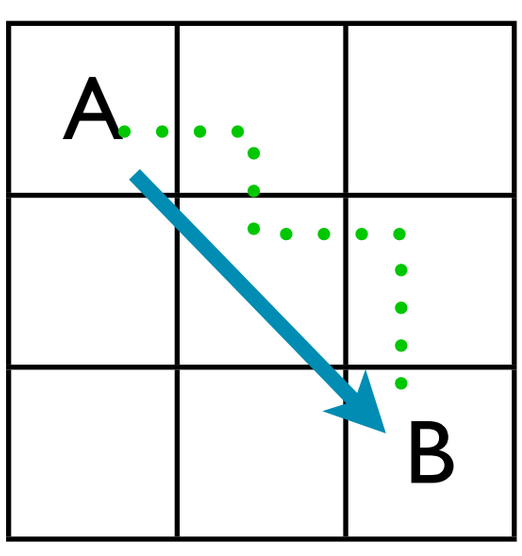
messages may convey *multiple* particles



because each processor need only

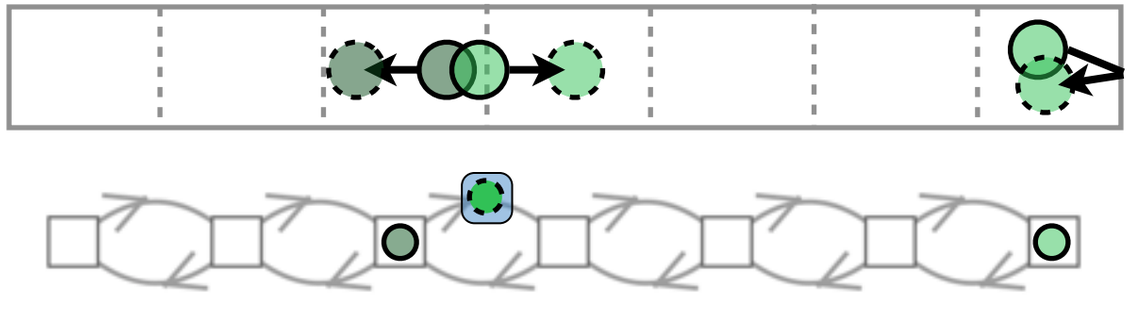
- *receive* at most one and
- *send* at most one message

per neighbor during each loop cycle,



communication cost is *the minimum possible*, under these assumptions:

- a one-patch-per-step information speed limit
- message cost independent of message size
- local computation is free



collision sort: performance

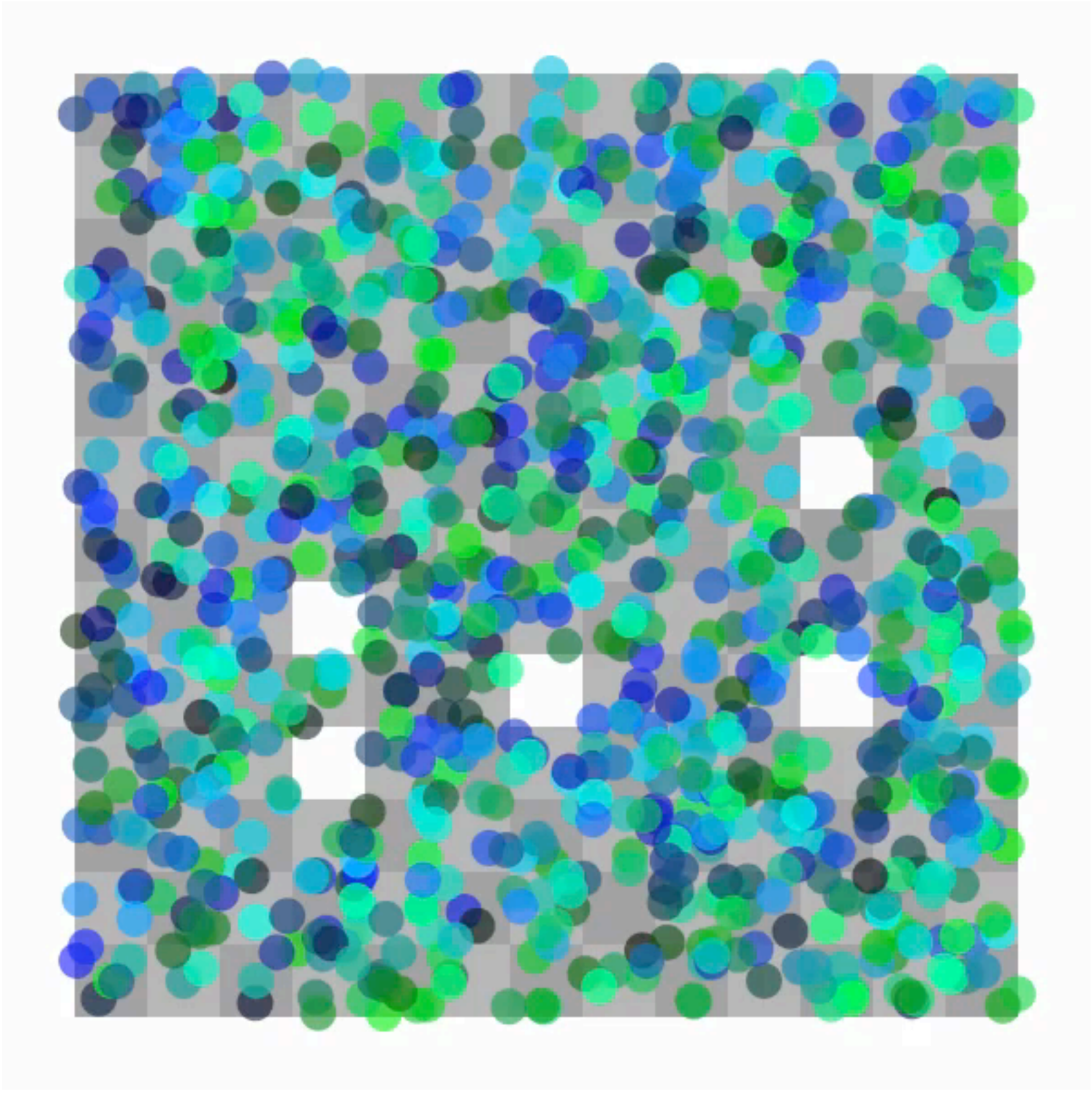
what about *component failure*?

if messages are *acknowledged*, then any 'undeliverable' particles can simply have their velocities reversed, just as in a boundary collision:

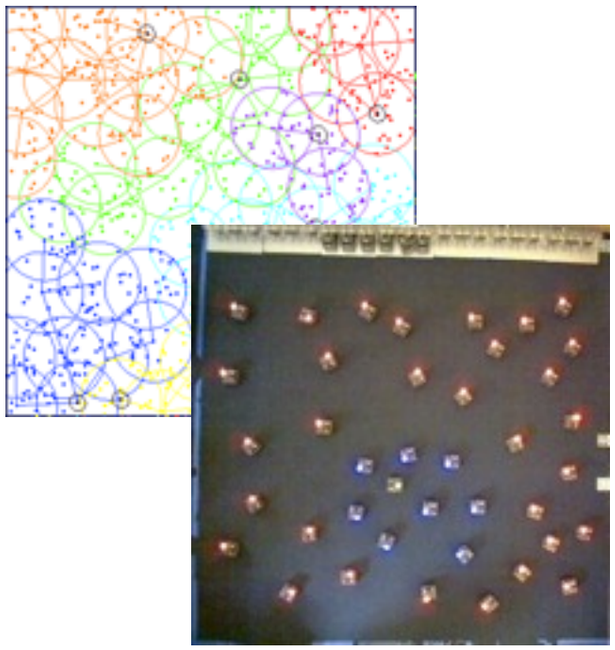
particle bounce ← message bounce

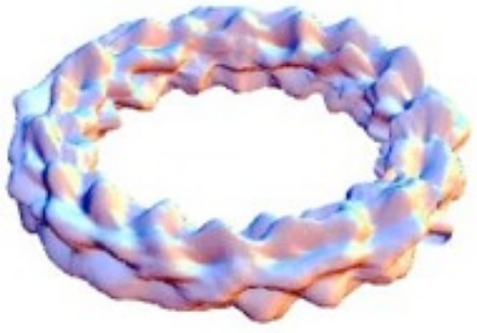
fault tolerance is an 'emergent property' of this rule!

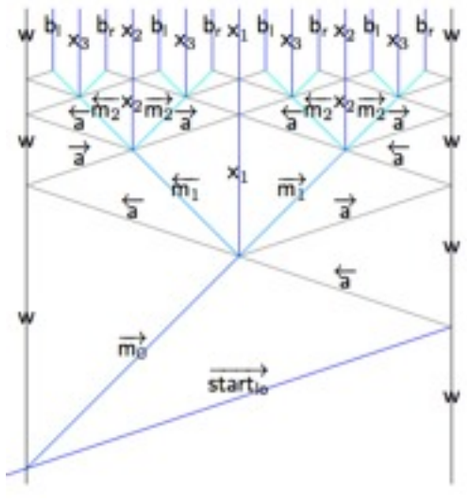
collision sort: fault-tolerance



related work: spatial computing







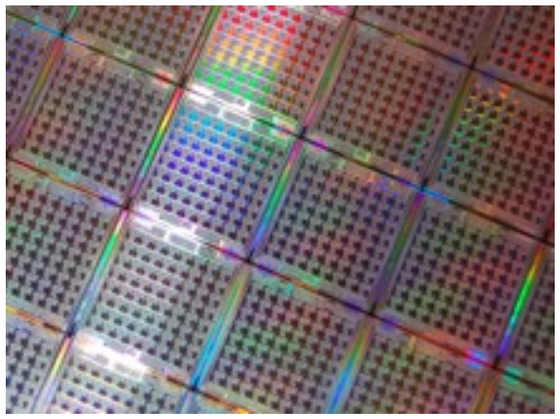
research efforts:

- MIT CSAIL *amorphous computing* project, c. 2000
  → BBN *Proto* language
- Laboratoire IBISC (Computer Science, Systems Biology and Complex Systems) at U. d'Evry
  → *MGS* topological simulation language
- abstract geometric computation (*continuous signal machines*)
  D. Duchier, J. Durand-Lose, M. Senot at U. d'Orleans

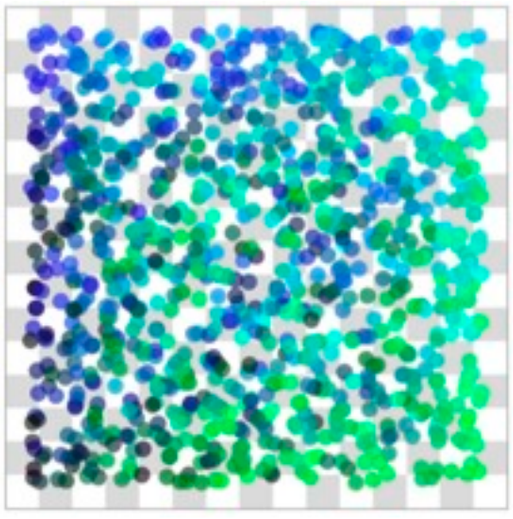
application domains:

- large-scale sensor networks
- distributed control systems
- mobile robot swarms
- biological modeling
- computability theory

conclusion



Tilera processor arrays



large-scale parallel computer arrays *can b*e spatial computer systems, if programmed in a way that respects physical distances.

spatial computing gives us a *cost model* that approximates the physics of information transmission, and that *leverages our intuition* about physical objects to design concurrent algorithms.

references:

- http://www.spatial-computing.org
- http://www.greenarraychips.com/
- http://www.tilera.com/
- http://www.xmos.com/

- Abelson *et al. Amorphous computing*. Communications of the ACM 2000.
- Gershernfeld *et al. Reconfigurable asynchronous logic automata (RALA)*. POPL 2010.
- Macias & Durbeck. *The Cell Matrix: an architecture for nanocomputing*. Nanotechnology 2001.
- Duchier, Durand-Lose, and Senot. *Solving Q-SAT in bounded space and time by geometrical computation*. 7th Intl. Conf. Computability in Europe 2011.