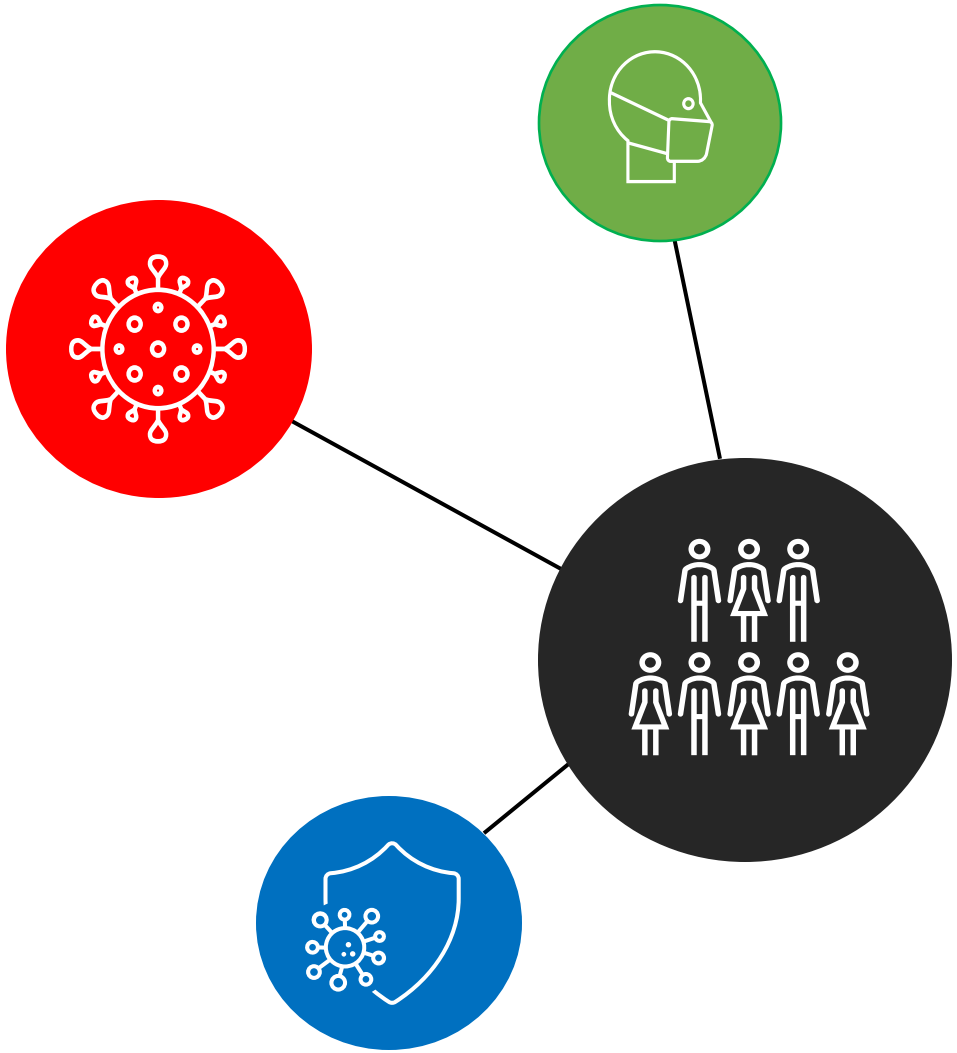
Federico Muscarà *Marco Parentin* *Leonardo Straccali* *Francesco Zanasi*

Modeling Infection Propagation in Populations

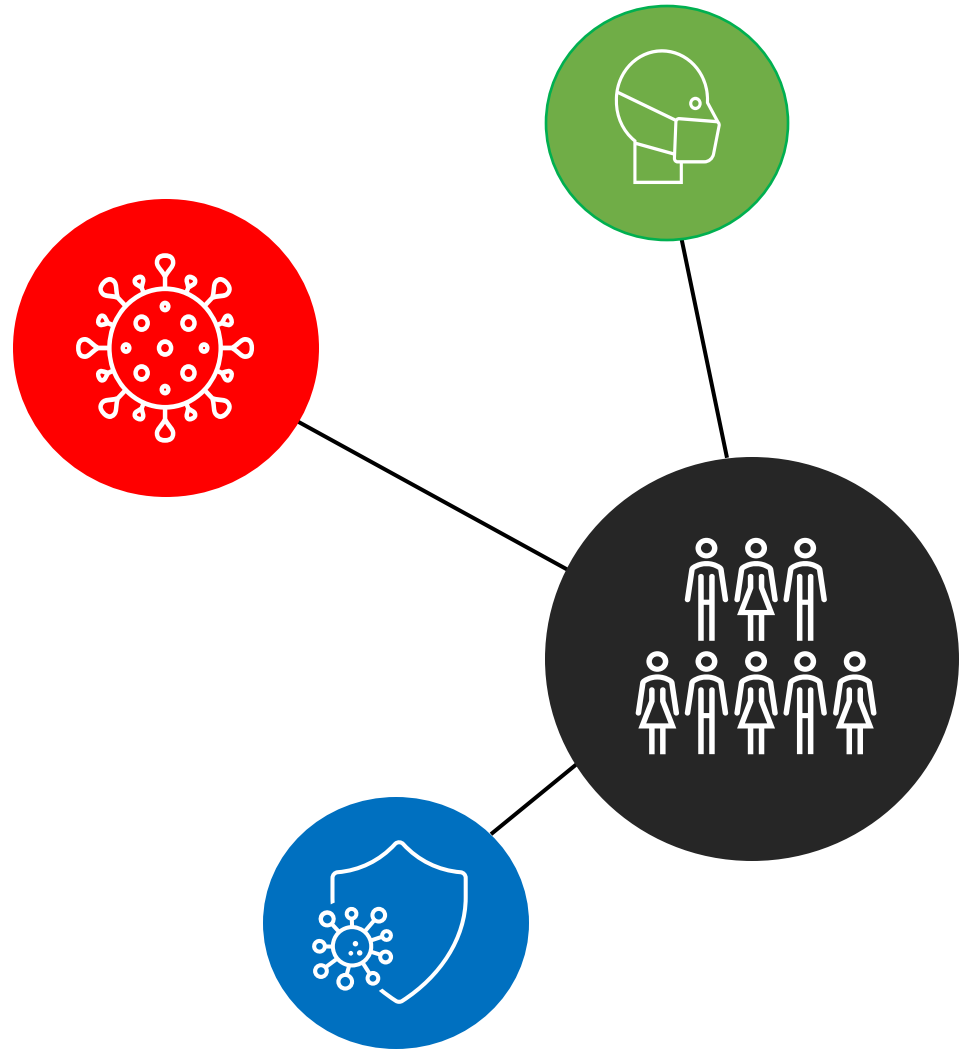


High Performance Computing Project

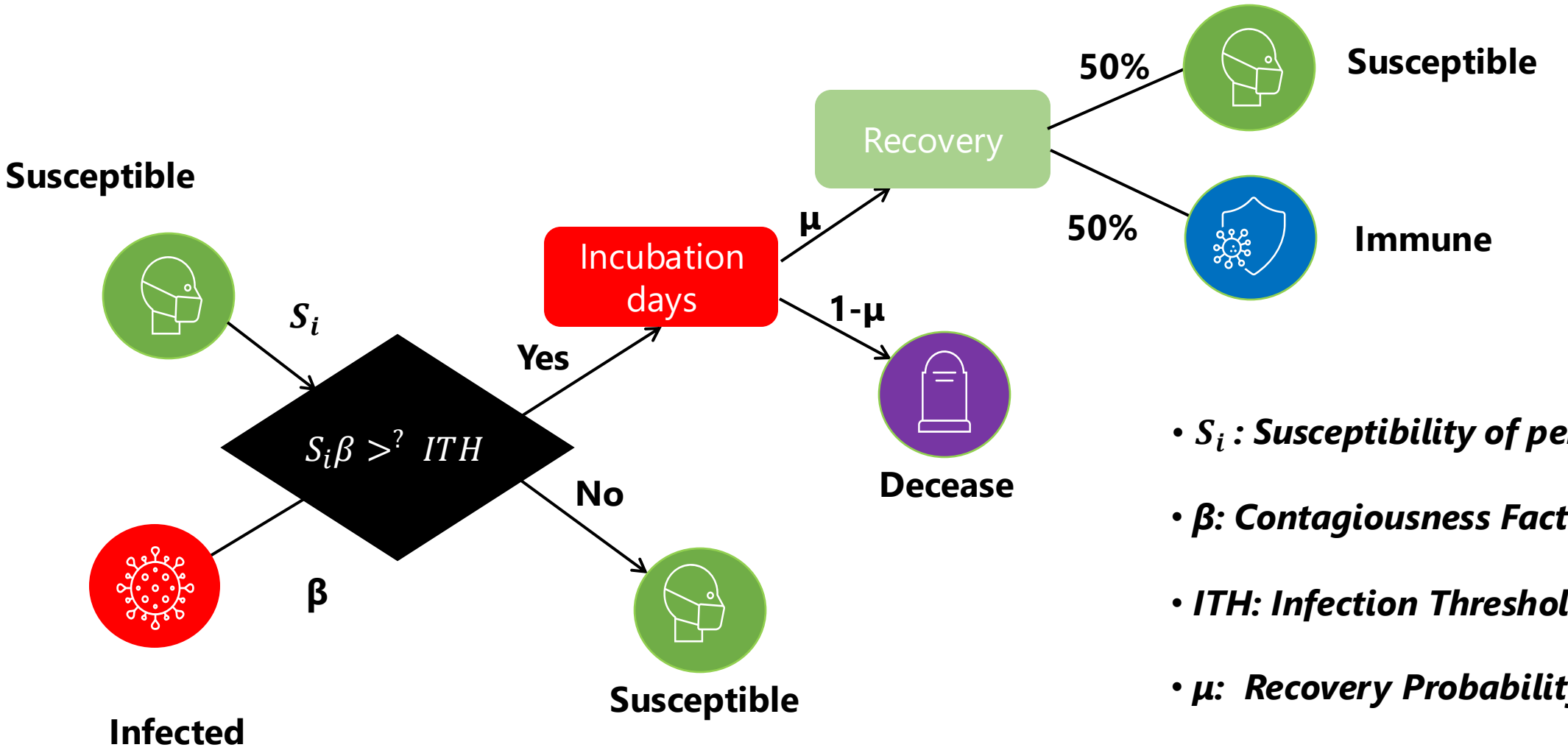
Politecnico di Torino

Presentation Overview

- Introduction
- Solution Description
- Experimental Results
- Conclusions

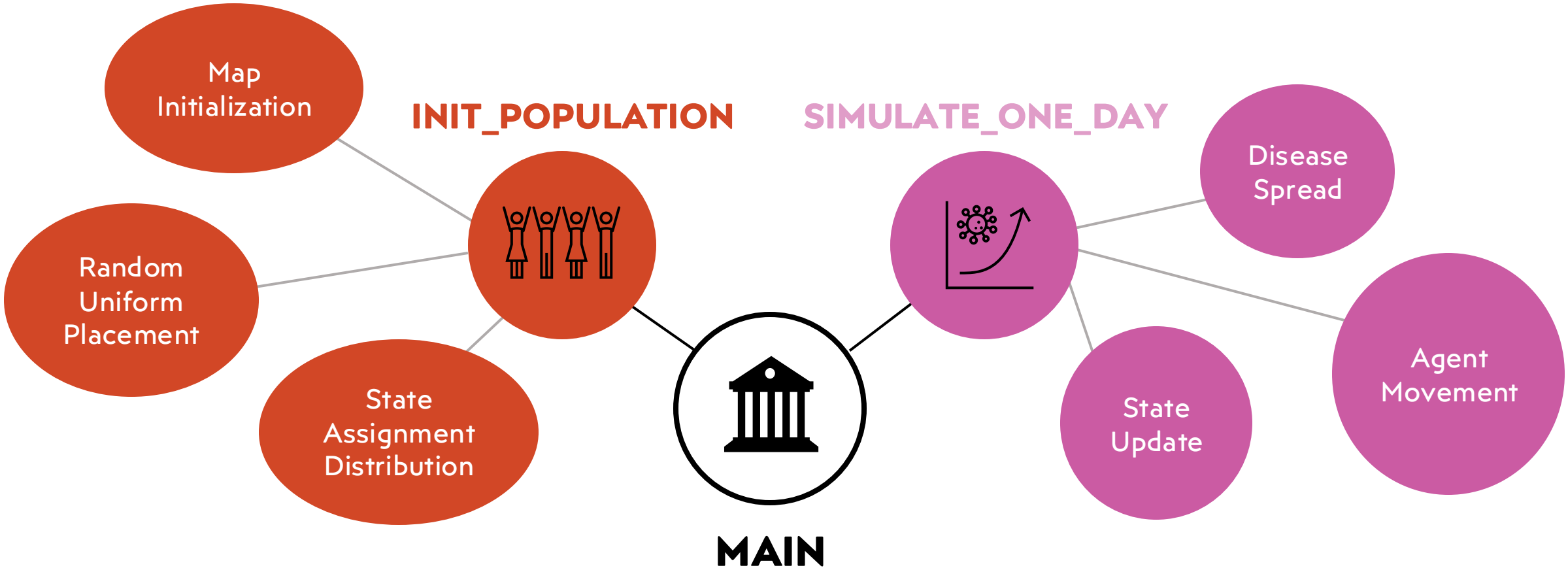


Introduction: The Kermack-McKendrick (SIR) Model

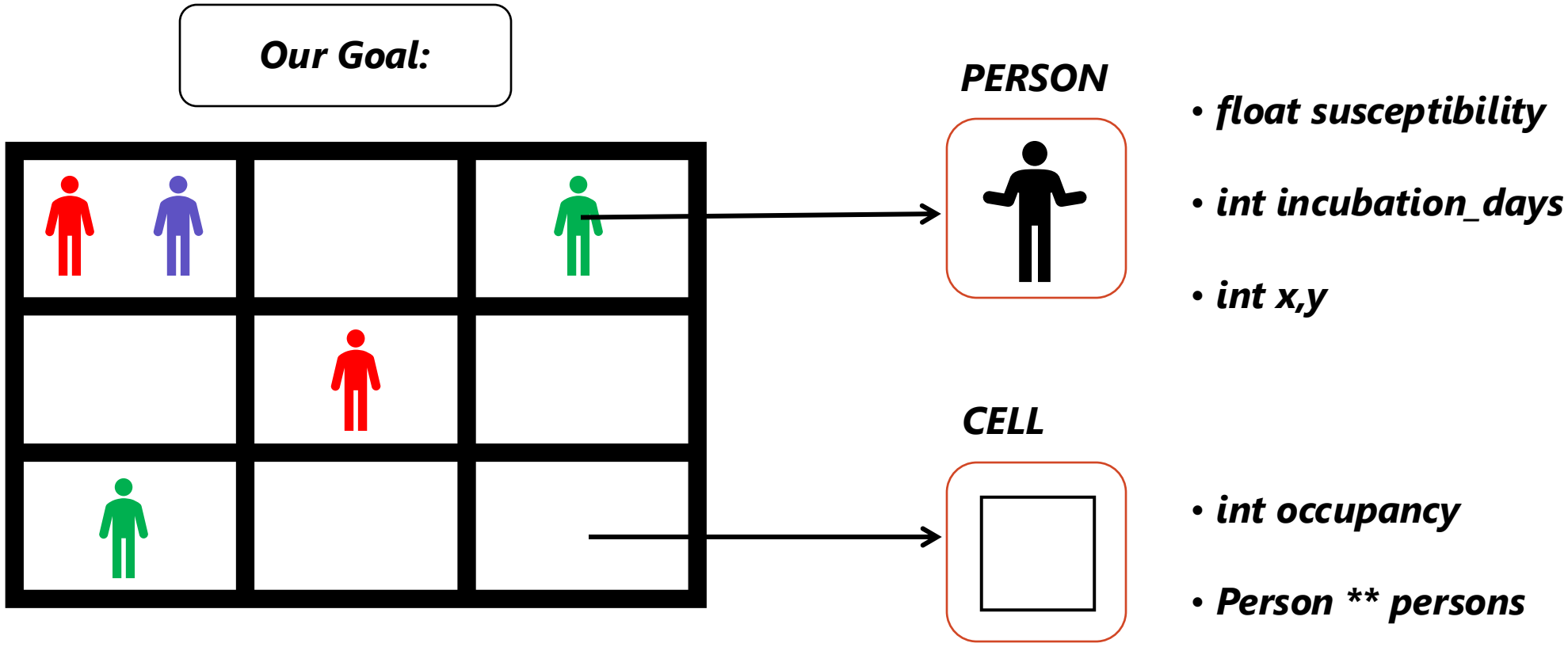


- S_i : *Susceptibility of person* i
- *β: Contagiousness Factor*
- *ITH: Infection Threshold*
- *μ: Recovery Probability*

Solution Description: SERIAL Version – Program Organization

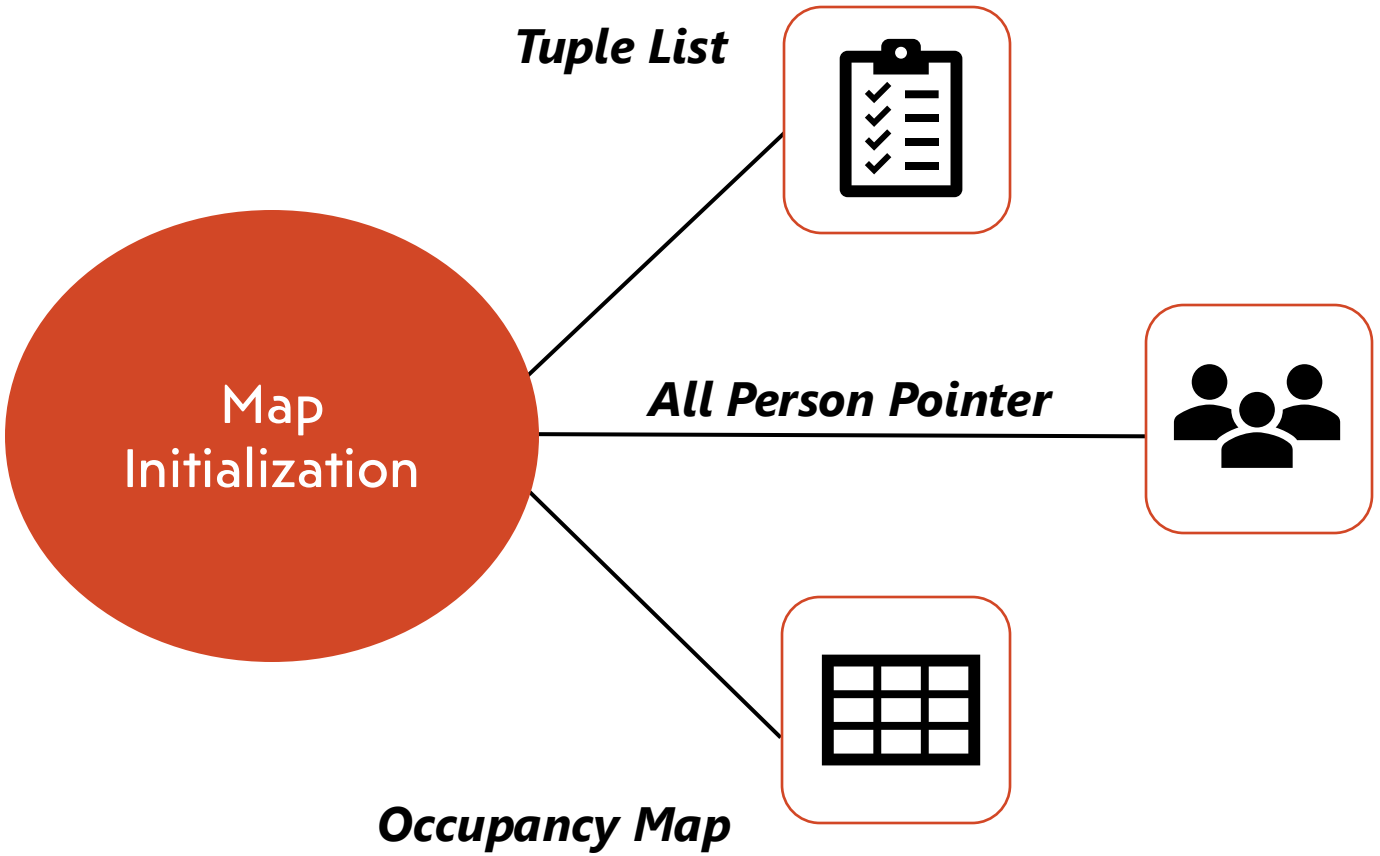


Solution Description: SERIAL Version – Population Initialization



- *Each Cell contains at maximum "MAXP_CELL" individuals*

Solution Description: SERIAL Version – Population Initialization

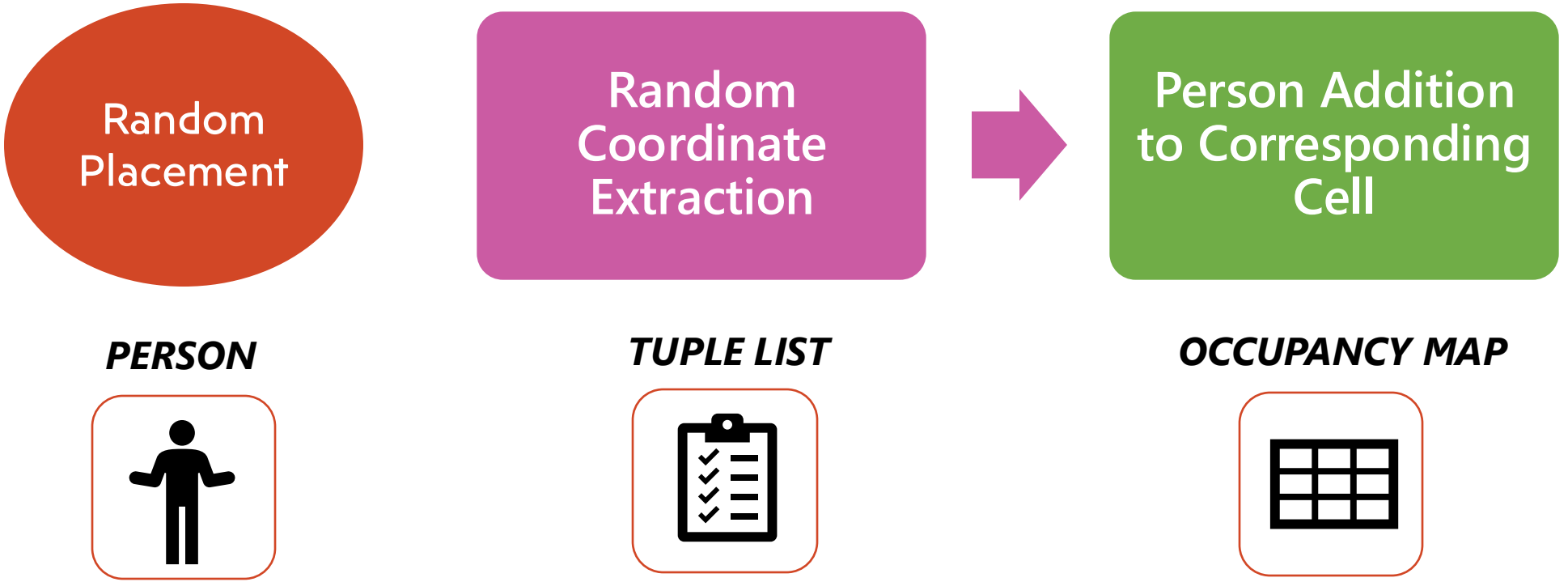


Generation af all WxH coordinates

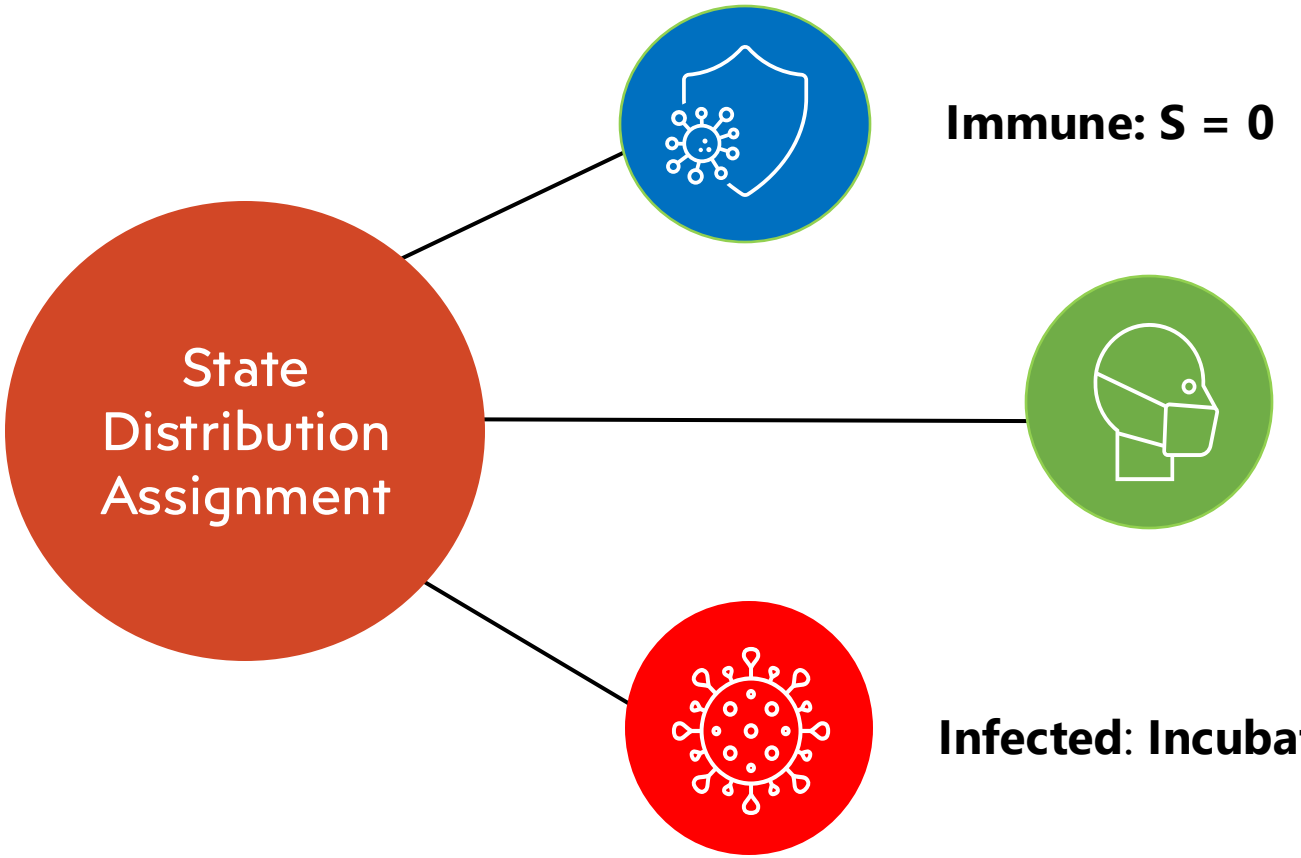
Cells occupancy set to 0

Each Cell points to MAXP_CELL slots of All Person Pointer

Solution Description: SERIAL Version – Population Initialization

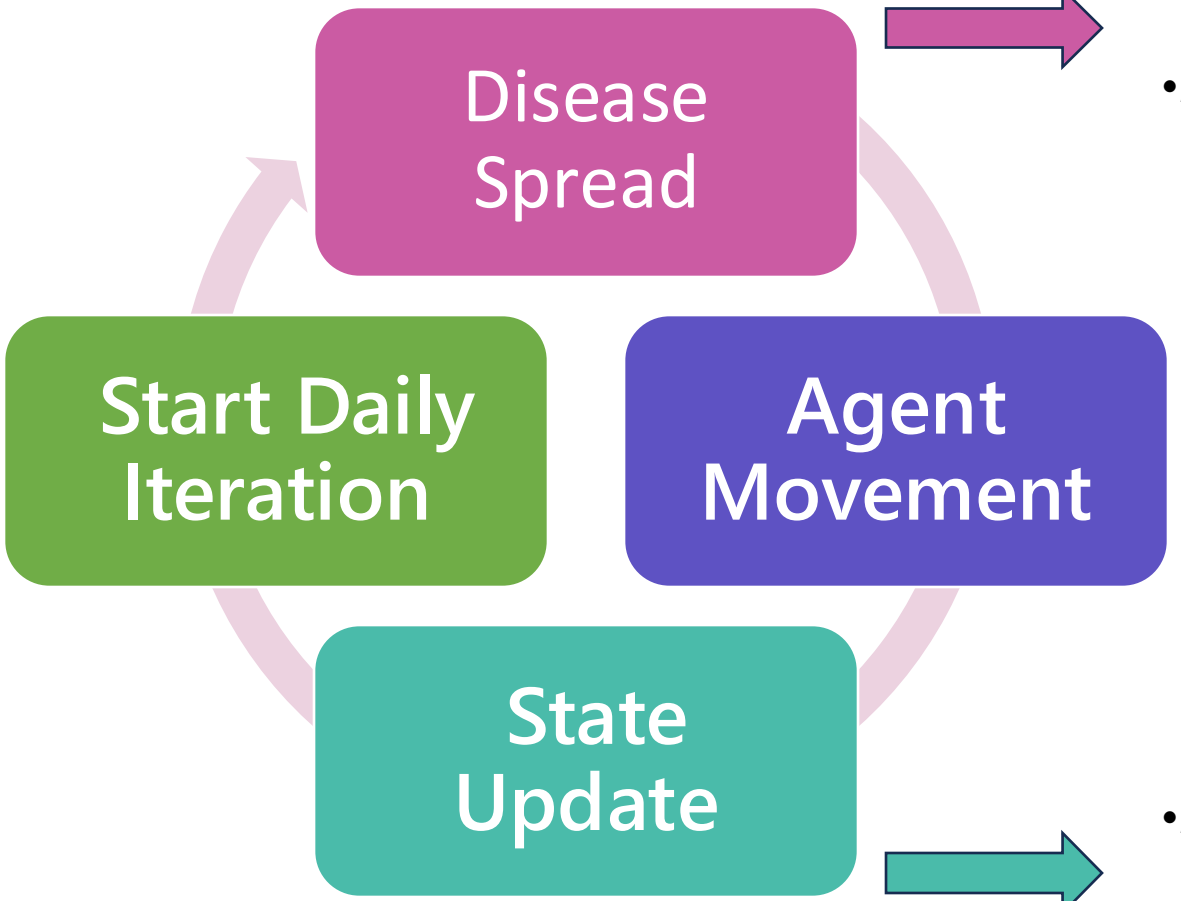


Solution Description: SERIAL Version – Population Initialization



- ***INFP: Initial infected percentage***
- ***IMM: Initial immune percentage***

Solution Description: SERIAL Version – Daily Simulation

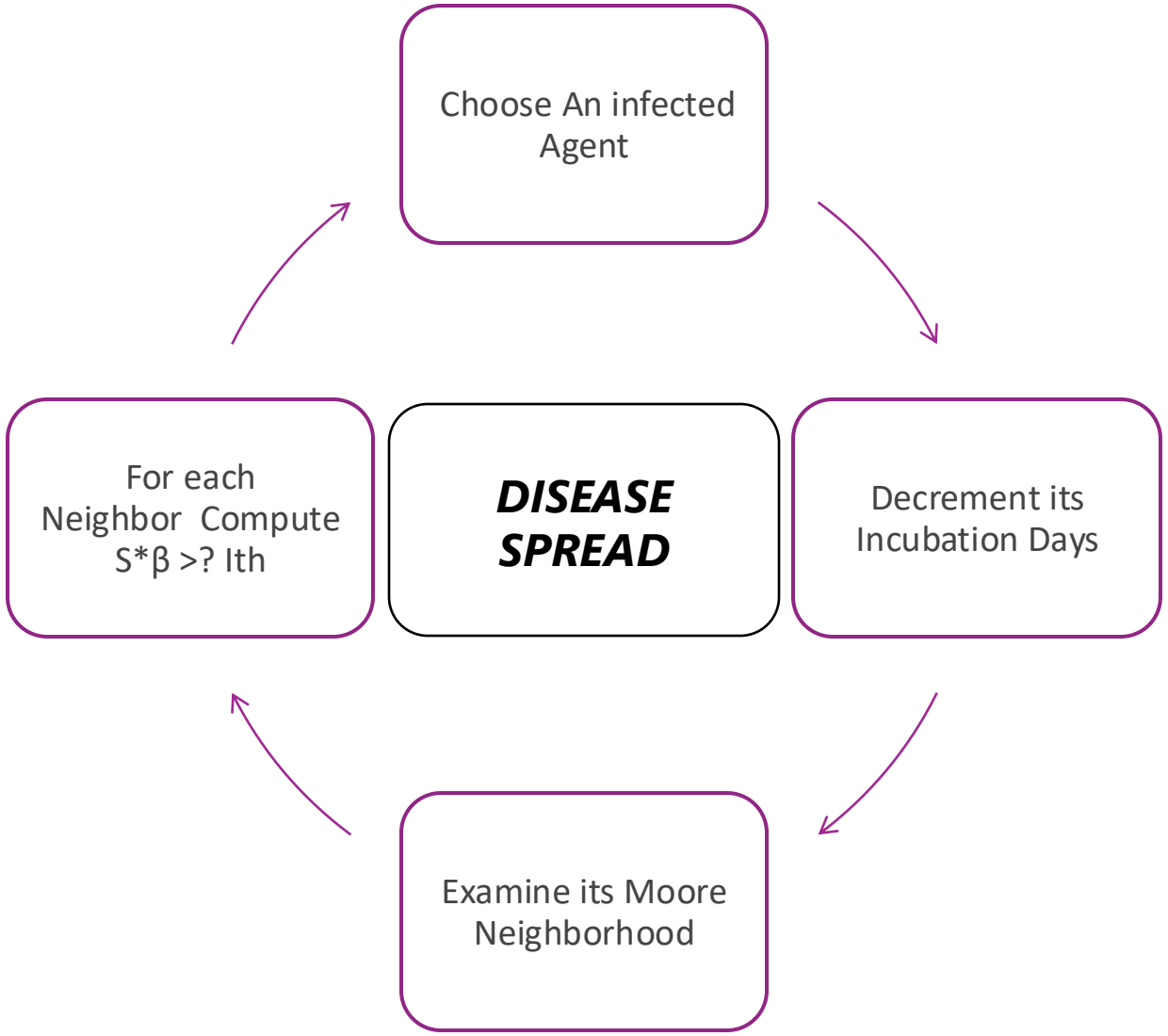


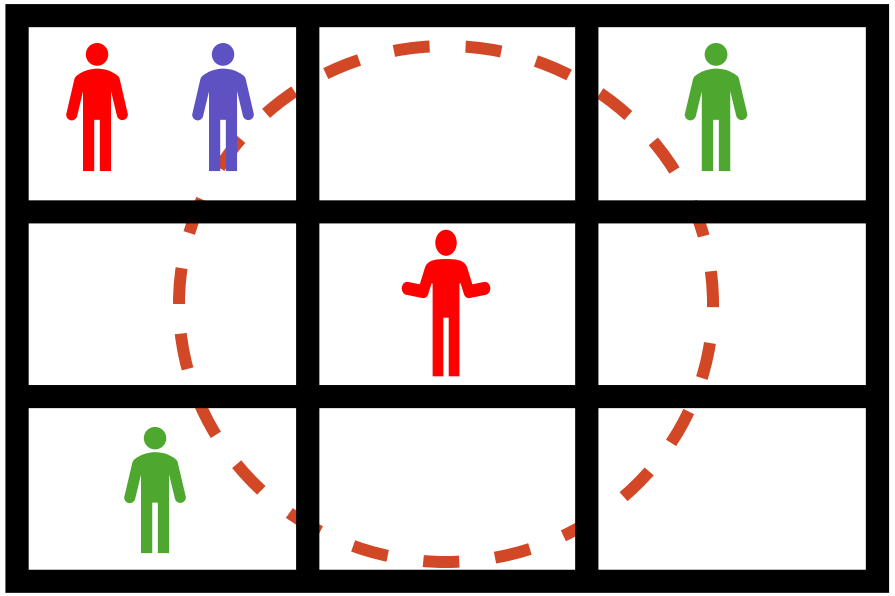
- ***Infected agent scanning***
- ***Probability-based infection***

- ***Random displacement generation***
- ***Boundary/occupancy validation***

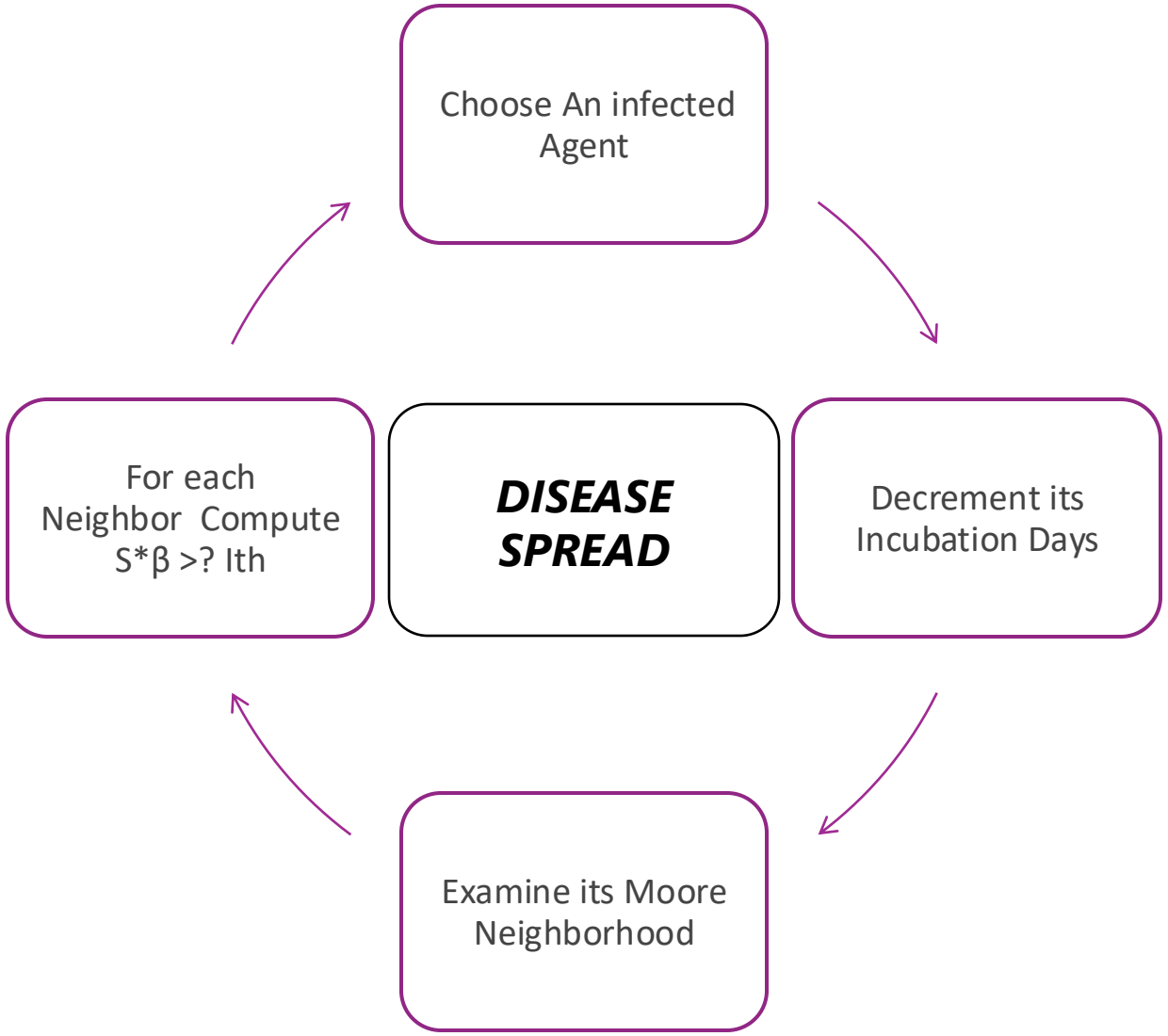
- ***Incubation completion***
- ***Recovery/death probability***

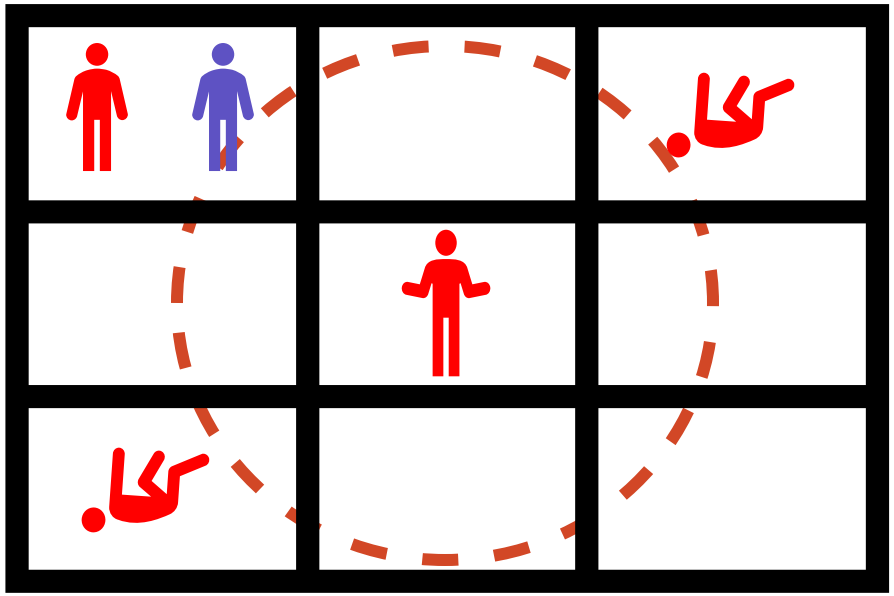
Solution Description: SERIAL Version – Daily Simulation





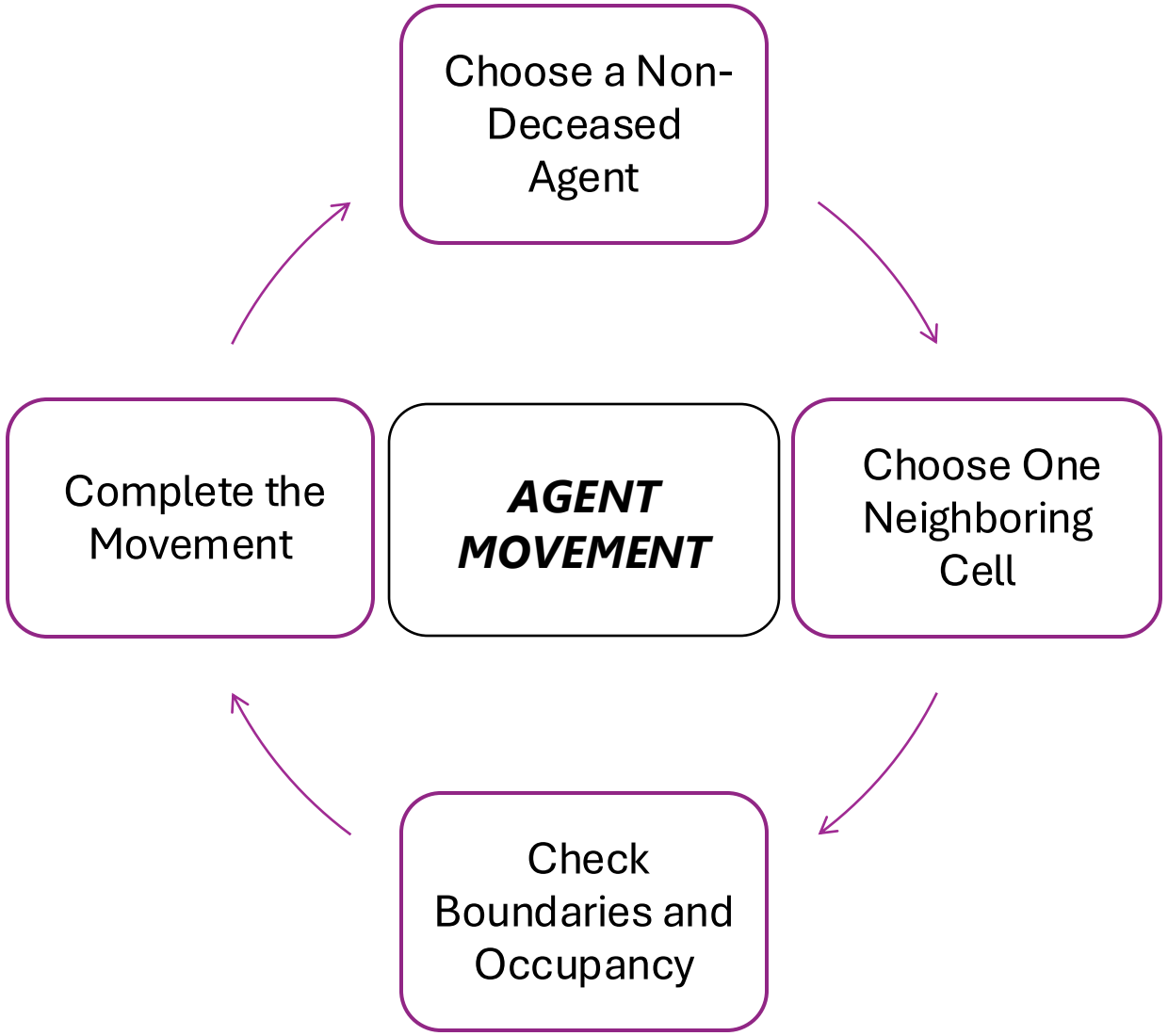
Solution Description: SERIAL Version – Daily Simulation

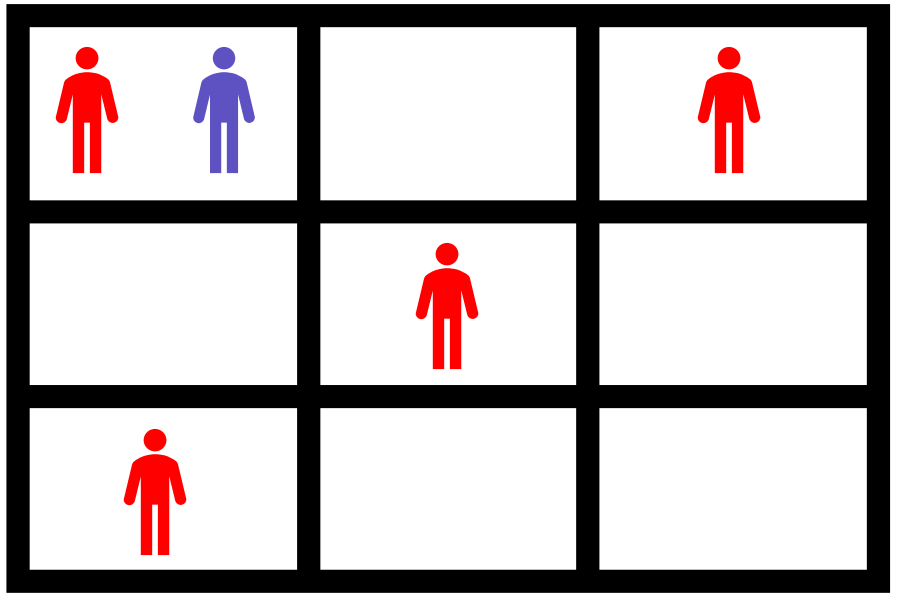




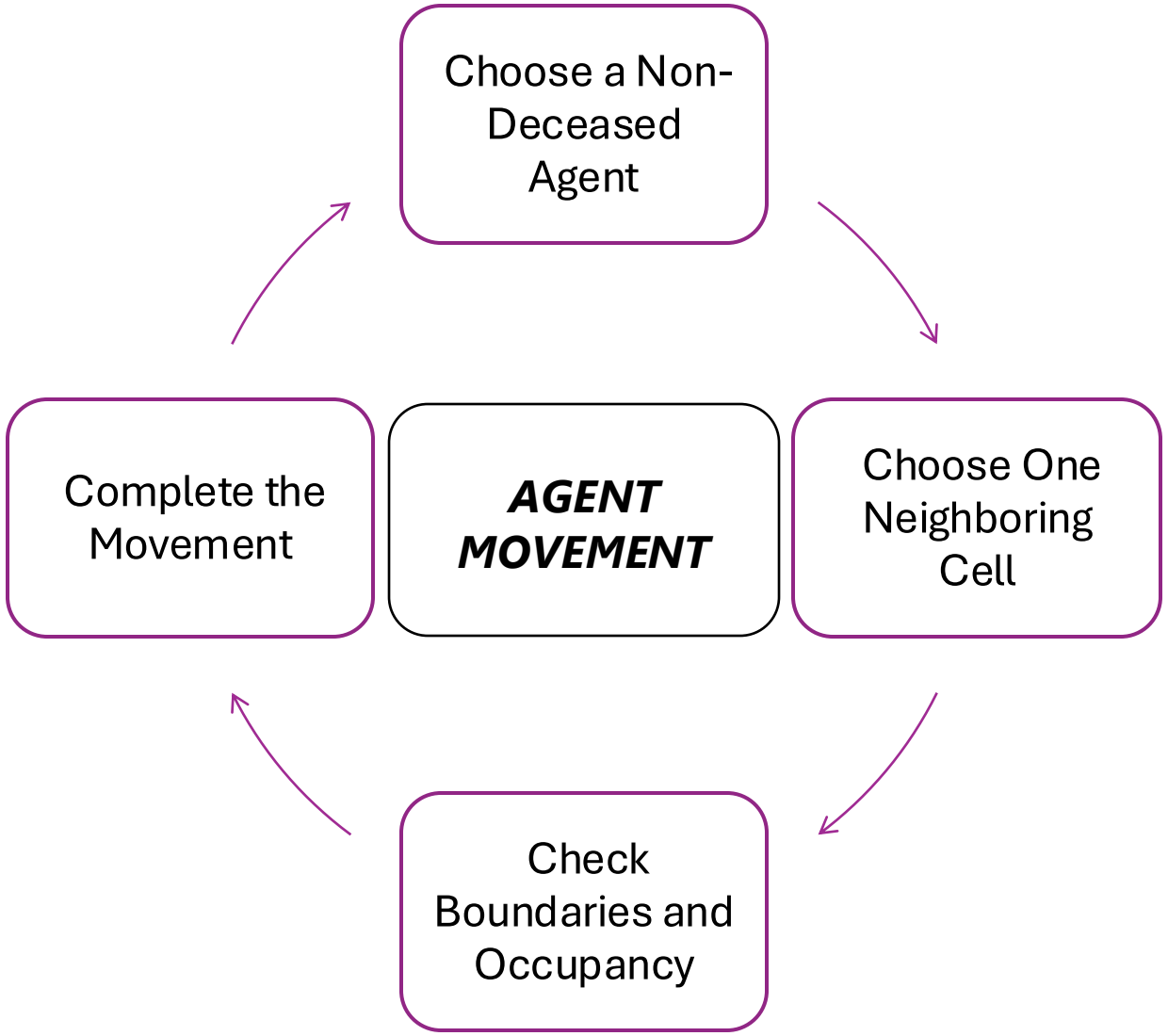
- ***Newly infected***
- ***Incubation_days = -1***

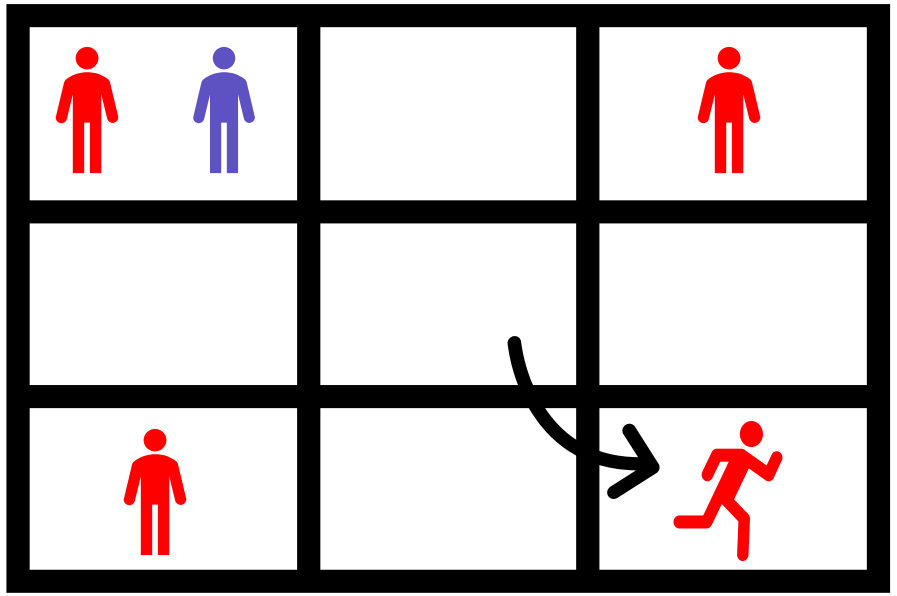
Solution Description: SERIAL Version – Daily Simulation





Solution Description: SERIAL Version – Daily Simulation





Solution Description: SERIAL Version – Daily Simulation

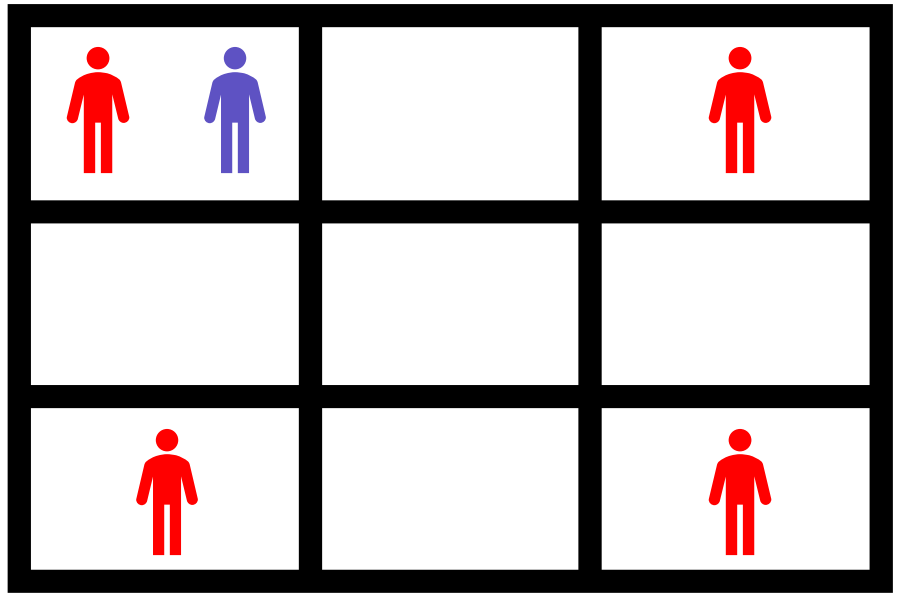
STATE UPDATE

Choose an Infected with INC = 1

Probability μ: Recover

Probabilty 1-μ: Death

50% Chance: Immunity



Solution Description: SERIAL Version – Daily Simulation

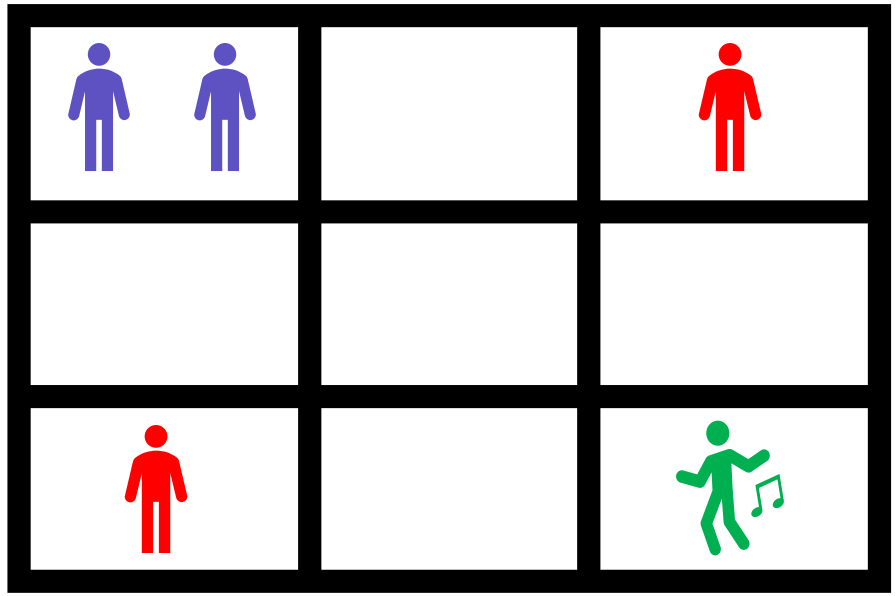
STATE UPDATE

Choose an Infected with INC = 1

Probability μ: Recover

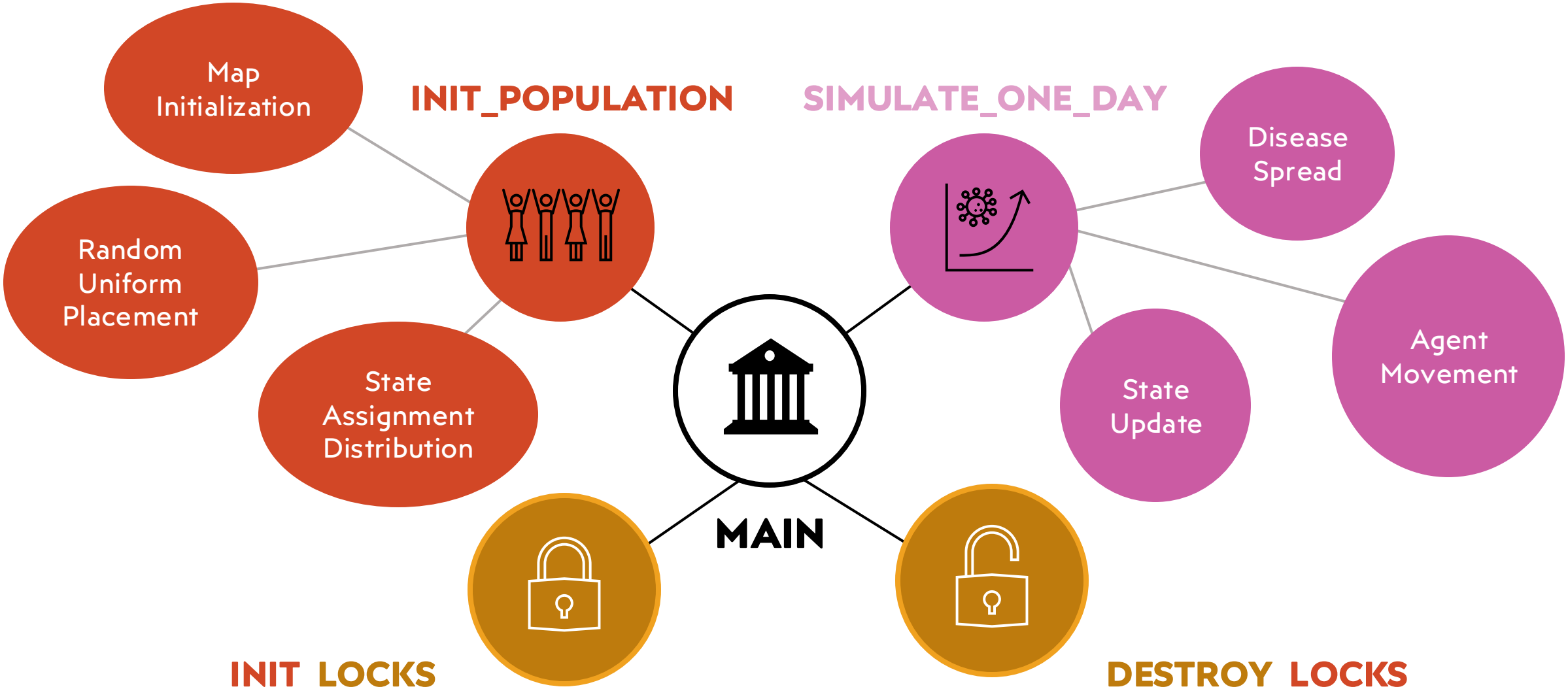
Probabilty 1-μ: Death

50% Chance: Immunity

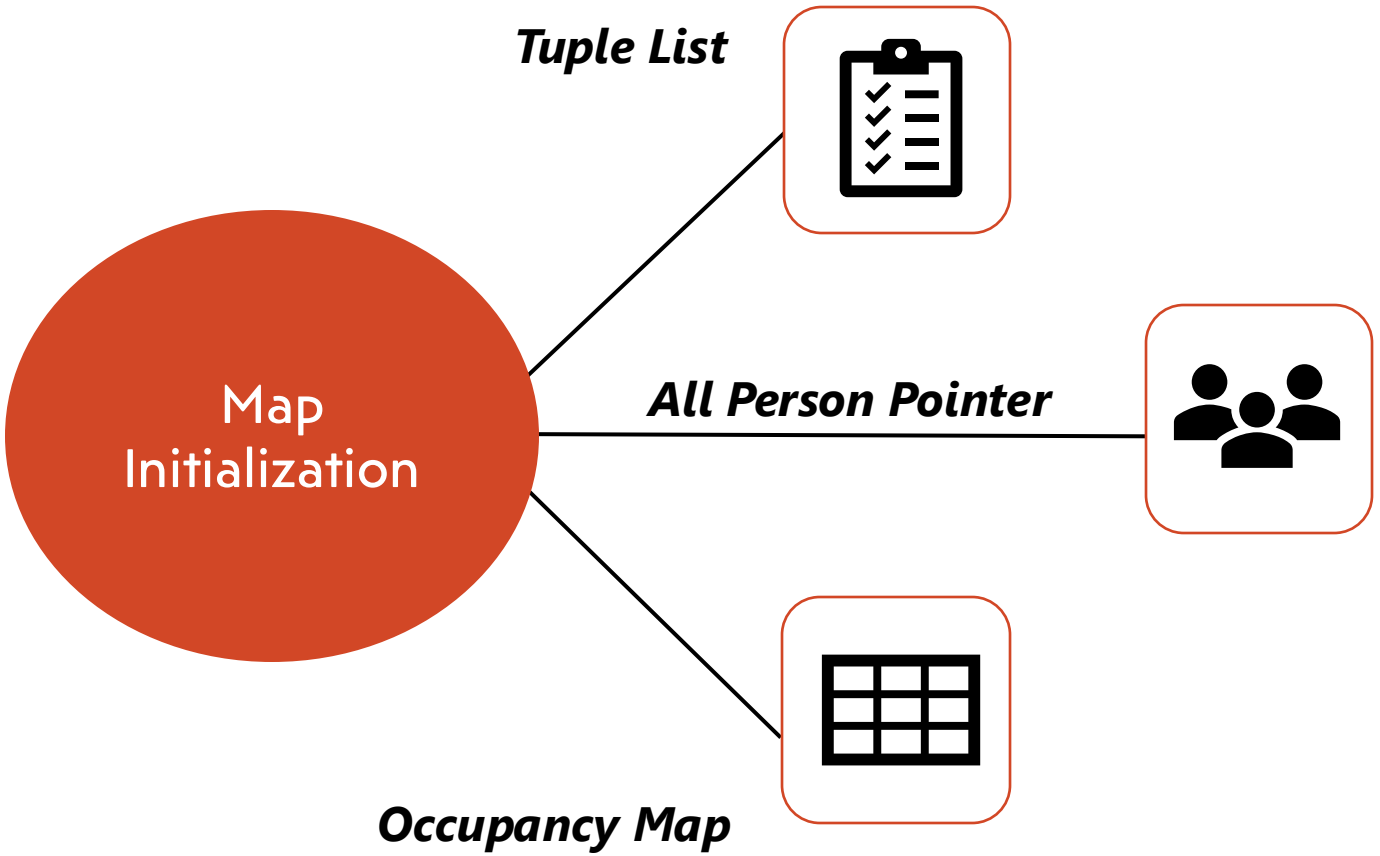


- ***Newly recovered***
- ***Incubation_days = -2***

Solution Description: OpenMP Version – Program Organization



Solution Description: OpenMP Version – Population Initialization



#pragma omp parallel for collapse(2)

Generation af all WxH coordinates

Cells occupancy set to 0

Each Cell points to MAXP_CELL slots of All Person Pointer

Solution Description: OpenMP Version – Population Initialization

#pragma omp parallel for

Random Placement

RACE CONDITION

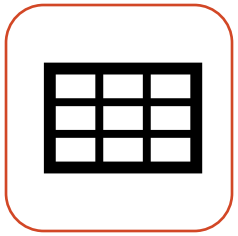
~~Random Coordinate Extraction~~

Person Addition to Corresponding Cell

PERSON

TUPLE LIST

OCCUPANCY MAP



Solution Description: OpenMP Version – Population Initialization

#pragma omp parallel for

Random Placement

Each thread selects its tuples

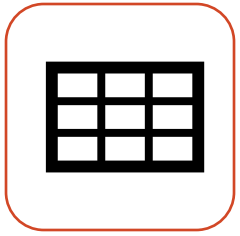
Person Addition to Corresponding Cell

OMP people distribution not aligned with coordinates distribution

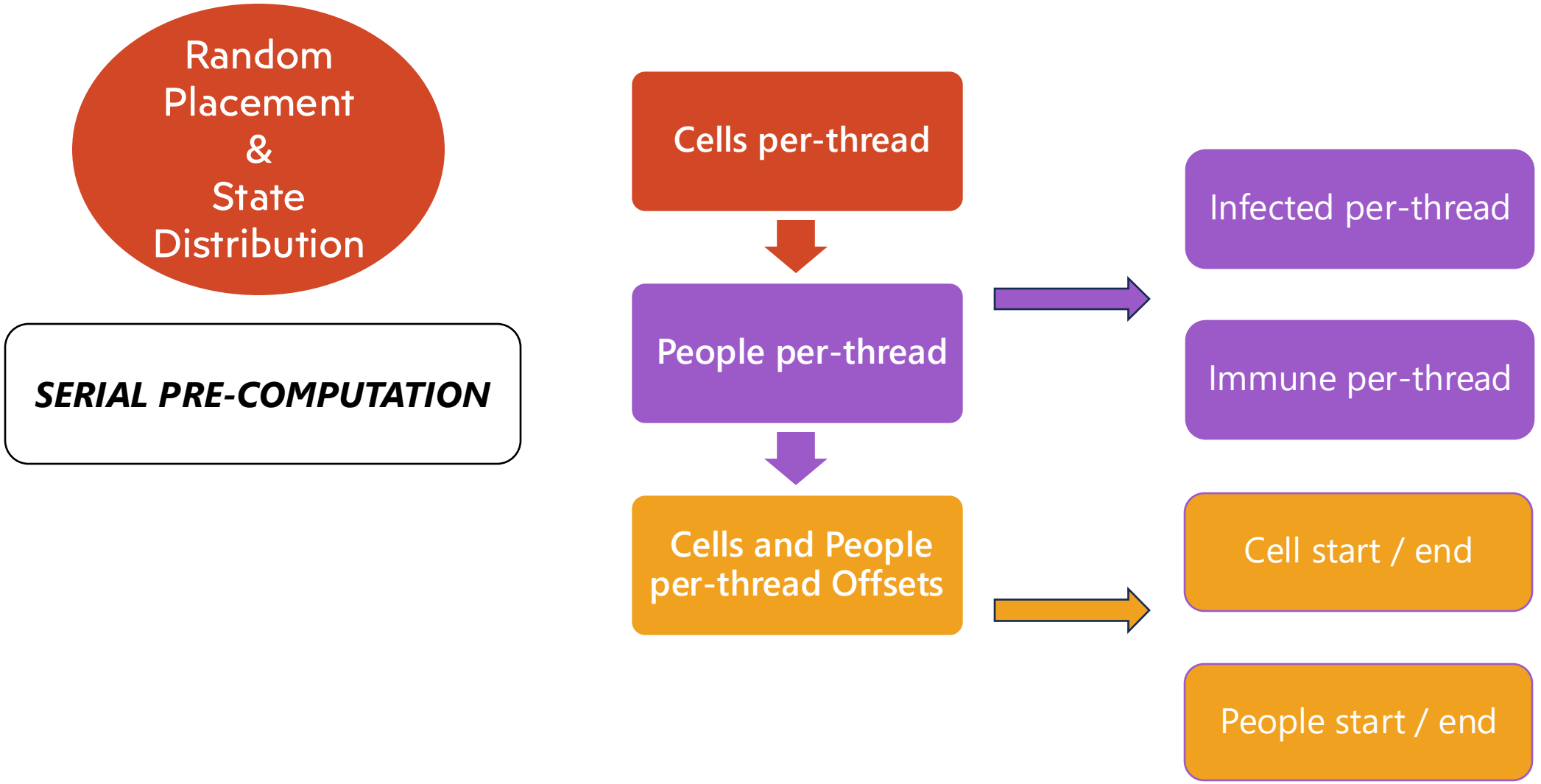
PERSON

PER-THREAD TUPLE LIST

OCCUPANCY MAP



Solution Description: OpenMP Version – Population Initialization



Solution Description: OpenMP Version – Population Initialization

PARALLEL SECTION

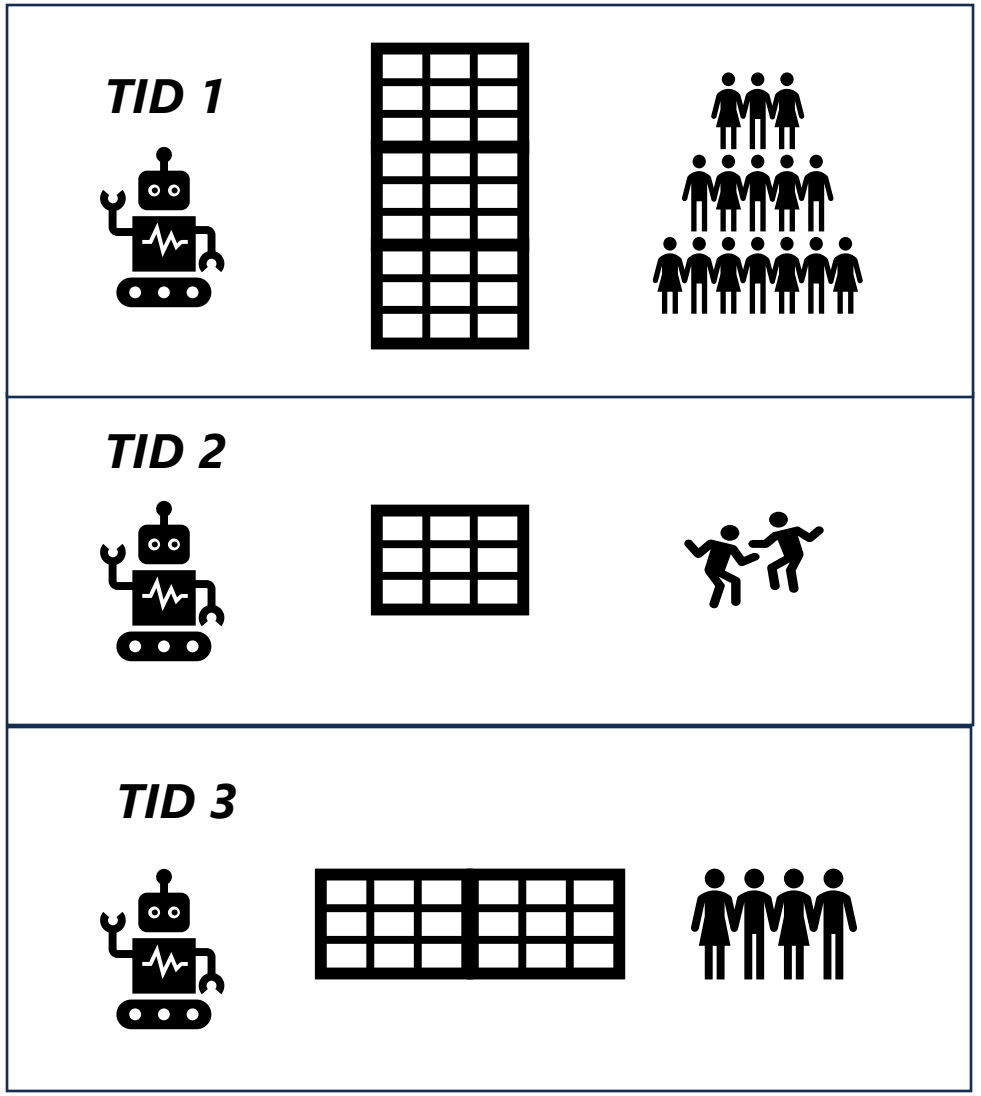
Cell start / end

```
for (int i = cell_start; i < cell_end; i++)
{
    int x = i / H;
    int y = i % H;
    addTuple(local_coords, x, y);
}
```

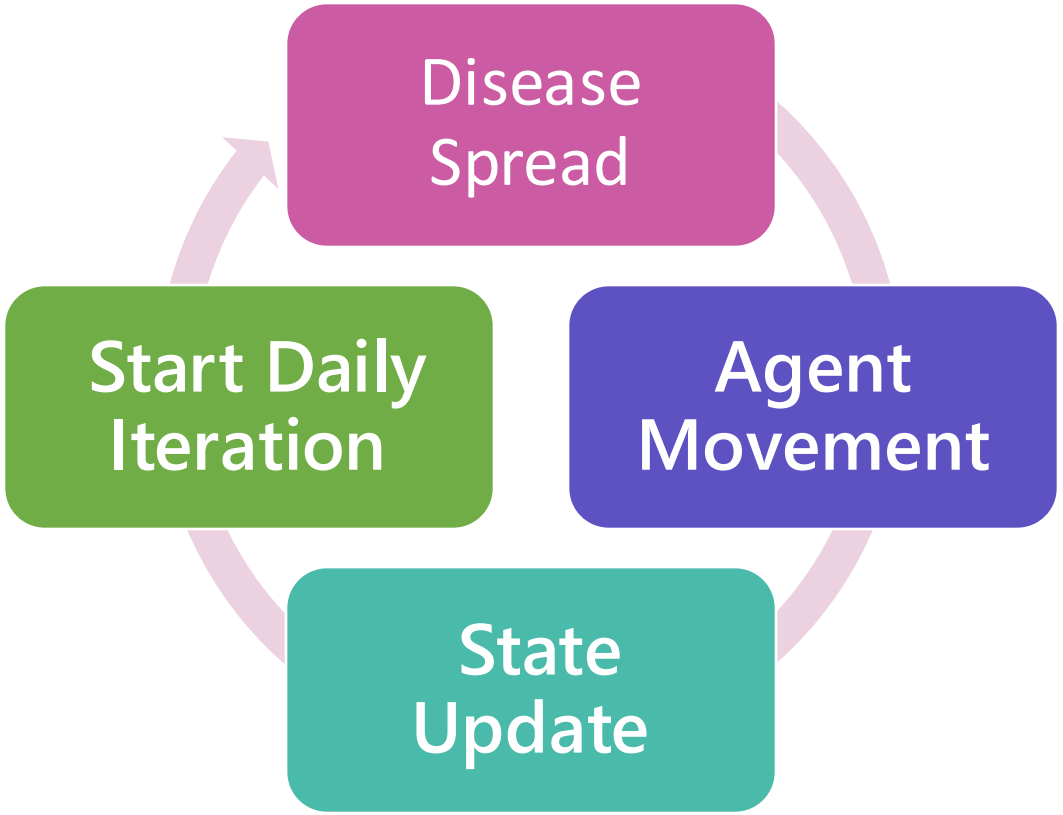
Random Placement & State Distribution

```
for (int i = people_start; i < people_end; i
    ++)
{
    Person *p = &population[i];
    // ...
}
```

People start / end

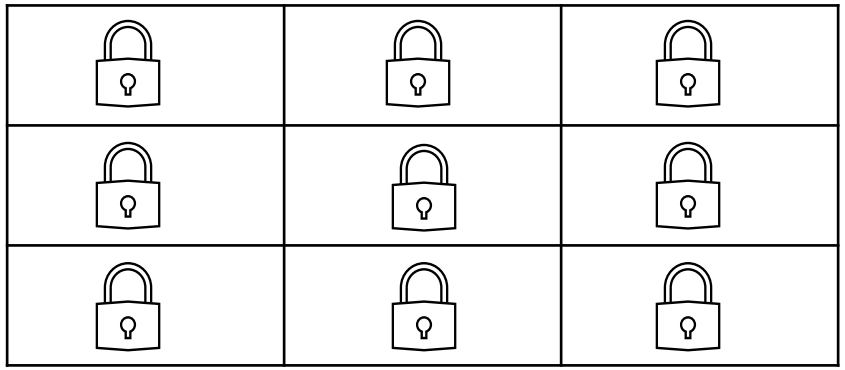


Solution Description: OpenMP Version – Daily Simulation



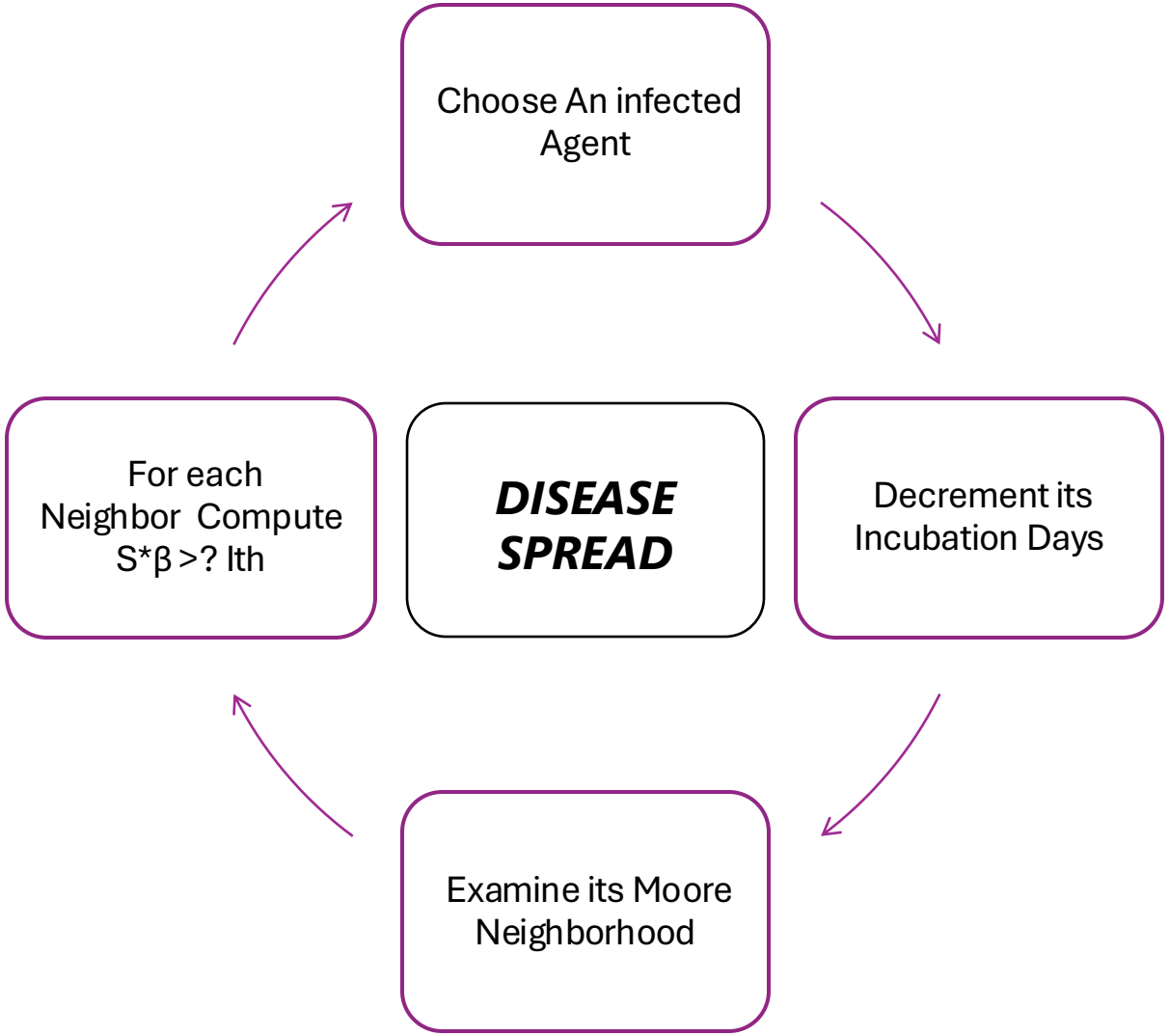
+

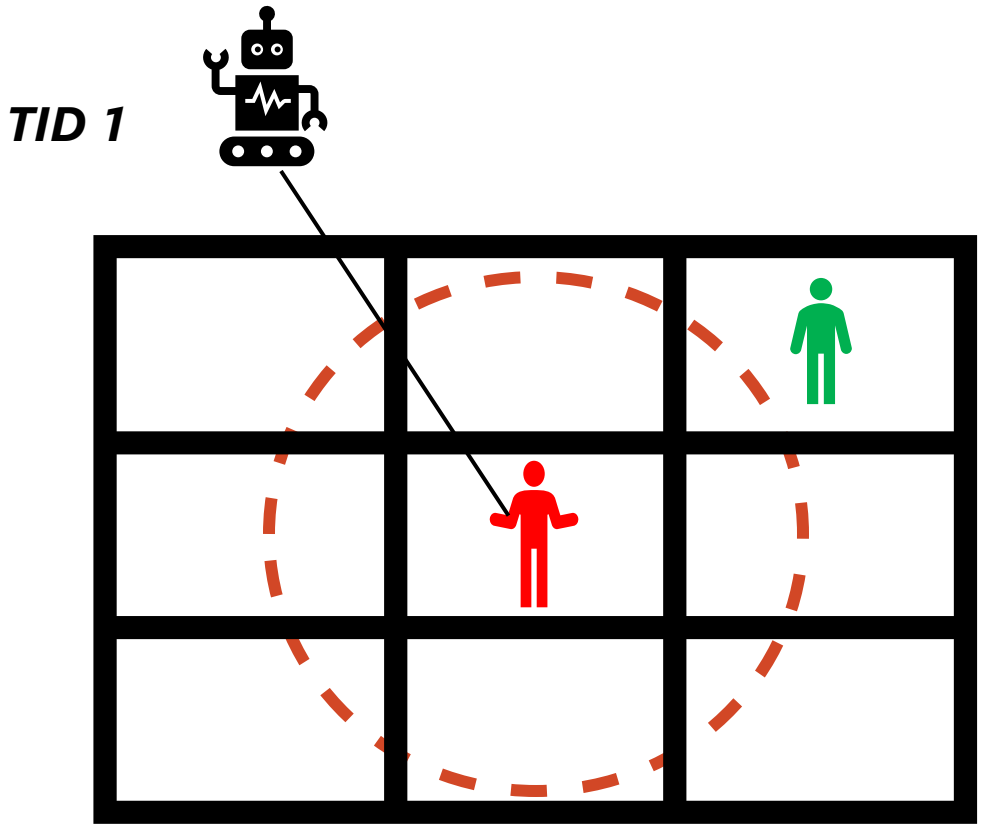
CELL LOCKS



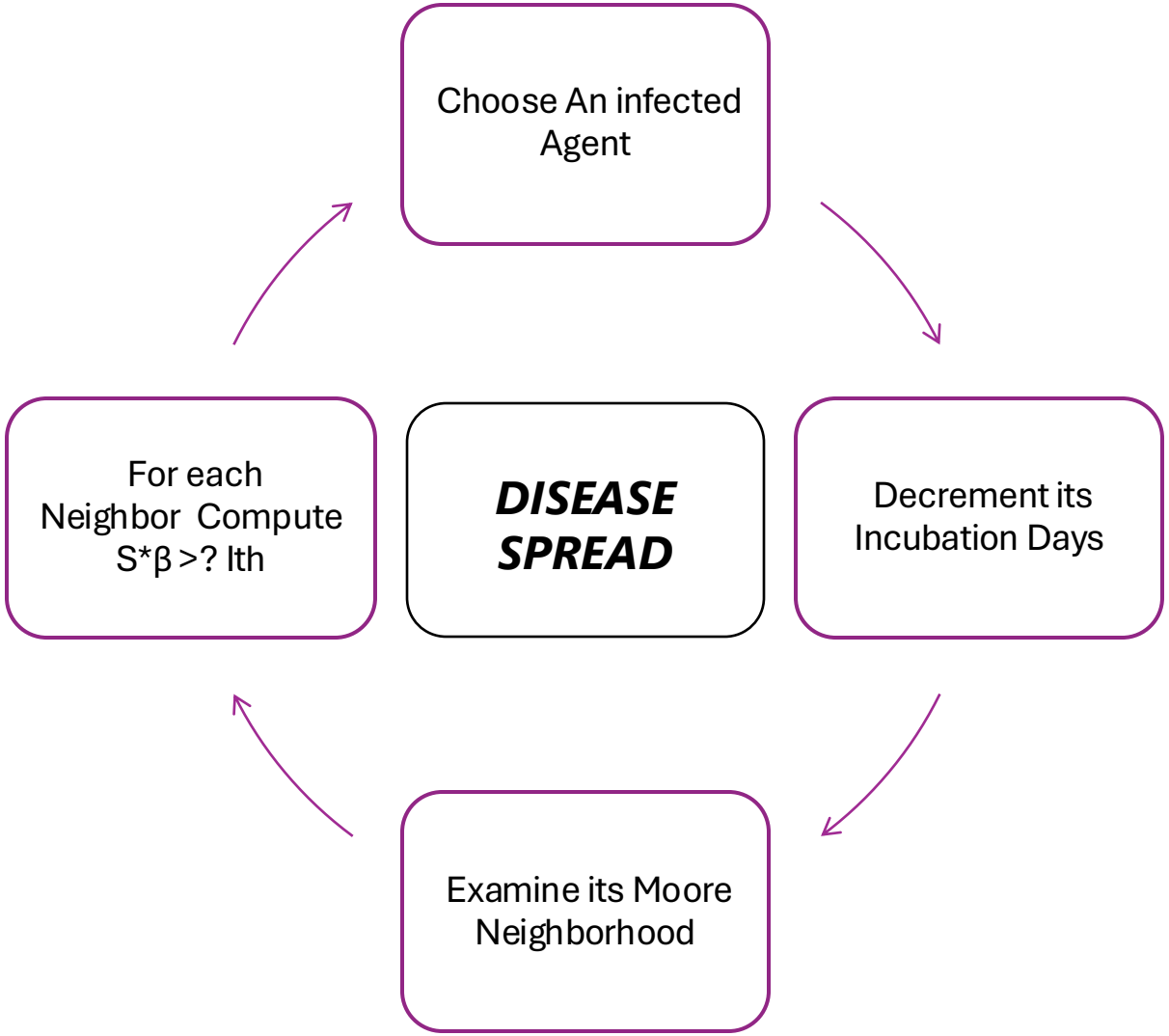
- ***Array of "omp_lock_t"***
- ***Each lock is associated to a tuple***

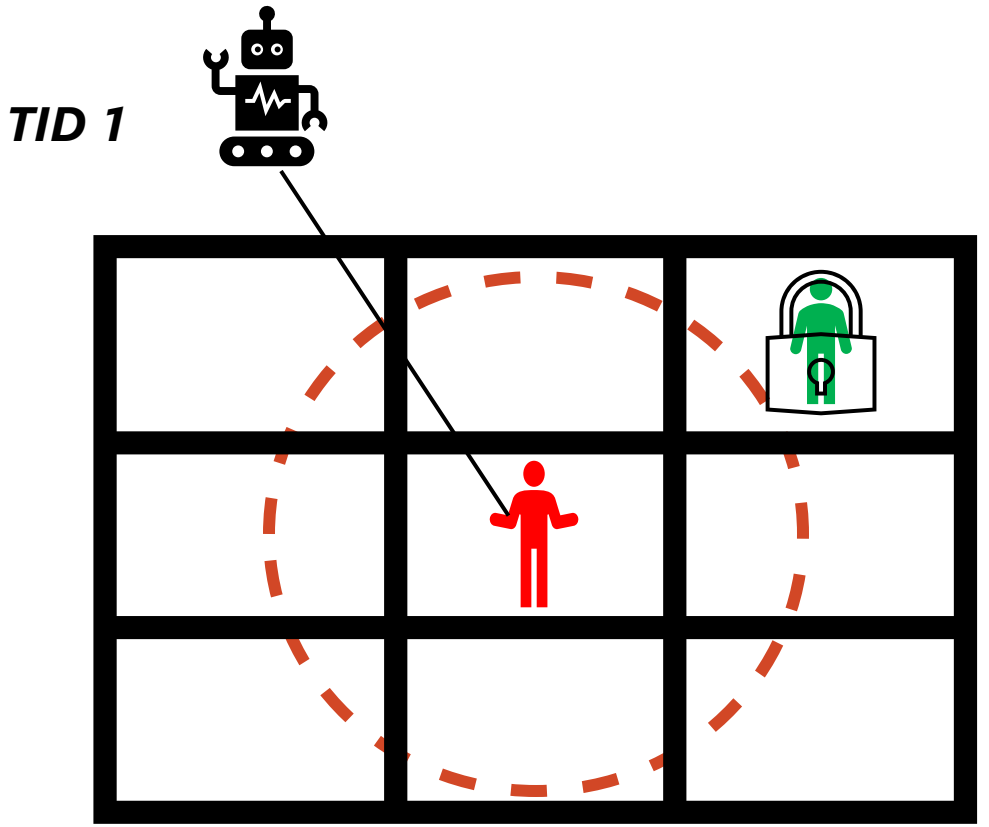
Solution Description: OpenMP Version – Daily Simulation



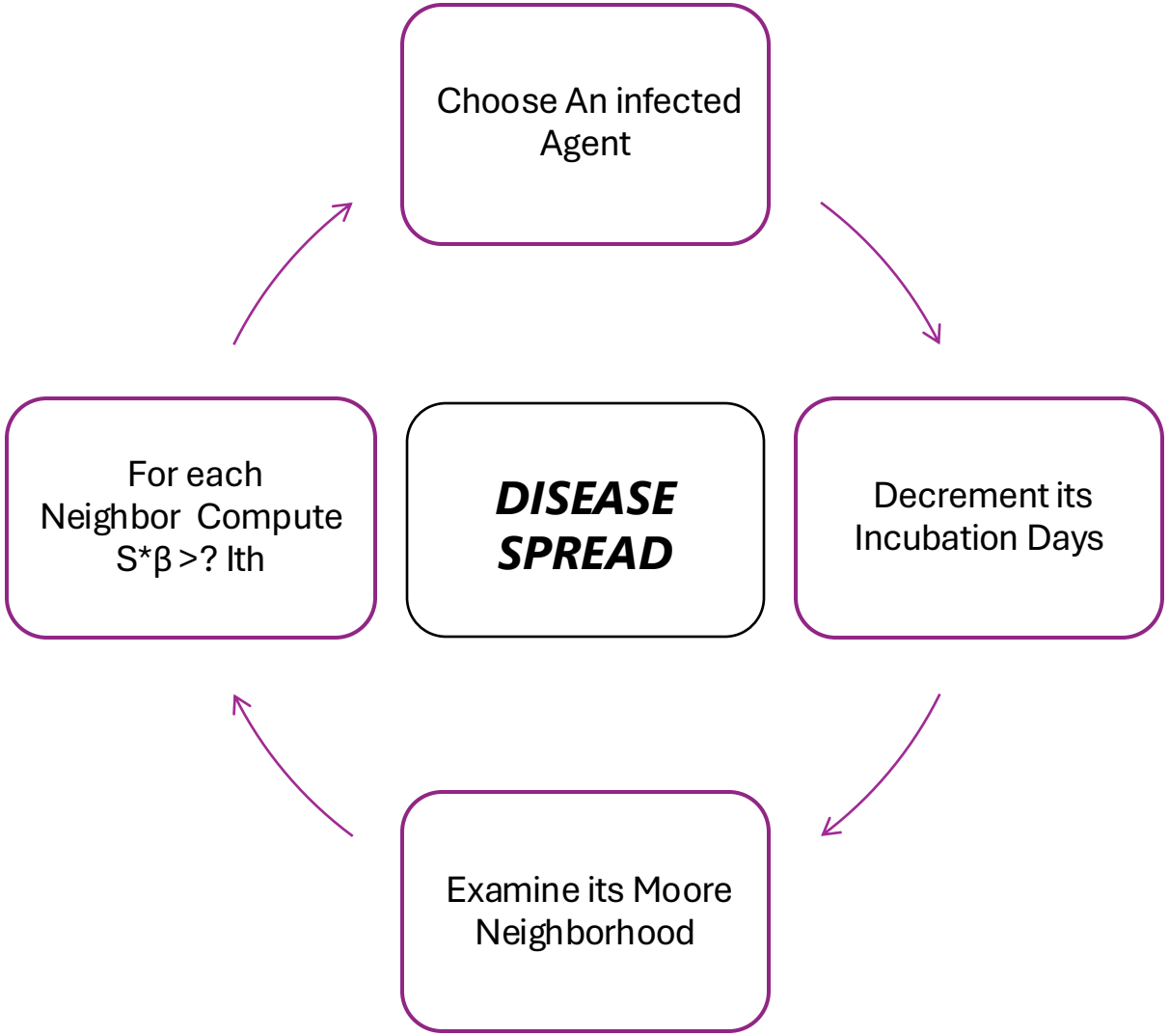


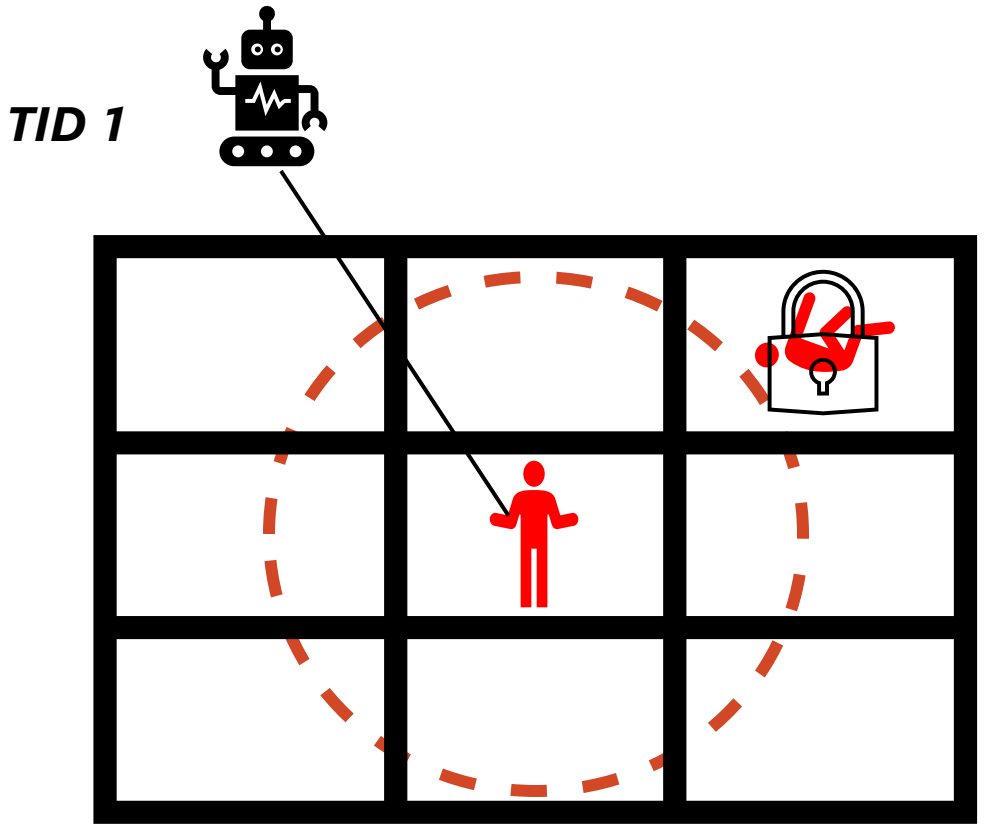
Solution Description: OpenMP Version – Daily Simulation



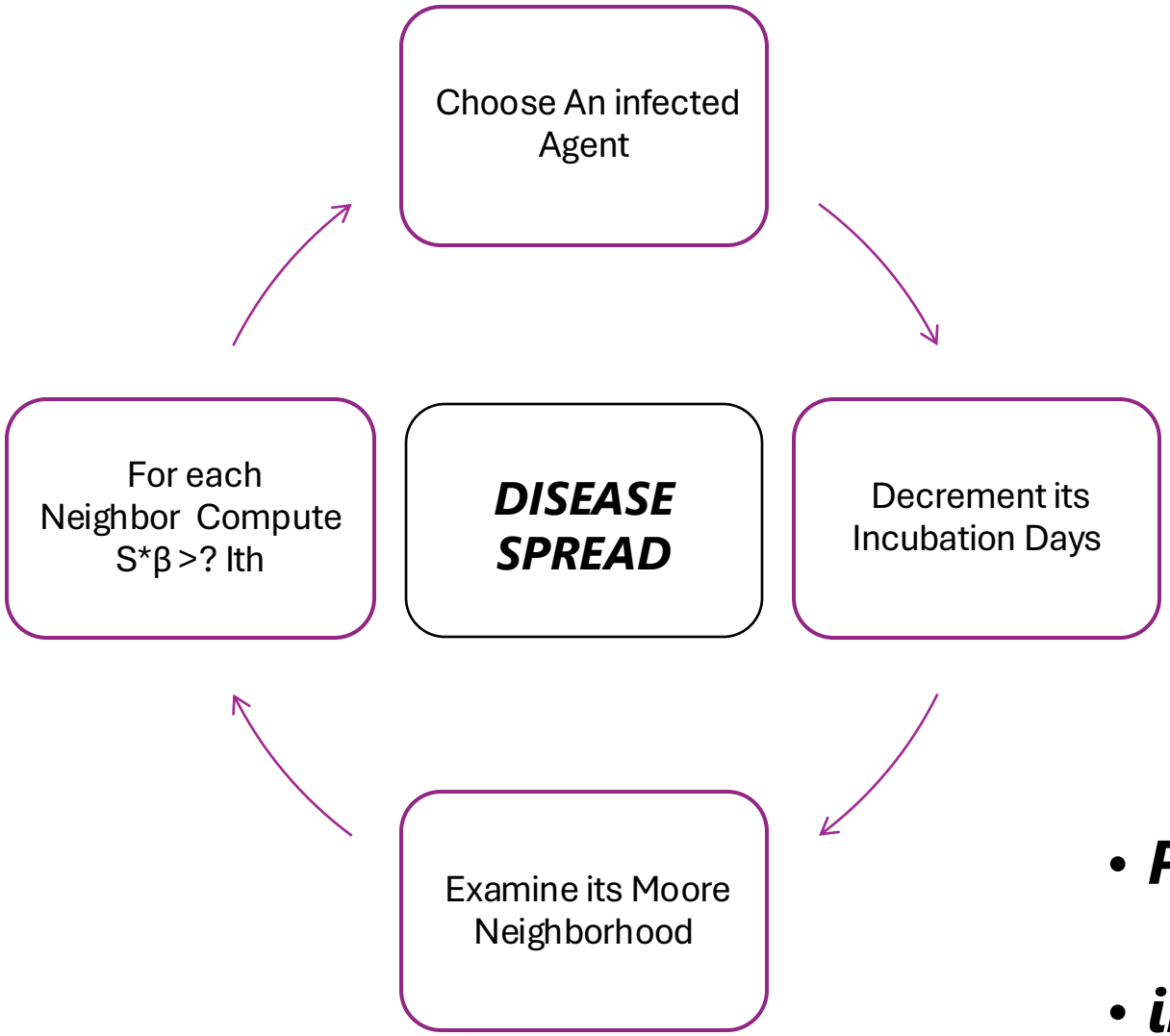


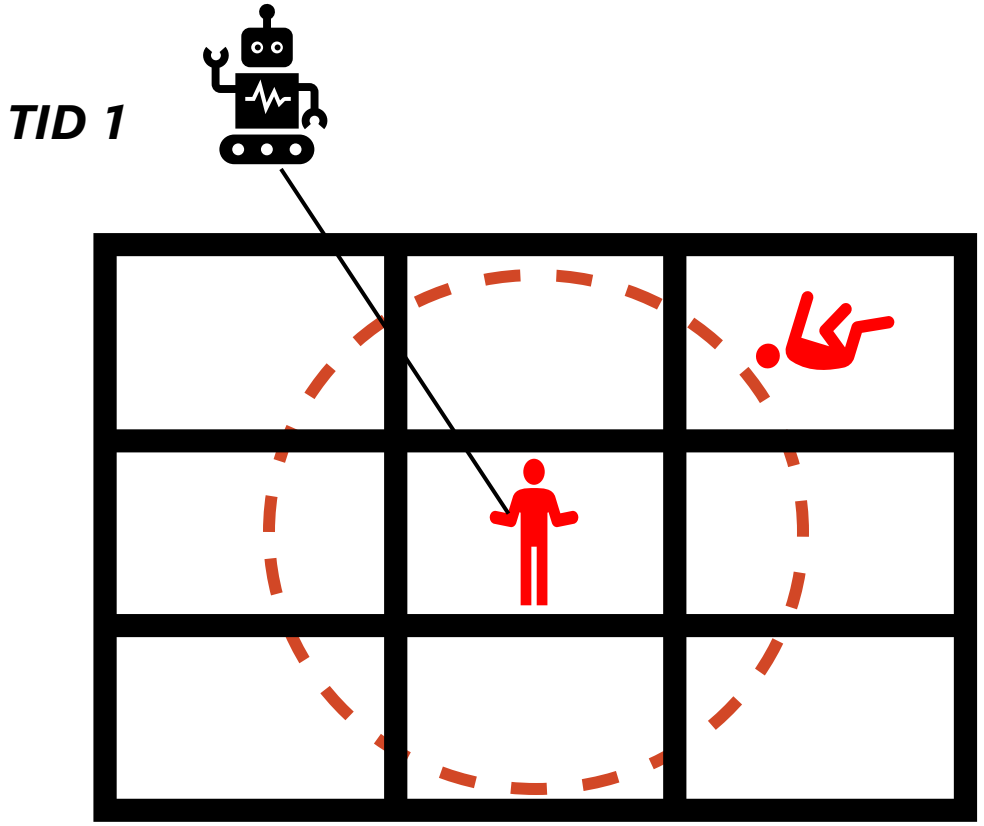
Solution Description: OpenMP Version – Daily Simulation





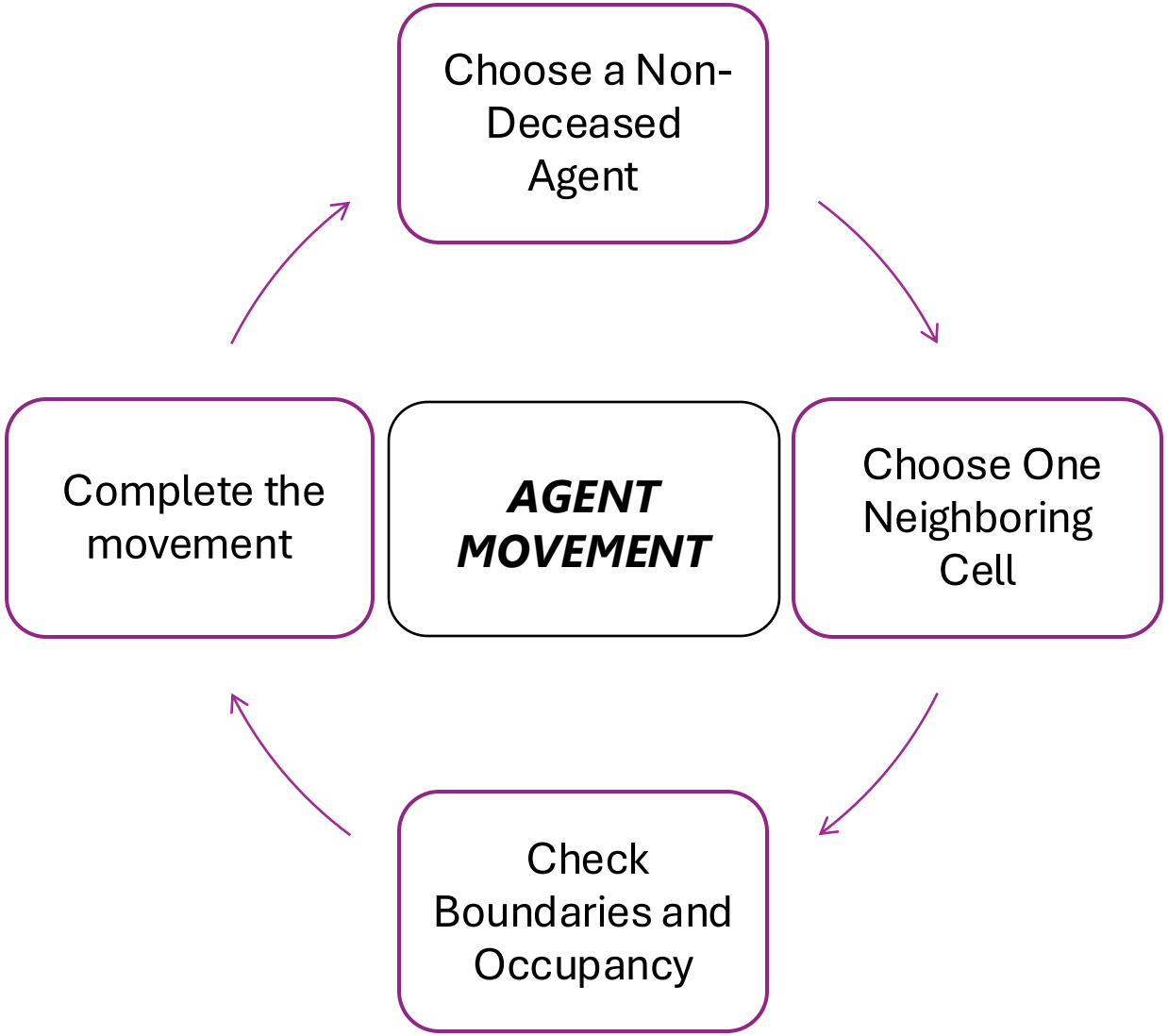
Solution Description: OpenMP Version – Daily Simulation

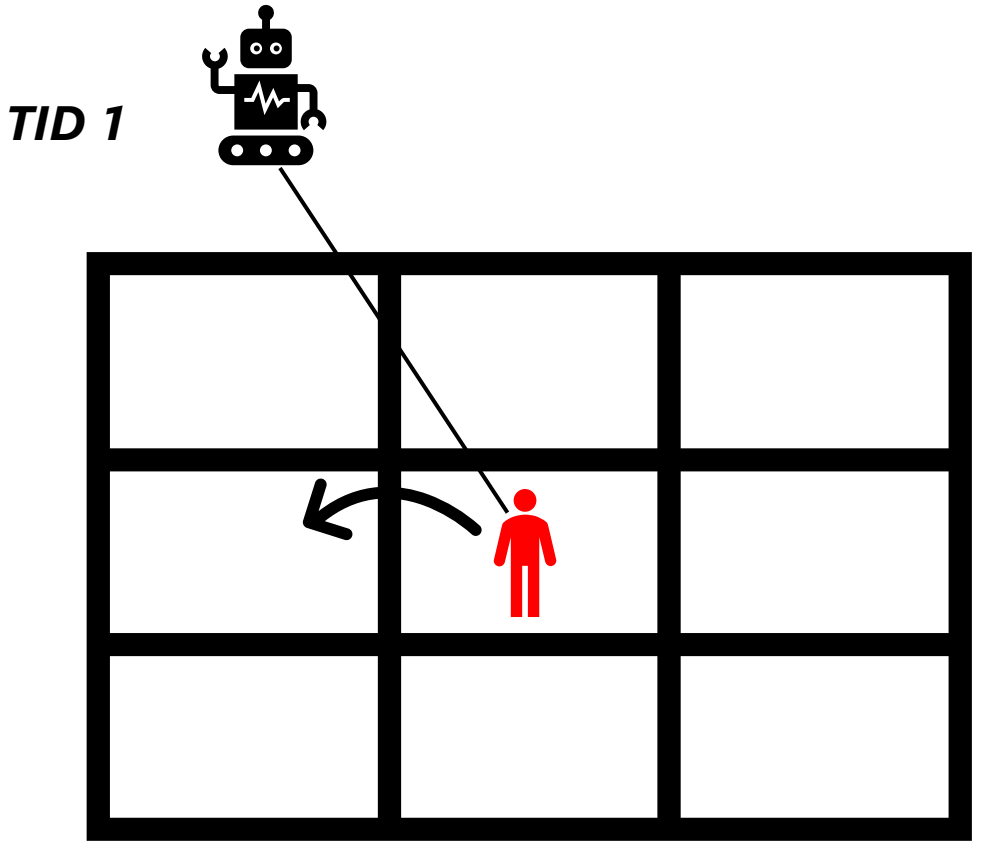




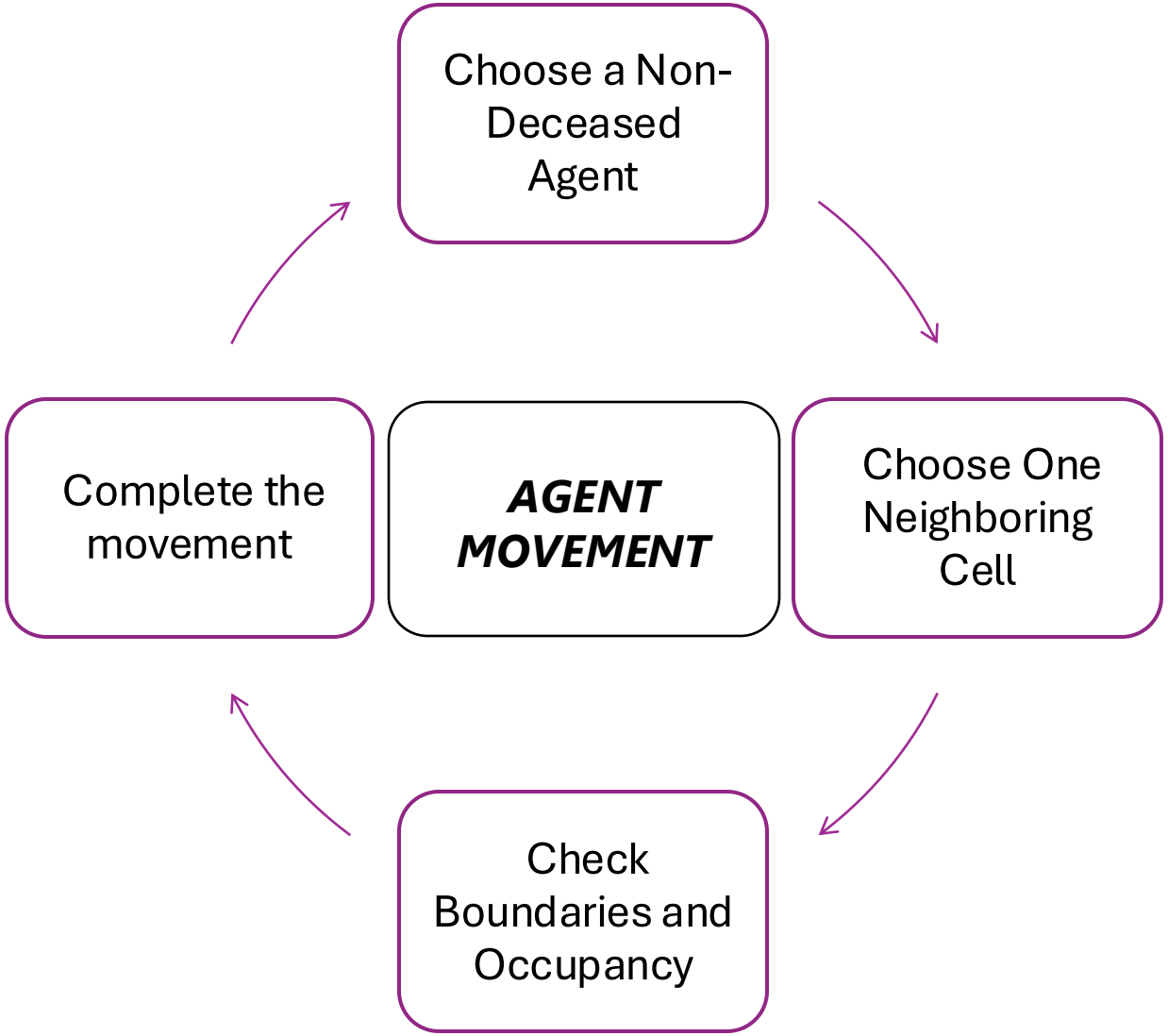
- ***Per-Thread Newly infected***
- ***incubation_days = -1***

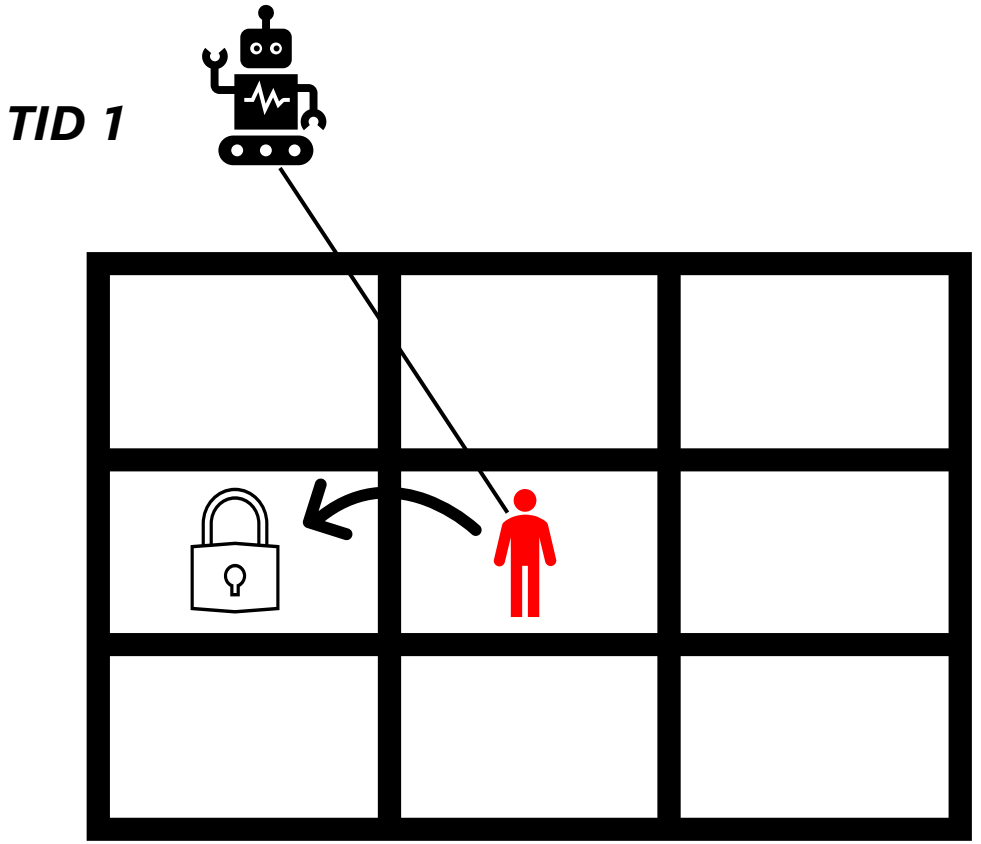
Solution Description: OpenMP Version – Daily Simulation



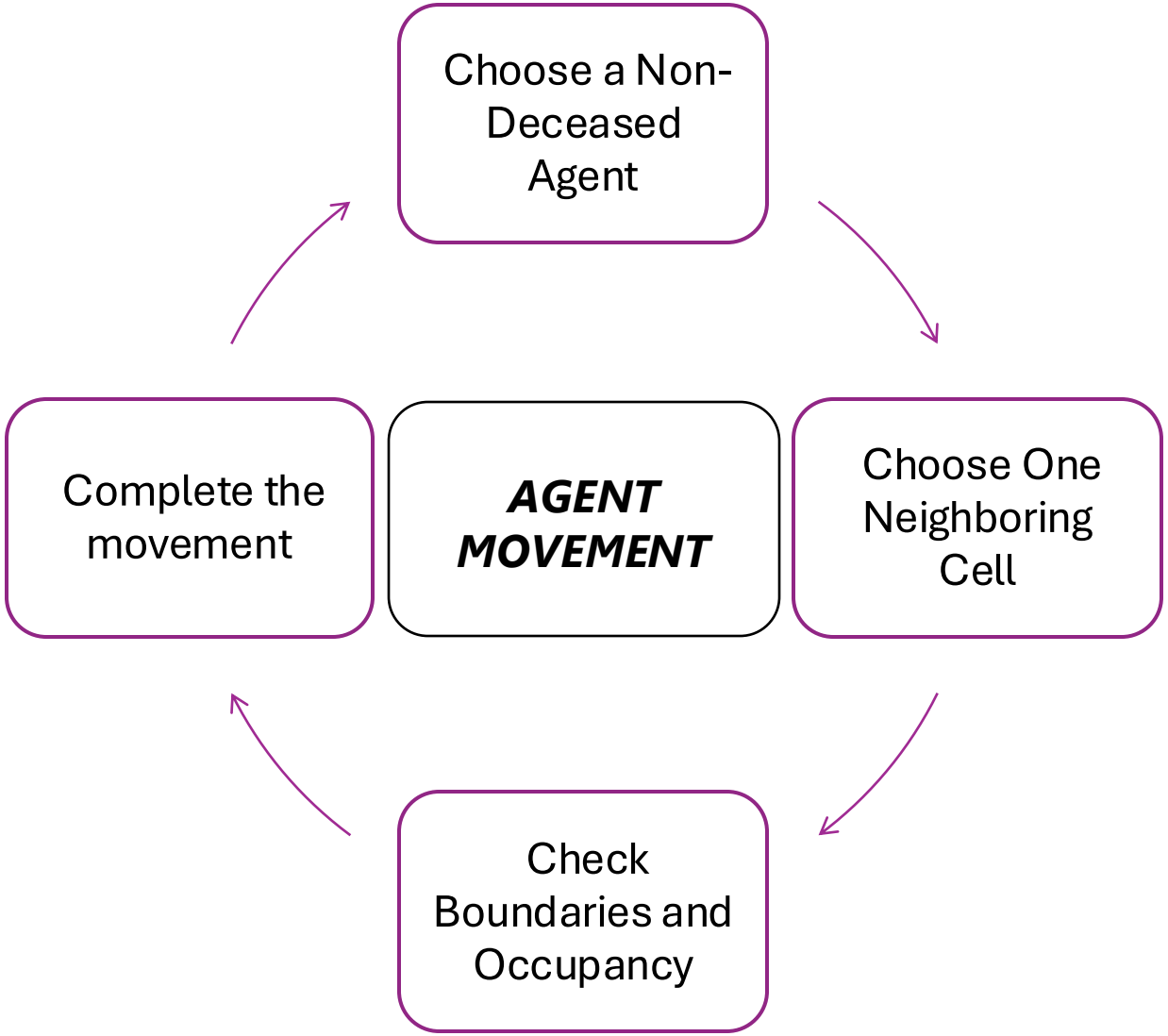


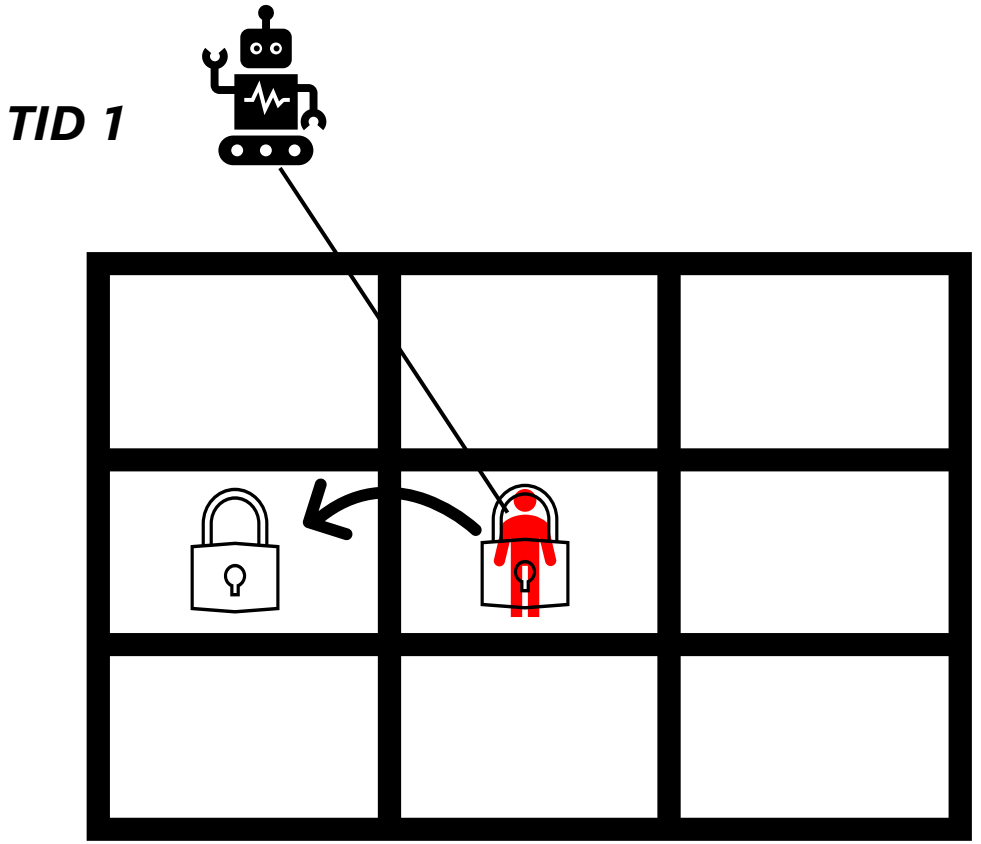
Solution Description: OpenMP Version – Daily Simulation



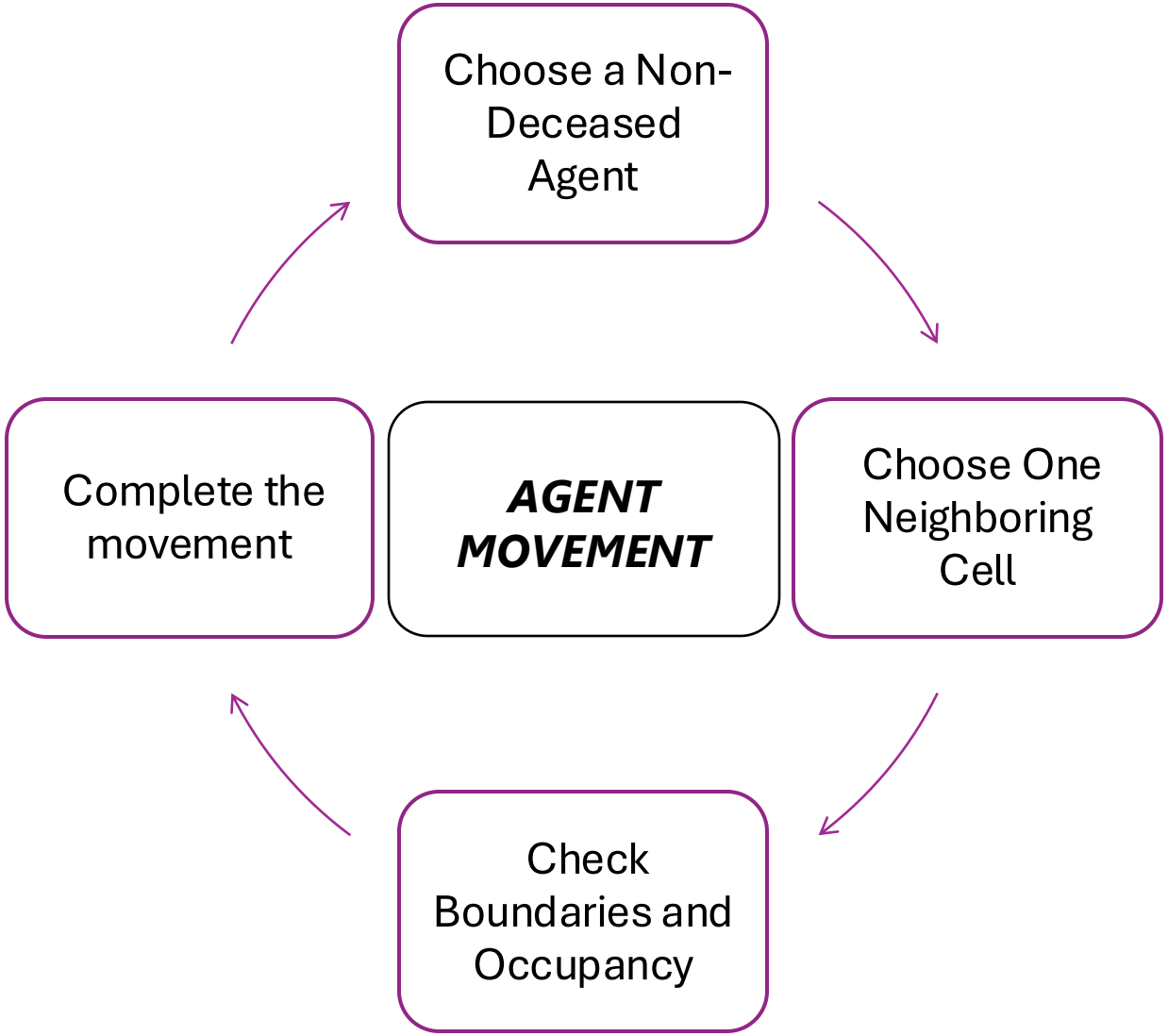


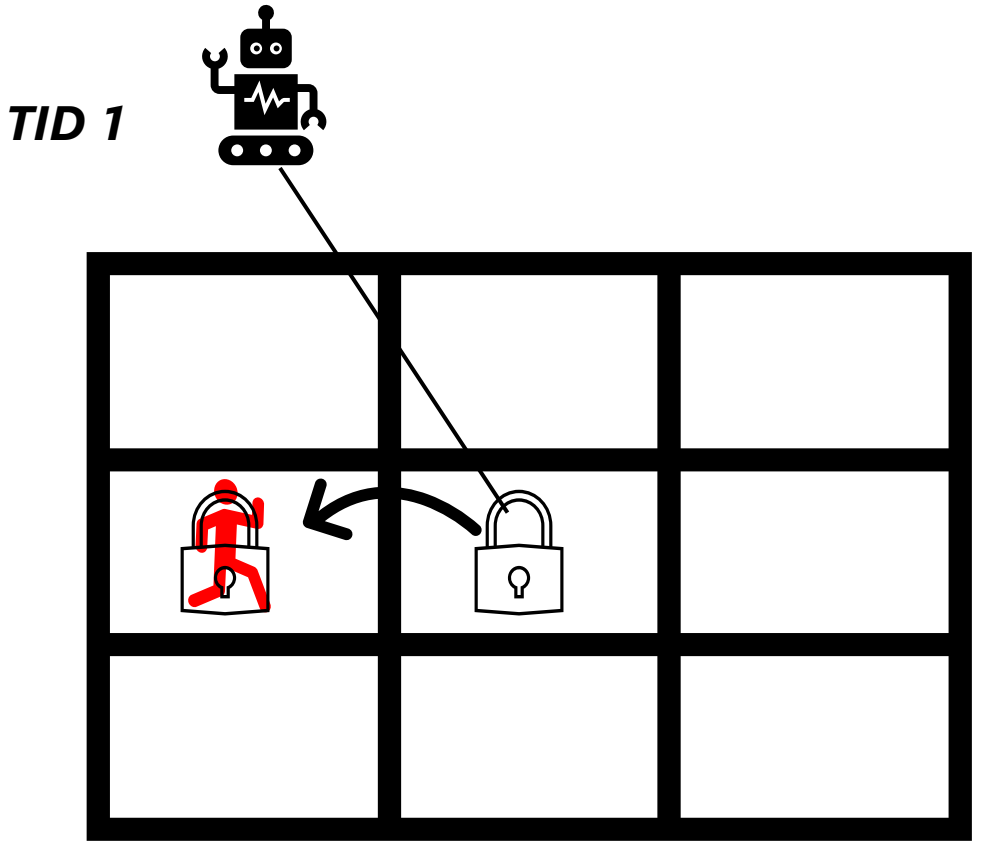
Solution Description: OpenMP Version – Daily Simulation



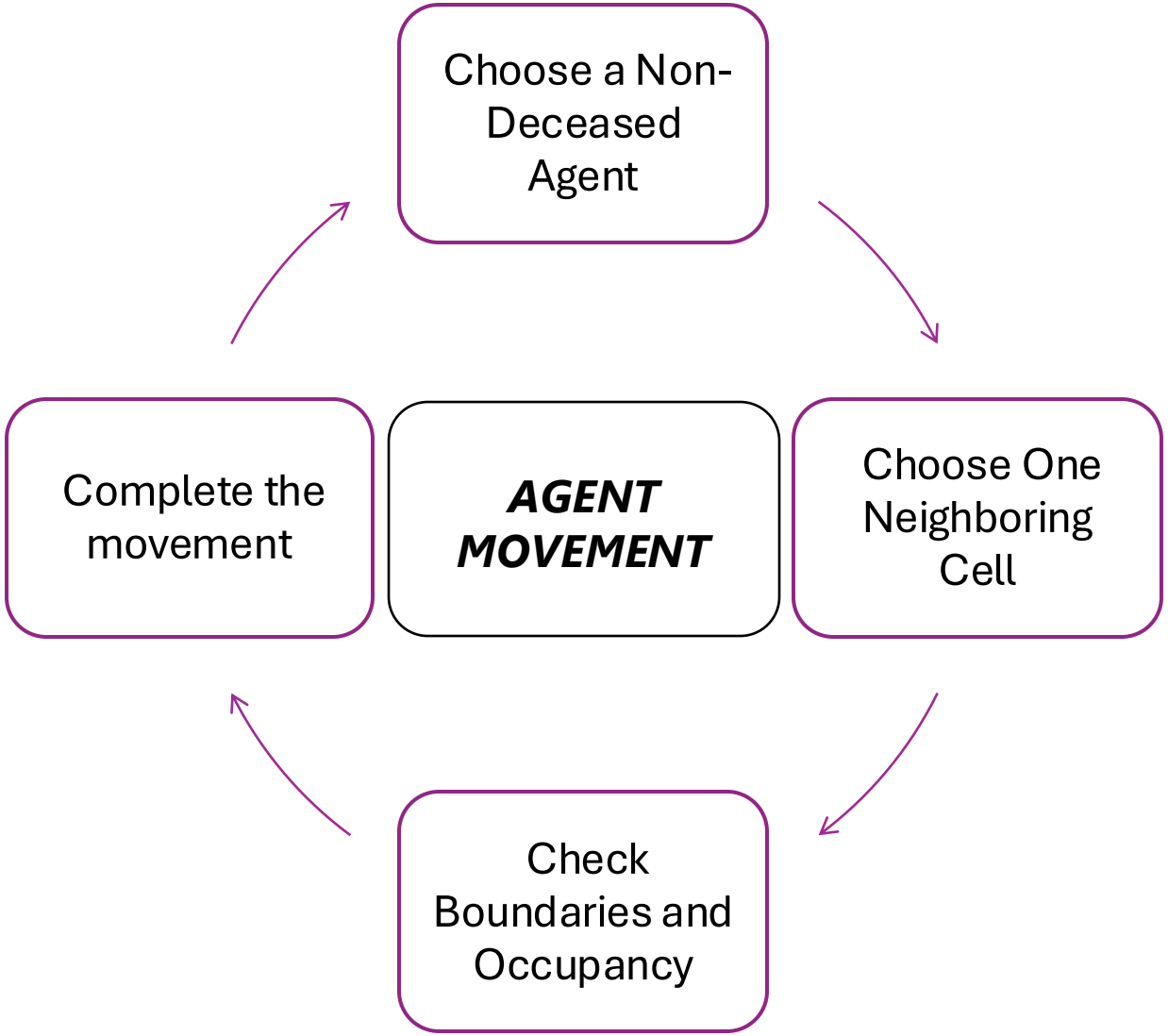


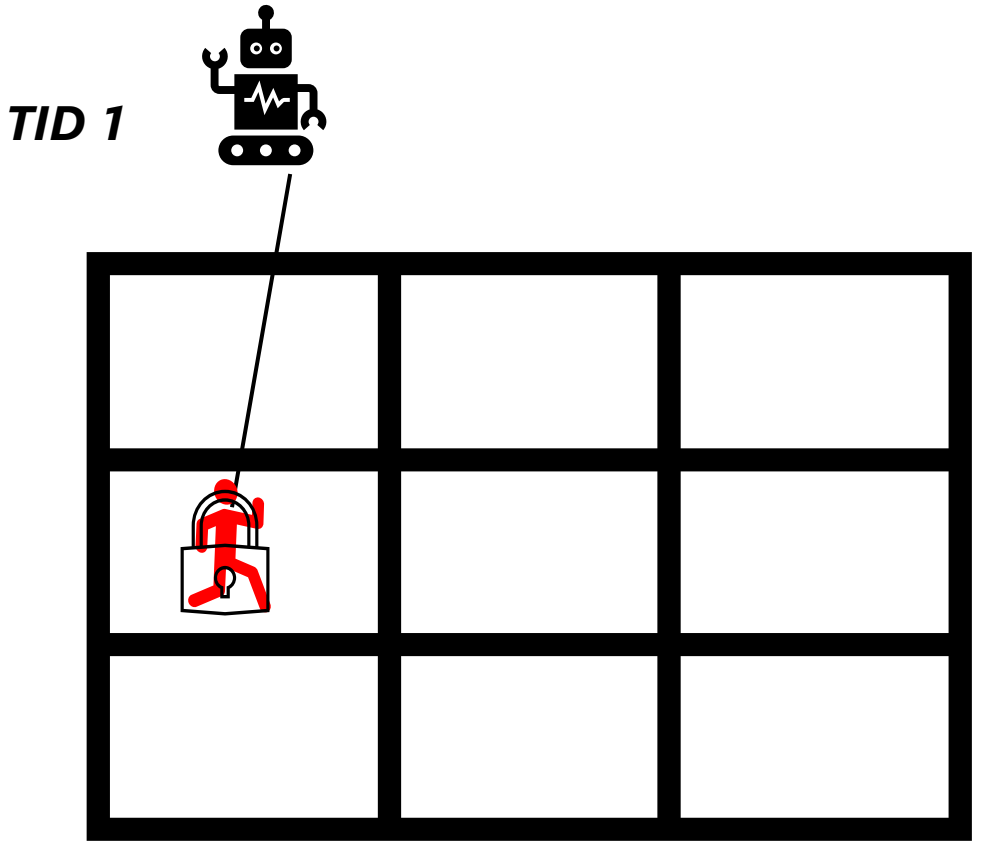
Solution Description: OpenMP Version – Daily Simulation



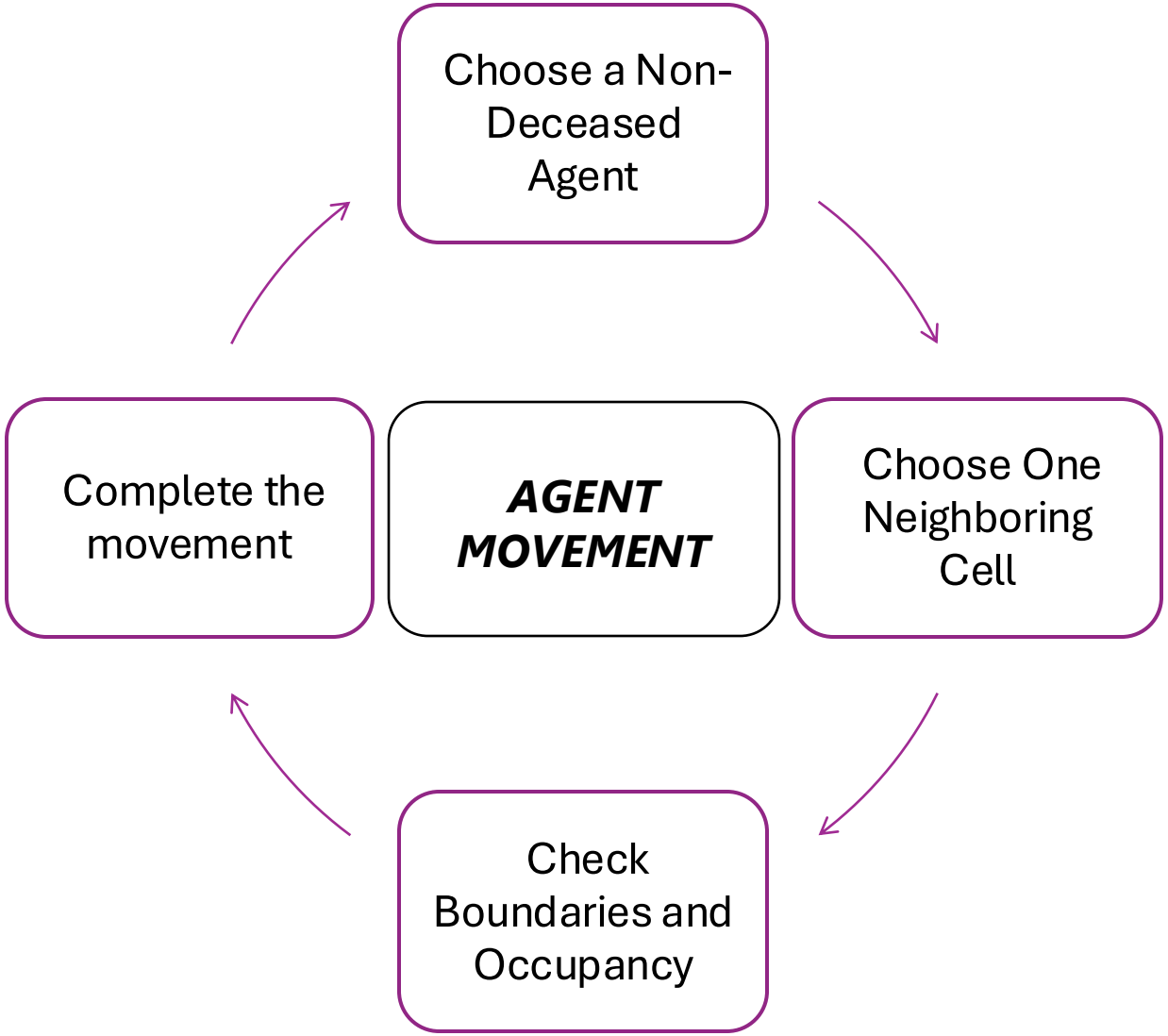


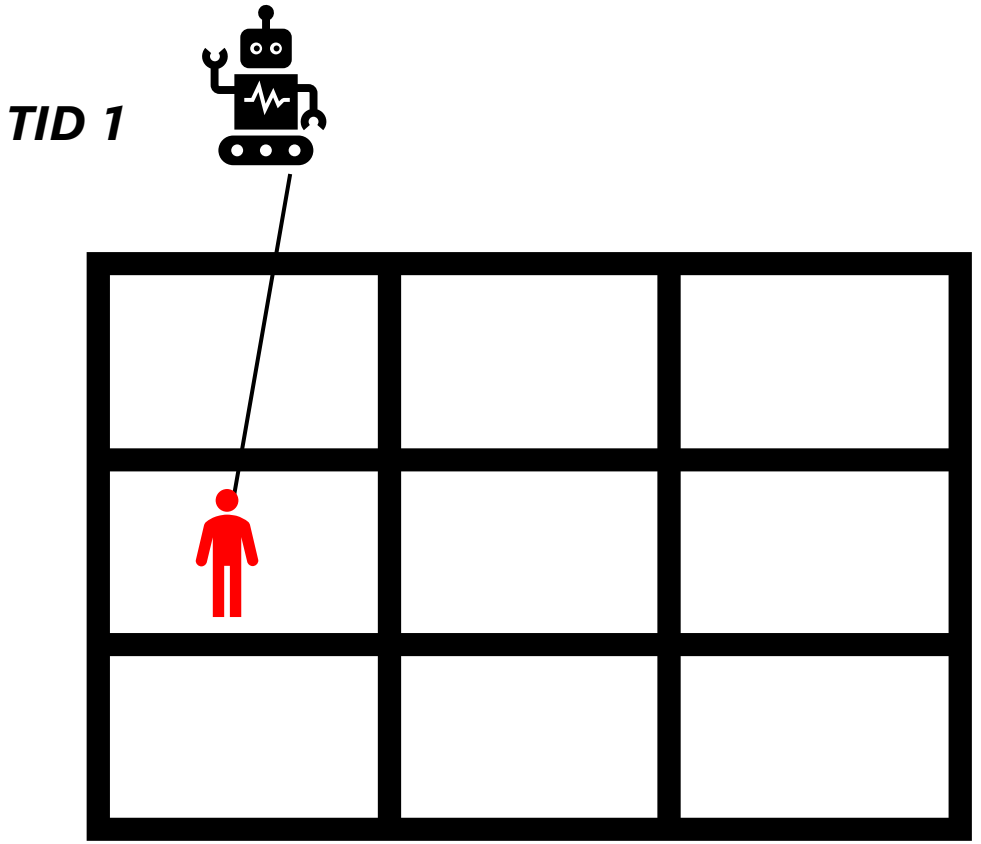
Solution Description: OpenMP Version – Daily Simulation





Solution Description: OpenMP Version – Daily Simulation





Solution Description: SERIAL Version – Daily Simulation

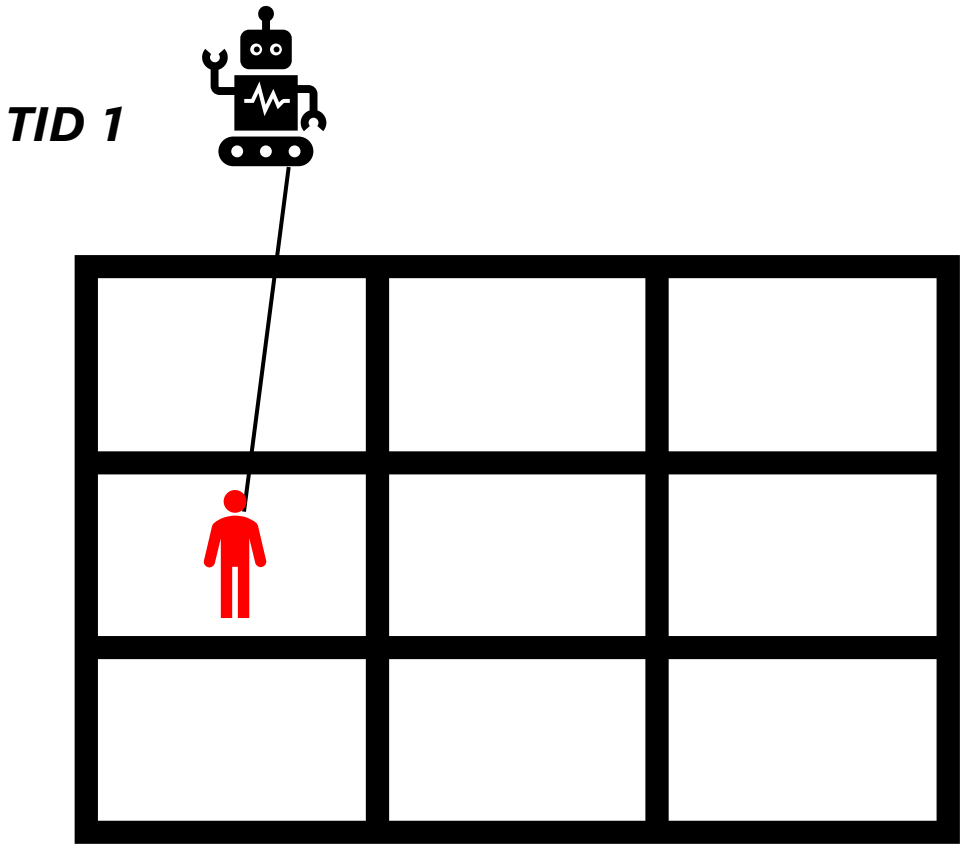
STATE UPDATE

Choose an Infected with INC = 1

Probability μ: Recover

Probabilty 1-μ: Death

50% Chance: Immunity



Solution Description: SERIAL Version – Daily Simulation

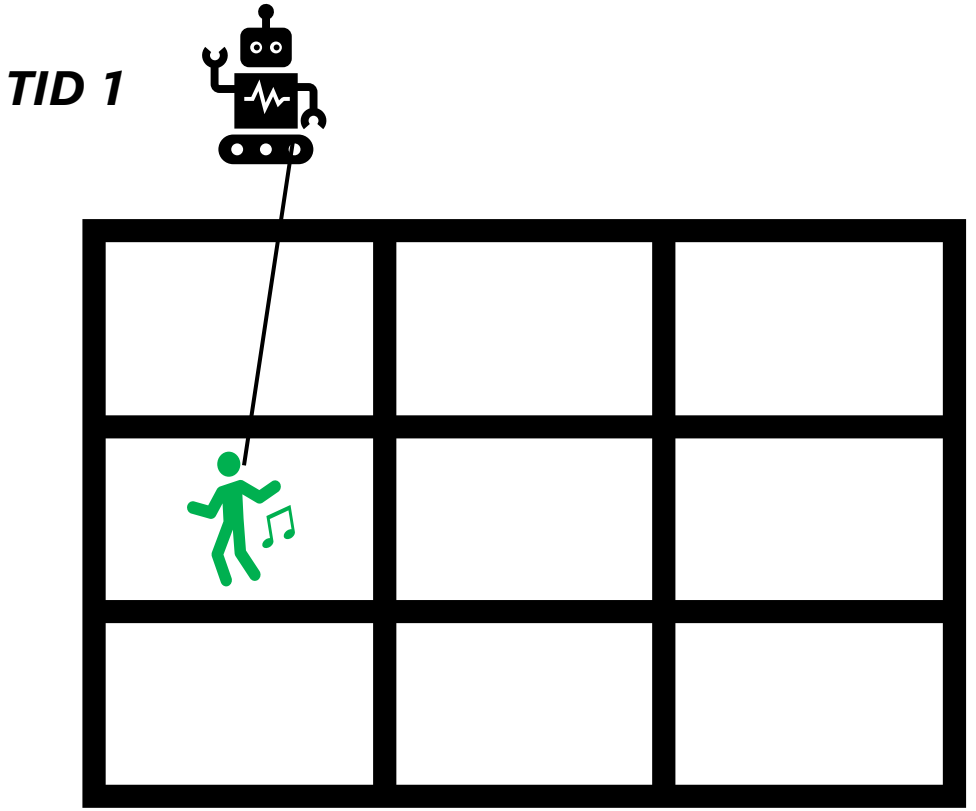
STATE UPDATE

Choose an Infected with INC = 1

Probability μ: Recover

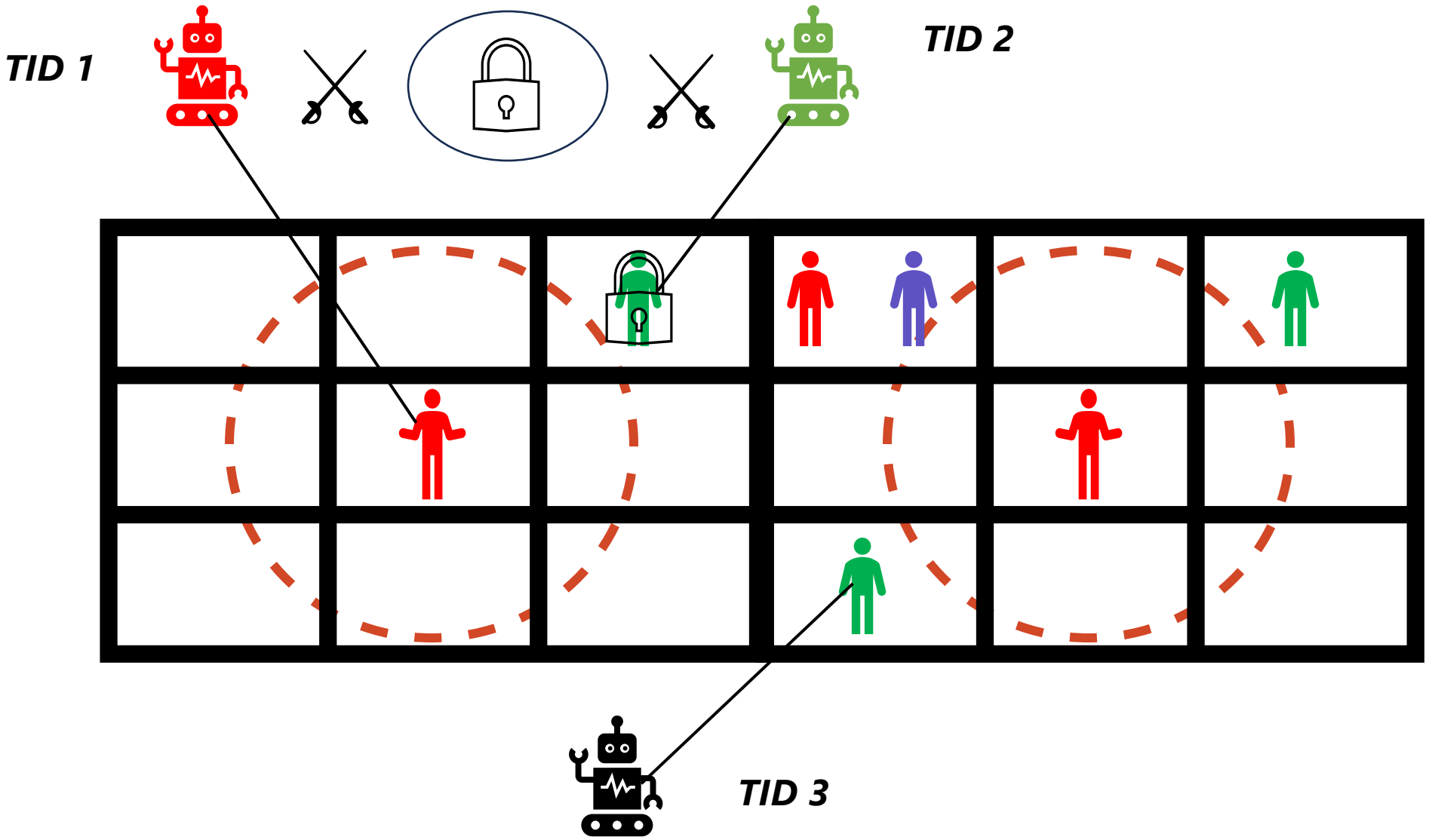
Probabilty 1-μ: Death

50% Chance: Immunity



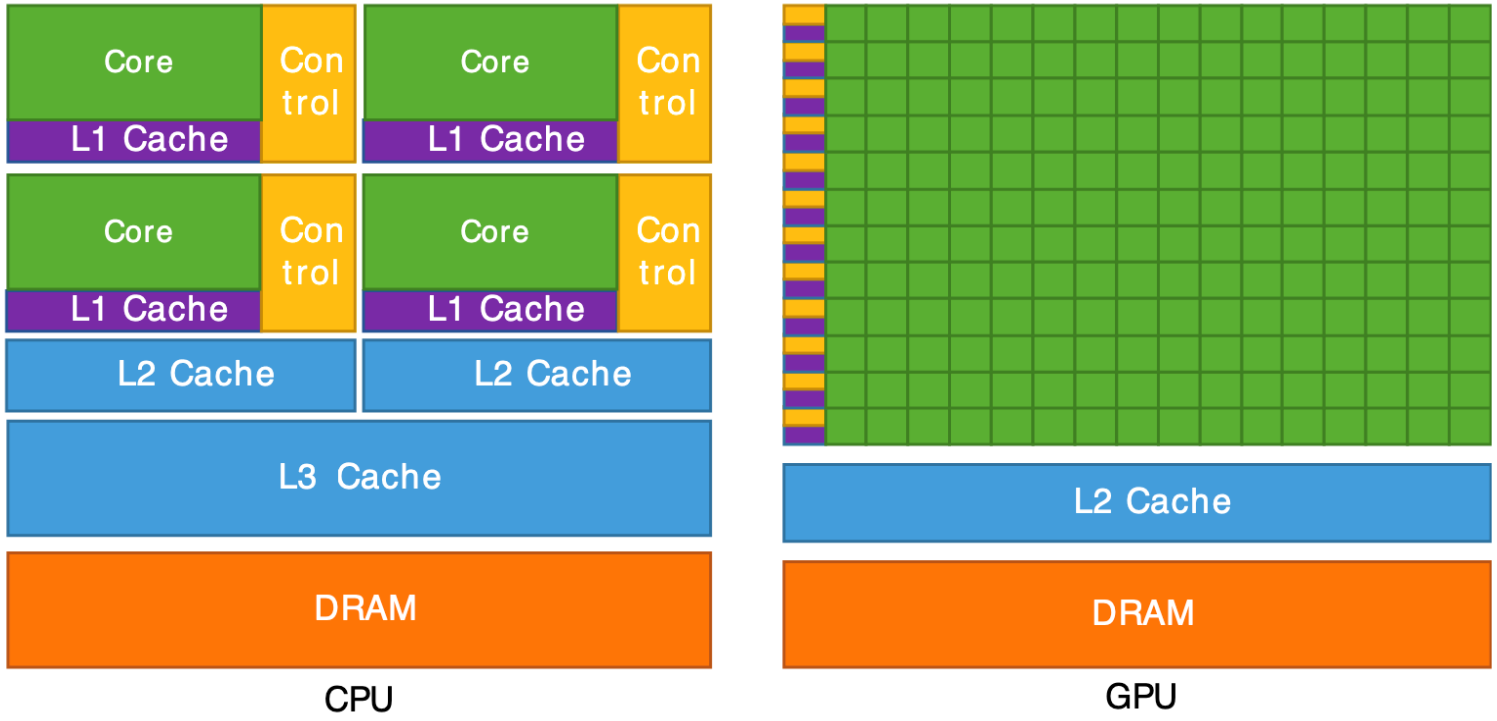
- ***Per-thread Newly recovered***
- ***Incubation_days = -2***

Solution Description: OpenMP Version – Daily Simulation

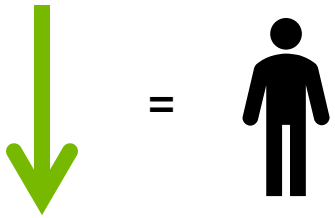


Solution Description: CUDA Version

Change of architecture



Solution Description: CUDA Version

Thread  ***Person***

```
int threads = # multiple of a warp
int blocks = (NP + threads - 1) / threads;
```

Grid = Population

Solution Description: CUDA Version

On-chip Memory

Block shared memory

Fast, Low latency

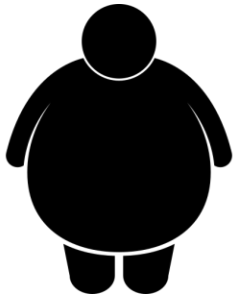
small

Off-chip Memory

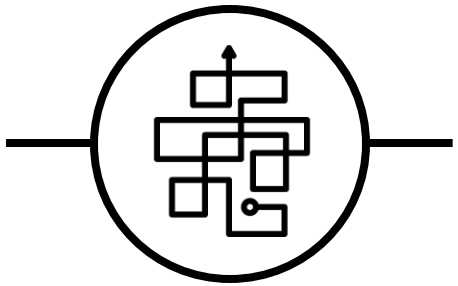
Global memory

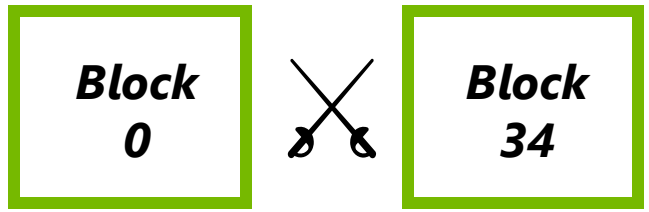
High latency

large



Tiling





Global synchronization

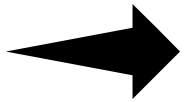
Solution Description: CUDA Version --> Togliere codice, rimettere le icone di slide 6

coalesced memory access

Array Of Structures

```
typedef struct {
    int x, y;
    float susceptibility;
    int incubation_days;
} Person;

typedef struct {
    int occupancy;
    Person **persons;
} Cell;
```



Structure of Arrays

```
int *d_x;
int *d_y;
int *d_incub;
int *d_newInf;
float *d_susc;

int *d_cellCount;
int *d_slotIndex;
int *d_cellSlots;
```

Solution Description: CUDA Version

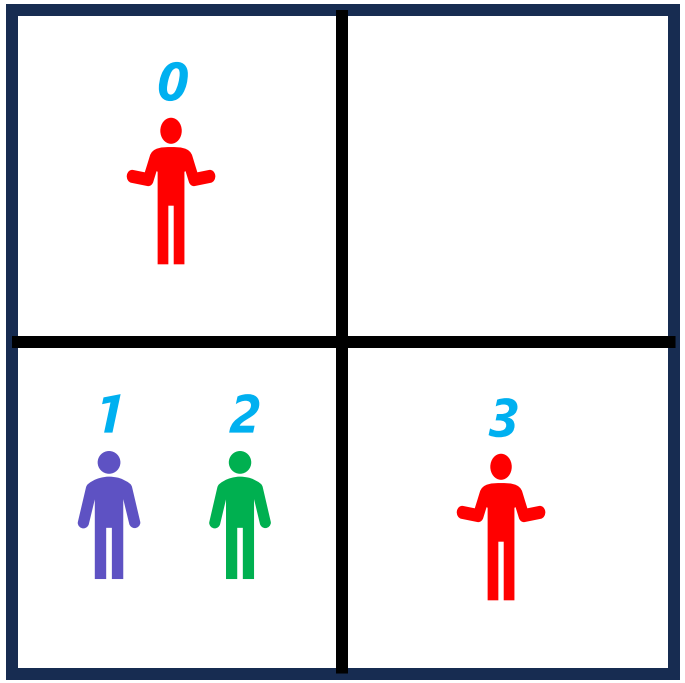
*cell count (W*H)*

1	0	2	1

slot index (NP)

0	0	1	0

Max People per cell = 2



*cell slots (W*H*MaxP)*

0	#	#	#	1	2	3	#

Solution Description: CUDA Version

*cell count (W*H)*

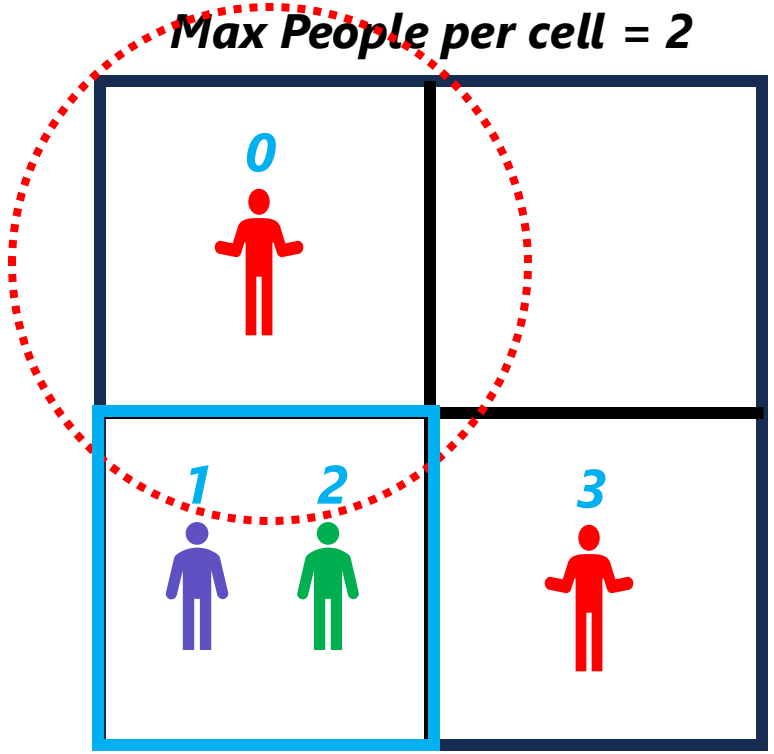
1	0	2	1

slot index (NP)

0	0	1	0

*cell slots (W*H*MaxP)*

0	#	#	#	1	2	3	#



Solution Description: CUDA Version

*cell count (W*H)*

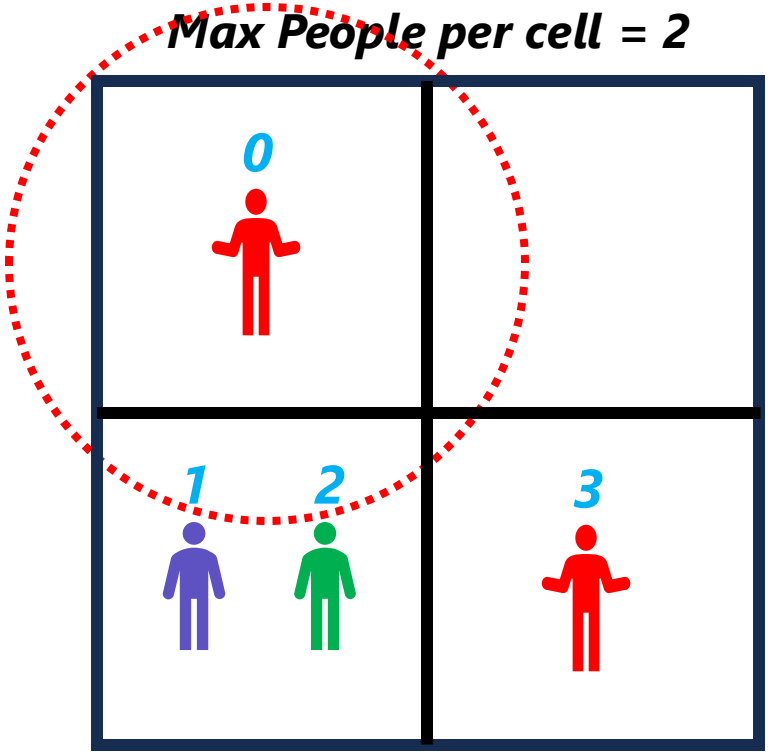
1	0	2	1

slot index (NP)

0	0	1	0

*cell slots (W*H*MaxP)*

0	#	#	#	1	2	3	#



Solution Description: CUDA Version

*cell count (W*H)*

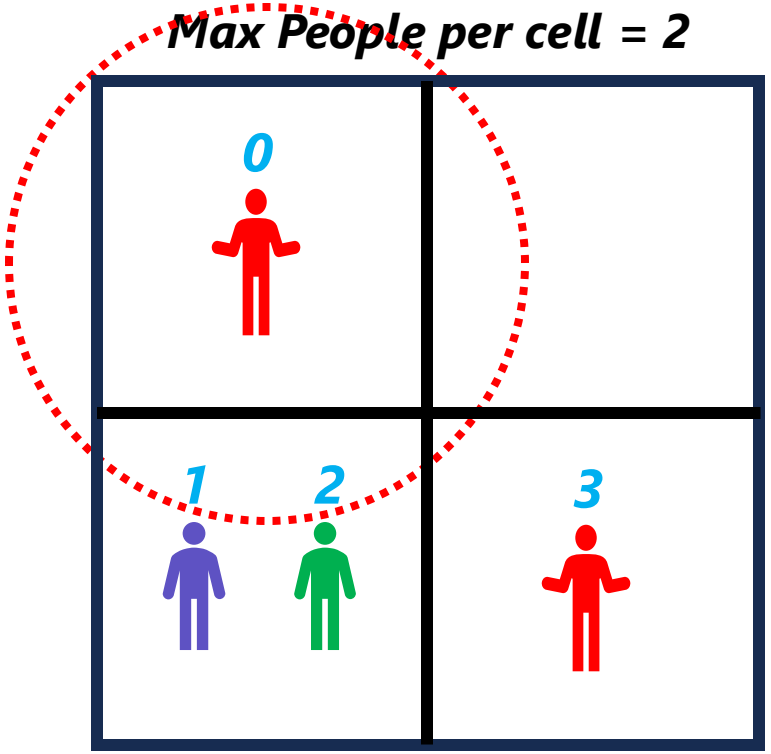
1	0	2	1

slot index (NP)

0	0	1	0

*cell slots (W*H*MaxP)*

0	#	#	#	1	2	3	#



Solution Description: CUDA Version

*cell count (W*H)*

1	0	2	1

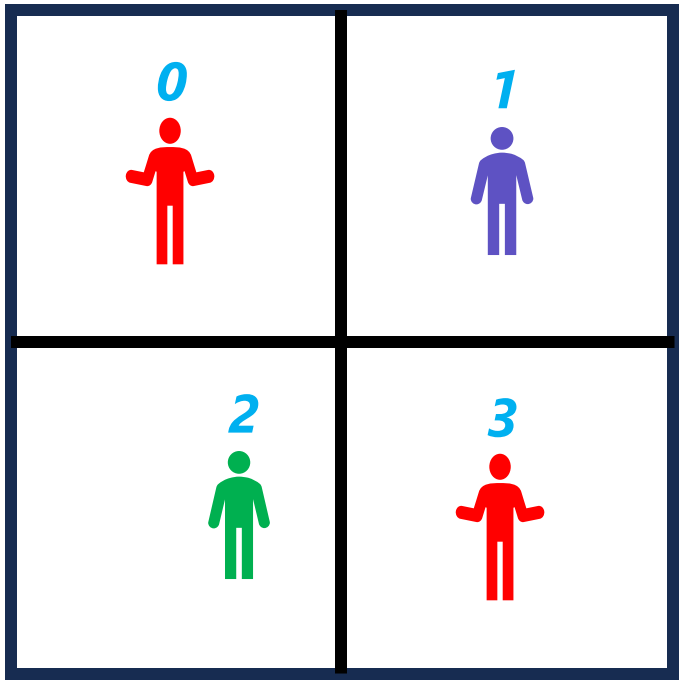
slot index (NP)

0	0	1	0

old Slot index

0

Max People per cell = 2



*cell slots (W*H*MaxP)*

0	#	1	#	1	2	3	#

Solution Description: CUDA Version

*cell count (W*H)*

1	1	1	1

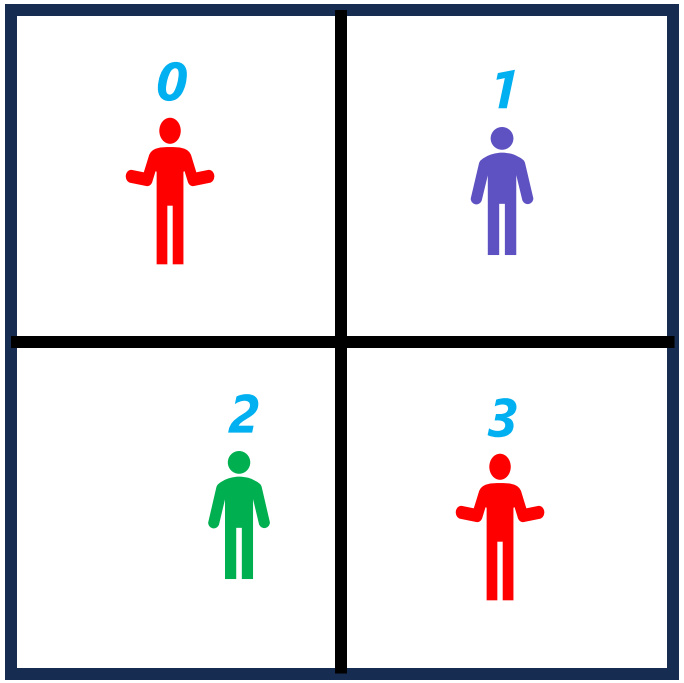
slot index (NP)

0	0	0	0

old Slot index

0

Max People per cell = 2

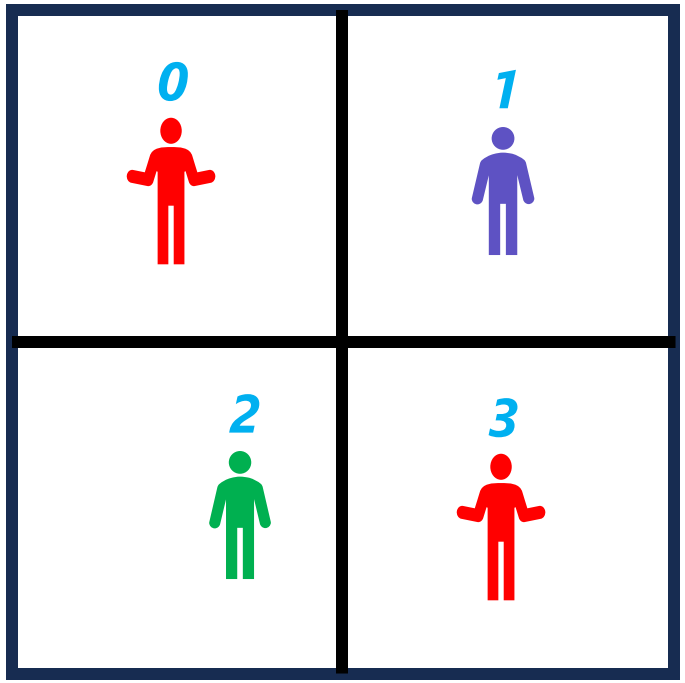


*cell slots (W*H*MaxP)*

0	#	1	#	1	2	3	#

Solution Description: CUDA Version

Max People per cell = 2



*cell count (W*H)*

1	1	1	1

slot index (NP)

0	0	0	0

old Slot index

0

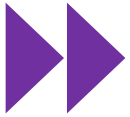
*cell slots (W*H*MaxP)*

0	#	1	#	2	#	3	#

Solution Description: CUDA Version

Infection kernel

status update kernel



movement kernel

Solution Description: CUDA Version

Infection kernel

```
if (infec > ITH)
{
    atomicCAS(&d_newInf[i], 0, 1);
}
```

status update kernel

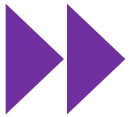
movement kernel

Solution Description: CUDA Version

Infection kernel

```
if (infec > ITH)
{
    atomicCAS(&d_newInf[i], 0, 1);
}
```

status update kernel



movement kernel

```
int pos = atomicAdd(&d_cellCount[newC], 1);
if (pos < MAXP_CELL)
{
    ...
}
else
{
    // rollback
    atomicSub(&d_cellCount[newC], 1);
}
```

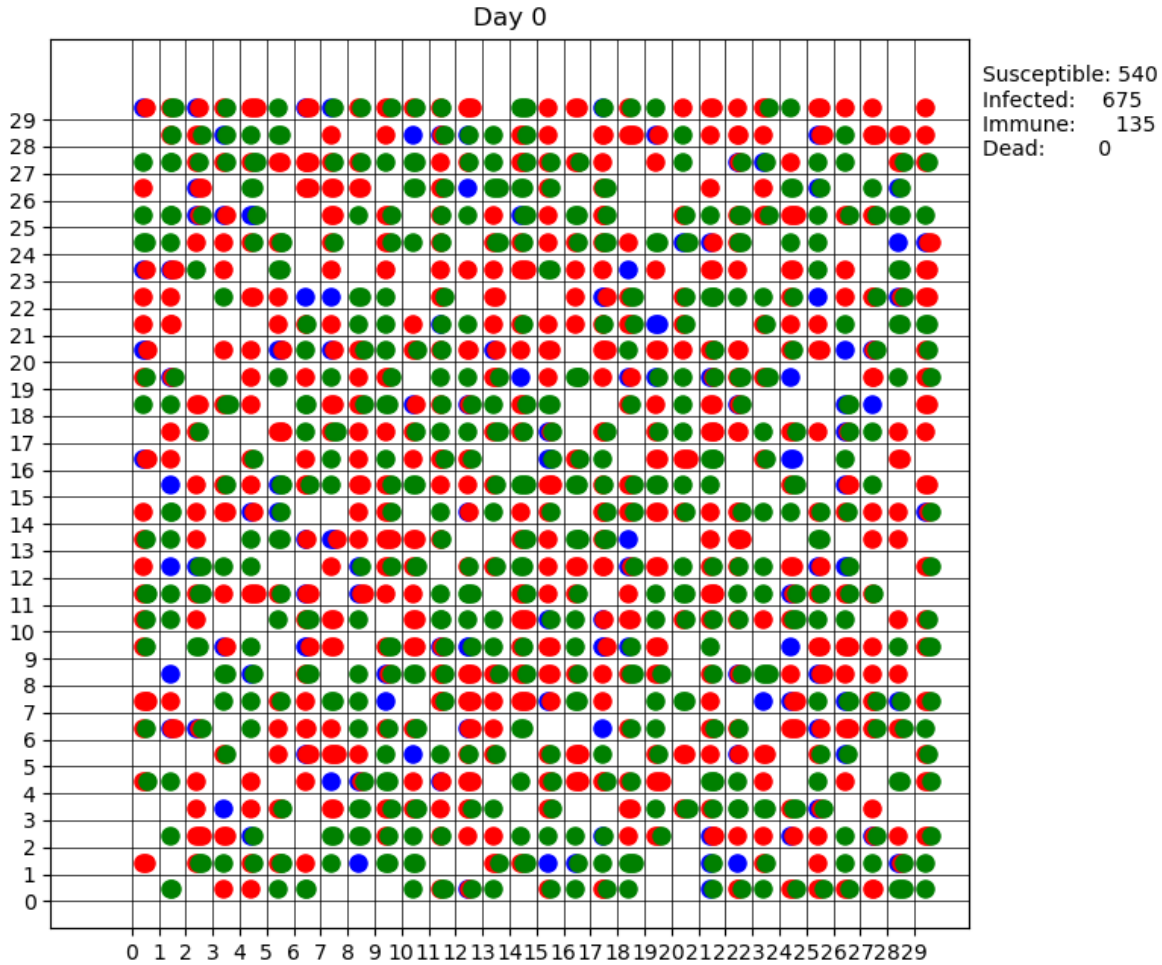
Solution Description: CUDA Version

all data structures are on device except for

```
clock_gettime(CLOCK_MONOTONIC, &ts_rm1);
cudaMemcpy(d_x, h_x, NP * sizeof(int), cudaMemcpyHostToDevice);
cudaMemcpy(d_y, h_y, NP * sizeof(int), cudaMemcpyHostToDevice);
clock_gettime(CLOCK_MONOTONIC, &te_rm1);
```

we expect
Cuda ≅ Cuda Evo

Solution Description: Report



- ***Report of the simulation for each day***
- ***Saving data about state and positions***
- ***Saving data to binary files: implementation-agnostic manner***

Solution Description: Report

Unit test

- cell occupancy limit
- deaths never decrease
- initial immune and infected counts
- incubation duration and transition
- cell movement
- state counts sum to NP
- positions valid or dead
- entries per day

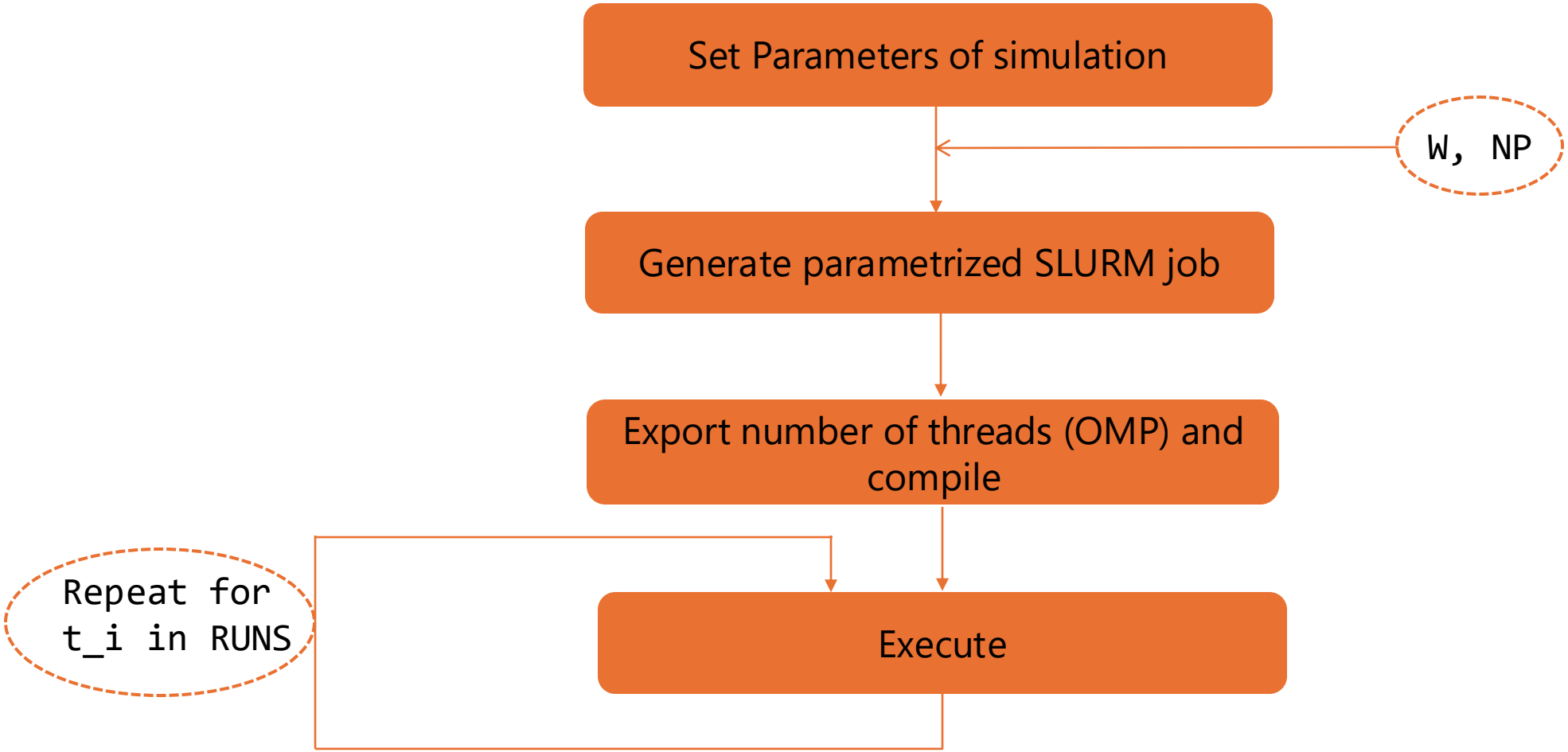
Experimental Results: Platform

- ***HPC Polito (Hactar)***
  - ***CPU Node:*** 2x Xeon E5-2680 v3 2.50 GHz (turbo 3.3 GHz) 12 cores
  - ***GPU Node:*** 2x Tesla K40 – 12GB – 2880 cuda cores
  - **Job Scheduler:** SLURM

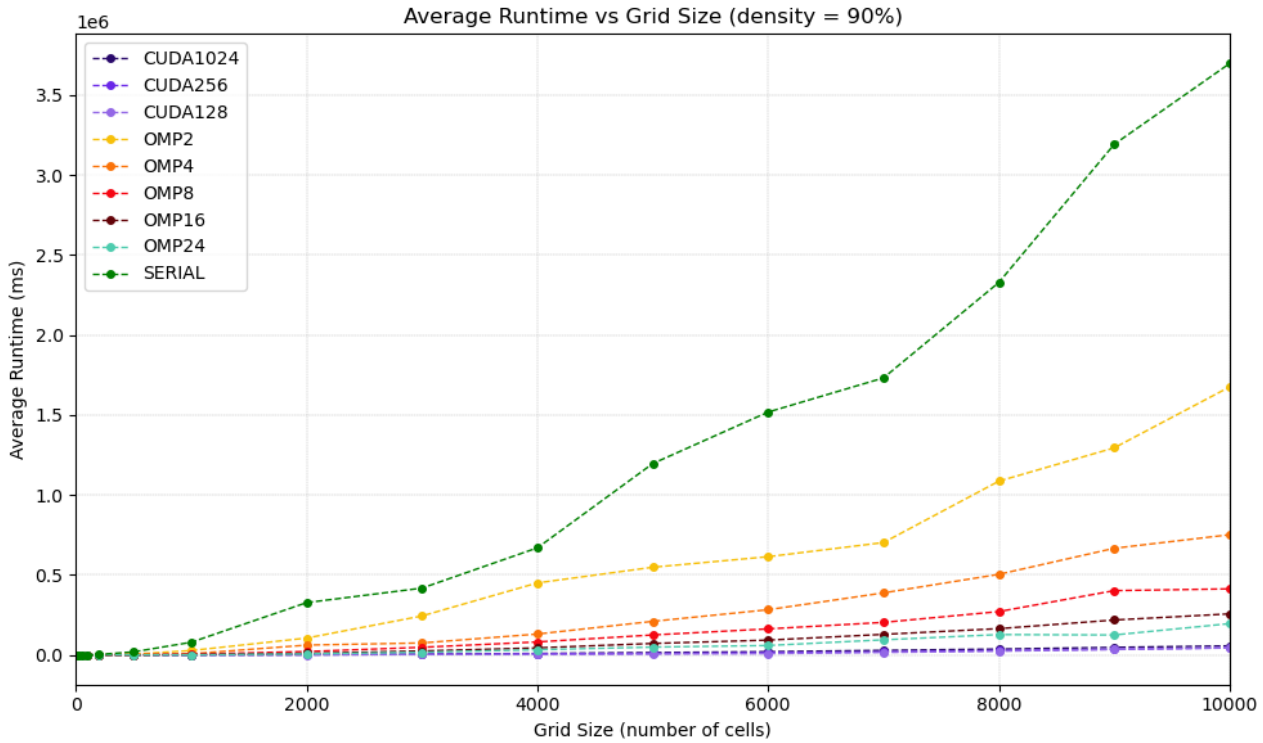
slurm
workload manager

Experimental Results: **Simulation File Generation**

- Study **performance** of the three implementations with increasing input dimension: Serial, OpenMP and Cuda



Experimental Results: Performance Analysis



DATASET:

- RUNS: 10
- Logical parameters: default values
- Grid Sizes: from 10x10 to 10000x10000
- Population Densities: 0.1, 0.5 and **0.9**

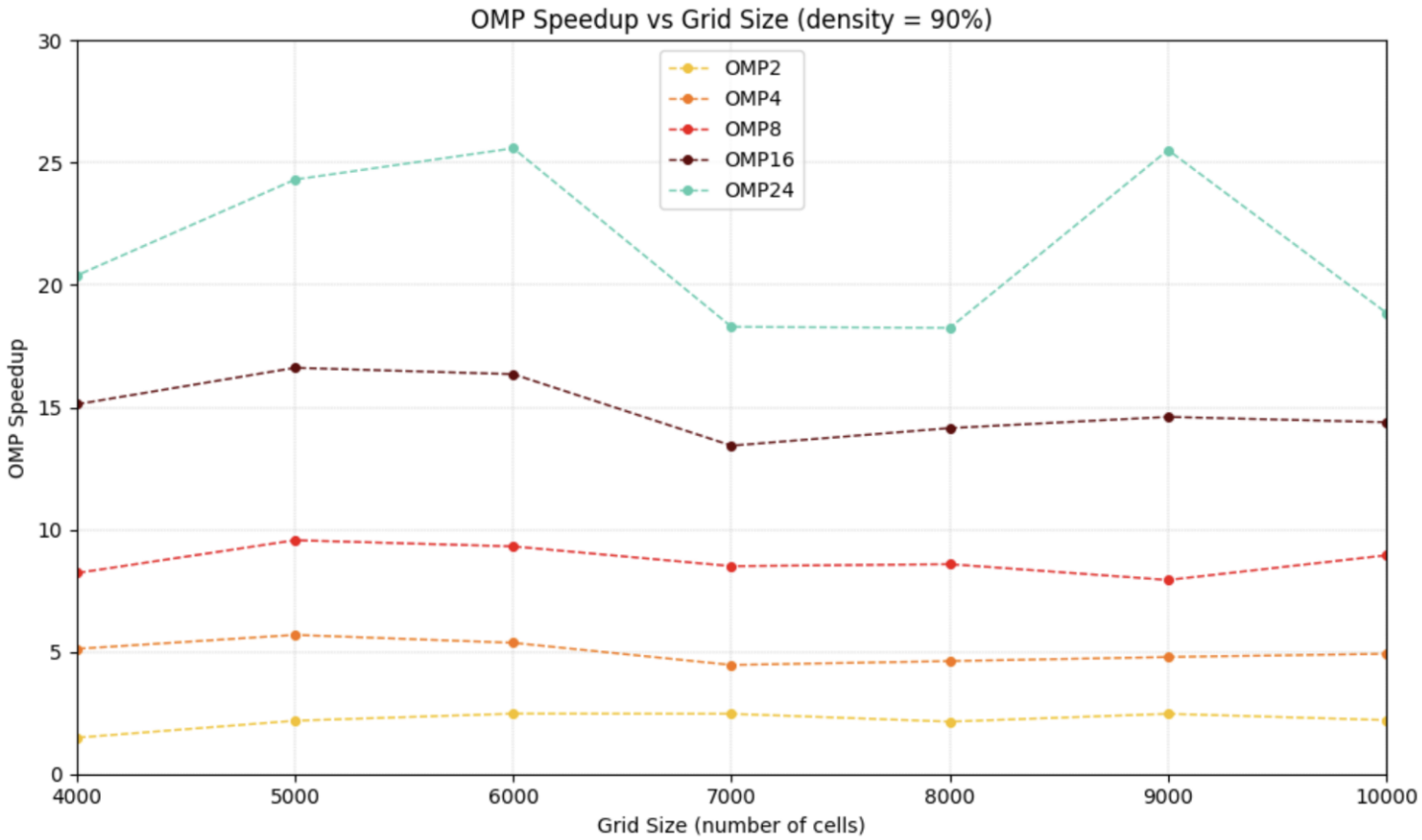
OPENMP:

- Number of Threads = {2, 4, 8, 16, 24}

CUDA:

- Number of Threads per Block = {128, 256, 1024}

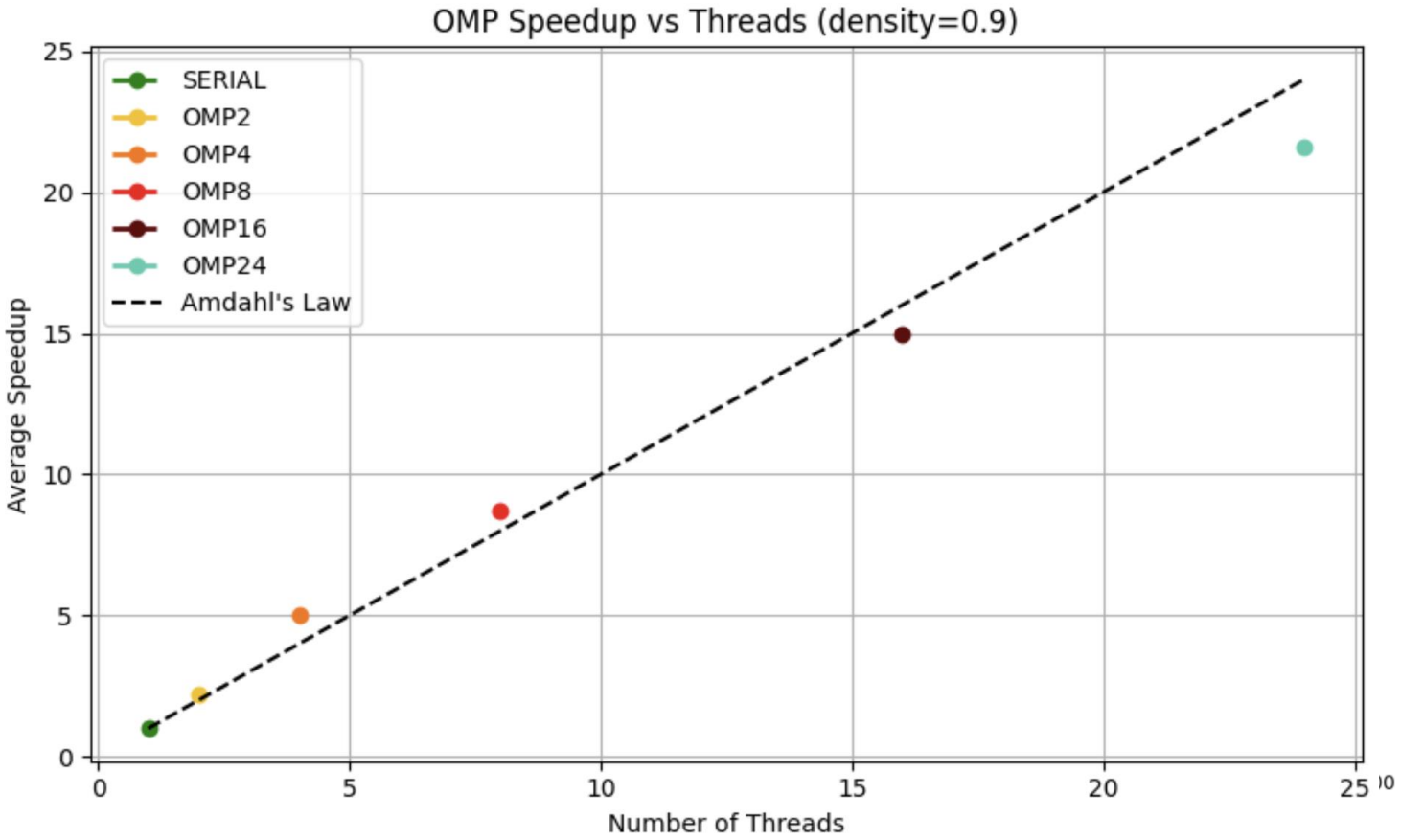
Experimental Results: Speedup - OpenMP



SPEEDUP $\in [2, 24]$

- $\frac{t_{serial}}{t_{OMP(N)}} \approx N$, with N number of threads

Experimental Results: Speedup - OpenMP



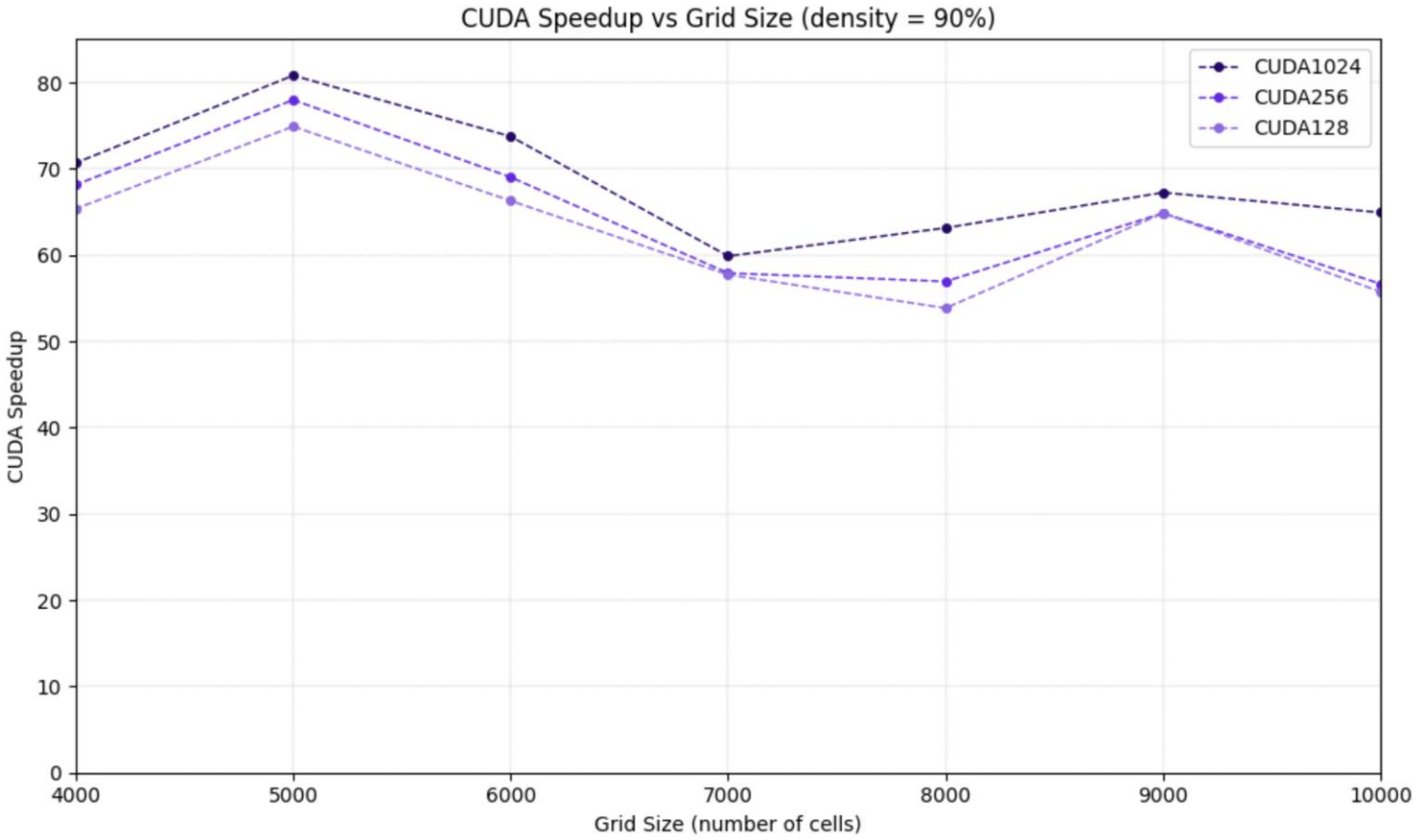
SPEEDUP $\in [2, 24]$

- $\frac{t_{serial}}{t_{OMP(N)}} \approx N$, with N number of threads
- **Linear trend** in line with Amdahl Law:

$$\boldsymbol{t(N) = t_{serial} + \frac{t_{parallel}}{N}}$$

  - $\boldsymbol{t_{serial} \ll t_{parallel}}$
  - Slight fluctuations due to inherent randomness

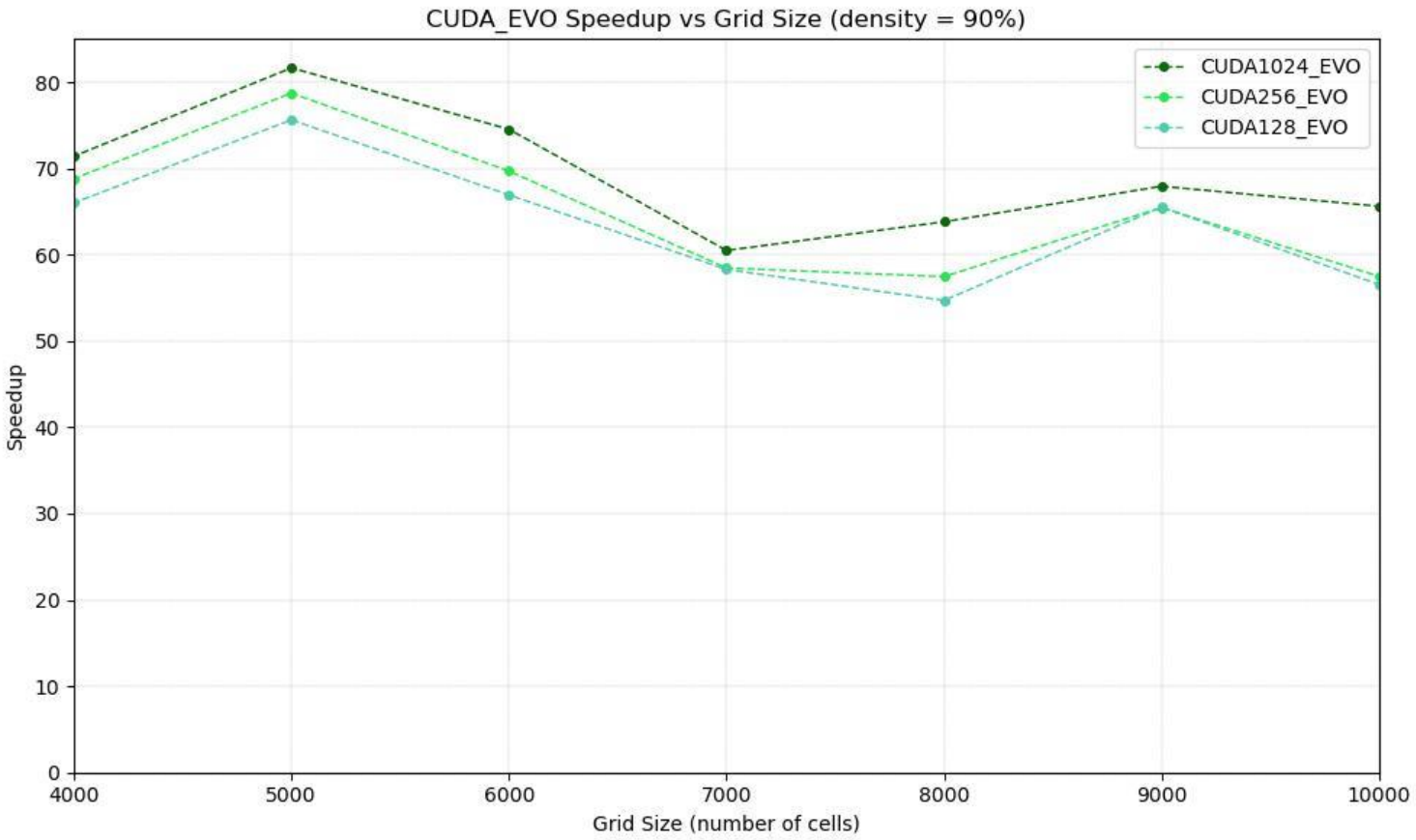
Experimental Results: Speedup - CUDA



SPEEDUP $\in [55, 80]$

- Huge performance **gain** for large grids thanks to thousands of threads
- Speedup almost **independent** of number of threads per block

Experimental Results: Speedup - CUDA

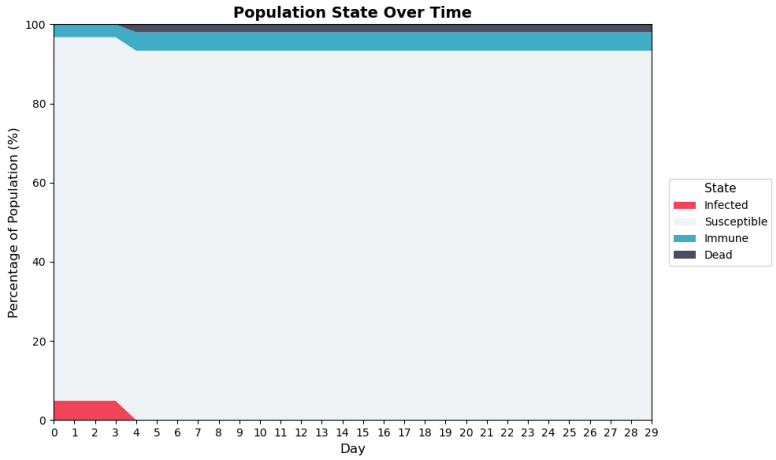


SPEEDUP $\in [55, 80]$

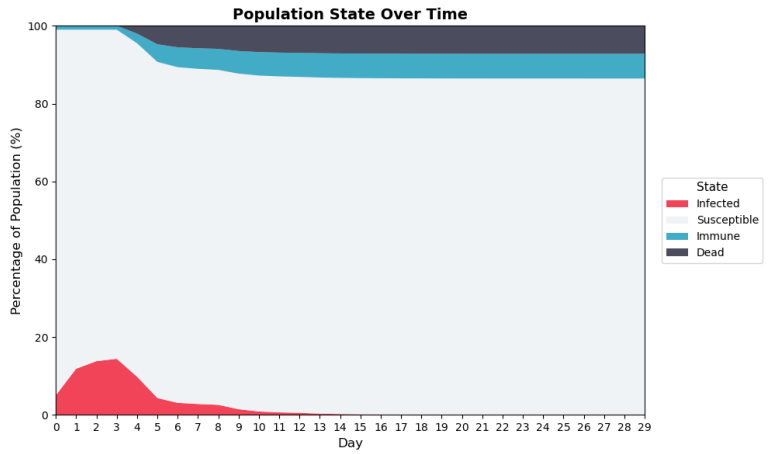
- Huge performance **gain** for large grids thanks to thousands of threads
- Speedup almost **independent** of number of threads per block
- **Negligible impact** of data transfers on performance (CUDA ≈ CUDA EVO)

Experimental Results: Logical Simulation

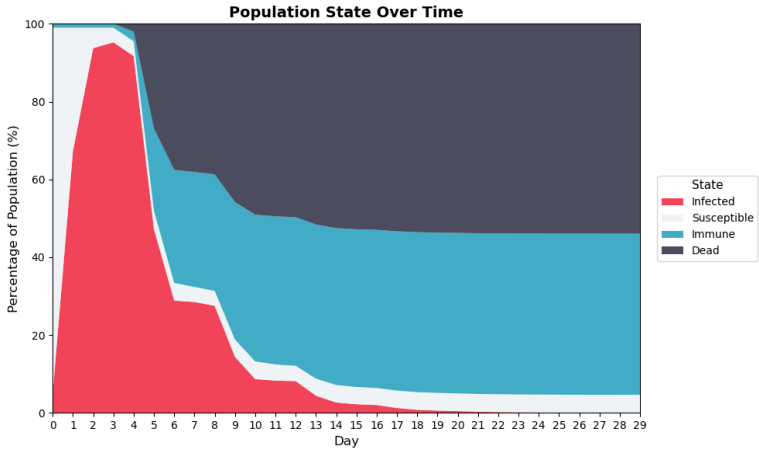
- **Q**: Most **critical** parameter for population?
  - ➢ A: Fix initial conditions and simulate evolution of population for varying S_{AVG} and μ



$S_{AVG} = 0.3, \mu = 0.6$



$S_{AVG} = 0.6, \mu = 0.6$



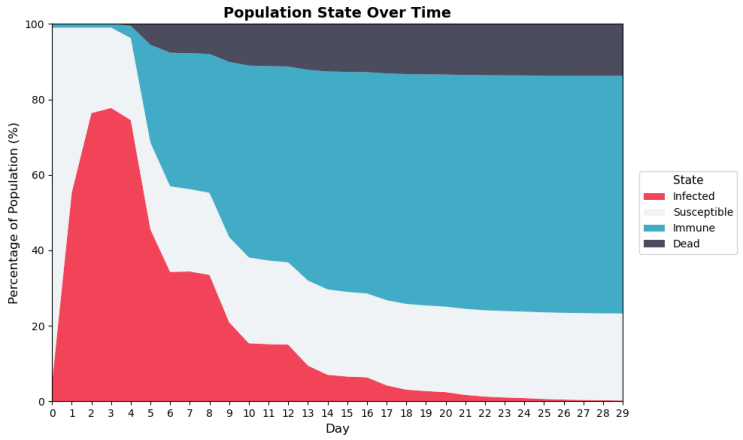
$S_{AVG} = 0.9, \mu = 0.6$

→ Threshold – behaviour vs S_{AVG} due to infection condition:

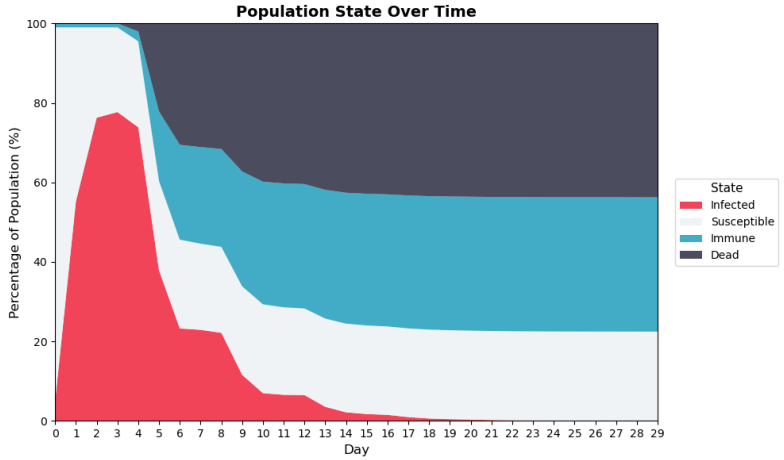
$$S_i >^{?} \frac{ITH}{\beta}$$

Experimental Results: Logical Simulation

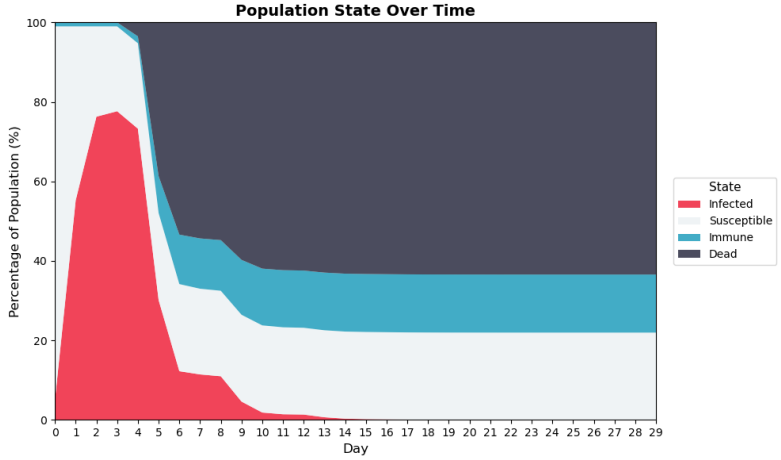
- **Q**: Most **critical** parameter for population?
  - ➢ A: Fix initial conditions and simulate evolution of population for varying $\mathbf{S_{AVG}}$ and $\boldsymbol{\mu}$



$S_{AVG} = 0.7, \mu = 0.9$



$S_{AVG} = 0.7, \mu = 0.6$



$S_{AVG} = 0.7, \mu = 0.3$

→ By the Law of Large numbers:

$$\boldsymbol{num\_Immune} \approx \boldsymbol{\mu} \times \boldsymbol{num\_Dead} \quad \text{as} \quad t \to \infty$$

Conclusions

- **Problem:** Simulate propagation of epistemics in a population

- **Three solutions**: Serial, OpenMP and Cuda:
  - Testing
  - Performance Evaluation on HPCPolito
  - Logical Simulation Analysis

- **Parallel Speedup**:
  - **OMP**: $\in [2, 24]$ linear with the number of threads
  - **CUDA**: ≈ 60, approx. constant with number of threads per block, no impact of memory transfer

Thank you for your attention!

Federico Muscarà *Marco Parentin* *Leonardo Straccali* *Francesco Zanasi*