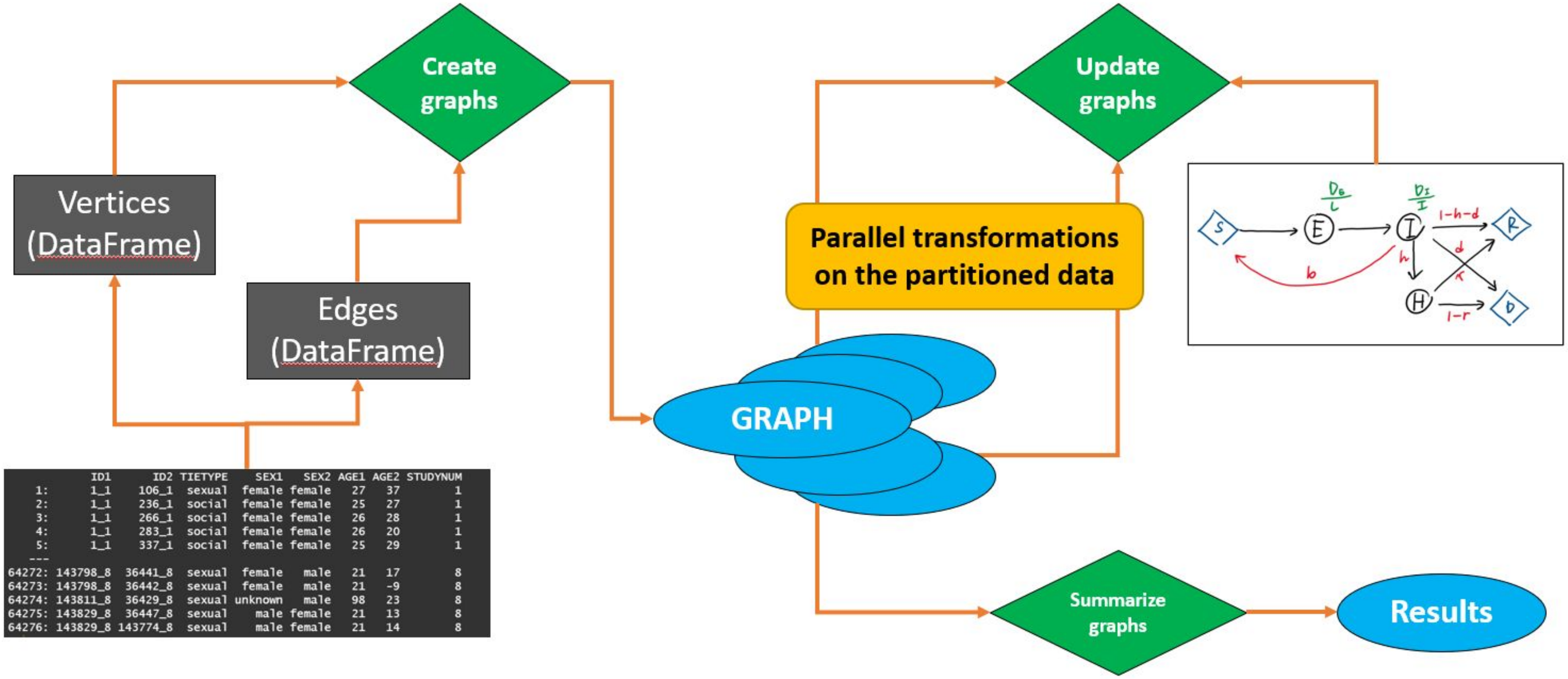
Efficient simulation and graphical modeling of Covid-19 spread

Group 13
Camille Bean, Intekhab Hossain, Hui Li and Stephanie Wu

Overall approach

- ➢ **Type of application:** Data-intensive
- ➢ **Programming model:** Functional dataflow parallel processing (Spark)
- ➢ **Levels of parallelism:** Coarse-grained transformations on partitioned data
- ➢ **Parallel execution model:** Single program - multiple data

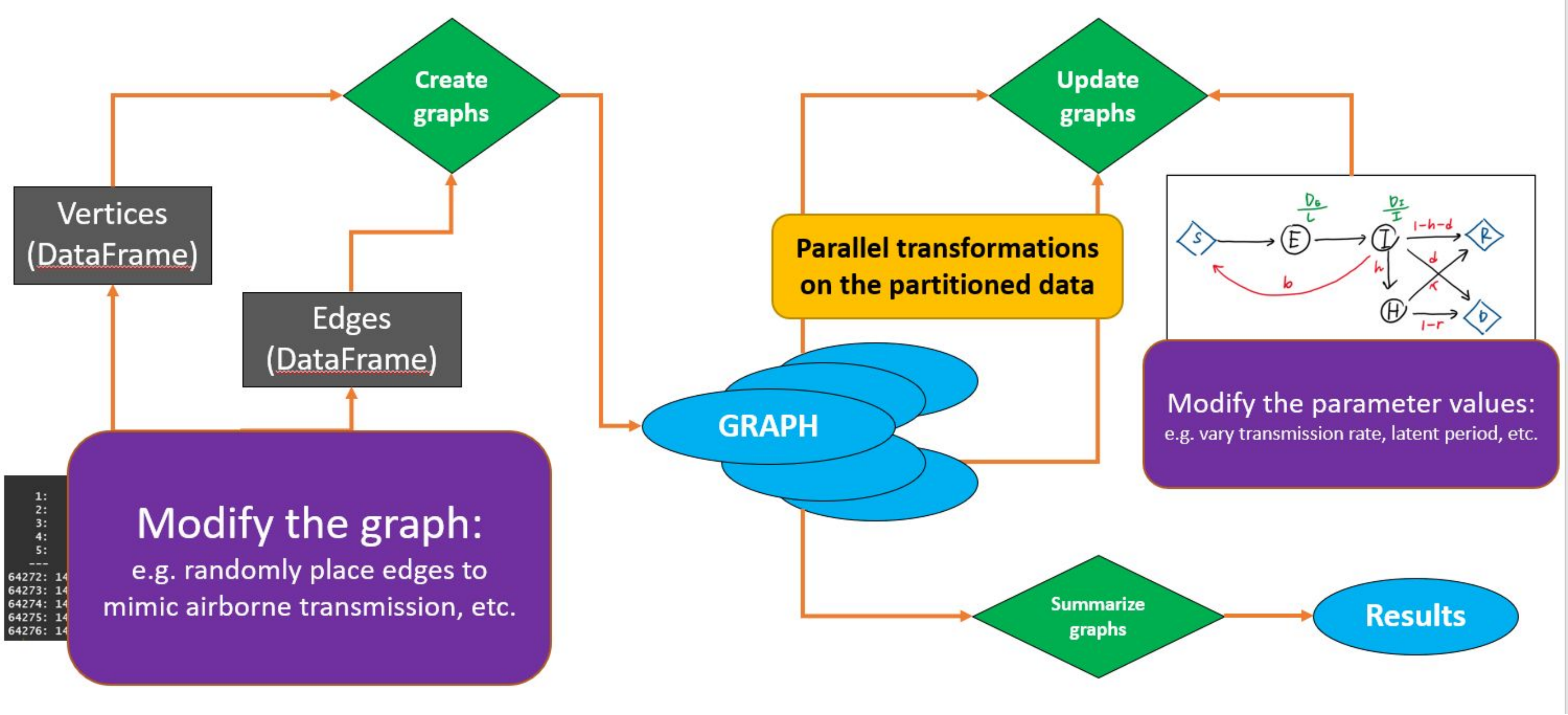


	ID1	ID2	TIETYPE	SEX1	SEX2	AGE1	AGE2	STUDYNUM
1:	1_1	106_1	sexual	female	female	27	37	1
2:	1_1	236_1	social	female	female	25	27	1
3:	1_1	266_1	social	female	female	26	28	1
4:	1_1	283_1	social	female	female	26	20	1
5:	1_1	337_1	social	female	female	25	29	1

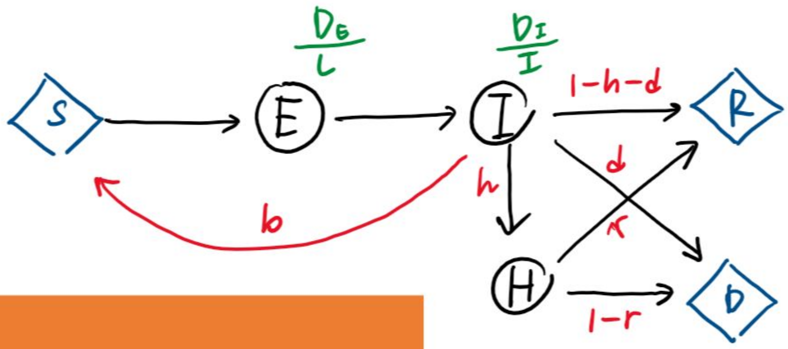
64272:	143798_8	36441_8	sexual	female	male	21	17	8
64273:	143798_8	36442_8	sexual	female	male	21	-9	8
64274:	143811_8	36429_8	sexual	unknown	male	98	23	8
64275:	143829_8	36447_8	sexual	male	female	21	13	8
64276:	143829_8	143774_8	sexual	male	female	21	14	8

Overall approach

- **Type of application:** Data-intensive
- **Programming model:** Functional dataflow parallel processing (Spark)
- **Levels of parallelism:** Coarse-grained transformations on partitioned data
- **Parallel execution model:** Single program - multiple data

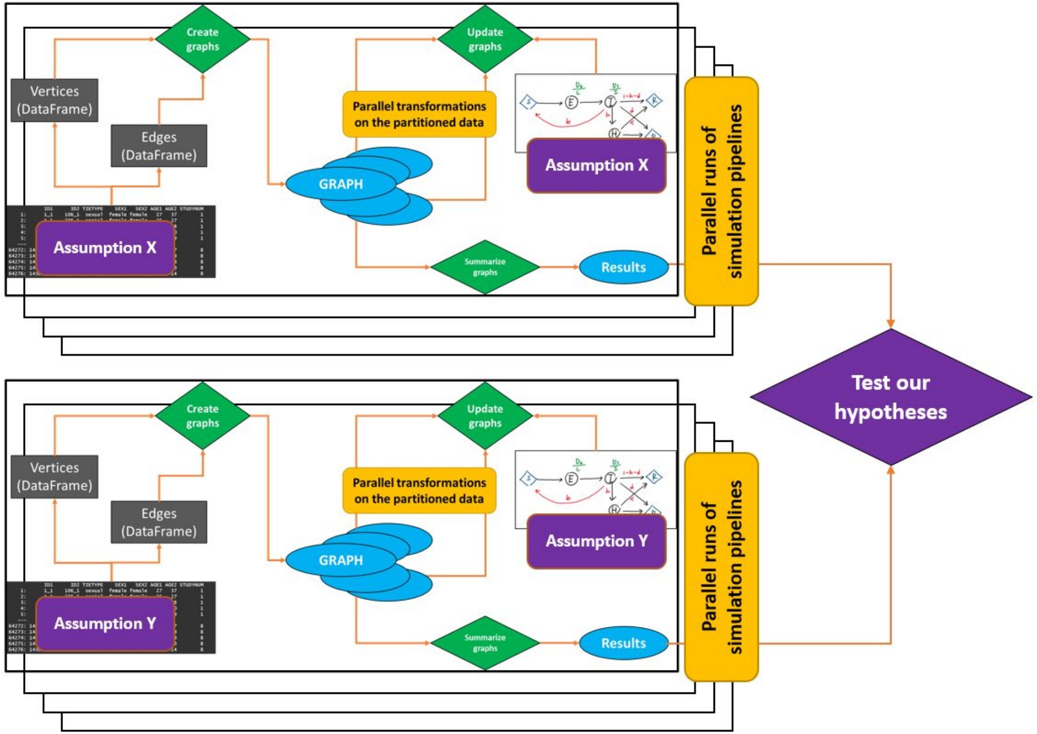


Extended SEIR model



	Estimate	Distribution	Estimation procedure
Transmission rate	1.596 (time-series data available)	Gamma	Used EpiEstim software, assuming a serial interval of 3.96 (4.75) from Du et al. 2020. U.S. national level data from Covid Tracking Project, dates ranging from 2/19 to 4/18.
Hospitalization rate	0.0678	Beta	Daily hospitalized increase / Daily case increase. U.S. national level data from Covid Tracking Project, averaged across available dates from 2/19 to 4/18.
Death rate	0.0419	Beta	Daily death increase / Daily case increase. U.S. national level data from Covid Tracking Project, averaged across available dates from 2/19 to 4/18.
Recovery rate	0.3945	Beta	Daily new recovery / Daily hospitalized increase. U.S. national level data from Covid Tracking Project, averaged across available dates from 2/19 to 4/18.
Latent period	5.2 days	Poisson	Taken from the literature (An et al. 2020)
Infectious period	2.3 days	Poisson	Taken from the literature (An et al. 2020)

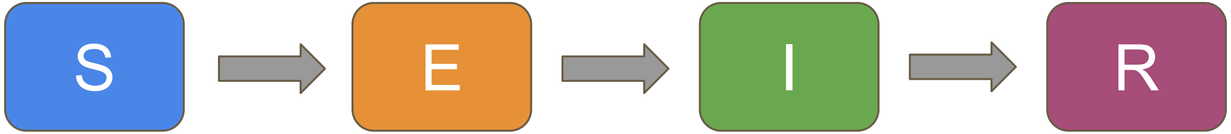
Parallel Execution Model



- **Type of application:**
  - Data-intensive
- **Programming model:**
  - MapReduce to process summary statistics
- **Levels of parallelism:**
  - Coarser-grained distributed data processing
- **Deployment mode:**
  - Local mode for prototyping
  - Cluster mode for scaled-up application

Programming Model and Infrastructure

- Spark is used for big data processing within each graph
  - SEIR model implemented using 2 Spark dataframes, supported by GraphFrames.
  - Run using AWS EC2 instance, or AWS EMR Spark-configured cluster for multiple worker nodes.
  - Written in Python, pyspark used for Spark
- AWS instances modified for parallelization within graphs
  - MPI through mpi4py for parallel processing, while avoiding serial steps
- Parallelization across simulations carried out by EMR cluster



Potential Bottlenecks

- Algorithm is composed of three nested for loops:
  - <u>Outermost</u>: loops over each Monte Carlo simulation
  - <u>Intermediate</u>: loops over each time step
  - <u>Innermost</u>: loops over each nodes
- Outermost loop can be parallelized easily (independent iterations)
- Innermost loop can be broken-up to deal with each of the four states separately, and can be parallelized for *within* each state.
- Intermediate loop **cannot** be parallelized since the results of a given time-step will depend on the previous time-step (**primary bottleneck**).
- Other bottlenecks = communication, synchronization, balancing, b/w units

Complexity and Theoretical Speed-up

Complexity: O(MTN), where

M = # Monte Carlo simulations

T = # of time steps

N = # of vertices in graph

```
for sim in M:
    for time in T:
        for p in H_nodes:
            # update H -> R, D
        for p in I_nodes:
            # update I -> R, H, D
        for p in E_nodes:
            # update E -> I
        for p in S_nodes:
            # update S -> E
```

Asymptotic S_T for large p => $\dfrac{1}{1-c}$

```
#serial runtime:
O(M*T*(S+E+I+H)) ~= O(M*T*N)

#parallel runtime:
O(M/M * T * (S/S + E/E + I/I + H/H)) = O(4T)

#max theoretical speedup:
O(MTN) / O(4T) = O(MN/4)

# Serial fraction, 1-c:
1-c = 1/max_speed_up = 4/O(MN)
```

Large number of vertices (N) and simulations (M)

=> serial fraction -> 0

=> speed-up should be approximately linear