Comparando Estrategias de Vacunacion SIRV con EB-DEVS

Emanuel Ferreyra Ivan Postolski

Abstract

Este trabajo presenta un modelado SIR [Wei13] con distintas estrategias de vacunación en EB-DEVS [FHUC21], un formalismo que extiende DEVS [CZ88] (manteniendo compatibilidad) y permite modelar propiedades emergentes.

1 Introducción

Los modelos SIR [Wei13, JLW14, Vol08] son modelos desarrollados para entender propagación de enfermedades, donde una población finita se divide en tres estados: Susceptibles, Infectados y Recuperados. Los individuos Susceptibles pueden contraer la enfermedad y pasar a ser Infectados, y continuar la propagación de la enfermedad a sus contactos hasta que eventualmente se recuperan y transicionan a un estado de Recuperado.

A su vez, los modelos SIR pueden ser extendidos con un cuarto estado de Vacunados (i.e., SIRV). En SIRV los individuos Susceptibles pueden transicionar dependiendo de una estrategia (que es la tasa de saltos del proceso) al estado de Vacunado, en donde no es posible infectarse.

Los contactos de los individuos (agentes en EB-DEVS) se pueden modelar con grafos aleatorios que siguen una determinada distribución de grados (conocidos como *Configuration Models* [FLNU18]). Estos grafos se pueden construir simultáneamente en EB-DEVS con la propagación de la enfermedad siguiendo un proceso de *Preferential Attachment*[MU17] conectando puertos de agentes de forma aleatoria dinámicamente [FHUC21].

Dado un modelo SIRV cuyos contactos siguen un *Configuration Model* con cierta distribución, es interesante preguntarse si existe una estrategia de vacunación óptima, donde una estrategia de vacunación ξ es mejor que otra ξ' cuando ξ reduce en promedio la cantidad total de individuos que se infectan usando la estrategia ξ'. Por supuesto, una estrategia óptima trivial consiste en vacunar a todos los individuos previamente a la primer infección. Por otro lado, es posible imaginar un escenario comparativo entre estrategias más interesante, en el que todos los agentes se vacunan con la misma esperanza [FJP21] modelando implícitamente el manejo de un recurso (e.g, costos en logística o centros de vacunación).

En este sentido, un conjunto interesante de estrategias a explorar consiste en ponderar el grado de un agente en la probabilidad de ser vacunado. Intuitivamente, la hipótesis es que los individuos con grados más altos de un grafo tienen un rol clave en la propagación de la enfermedad. Por lo

tanto, una estrategia que consuma los mismos recursos en promedio, pero que se apure a vacunar a los individuos de grado más alto podría, según la distribución de la red, ser mejor que una estrategia que no distingue grados.

De hecho, trabajos recientes muestran mediante análisis númericos que esto es factible [FJP21]. En este trabajo abordaremos esta pregunta desde la simulación, concretamente:

RQ1: Cómo se comparan las estrategias de vacunación ponderadas por los grados de los agentes con estrategias cuya intensidad no depende del grado?

Además, planteamos otro escenario interesante donde la cantidad de vacunas tiene un límite fijo global, que es posible de implementar por la arquitectura de EB-DEVS, donde un Agente puede acceder a un estado "global" de la simulación y viceversa.

RQ2: Cómo se comparan las estrategias de vacunación ponderadas por los grados de los agentes con estrategias cuya intensidad no depende del grado cuando hay un limite finito de vacunas disponibles?

2 Ejemplo Motivador

Presentamos en esta sección un ejemplo motivador. Supongamos que tenemos un grafo de conectividad entre agentes como el que se muestra en la Figura 2.1.

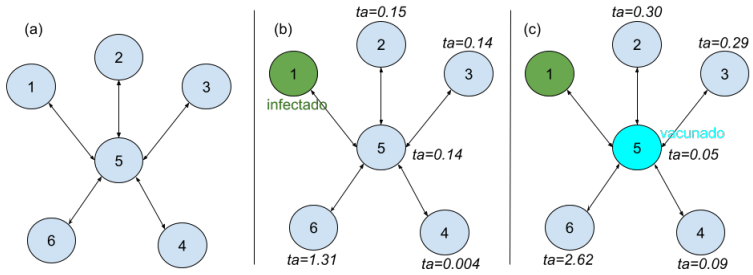

Figure 2.1: (a) Ejemplo de grafo de conectividad, (b) nodo 1 Infectado, ta de vacunación con probabilidad exponencial con media 0.2 independiente del grado, (c)ta de vacunación con probabilidad exponencial con media 0.2 dependiente del grado.

En este ejemplo la red se conforma con un nodo "hub", el 5 de grado 5, mientras los demás nodos tienen grado 1 y solamente se comunican con 5. En la estrategia que no pondera grados (b), se le asigna a todos los nodos un ta de vacunación exponencial con media 0.2, sin embargo, en este grafo es claro que vacunar a 5 rápidamente es lo más conveniente para detener la propagación de la enfermedad lo antes posible. Una de nuestras estrategias produciría, por ejemplo la asignación de ta como la que se puede ver en (c), donde el tiempo de vacunación mantiene la media 0.2 pero depende del grado, y el ta de vacunación de 5 se reduce tres veces. Esto podría lograr que 5 se vacune antes de recibir una señal de infección por parte de 1.

Por supuesto que el impacto de dicha estrategia depende del tiempo de propagación de la enfermedad, pues si la enfermedad se propagase a velocidad 0.001 todos los nodos serían infectados antes

de ser vacunados.

Cabe aclarar también que si bien en este ejemplo nuestra red de contactos es fija, en la implementación que usamos la misma se construye mediante un proceso de Preferential Attachment donde primero se sortean los grados de los nodos, y luego dinámicamente a partir de como se va propagando la enfermedad se van estableciendo los puertos entre los agentes.

3 Trabajos Previos

Basamos los modelos de este trabajo en [FJP21] donde encontramos el análisis de estrategias de vacunación para el problema SIRV en configuration models. En este trabajo se deriva un sistema diferencial de dimensión finita que describe la evolución de las principales variables que describen las epidemias, a saber, el número de individuos en cada compartimento (S, I, R, V) y el probabilidad de interacción entre la población susceptible con agentes en diferentes compartimentos.

En particular, p^X denotará la probabilidad de que una arista conecte un susceptible con un nodo en estado X, para $X = S$, I, R o V. Usando una generalización de la función generadora de probabilidad g, que involucra tanto el grado inicial de agentes como la probabilidad de que un nodo de grado uno permanezca susceptible, descrita por α y θ. El sistema cerrado que se obtiene es el siguiente (consultar [FJP21] para obtener las definiciones precisas y la notación):

$$\begin{cases} \dot{\alpha} = -rp^I\alpha \\ \dot{\theta} = -\pi\theta \\ \dot{I} = -\gamma I + rp^I\alpha\partial_\alpha g(\alpha,\theta) \\ \dot{V} = \pi\theta\partial_\theta g(\alpha,\theta) \\ \dot{p^S} = rp^Ip^S\left(1 - \dfrac{\alpha\partial_{\alpha\alpha}g(\alpha,\theta)}{\partial_\alpha g(\alpha,\theta)}\right) - \pi p^S - \theta\pi p^S\dfrac{\partial_{\alpha\theta}g(\alpha,\theta)}{\partial_\alpha g(\alpha,\theta)} \\ \dot{p^I} = -\gamma p^I + rp^Ip^S\dfrac{\alpha\partial_{\alpha\alpha}g(\alpha,\theta)}{\partial_\alpha g(\alpha,\theta)} - rp^I(1-p^I) \\ \dot{p^V} = rp^Ip^V + \theta\pi p^S\dfrac{\partial_{\alpha\theta}g(\alpha,\theta)}{\partial_\alpha g(\alpha,\theta)} \end{cases} \tag{1}$$

donde r, γ y π_t son las tasas de infección, recuperación y vacunación, respectivamente.

La heterogeneidad en las conexiones de les agentes, algo que es una característica importante para modelos de propagación de epidemias, está presente a través de la función g definida a partir de la distribución de grados; en particular en el término $\frac{\alpha\partial_{\alpha\alpha}g(\alpha,\theta)}{\partial_\alpha g(\alpha,\theta)}$.

Estos modelos sobre grafos heterogéneos presentan una característica de formalización en la que es posible describir la propagación de la epidemia y la estrategia de vacunación simultáneamente con la construcción de la red de contactos. Para eso es importante poder tener un registro dinámico de la estructura en un nivel superior al de les agentes. Esto lo haremos con el formalismo EB-DEVS que comentamos a continuación.

En EB-DEVS se introduce un estado a nivel macro en el formalismo que es clave para identificar, analizar y modelar la emergencia. El esquema conceptual en Figura 3.1 muestra el flujo de información entre los niveles macro y micro del sistema. Un macroestado (global) s_G aparece almacenado en el modelo acoplado, como un estado agregado que emerge de una red de microestados (individuales) en los modelos atómicos (es decir, los bloques de construcción que hacen posible la emergencia). El macroestado será producido por una nueva función global aplicada a través de microestado. Podemos considerar que en cualquier instante de la línea de tiempo en la simulación, los modelos atómicos presentan un conjunto de propiedades visibles mediante un mecanismo de causalidad ascendente. Esto debería conformar una secuencia de estados colectivos microscópicos que confluye en un estado a nivel macro a traves de la función global. Al mismo tiempo, el estado a nivel macro puede ser siempre compartido con los sub-modelos (posiblemente) influenciando las transiciones a nivel micro. Esta generalización de DEVS clásico puede encontrarse en [FHUC21] explicada con riguroso detalle y formalidad.

Por un lado, nos parece interesante del formalismo computacional la estructura dinámica, que nos permite modelar experimentos complejos como el SIRV. Por otro, la implementación del *preferential attachment* recurre a la capacidad de que haya una causalidad vertical entre las distintas estructuras del sistema, permitiendo la emergencia de propiedades en un nivel más alto: del comportamiento individual de los nodos emerge la distribución de grados de la red. Esta cualidad tiene un valor de por sí dentro de las diferentes herramientas y formalismos de modelado de sistemas complejos.

Más aún, con este modelo básico se explicita que el armado de redes dinámicas puede ser expresado con patrones simples, permitiendo implementar situaciones mucho más complejas en las que la estructura de contactos de les agentes se construya junto con la dinámica. En general, estos modelos en grafos consideran redes prefijadas, perdiendo una parte de la aleatoriedad asociada con el *Configuration Model*.

El modelo que implementamos sobre un grafo de configuraciones se sustenta en el principio de decisiones diferidas [MU05], revelando las conexiones en el grafo simultáneamente con la propagación de la enfermedad. Este truco es posible dado el ambiente aleatorio que nos provee el CM, que es construido siguiendo el mecanismo de *Preferential Attachment*.

El formalismo de modelado EB-DEVS nos permitirá tener dos tipos de modelos internos: agentes atómicos individuales y un modelo acoplado que contiene a les agentes atómicos junto con la estructura dinámica del proceso de contacto, que pensamos de manera vertical.

4 Estrategias y Distribuciones de grado

Para comparar como se comportan distintas estrategias que ponderan o no el grado de los agentes presentamos las siguientes cuatro estrategias:

1. Default: Todos los agentes se vacunan con una probabilidad exponencial con media $\lambda = \frac{1}{\psi * scale}$ donde ψ es el grado medio del grafo de conectividad, y *scale* una constante fija para escalar la

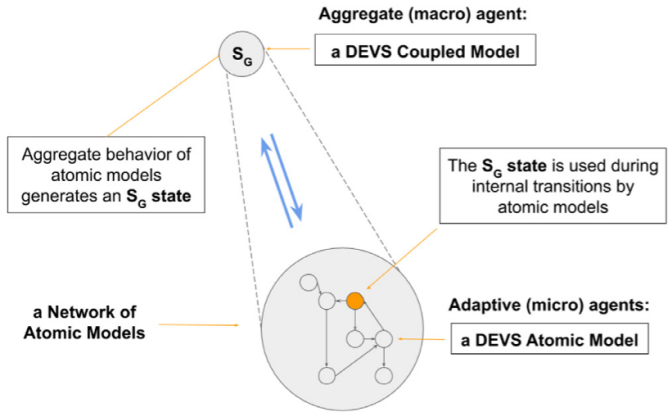

Figure 3.1: Estructura del formalismo EB-DEVS

probabilidad a los tiempos de la simulación.

2. Degree Based: Cada agente se vacuna con una probabilidad exponencial con media $\lambda = \frac{1}{deg*scale}$ donde *deg* es el grado del agente, y *scale* una constante fija para escalar la probabilidad a los tiempos de la simulación.

3. Double Degree Based: Cada agente se vacuna con una probabilidad exponencial con media $exp(\lambda)$ con $\lambda = \frac{1}{deg*scale}$ donde *deg* es el grado del agente, y *scale* una constante fija para escalar la probabilidad a los tiempos de la simulación.

4. Triple Degree Based: Cada agente se vacuna con una probabilidad exponencial con media $exp(exp(\lambda))$, con $\lambda = \frac{1}{deg*scale}$ donde *deg* es el grado del agente, y *scale* una constante fija para escalar la probabilidad a los tiempos de la simulación.

Es claro que dado el grado medio del grafo de conexiones ψ, todas las estrategias tienen la misma esperanza $\frac{1}{\psi*scale}$, una propiedad que se puede pensar como el costo promedio de un recurso finito.

Además elegimos algunas distribuciones de grado (con el mismo grado medio), utilizadas previamente en estudios numéricos de estrategias de vacunación [FJP21], para estudiar como varían los comportamientos de las estrategias según la distribución. Proponemos estudiar cuatro redes distintas de tamaño $N = 300$, asociadas a sus correspondientes funciones generadoras para la distribución de grados. Elegimos cuidadosamente los parámetros para que las cuatro tengan el mismo grado medio $\psi = 5$:

(a) Poisson: los grados se distribuyen según una variable aleatoria $\mathcal{P}(\lambda)$ con parámetro $\lambda = 5$. (Figura 4.2)

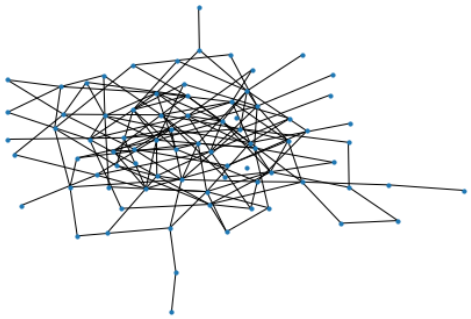

Figure 4.1: Ejemplo de grafo con distribución Bimodal.

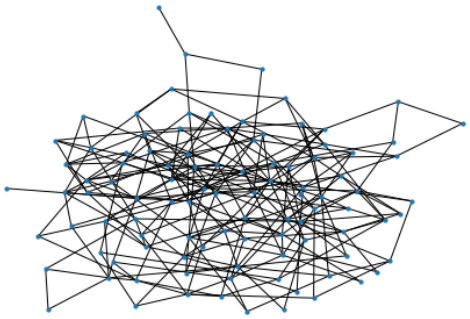

Figure 4.2: Ejemplo de grafo con distribución Poisson

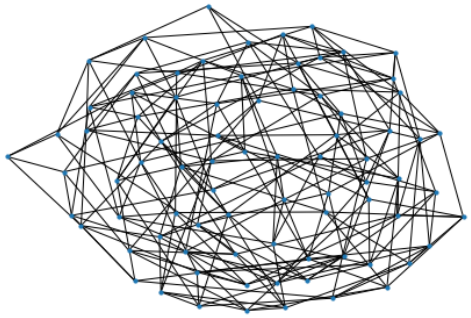

Figure 4.3: Ejemplo de grafo con distribución Regular.

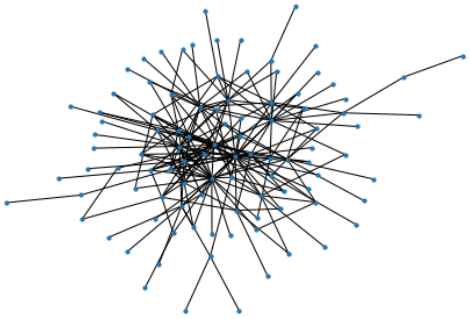

Figure 4.4: Ejemplo de grafo con distribución PowerLaw

(b) Bimodal: una proporción $p = 4/5$ de los nodos tiene grado $\mathcal{P}(3)$ y el resto sigue una distribución delta de parámetro $L = 13$. (Figura 4.1)

(c) Regular: todos los nodos tienen grado 5. (Figura 4.3)

(d) Power Law: la probabilidad de que un nodo tenga k viene dada por $p_k = \frac{k^{-\alpha}e^{-k/\kappa}}{Li_\alpha(e^{-1/\kappa})}$ para k en $\mathbb{N}$, $\alpha = 1.474$ y $\kappa = 100$, resultando $\psi(z) = \frac{Li_\alpha(z)(ze^{-1/\kappa})}{Li_\alpha(e^{-1/\kappa})}$ donde $Li_s(z)$ es el $s-$polilogaritmo de z. Consideramos un *cut-off* exponencial alrededor de $\kappa = 100$ para tener momentos finitos. (Figura 4.4)

5 Implementación

Nuestra implementación esta basada en EB-DEVS, allí implementamos las estrategias, las distribuciones, la propagación de una enfermedad y el limite global de número de vacunas.

5.1 Estrategias de vacunación y distribuciones

Separamos las estrategias de vacunación, en donde desacoplamos las estrategias propiamente dichas, del resto de la lógica del simulador. Cada estrategia es una función que recibe un agente, un entorno, y un conjunto de propiedades (parámetros iniciales de la simulación) y debe retornar el *ta* que se asignara para la vacunación del agente susceptible.

```
# Vaccination Strategies

def default_strategy_fun(agent,env,props):
    if agent.deg == 0:
        return INFINITY
    else:
        return np.random.exponential(float(1)/(float(props.PHI)*env.vaccine_scale))

default_strategy = {"name": "default_strategy", "fun": default_strategy_fun}


def double_default_strategy_fun(agent,env,props):
    if agent.deg == 0:
        return INFINITY
    else:
        return np.random.exponential(np.random.exponential(float(1)/(float(props.PHI)*env.vaccine_scale)))

double_default_strategy = {"name": "double_default_strategy", "fun": double_default_strategy_fun}


def triple_default_strategy_fun(agent,env,props):
    if agent.deg == 0:
        return INFINITY
    else:
        return np.random.exponential(np.random.exponential(np.random.exponential(float(1)/(float(props.PHI)*env.vaccine_scale))))

triple_default_strategy = {"name": "triple_default_strategy", "fun": triple_default_strategy_fun}

def degree_based_strategy_fun(agent,env,props):
    if agent.deg == 0:
        return INFINITY
    else:
        return np.random.exponential(float(1)/((float(agent.deg)*env.vaccine_scale)))

degree_based_strategy = {"name": "degree_based_strategy", "fun": degree_based_strategy_fun}


def double_degree_based_strategy_fun(agent,env,props):
    if agent.deg == 0:
        return INFINITY
    else:
        return np.random.exponential(np.random.exponential(float(1)/((float(agent.deg)*env.vaccine_scale))))

double_degree_based_strategy = {"name": "double_degree_based_strategy", "fun": double_degree_based_strategy_fun}


def triple_degree_based_strategy_fun(agent,env,props):
```

```
    if agent.deg == 0:
        return INFINITY
    else:
        return np.random.exponential(np.random.exponential(np.random.exponential(float(1)/((float(agent.deg)*env.vaccine_scale))

triple_degree_based_strategy = {"name": "triple_degree_based_strategy", "fun": triple_degree_based_strategy_fun}

# Degree distributions

def poisson_5_gen(agent):
    return np.random.poisson(5)


poisson_5 = {"generator":poisson_5_gen, "mean":5.0, "name": "poisson_5"}

def bimodal_poisson_3_L13_gen(agent):
    if np.random.random() <= 0.8:
        return np.random.poisson(3)
    else:
        return 13

bimodal = {"generator":bimodal_poisson_3_L13_gen, "mean": 5.0 ,"name": "bimodal"}

xk1=np.arange(0,30,1)
pk1=np.zeros(30)
for i in range(30):
  pk1[i]=5**i*np.exp(-5)/math.factorial(i)

custm1 = stats.rv_discrete(name='custm', values=(xk1, pk1))

def power_law_gen(agent):
    return custm1.rvs(size=1)[0]


power_law = {"generator":power_law_gen, "mean": 5.0 ,"name": "power_law"}

regular_5 = {"generator":lambda x: 5, "mean": 5.0 ,"name": "regular_5"}


default_vaccine_props = {"name":"s=0.05_limit=inf", "vaccine_scale": 0.05, "vaccine_limit": INFINITY }
```

5.2 Modelado en EB-DEVS con soportes de estrategias de vacunación y limite global de vacunas

La implementación completa del modelo se puede ver en `https://github.com/ivanpostolski/sirv_final`, aquí mostraremos algunas funciones clave del uso de la estrategia de vacunación y del uso de EB-DEVS. Para simplificar la explicación omitimos parte del código que se relaciona con la propagación de la infección.

Agente (Atómico EB-DEVS)

```
SIRStates = enum(S='Susceptible', I='Infected',  R='Recovered',V='Vaccinated')
ENVProps = enum(DECAY='decay_rate', AGENT_STATES='log data'\
```

```
3       , QUARANTINE_CONDITION='If above threshold, agents do quarantine.',\
4            AVAILABLE_VACCINES='If there is available vaccines')
5
6
7   class AgentState(object):
8       def __init__(self, model, name, id, state, env):
9           """TODO: to be defined1. """
10          self._name = name
11          self.current_time = 0.0
12          # ...
13          self.free_deg = env.degree_gen(self) if id!=0 else 0
14          self.deg = int(self.free_deg)
15          # ...
16
17      def timeAdvance(self):
18          #...
19          if self.state.state == SIRStates.S and (self.state.ta == 0 or self.state.ta == INFINITY):
20              # vaccination strategy
21              self.state.ta = self.parent.vaccine_strategy(self.state,self.parent,Parameters)
22          return self.state.ta
23
24
25      def intTransition(self):
26          self.state.current_time += self.timeAdvance()
27          # ...
28          if self.state.state == SIRStates.S and self.state.ta > 0:
29              if (self.parent.getContextInformation(ENVProps.AVAILABLE_VACCINES)):
30                  self.state.state = SIRStates.V
31                  self.state.share = True
32              else:
33                  self.state.ta = INFINITY
34          #...
35          self.y_up = self.state
36
37          return self.state
38
39      def modelTransition(self, state):
40          # ...
41          if self.state.state == SIRStates.V:
42              state["V"] = self.state
43              return True
44          return False
```

En el agente podemos ver en principio la asignación de un grado y grado-libre (linea 13 y 14) aleatoriamente respecto de la función de distribución generadora del grado, seteada en el entorno de EB-DEVS `env`. Luego en las lineas 19 a 21 podemos ver como en la función de time advance introducimos el llamado a la estrategia de vacunación asignada al `parent`, que en nuestro caso es el Environment. Allí llamaremos a las diferentes estrategias diseñadas para setear el tiempo elegido para vacunar este agente. En las lineas 28 a 31 podemos observar el comportamiento introducido a un agente que acciona la vacunación en su transición interna (una vez que suena su ta), y allí se ejecutan dos comportamientos relevantes, el primero es la obtención de información sobre las

vacunas disponibles (propiedad AVAILABLE_VACCINES) al entorno. Si este entorno efectivamente aún tiene vacunas disponibles podemos vacunar a este individuo y accionar el mecanismo de transición de modelo de EB-DEVS. mediante la asignación de la propiedad `share` en True, y el atributo `y_up`.

Finalmente, en la transición de modelo, asignamos la clave "V", y retornamos True, para continuar con la transición de modelo del Environment que describiremos a continuación.

Environment EB-DEVS

```
class Environment(CoupledDEVS):
    def __init__(self, degree_gen, vaccine_props, vaccine_strategy,name=None):
        # ...
        self.nodes_free_deg = {}
        self.degree_gen = degree_gen["generator"]
        Parameters.PHI = degree_gen["mean"]
        # ...
        self.vaccine_scale = vaccine_props["vaccine_scale"]
        self.vaccines_available = vaccine_props["vaccine_limit"]
        self.vaccine_strategy = vaccine_strategy["fun"]
        #...

    def getContextInformation(self, property, *args, **kwargs):
        super(Environment, self).getContextInformation(property)
        if(property == ENVProps.AGENT_STATES):
            return self.agent_states

        if(property == ENVProps.AVAILABLE_VACCINES):
            return self.vaccines_available > 0

    def modelTransition(self, state):
        # Available vaccination decrease
        if "V" in state:
            self.vaccines_available = self.vaccines_available - 1
            del state["V"]
            return False

        # Preferential Attachment Process
        newly_inf=state["newly_infected"]
        current_time=newly_inf.current_time
        newly_inf_id=newly_inf.id
        newly_inf_deg = newly_inf.free_deg
        self.nodes_free_deg[newly_inf.id] = 0
        grados = list(self.nodes_free_deg.values())

        xk = np.array(grados)
        xk[newly_inf_id]=0

        if sum(xk) == 0:
            return False

        pk = xk / float(sum(xk))

        deg=max(0,newly_inf_deg+Parameters.K*np.random.binomial(1,Parameters.p))

        if deg > sum(pk>0):
```

```
47              return False
48
49          selected_agents = np.random.choice(max([0],list(self.nodes_free_deg.keys())), int(deg), p=pk,replace=False)
50
51
52          neighbor_states = {}
53
54          # Connect ports from/to that node
55          for i in selected_agents:
56              i0, o0 = self.agents[newly_inf_id].add_connections(i)
57              i1, o1 = self.agents[i].add_connections(newly_inf_id)
58
59              self.connectPorts(o0, i1)
60              self.connectPorts(o1, i0)
61
62              # Update the nodes free degrees list
63              self.nodes_free_deg[i] = self.nodes_free_deg[i]-1
64              self.G.add_edge(int(newly_inf_id), int(i), timestamp=current_time)
65              neighbor_states[i] = self.agents[i].state.state
66              self.agents[i].state.neighbors_state[newly_inf_id] = self.agents[newly_inf_id].state.state
67
68          self.agents[newly_inf_id].state.neighbors_state = neighbor_states
69          self.agents[newly_inf_id].state.free_deg = 0
70          return False
```

En el código del Environment vemos en principio la asignación de la distribución de grado y el parámetro PHI en las lineas 5 y 6, y luego la asignación de las propiedades de escala, limite y estrategia elegidos para esta simulación (linea 8 a 10). En la función `getContextInformation` podemos ver la consulta de vacunas disponibles (linea 18). Y para terminar `modelTransition` es el encargado de unir las transiciones de los Agentes con las del Entorno (EB-DEVS), allí es donde se actualizan la cantidad de vacunas disponibles y donde se realiza el procedimiento de preferential attachment para armar el configuration model dinámicamente.

Es importante aclarar que a diferencia de los valores que son fijados en el comienzo de la simulación tanto en Agentes como en el Entorno (e.g, distribución de grado de nodos, estrategia de vacunación, parámetros de vacunación) utilizamos la estructura de transiciones de EB-DEVS para la mecánica de límite de vacunación finita, dado que los Agentes no acceden al estado del Entorno, sino que su comunicación se realiza plenamente con las transiciones definidas en el modelo EB-DEVS. Permitiendo por ejemplo cambios futuros en políticas de limites de vacunas en el Entorno independientemente de la lógica que se encuentra en los Agentes.

6 Evaluación con Simulaciones

Recapitulando las preguntas que nos hicimos en la introducción:

RQ1: Cómo se comparan las estrategias de vacunación ponderadas por los grados de los agentes con estrategias cuya probabilidad no depende del grado?

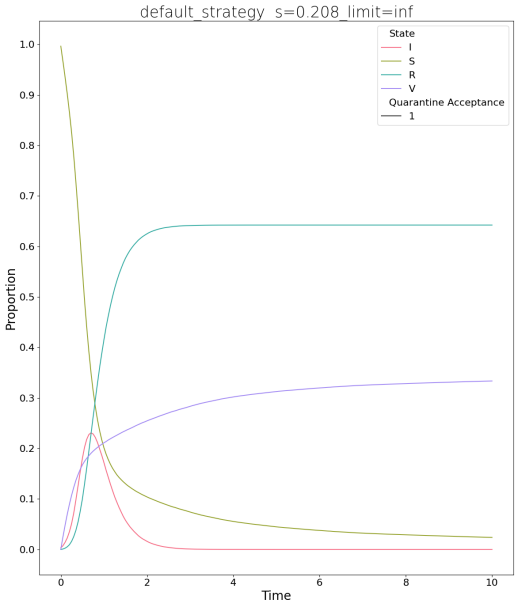

Figure 7.1: Poisson 5, Estrategia de Vacunación Default, Factor de escala 0.208

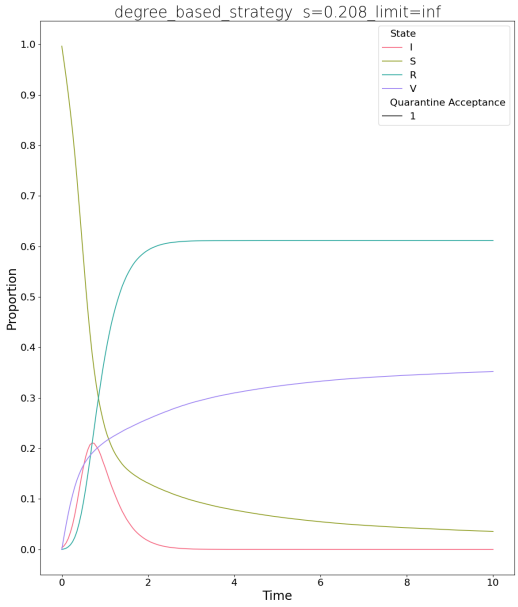

Figure 7.2: Poisson 5, Estrategia de Vacunación Degree, Factor de escala 0.208

RQ2: Cómo se comparan las estrategias de vacunación ponderadas por los grados de los agentes con estrategias cuya probabilidad no depende del grado cuando hay un limite finito de vacunas disponibles?

Para responder estas preguntas realizamos simulaciones utilizando nuestra implementación de las estrategias y distribuciones presentadas en las secciones previas en EB-DEVS. Básicamente, instanciamos cuatro parámetros: estrategia, distribución, factor de escala y límite finito de vacunas, y promediamos 20 re-ejecuciones para cada instanciación. Para responder **RQ1** no instanciamos el límite de vacunas, y para **RQ2** instanciamos el limite de vacunas en 10%, 25% y 50%. Para todos nuestros experimentos seteamos 100 factores de escala linealmente distribuidos entre 0.001 y 0.5.

7 Resultados RQ1

7.1 Poisson

En la simulación de red de grado 5 Poisson se ven diferencias sutiles entre Default y Degree (Figuras 7.3 y 7.4), que se acentúan en factores de escala a partir de 0.389 (Figuras 7.5 y 7.6), mientras las estrategias Double y Triple degree based logran contener la epidemia mas rápidamente que las estrategias Default y Degree. Se ven algunas diferencias entre Double y Degree en factores altos, posiblemente por el balance que ofrece Double entre la vacunación de agentes temprana (grado más alto) pero la acentuada postergación de agentes de grado bajo.

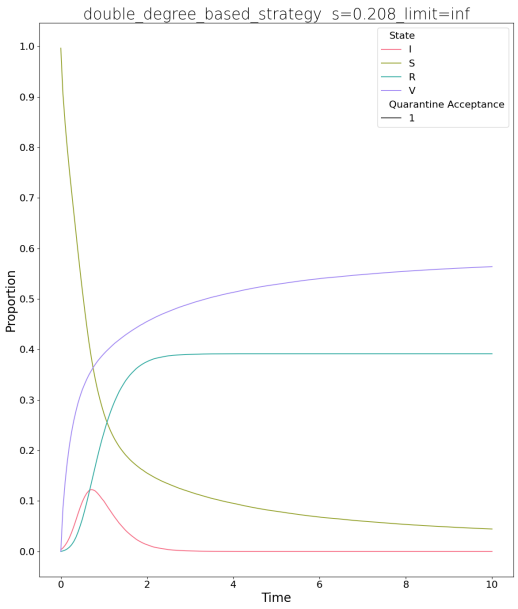

Figure 7.3: Poisson 5, Estrategia de Vacunación Double, Factor de escala 0.208

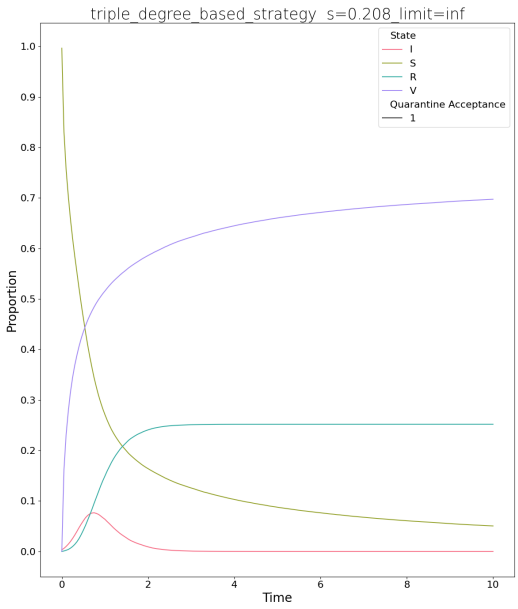

Figure 7.4: Poisson 5, Estrategia de Vacunación Triple, Factor de escala 0.208

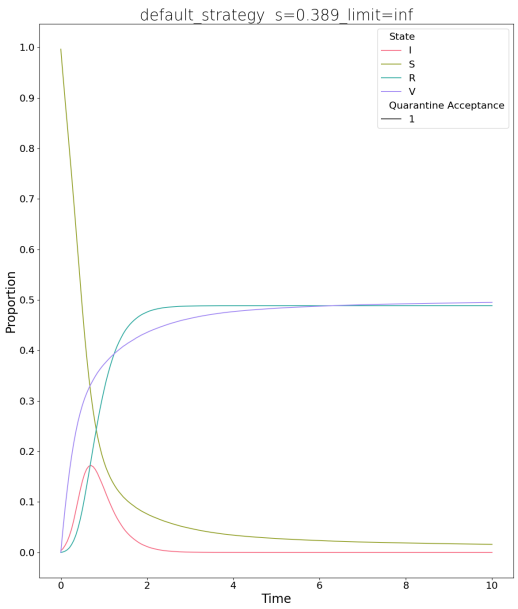

Figure 7.5: Poisson 5, Estrategia de Vacunación Default, Factor de escala 0.389

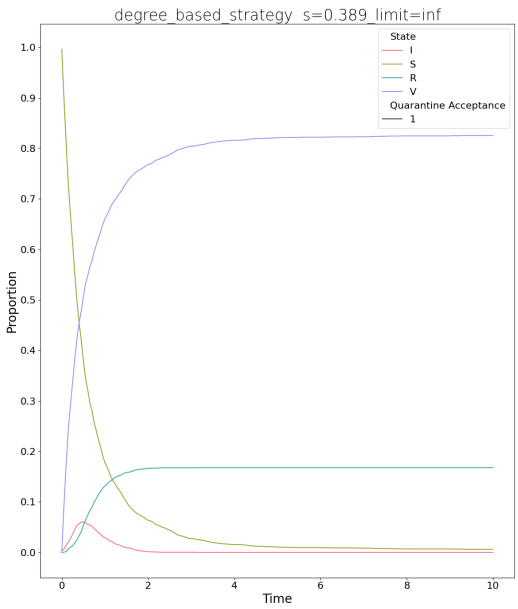

Figure 7.6: Poisson 5, Estrategia de Vacunación Degree, Factor de escala 0.389

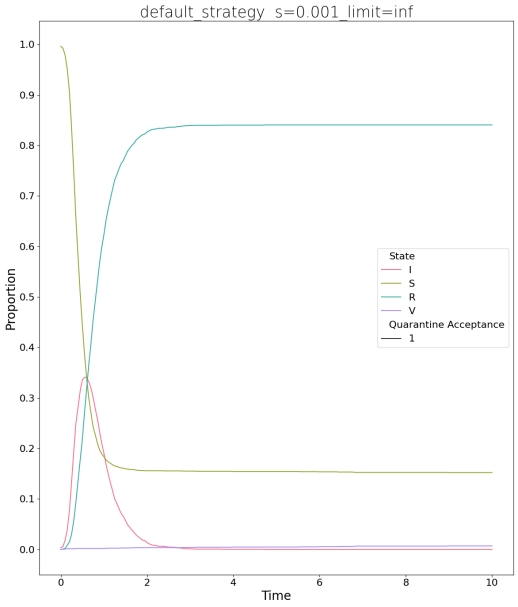

Figure 7.7: Bimodal, Estrategia de Vacunación Default, Factor de escala 0.001

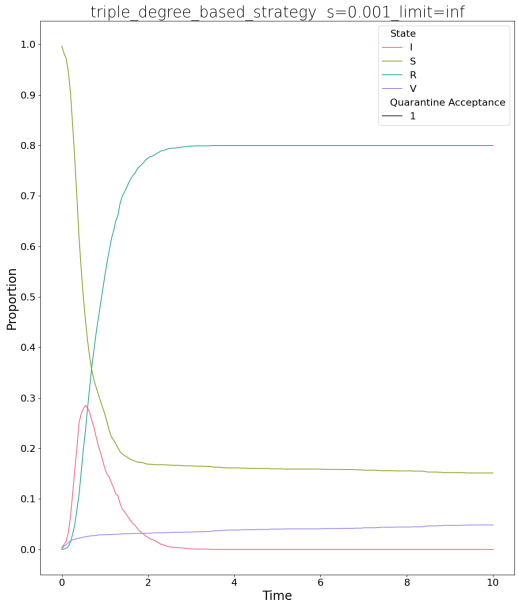

Figure 7.8: Bimodal, Estrategia de Vacunación Triple, Factor de escala 0.001

7.2 Bimodal

Para valores chicos de escala como 0.001, vemos que estrategias como la Triple logran achicar el pico de infectados (vacunando rápidamente el 5% de la población) pero no reducen en valor absoluto la cantidad de infectados de la enfermedad (Figuras 7.7 y 7.8). Y un comportamiento similar a la distribución de Poisson 5 en factores de escala mayores.

7.3 Regular

Observamos que la estrategia Default y Degree tienen el mismo comportamiento invariable al factor de escala (Figuras 7.9 y 7.10), esperable dado que ambos tienen la misma distribución (grado constante y grado medio son iguales). Además vemos un comportamiento similar en Double y Triple al resto de las distribuciones (Figuras 7.11 y 7.12), esto es interesante porque al ser una distribución regular, depender del grado no debería dar una ventaja comparativa y las diferencias deberían atribuirse a vacunar más rápido algunos agentes que las otras estrategias más allá de depender o no del grado.

7.4 PowerLaw

Vemos comportamiento similar en las diferencias entre Default y Degree. A su vez, las estrategias Double y Triple mantienen un comportamiento similar al de las otras distribuciones.

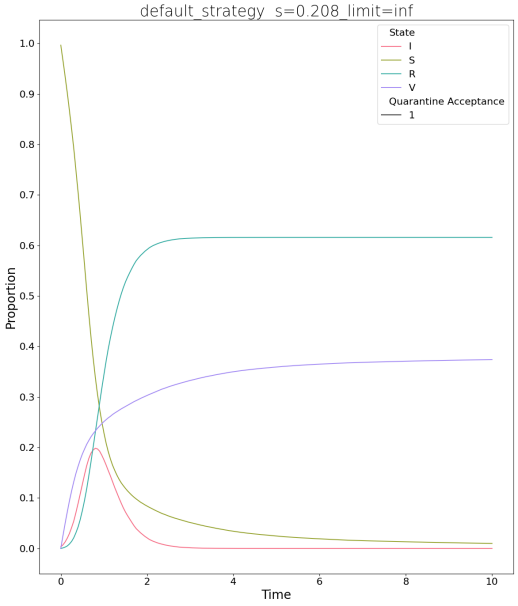

Figure 7.9: Regular 5, Estrategia de Vacunación Default, Factor de escala 0.208

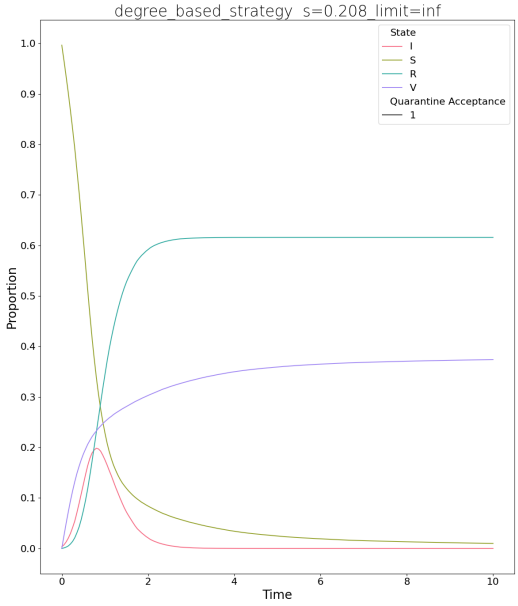

Figure 7.10: Regular 5, Estrategia de Vacunación Degree, Factor de escala 0.208

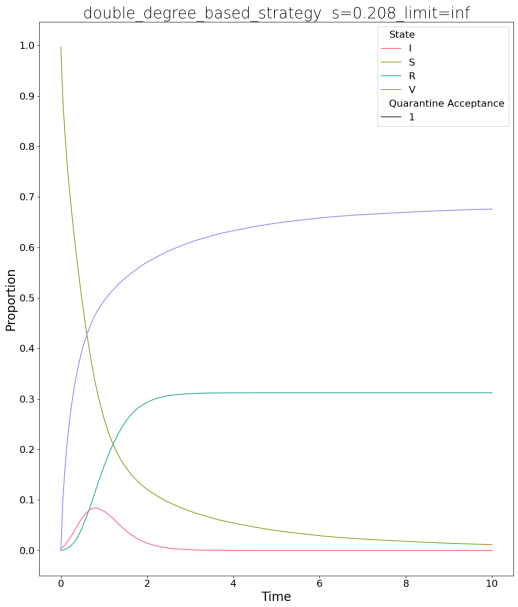

Figure 7.11: Regular 5, Estrategia de Vacunación Double, Factor de escala 0.208

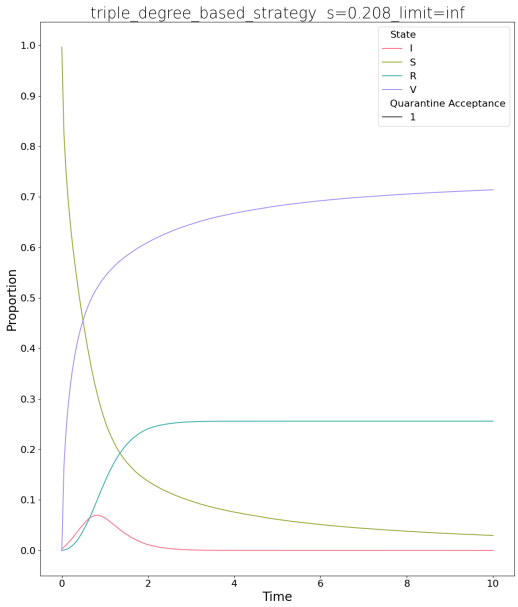

Figure 7.12: Regular 5, Estrategia de Vacunación Triple, Factor de escala 0.208

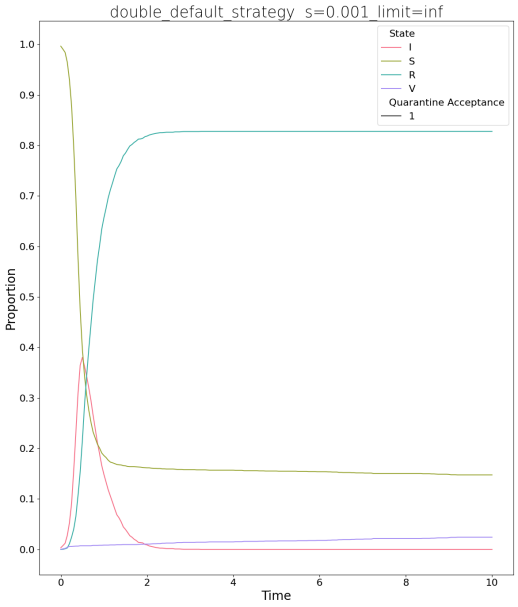

Figure 7.13: Bimodal, Estrategia de Vacunación Double **Default**, Factor de escala 0.208

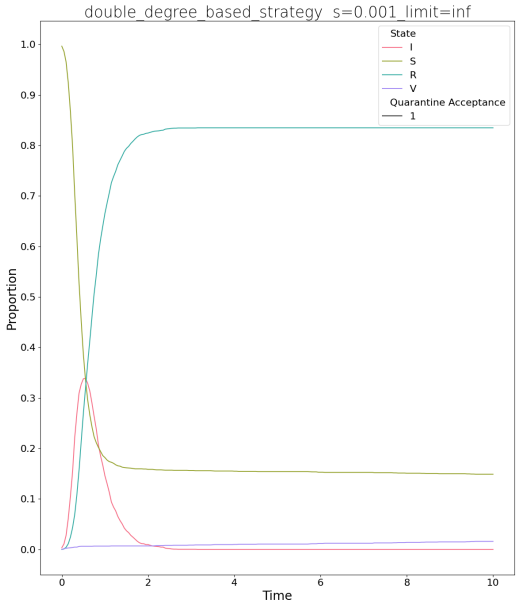

Figure 7.14: Bimodal, Estrategia de Vacunación Double, Factor de escala 0.208

7.5 Comparación con Double y Triple Default

Al observar el comportamiento de las estrategias Double y Triple en la distribución regular, nos preguntamos que rol tienen efectivamente los grados en la performance de estas estrategias. Por lo que realizamos una simulación extra, en la que comparamos hacer Double en Default y Triple en Default (una exponencial que no depende del grado de cada agente sino que del grado medio de la red), en una distribución de grado Bimodal.

Vemos que para factores bajos $s = 0.001$ el pico de infectados es menor cuando se vacuna proporcional al grado de los agentes (Figuras 7.17 y 7.18), sin embargo el numero de infectados totales se mantiene igual. A medida que el factor de escala aumenta se ven comportamientos similares (al parecer independientes del grado), y a partir de factores 0.283 se ve un salto de fase que logra una diferencia a favor de las estrategias basadas en el grado de los agentes. Sin embargo, las diferencias son más parecidas a las obtenidas entre Default y Degree, es decir que el impacto de elegir el grado de cada agente para determinar su estrategia de vacunación no parecería ser el factor determinante en las comparaciones entre Default, Degree y Double, Triple.

Resumen RQ1: Observamos en general que las estrategias que dependen en el grado (i.e, Degree,Double y Triple) realizan un mejor trabajo en reducir la propagación de una enfermedad. Sin embargo, las mejoras obtenidas por las estrategias Double y Triple (que fueron las más acentuadas) no parecen ser debidas a su dependencia en el grado de los agentes vacunados, sino a la varianza entre los tiempos de vacunación (más sujetos se vacunan rápidamente, y más sujetos se vacunan mucho

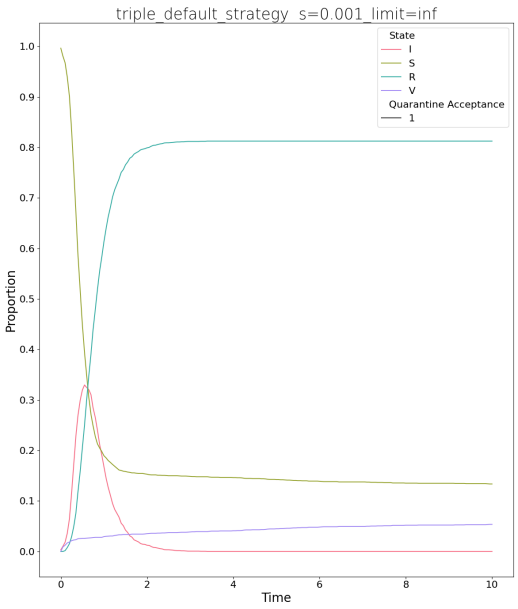

Figure 7.15: Bimodal, Estrategia de Vacunación Triple **Default**, Factor de escala 0.208

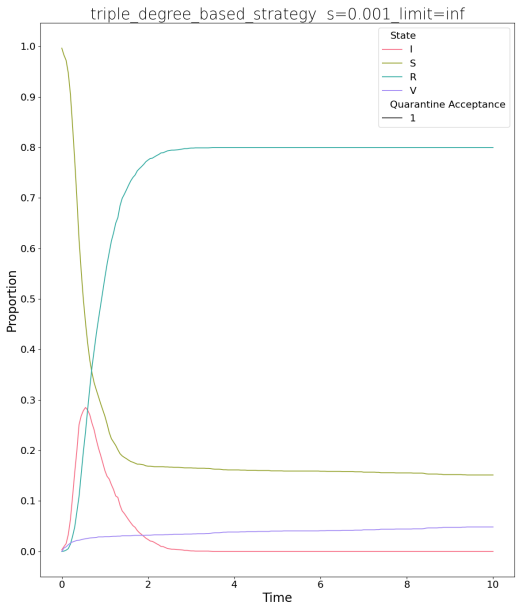

Figure 7.16: Bimodal, Estrategia de Vacunación Triple, Factor de escala 0.208

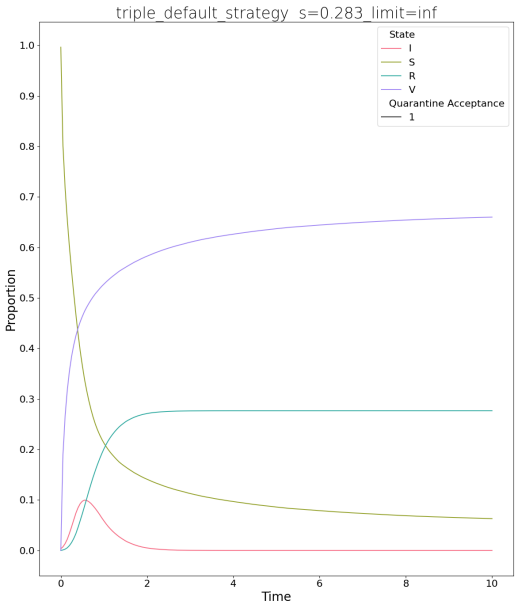

Figure 7.17: Bimodal, Estrategia de Vacunación Triple **Default**, Factor de escala 0.283

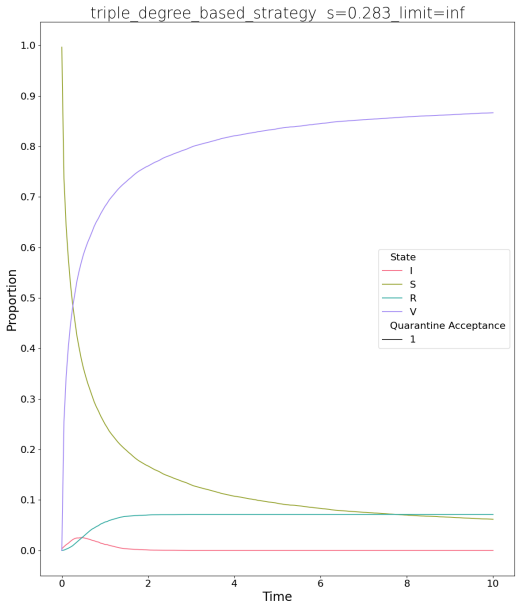

Figure 7.18: Bimodal, Estrategia de Vacunación Triple, Factor de escala 0.283

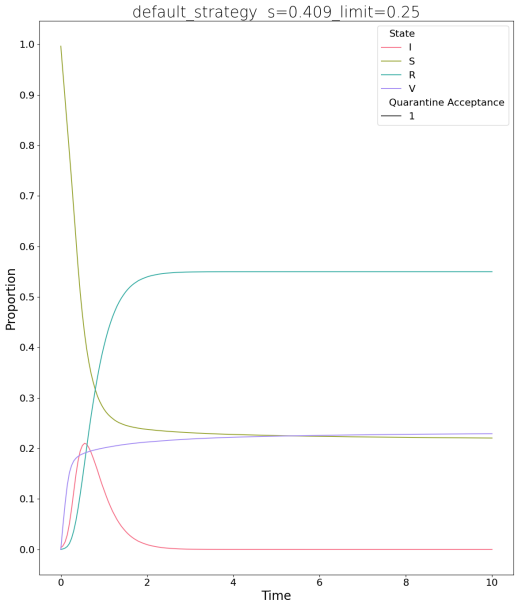

Figure 8.1: Bimodal, Estrategia de Vacunación Default, Factor de escala 0.409,Limite de vacunacion 25%

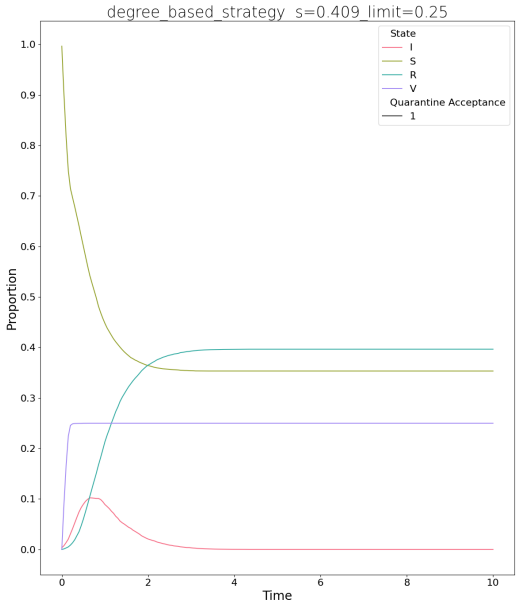

Figure 8.2: Bimodal, Estrategia de Vacunación Degree, Factor de escala 0.409, Limite de vacunacion 25%

más tarde).

8 Resultados RQ2

8.1 Limite finito de Vacunación

En general observamos que imponiendo un limite de vacunación bajo se pueden observar comportamientos similares a partir de ciertos factores de escala entre Degree, Triple y Default. Sugiriendo que la mayor parte de las diferencias entre las estrategias se deben a la velocidad de vacunación inicial y no tanto a la ponderación de grado por agente. Además, vemos que en límites entre (25% y 50%) se acentúan las diferencias entre Default y Degree. Mostraremos en esa sección algunas de estos comportamientos por cada distribución de grado.

8.1.1 Poisson

Al fijar un limite de 10% de vacunas disponibles, observamos que tanto Degree, Double y Triple presentan un comportamiento similar mientras que Default tiene un pequeño incremento en el pico de infectados y mayor cantidad total de infectados (Recuperados) al terminar la epidemia, la simulación se estabiliza en valores de escala alrededor de $s = 0.3$.

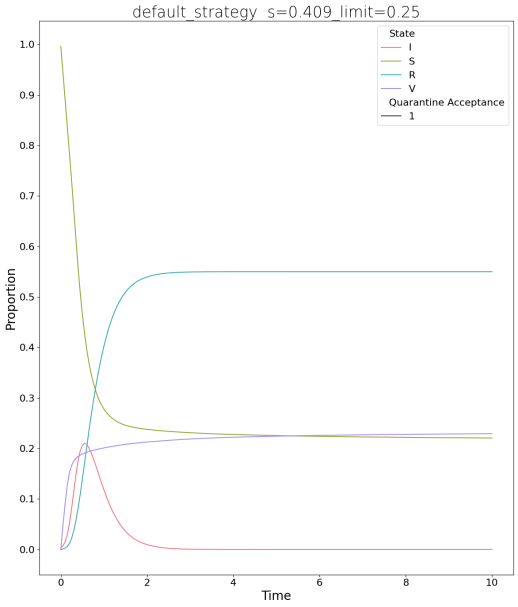

Figure 8.3: Bimodal, Estrategia de Vacunación Default, Factor de escala 0.409,Limite de vacunacion 25%

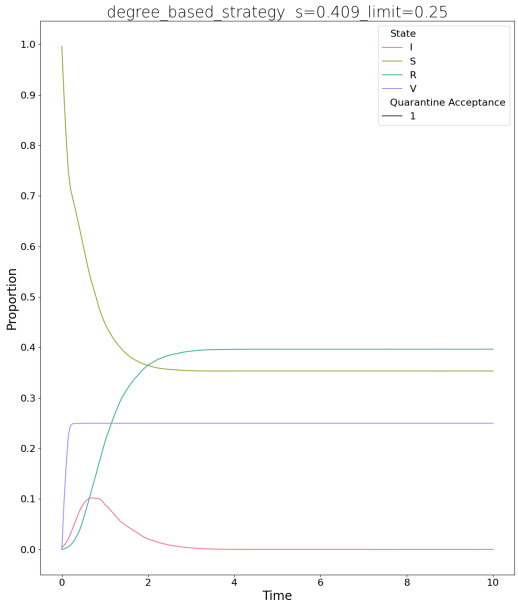

Figure 8.4: Bimodal, Estrategia de Vacunación Degree, Factor de escala 0.409, Limite de vacunacion 25%

8.1.2 Bimodal

Al fijar un limite de 25% de vacunación vemos que Degree se comporta parecido a Double y Triple, pero mucho mejor que Default, con aproximadamente 10% menos de Recuperados y menos de la mitad de pico de infectados durante la epidemia (Figuras 8.3 y 8.4). En otras palabras, vemos que la velocidad de Double y Triple para vacunar logran utilizar todas las vacunas disponibles (25%) mientras que Default y Degree no logran aprovechar todo el stock de vacunas, indicando que la velocidad para vacunar puede ser un factor más relevante que el stock disponible.

8.1.3 Regular

Fijando un límite bajo de vacunación en la distribucion Regular, observamos que en factores de escala grandes, cercanos a 0.5, todas las estrategias se comportan igual, ofreciendo un chequeo de sanidad para el desarrollo propuesto, ya que Degree, Double y Triple no deberían comportarse de forma distinta a Default cuando la red tiene grado regular, ya que no hay ventaja en la información que utilizan, excepto la velocidad que pueden alcanzar debido a una mayor varianza en Double y Triple.

8.1.4 PowerLaw

Similar a lo reportado en la distribución Bimodal, vemos que en valores altos de factor de escala Degree tiene mejor performance que Default, y parecida a Double y Triple. Indicando una mejoría

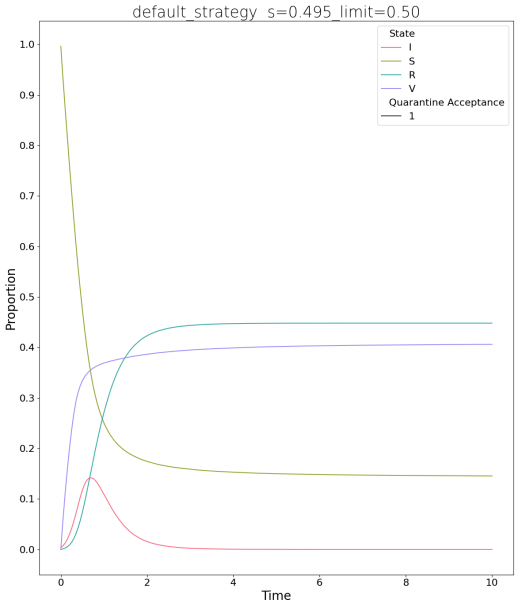

Figure 8.5: PowerLaw, Estrategia de Vacunación Default, Factor de escala 0.409,Limite de vacunacion 50%

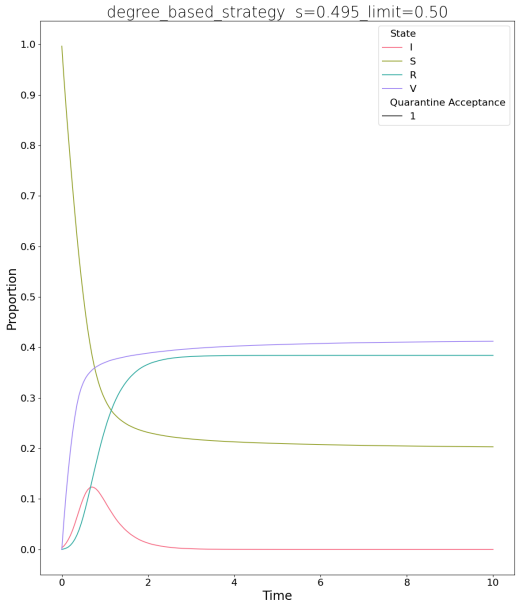

Figure 8.6: PowerLaw, Estrategia de Vacunación Degree, Factor de escala 0.409, Limite de vacunacion 50%

consistente en las distribuciones que otorgan mayores grados con una velocidad de vacunación alta. (Figuras 8.5 y 8.6).

Resumen RQ2: Vemos que al fijar un límite en la cantidad de vacunas, las diferencias entre Double y Triple se reducen en factores de escala mayor, apoyando la hipótesis de que las diferencias entre estrategias Degree, Double y Triple está en la velocidad en la que Double y Triple vacunan al inicio de la propagación. Más aún, las estrategias Default y Degree muestran diferencias mayores cuando la velocidad de vacunación inicial se acelera, indicando que en efecto una estrategia de vacunación dependiendo del grado es más efectiva para elegir un numero finito de vacunados iniciales.

9 Conclusión

En este trabajo final hemos evaluado por medio de simulaciones diferentes estrategias de vacunación en poblaciones que siguen distintas distribuciones de grado, vemos que las diferencias entre una estrategia de vacunación que depende del grado de un agente o una que no, depende en buena parte de la velocidad y el momento en el que cada estrategia puede ser aplicada, ya que existen casos donde las diferencias son grandes o pequeñas dependiendo del límite fijo de stock que limita el tiempo de vacunación, o también diferencias asociadas a la varianza en tiempos de vacunación.

10 Posibles Líneas Futuras

Nuestro trabajo es fácilmente extensible respecto de agregar nuevas estrategias de vacunación prometedoras, y explorar rápidamente su desempeño bajo diferentes condiciones, que a su vez pueden ser extensibles como *i*) la distribución de la red y *ii*) los parámetros del entorno.

Otra línea posible de extensión vinculada al uso de EB-DEVS puede ser funciones de costo o restricciónes del entorno, por ejemplo un límite de vacunas por intervalo tiempo, o incluso con una dinámica cambiante en la provisión de las mismas (e.g, agentes que producen vacunas) y las proveen al entorno. Por otro lado, una extensión más compleja podría ser la incorporación de particiónes de la red (e.g regiones) donde las mismas a su vez podrían tener distintos parámetros de cantidad de vacunas, etc.

11 Material Adicional

El código que desarrollamos para realizar la experimentación de este trabajo esta disponible públicamente en el repositorio `https://github.com/ivanpostolski/sirv_final`. Debido a que en total fueron más de 7 mil simulaciones es imposible almacenar todos los resultados (datos y gráficos), sin embargo realizamos algunos vídeos mostrando los comportamientos relevantes que mencionamos en el trabajo.

References

[CZ88] Arturo I. Concepcion and Bernard P. Zeigler. Devs formalism: A framework for hierarchical model development. *IEEE Transactions on Software Engineering*, 14(2):228–241, 1988.

[FHUC21] Daniel Foguelman, Philipp Henning, Adelinde Uhrmacher, and Rodrigo Castro. Eb-devs: A formal framework for modeling and simulation of emergent behavior in dynamic complex systems. *Journal of Computational Science*, 53:101387, 2021.

[FJP21] Emanuel Javier Ferreyra, Matthieu Jonckheere, and Juan Pablo Pinasco. Sir dynamics with vaccination in a large configuration model. *Applied Mathematics & Optimization*, pages 1–50, 2021.

[FLNU18] Bailey K Fosdick, Daniel B Larremore, Joel Nishimura, and Johan Ugander. Configuring random graph models with fixed degree sequences. *Siam Review*, 60(2):315–355, 2018.

[JLW14] Svante Janson, Malwina Luczak, and Peter Windridge. Law of large numbers for the sir epidemic on a random graph with given degrees. *Random Structures & Algorithms*, 45(4):726–763, 2014.

[MU05] M. Mitzenmacher and E. Upfal. Probability and computing: Randomized algorithms and probabilistic analysis. 2005.

[MU17] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[Vol08] Erik Volz. Sir dynamics in random networks with heterogeneous connectivity. *Journal of mathematical biology*, 56(3):293–310, 2008.

[Wei13] Howard Howie Weiss. The sir model and the foundations of public health. *Materials matematics*, pages 0001–17, 2013.