Optimization of limited budget allocation for Cybersecurity Operations using Complex Networks

Master in Computational Engineering and Mathematics
Master's Thesis

Nicolas Leconte-Demarsy
Advisor: Dr. Sergio Gómez

May 30, 2024

Abstract

A typical Cybersecurity Operations department has to sometimes make *difficult choices* with limited budgets. Such choices and decisions are often based on concepts such as "systems exposure", "impact of compromise" and other risk-related ideas. We propose to use epidemiology-inspired simulations of infectious processes on graph/networks to provide an alternative, additional, decision support system.

We first implement a SIS simulator to run on graphs, which we re-configure to allow for the distribution of an improved "protection" and/or improved "detection and recovery" measure(s) per node. We then supplement it with a Genetic Algorithm, coupled with Monte-Carlo (given the stochastic nature of our SIS simulation outputs), to try and optimize the use of such infection control measures, on any given network, by minimizing the resulting infection prevalence. Importantly, said infection control measures are scarce, depending on limited budget. We construct such a simulator and optimization process mostly from the ground up, as a personal challenge — exception made of some key and useful supporting programming libraries for graphs manipulations and visualizations.

After generating several simulation outputs, for varying inputs, in this report we analyze and demonstrate how our suggested approach of computer-informed recommendations can indeed generally be valuable. Our program provides improved constrained budget allocation proposals, compared to a status-quo that would fail to consider interactions between nodes on a given network (or more basically its structure). The number of simulations needed to obtain valuable results was a challenge that forced several improvements of the implemented program, including a highly distributed configuration to allow for linear horizontal scalability. An alternative to Monte-Carlo was also implemented with relevant gains in processing speeds, while maintaining valuable results. The outcome (recommendations) generated by our simulations and optimization processes in a real-world setting would have to be used as a supplementary decision support tool, never fully replacing pre-existing context information such as the value or exposure of assets.

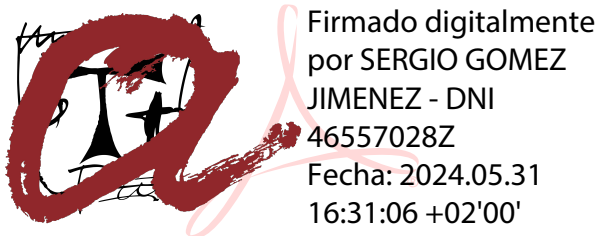
A few theoretical results for "epidemic control" have been re-discovered through our Project, thereby further validating our implementation.

Keywords: Optimization, Meta-heuristics, Genetic Algorithms, Monte-Carlo simulation, Epidemiology, Susceptible-Infected-Susceptible infectious process, Network/Graph.

Contents

Dr. Sergio Gómez, certifies that the student Nicolas Leconte Demarsy has elaborated the work under his/her direction and he/she authorizes the presentation of this Master's Thesis for its evaluation.

Advisor/Director's signature:



Final Project Sheet

Title:	Optimization of limited budget allocation for Cybersecurity Operations using Complex Networks
Author:	Nicolas Jean Leconte Demarsy
Advisor/Director:	Dr. Sergio Gómez
Date:	May 2024
Program:	Master in Computational Engineering and Mathematics
Area:	Complex Networks
Language:	English
Keywords (top 3):	Simulation, Genetic Algorithms, HPC

Dedicatory and Quote

To my wife: Thank you for being so understanding, particularly these past 3 years.

To my mother: I wouldn't be who I am without you.

And to my (true) father: My interest in Science I owe to you.

"All models are wrong but some models are useful."

George E. P. Box, "Statistical Control: By Monitoring and Feedback Adjustment".

Acknowledgement

Many colleagues at my current company have been nice enough to withstand rather long explanations of the concepts of this Project. Just knowing it all made some sort of sense to them was a comfort and helped me progress.

Some in particular not only listened and understood, but even provided some valuable, constructive feedback: I couldn't possibly name them all, and if on top of that, you feel I have ignored some of your comments, I apologize; I had to limit the scope of the Project.

My thesis advisor, Sergio, has dedicated time but also given key insights that I wouldn't have thought of, from a more academic perspective, which not only have enriched the resulting outcome, but also improved substantially the output of the developed code.

The UOC/URV Master Degree Directors have been attentive for the past 3 years to my doubts and helped guide my organization towards completing the Master degree, purposefully spread across different years. In 3 separate years, them being certainly otherwise occupied, I have always been amazed by their kind and timely responses.

Acronyms, Abbreviations and Definitions

Acronym / Abbreviation	Definition
BA	Barabási-Albert (Graph)
CYS	Cybersecurity
CYS Ops	Cybersecurity Operations (i.e., Business Unit)
EDR	Endpoint Detection and Response (usually a Cybersecurity Software component run on Computers)
ER	Erdős-Rényi (Graph)
GA	Genetic Algorithm
IT	Information Technology
MC	Monte-Carlo (in context of Simulation here)
MMCA	Microscopic Markov-Chain Approach (see (Gómez 2010))
MSc.	Master of Science
NIST	National Institute of Standards and Technology (USA)
SIRS	"Susceptible, Infected, Recovered, Susceptible" (Epidemiology Simulation)
SIS	"Susceptible, Infected, Susceptible" (Epidemiology Simulation)
SW	Software
WS	Watts-Strogatz (Graph - Small World Model)

List of Figures

1. Introduction

1.1. Context and Project Justification

Depending on sources, the Cybersecurity Market is considered to be anywhere in the order of 50 up to 200 hundreds billion US dollars today. Cybersecurity Operations ("CYS Ops" from now on) units in large (and not-so-large) companies need to choose how to best dedicate/spread limited resources (sometimes very limited) across several potential "efforts" (Venables 2022), all meant to increase the cybersecurity "posture" of their company's IT assets, to protect themselves against an increasingly challenging environment.

The question motivating this Project was:

"How to *best* choose to dedicate *scarce* resources to maximize the IT Security of a company, provided one is offered several competing options to choose from?"

Traditional Cybersecurity management would consider mainly the exposure (i.e., Is a system reachable from any device on the Internet, without controls?) and the business-criticality (i.e., How much can we loose if the system was lost/unresponsive?) of systems/assets, although there are of course other considerations. We here propose to add a third approach for decision support, by analyzing the interrelations of our IT systems, modeled as graphs, when designing and implementing cybersecurity controls.

Different from other similar work, we simulate two possible actions per node on our graphs, which will affect how fast (i.e., how probable) a node gets infected by any of its neighbours (is the computer duly patched or not? Is it protected with an EDR? ...), and/or how fast (how probable) a node gets cleaned (was the node duly inventoried? Was it monitored for behavior anomalies?). We propose an approach based on simulations on graphs, representing actual IT networks, and so-called viruses (based on epidemiology concepts), using a simplistic SIS model as a base component. In such a graph the different nodes are more or less "connected", depending on whether they sit in the same or different "subnets" and communicated/see each other. The "virus" in the SIS simulation then spreads following simple rules (summarily discussed later in this report). Our program then uses a Genetic Algorithm to propose a better than random assignation of a limited budget.

Note: We were very much inspired by the discovery of the fields of "Cybernetics", "Systems Theory" — and "Complexity" at large —, during the Master courses.

1.2. Project Objectives

We aim to demonstrate that Cybersecurity Operations (for instance) have the option to leverage computer-based optimization on simulations to supplement their decision process, which consider factors such as network structure, when it comes to limited budget allocation.

Note: Although not an initial objective, the proposed work and its results can also in concept be easily "translated" and applied back to more traditional Epidemiology scenarios, namely that of containment of human infectious processes.

1.3. Approach and Methodology

Several months were spent, starting around June 2023, on the project conceptualization and design. We would design and implement, "from scratch", a program that would run epidemiology-inspired simulations, and optimization of a fitness metric, in the form of a meta-heuristic algorithm, on top of said simulations, to propose recommendations to contain infections on networks.

Note: the spreading of the master courses over a period of three years has allowed an unusual time to digest the concepts presented in the different topics studied for this MSc.

Starting roughly in November 2023, the approach to the implementation of the underlying program for our Project was then that of iterative and incremental implementation, followed by the corresponding results analysis. After an initial concept was put forth, the methodology followed was in principle essentially what is called "Agile".

Five main fronts were considered:

- "Functional" Simulator implementation (including SIS Simulator, variable nodes "security measures" assignation, Monte-Carlo setup, then Genetic Algorithm implementation, with budget limitations). A point was made to implement as much as possible in the programming language R, unless deemed unsuitable (then moving to C++), and as much as possible directly from theoretical concepts.
- Speed considerations: Processing speed and later horizontal scalability (as described later on in this report).
- Visual Analysis tools (first in the form of simple visuals, then more complex and "original" visualizations, and finally the creation of an interactive dashboard, as part of the Project outcome).
- Results analysis and interpretation, which would feed a new iteration with improvements on both the simulator and at times the visualization tools.
- Report redaction. With the support of timely notes on a separate support, the contents of the report were being redacted alongside all the former activities.

An important supplement to the initial considerations was that of the implementation of a "Mean-Field" approximation, as an alternative to the Monte-Carlo approach, motivated by potential simulation speed gains and results validation.

Note that a relevant threshold was passed in April/May 2024, with the redesign of the Genetic Algorithm, which invalidated some of the former results (as "too poor", albeit not invalid), and offered faster and better results to be analyzed. This was made possible by a deeper review of the overall project status by our Project advisor. It also forced a re-run of many simulations and a new analysis of results, alongside a complete review of this report.

1.4. Project Planning

The activities we pursued during the Project for the ellaboration of the Thesis were:

- Project conceptualization and design: From June 2023 to October 2023 (main SIS simulation and GA), then from December 2023 to end of January 2024 (supplementary MMCA component), and GA re-design/implementation (April 2024).

- Implementation of Main Simulator and Optimization Process: From October 12th, 2023 to end of January 2024, both inclusive.
- Graphical results analysis code implementation: From December 2023 to March 2024, both inclusive.
- Mean-Field approach coding: January/February 2024 (supplement, suggested by our advisor).
- Interactive dashboard implementation (for support for final presentation video elaboration): From February 2024 to April 2024, both inclusive.
- Re-design of the GA and re-run of simulations: End of April, to mid-May, 2024.
- Final dissertation writing: From mid-December 2023 to May 2024.
- State-of-the-art review: From mid-November 2023 to March 2024.

Against best-practices, "State of the Art" came as a second thought at the beginning of the project, as the goal was to validate a personal idea through implementation. However lots of information had been collected previously that supported the Project concepts early on.

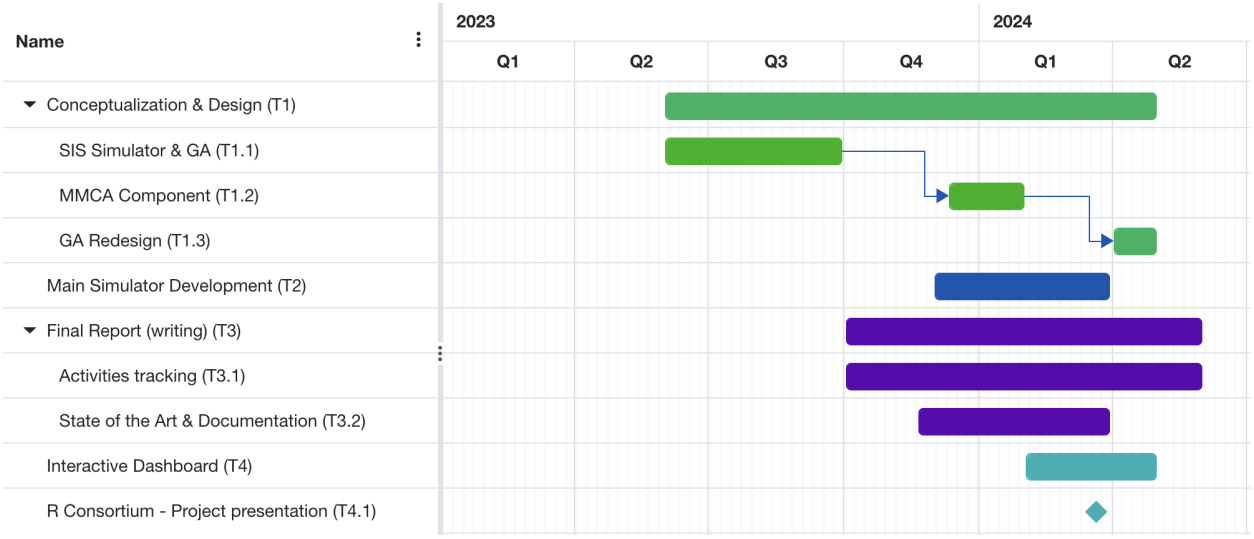


Figure 1: Overview of Project timeline

1.5. Summary of outcomes

The main outcome of the proposed Project is a simulator coupled with an optimization process, made of a functional (albeit simple) binary Genetic Algorithm (meta-heuristic) with mutation, tournament parents selection and one-point cross-over offspring generation, respecting a limited "budget", applied to a Monte-Carlo implementation of network/graph-based SIS infectious process simulations. It is geared towards optimizing (minimizing) the infection prevalence on said network(s), by selecting nodes on which to apply one or two potential improvements — better protection and/or better recovery —, while never modifying the underlying network structure. The above was implemented as code with several improvements to reduce overall processing times given the high volume of operations (congruent with networks size).

A second "approach" to the product was also implemented and included, with an underlying Mean-Field algorithm, which leverages direct probabilities, replacing the need for repetitive work imposed by the Monte-Carlo algorithm, and hence reducing drastically the required number of simulations ran to get to a valuable result.

All the above is initially coded in R, then in C++ for speed (for sequential operations), then made compatible with multi-core/thread local CPU leverage for local acceleration. A later version was soon implemented to support distributed processing over a containerized (Docker) API-based architecture, that allows to scale further (linearly) across multiple machines for different simulation configurations. All of that was implemented mostly in R, not using a third party product (albeit with the help of key libraries such as "igraph", "doParallel", "RCpp", and such).

Note that a modified version of the above simulator demonstrated other similar applications of the same approach, whereby instead of trying to reduce the overall prevalence of infection on the given network, the objective was set to protect a defined subset of its nodes. This has valid applications for real-world scenarios in the Cybersecurity world.

The results are presented in visual manner, and a supplementary part of the code was implemented to provide an interactive dashboard for simpler review of the results and facilitate the explanation of the involved concepts. To be noted, the project (and its initial results), in an earlier version, was presented (with permission) to the Madrid Chapter of the Spanish R-Consortium in March 2024.

The outputs of our simulations and optimisation processes are coherent:

- The higher the budget, the less overall infection prevalence, regardless of how the budget is distributed.
- For low probability of infection per contact, if the action of improving "detection and cleanup" makes an important difference, the better option is to spend some resources on detection and cleanup.
- For high probability of infection per contact, the better choice is to choose the protection measure instead, reducing said probability of infection per contact.

More importantly it is able to recommend solutions and provides insights such as:

- Network structure affects the ability of our product to propose impactful recommendations.
- Coherent with theoretical results, nodes centrality measures play a role in the choice for measures assignation.
- Under certain circumstances, even with only 40% of the optimal budget available, the better assignation of budget (as proposed by our program) can bring infection prevalence of nodes down to 0, after infection control has taken place, compared to 40% of them infected with random budget assignation.
- The assignation of budget, when too limited, is better spent spread among more nodes — with only one security measure per node —, than focused on a few nodes with both possible (available) security measures.
- And at a certain budget threshold, depending on the network topology, the value of a better-than-random assignation becomes irrelevant.

1.6. Ethical/Social impact, Sustainability and Diversity aspects

The proposed project in principle presents no ethical impact. Socially, a change in approach to cybersecurity management we hope would be positive.

Running many iterations of simulations on computers does have an associated carbon footprint. We hope that the aim of positively impacting cybersecurity management and the potential to reduce cyber risks and vulnerability exposure through our investigation will some day compensate the costs of running the simulations.

Finally, we expect no impact derived from this work regarding diversity aspects.

1.7. Report Outline

A short review of state-of-the art is presented first, including some notes on how this Project differs from said pre-existing work.

We describe the component of the code developed for this Project:

- We start with a simple simulator for a Susceptible-Infected-Susceptible infectious process on a network/graph. We also present the modifications on this base component, that will allow variable control measures per node — through the introduction of variable protection and detection/cleanup probabilities.
- We then proceed to introduce the implementation of the binary Genetic Algorithm, that will do the work of searching for recommendations.
- We mention in passing a possible modification to the fitness function, as an example of the value of our product with minimal modifications for different (similar) exercises.
- We discuss the challenges found in *processing capacity* and the different measure taken to face said challenges.
- The above justifies the description of an alternate approach to reduce processing needs, based on a Mean-Field approximation — instead of Monte-Carlo —, and we discuss a comparison of both approaches.
- We make a note of the visualization tools implemented as well for the Project.

We then analyze the results considering different aspects relevant to evaluate the overall Project, both in terms of the value and the validity of the implemented code.

We finalize this Master's Thesis with conclusions, a clarification of some assumptions, and we propose some directions of potential future work.

References used along the report, and finally annexes, are included last.

1.8. Computational Engineering and Mathematics: Topics covered

For the elaboration of this Project, we have used, in one capacity or another, concepts from the following Master topics:

- Simulations and Complex Networks theory, as all the results are obtained through network/graph-based discrete-time SIS simulations.
- Metaheuristic for optimization of decisions, in the form of a simple binary Genetic Algorithm.
- Graph theory to analyze findings.
- High Performance Computing, using multi-processor executions, including considerations of Gustafson's and Amdahl's laws to organize executions (C++ for faster sequential executions, batched per GA individuals parallelising). In fact, after a point, an implementation for multi-processor on multiple machines was implemented.

2. State of the Art & Related Work

Although the Project detailed in this dissertation was in fact developed independently — from "ideation" to implementation —, a necessary step in any scientific work is the study of similar and related work.

For instance, if only based on the abstract, it would seem clear that (Leyffer and Safro 2013) is in fact very much related to our work: "[. . .] We consider *an optimization problem that is based on the SIS epidemiological model. The objective is to detect the network nodes that have to be immunized in order to keep a low level of infection in the system.*" Indeed, that work is closely related to ours, although the approach in it differs in that it proposes "acting" on edges, and uses binary choices (i.e., either keep or de-activate an edge), all the above making assumptions of independence over time and proposing a pure numerical approach with step-wise time-related equations. In our work we instead shift focus towards the actual implementation of a simulator on actual networks to run our discrete-time SIS simulator, and we also act on the β and μ variables of each node with probabilities. Moreover, we use a Genetic-Algorithm as the meta-heuristic algorithm for approximating an optimization, while in the reference paper they choose to focus on "multiscale" methods, which aims at solving the issues related to scale that we, indeed, have faced in our Project. (Such "multiscale" methods could in fact maybe leveraged in future work related to our Project, as hinted in the reference's Appendix.)

In (Kim, Radhakrishnan, and Jang 2006), the focus is on the optimal control problem, in the sense of *minimizing treatment cost*, and working on delay introduced by an infection (with basis on a SIS model, like ours) and the corresponding treatment. Our work differs in that it focuses on *minimizing infections for a pre-set "treatment cost" by choosing the (approximate) best defense configuration.* This reference also leverages impacting the μ variable (recovery rate), while our work works on both β and μ. That is a key difference of our work, compared to many others, and in fact we have found no reference work that would act on both variables the way we did.

In (Neil Dhir 2021) a mention is made for the need for "AI-enabled security planning – AI-enabled planning systems which automatically enhance human decision making and action taking", and we make a note of that idea as it distinctly points to the goal of our Project. That paper also details an approach in which Reinforcement-Learning is used and considers both defender and attacker perspectives, which indeed we hint at in our "future work" section.

Building on it, (Alex Andrew 2021) the authors proceed to implement a simulator, which in that sense relates to our own Project. The proposed "YT" simulator is more complex than ours, and its actions are to either isolate or restore nodes, while it also simulates a "red" (attacker) agent, again hinted as an idea for future work.

Some other papers such as (Badham J 2010) discuss graphs structural aspects such as clustering and assortativity. Such concepts and their impact/value will be discussed in our report as well. More papers/abstracts were considered potentially relevant, like (Noam Goldberg 2012) in which a very similar problem is dubbed the "firefighter problem". We also reviewed (La 2016), which considers assortativity and approaches inspired in Game Theory, and (Sylvain P. Leblanc 2011), the last one hinting in particular at military applications of similar works. (Wagner et al. 2016) focuses on Network Segmentation and Heuristic search with agents.

Some NITRD programs are somewhat related to similar research as well and conceptually might have an interest in our Project ("Cybersecurity Game-Change Research & Development Recommendations" 2010). A few other consulted sources in search for inspiration were finally (Peter Grandits 2019), (Shaffer 2023).

From our personal library, other consulted references over the months and used in one capacity or another to inspire or elaborate this report included:

- (Newman 2018), critical for the overall concept and some key metrics.
- (Cortez 2021), (Enrique J. Carmona Suárez 2020), (Erik V. Cuevas 2017) were instrumental for the elaboration of the genetic algorithm.
- Most of the code was ellaborated thanks to the learnings over the years of support material such as (Adler 2009). Although the visualizations in the end we designed ad-hoc, the designs were also helped by the readings of (J. García 2018).
- Some ideas justifying the study of interactions (and regarding certain values of matrix multiplications) were found in (Ashby 2015).
- (Mitchell 2011) was an inspiration and provided context on the works of J. Holland.
- A background review on Complex Systems Modeling was supported among others by (Sayama 2015).

Note that this constitutes just a part of the study material reviewed, if we consider the rest of the last 3 years leading up to the elaboration of this Project.

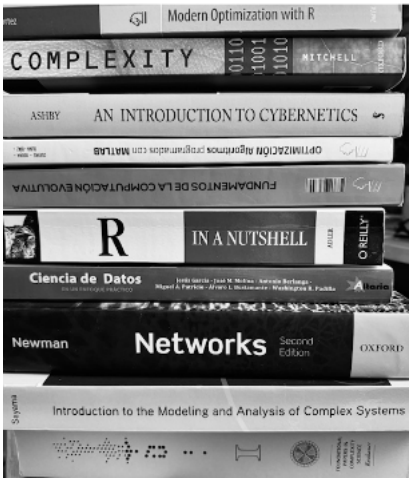


Figure 2: A few consulted sources

3. Development: Description

The basic component of the Project object of this Master's Thesis is an un-directed graph (a.k.a. "network") in which nodes connect, through edges, to some other nodes. We shall run "infection" simulations on such graph(s), and study how to best contain said infection, letting our computer(s) make recommendations. To facilitate the description of our simulator and optimization processes for this report, assume we choose a graph from a preset selection of them, on which to run the simulation — a discussion of such graphs choices is offered later in this dissertation. It could be any of:

- 50 nodes Erdős-Rényi.
- 50 nodes Watts-Strogatz (one "small-world" graph type).
- 50 nodes Barabási-Albert ("scale-free" network).
- 50 nodes "highly disassortative" network (to be discussed later).
- 31 nodes "IT Network", a manually created network to serve as a simple example network for IT practitioners — for reference mostly.

The simulator and optimization process can be used to run on any of the above or other networks. We shall discuss results with focus on the above networks for practical reasons only — but "complete", "tree" and "star" networks for instance were also used for testing.

Note: The order of the network is relevant for practical reasons, as will also be clarified later on.

3.1. Component one: SIS simulation on Undirected Networks

A "Susceptible-Infected-Susceptible" infectious process simulation on such a network starts with one or more nodes "infected", in a proportion $0 < \rho_0 \leq 1$ of initial infected population. For the remainder of this work, we fix certain parameters for our SIS simulations.

Parameter	Value	Comments
Initial Population Infected (ρ_0)	20% of nodes	Inspired by a past exercise.
Time Steps (T_{max})	N_g	After many experiments on small and big networks, we concluded this was sufficient.
Transitory phase (a.k.a. Warmup, T_{trans})	$0.9T_{max}$	90% of time steps are skipped.
Objective/Fitness measure ρ_f	$\frac{\sum_{i=T_{trans}}^{T_{max}}(\rho_{t_i})}{T_{max}-T_{trans}}$	Mean infection prevalence for the "stable" period in a simulation.

In our scenario, we use a discrete time-steps SIS. With the "passing of time", for each time step:

- We review nodes that are not infected, and check *each* of their neighbors (connected by an edge). For each infected neighbor (if any), we generate a random number between 0 and 1. If that number is below a threshold, β, assigned per node (see below), we consider the node infected and move on to the next node (we stop checking neighbors). Otherwise we repeat with the remaining neighbors. We thus mark the node as infected or not for the next time step. The higher the parameter β and the higher the number of infected neighbors, the more probability of getting infected.
- We review all the nodes that are infected in the current time step. We generate a random number between 0 and 1. If that number is below a threshold μ, we mark this node as "clean" for the next time step. The higher the value of μ (independent of number of neighbors), the higher the probability of removing infection at the next step.

We then proceed with the next time step, and iterate until a number T_{max} of time steps is reached (see table above). Finally, we calculate a "final prevalence of infection of the network": ρ_f. We will use this key result as a fitness to be optimized in later stages. ρ_f is calculated as the simple arithmetic mean of prevalence for a certain number of the final time-steps. We do *not* consider for this averaging a set of initial T_{trans} "transition" time-steps. In simulation terms, this is meant to obviate a potential "warm-up" phase. For the purposes of our Project, we will normally select: $T_{trans} = 0.9 * T_{max}$, as indicated in the table above.

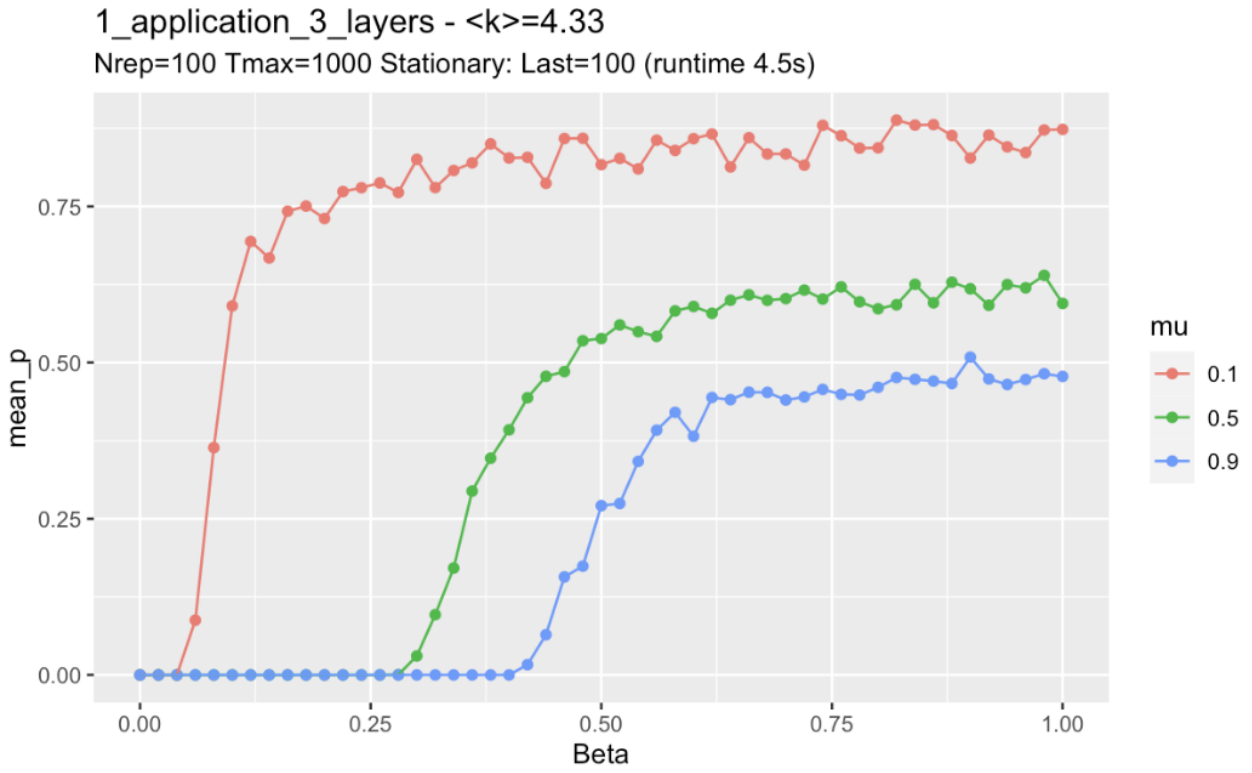


Figure 3: Running first SIS simulations on a "IT Network" inspired graph with different β and μ values

Notice in Figure 3 how for "higher protection" (lower β) of the nodes, the infectious process tends to a lower average final prevalence (left side of the visualization), while a higher detection (higher μ) also ensures a lower final prevalence of infection. And vice-versa.

This is an important conclusion for a good understanding of the rest of the Project.

An implementation of such a basic SIS simulator, programmed using a mix of R and C++, was previously

implemented during the course of the Master subject "Complex Systems", during the second semester of 2022-2023. That was in fact a precursor to the conceptualization of this Project.

Note: For the object of documenting this Project, we (later in time) proceeded to implement an interactive dashboard in an R "Shiny" application, that allows to show such an infection process in an animated manner (see Figure 4).

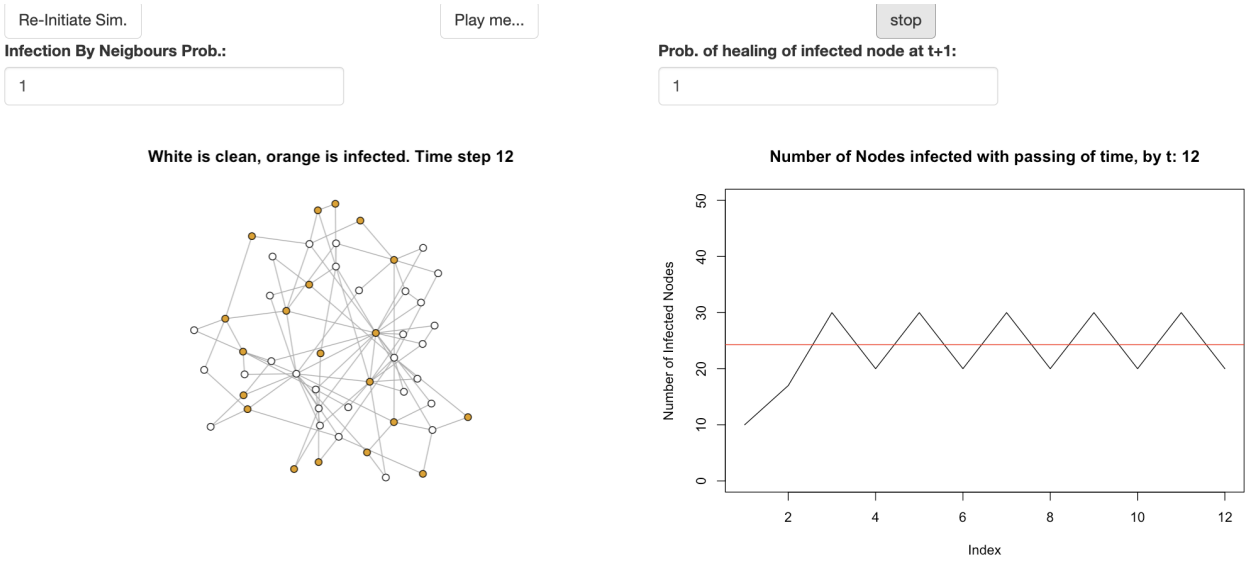


Figure 4: A basic SIS simulator in an interactive Shiny Dashboard with animated output. Example network is BA, 50 nodes. Initial Prevalence is set to 20% assigned randomly. Infectiousness and recovery probabilities adjustable.

3.2. Component two: Variable Protection of Network Nodes

In a normal discrete time-steps SIS epidemiology model, there is one type of nodes with one given β and one preset μ, respectively indicating for a given node its susceptibility (probability) of getting infected by an infected neighbour, and the probability of recovery if infected at the next time step.

We modify the basic SIS simulator to accept varying β, μ values per node. For all the simulations discussed in this report (unless otherwise expressly mentioned), we proceed to fix an "improved" value for both parameters. This is fixed for all simulations in all configurations for the rest of this discussion. These improved values correspond to the action of acting in "protecting from infection" (assigning β_i to a node) or ensuring better detection and cleanup (assigning μ_i to a node). For one particular SIS simulation, we define a "configuration" by setting 3 parameters: $Budget, \beta_b, \mu_b$:

- A Budget indicates how many nodes (in proportion of the nodes in the network) can be assigned the improved parameters, β_i, μ_i
- The "base Beta" β_b indicates the *default* probability of a node to get infected by an infected neighbour, *for the duration of one complete SIS simulation.* The above variables together are defining a "simulation configuration". This will be used for comparisons at later stages.

Parameter	Value	Comments
Budget multiplier	0.2, 0.4, 0.6, 0.8 ... 2	An indicator of the budget relative to the size of the network. E.g., For a budget multiplier of 0.2, $\|\mu_i\| + \|\beta_i\| <= Budget = 0.2 * N_g$.
Budget	Budget multiplier * N_g	A given budget indicates how many "actions" one can take for the simulation, as a number of nodes among those of a given graph.
β_i	0.05	Lowest probability of getting infected by an infected neighbour. Equivalent to receive a "vaccination" (epidemiology) or some protection (for a computer, e.g. patch or antimalware agent). Can be assigned to some nodes (based on other considerations)
μ_i	0.95	Highest probability of remediation, i.e., detecting and removing an infection from an infected node. Equivalent to receive a "cure" (epidemiology) or being closely monitored (for a computer, e.g. logs monitoring and re-deploy). Can be assigned to some nodes (based on other considerations).

3.2.1. Variable "Resistance"

In our scenario, we are interested in voluntarily acting on the *resistance to infection* of our machines — the nodes of the underlying graph —, in other words we want to increase the value $1 - \beta$ as much as possible, or equivalently: decrease the β. A first approach would be to arbitrarily set the varying β of each node before launching the simulation, fixing all other parameters. This was implemented, tested and validated. At this stage, we could set a fixed β value per node of a graph, at will, before running a SIS simulation. A video shows this level of progress here: https://www.kaizen-r.com/2023/10/Project-log-day-6/.

3.2.2. Variable "Detection and Remediation"

After modifying the code to accept varying β, effectively being able to set a chosen value per node, we just continued the development and modified the code so that we could do the same and arbitrarily set values of the μ parameter per node as well, thereby affecting per node *how probable it is to detect an infection and clean it at the next SIS simulation time step.*

3.3. Component three: Genetic Algorithm

In order to be able to run only a restricted number of simulations for posterior analysis, we fix for our GA the following parameters:

Parameter	Value	Comments
Population Size	4 * N_g	Most of the time, 200 individuals (50 nodes networks)
Max Iterations	3000	Too high value for more recent GA implementation
Max Iterations without Progress	4 * N_g	Fitness (ρ_f) without change for: the whole generation mean, top 20 individuals mean, best individual

Parameter	Value	Comments
Tournament Pressure	20 individuals	Fixed number (i.e., relatively higher for smaller network studied)
Mutation Rate	10%	One pair of gene exchanged, per selected child (10% of children pop.)

We discuss some of these parameters in the following subsections. The two possible "acts" described above (acting on β, μ), per node, will each correspond to an "effort". As a Cybersecurity Operations unit has limited resources/budget, such available efforts (time, personnel, spending) might be limited by said budget, thereby in some cases impeding the deployment of protection and detection measures on all nodes (which would be the ideal scenario) of a given network representation of an IT Infrastructure. *This limitation is a central concept of the Project.*

We now come to the idea of *optimizing the use of such limited resources*. This is a task for Operational Research methods, and in the present scenario, a simple (metaheuristic) *Binary Genetic Algorithm* was chosen.

3.3.1. Individual Fitness Evaluation: Monte-Carlo

A key consideration of the implementation (with many implications for the final results, as will be detailed later) of the upcoming component is that of the stochastic aspect of things thus far. Up until now, we have considered to run a simulation a number of time steps. At each step, however, for the SIS simulation we have chosen, the nature of the parameters β, μ is probabilistic: the infection progresses depending on random numbers calculated for each node at each time step. Moreover, we need to *also* consider the random nature of ρ_0, which adds to the complexity of the problem. A configuration might produce an averaged ρ_f that differs from another depending solely on the initial subset of chosen infected nodes. In order to have a correct understanding of the actual *expected value of* ρ_f, a Monte-Carlo approach is deemed necessary, which will average multiple iterations of the whole process.

For the rest of the discussion, hence, we assume it is clear that each configuration (choices of how to assign a limited budget, as will be explained) is in fact executed several times and for each such configuration the results is an average of the several iterations (N_{reps}).

3.3.2. GA: Simplistic description

This Genetic Algorithm is derived from the works of John Holland. We shall start with a simple Network as a basis, and implement the GA, using a binary approach. At this stage, essentially any small/simple, non-regular (see discussion later on) graph should do. We set a traditional configuration for a SIS, with a base β_b and a base μ_b, applicable to all vertices. We can iterate the evaluation of the objective/fitness using Monte-Carlo, all as described thus far.

Now suppose you have the possibility to choose to improve one or both parameters for some of the vertices, to a different β_i and μ_i. At the beginning, we will then set things up so that each vertex is assigned one of (β_b, β_i) and one of (μ_b, μ_i), such that $|\mu_i| + |\beta_i| <= Budget = 2 * N_g$, that is, one can assign up to a certain amount of improvements across the network. How to assign such improvements? A GA will now try to assign as best as possible the available Budget so as to minimize our objective/fitness, the final infection prevalence ρ_f (i.e., mean % of vertices infected in a graph).

For the Genetic Algorithm, a population of individuals are evaluated. Each individual is encoded as a set of "genes", one for each chosen β for each node, and one for each μ for each node. In other words, each

individual represents one chosen budget distribution, with twice as many "genes" per individual as the order of the graph (i.e., number of nodes). This will be reflected in the resulting budget: $0 < Budget \leq 2$. A simple GA will then try and mix and evolve the individuals towards an evaluation of overall ρ_f that is minimized, while respecting the limits of the budget.

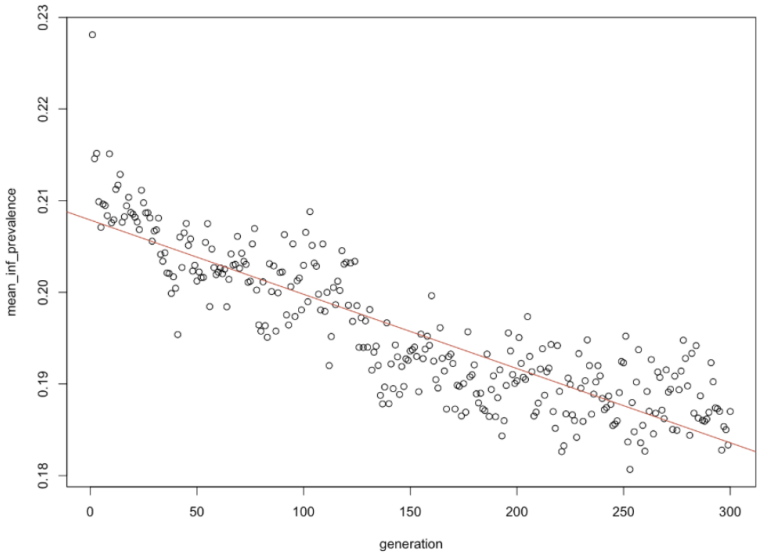


Figure 5: First ever confirmation of the overall Project value: reducing prevalence of infection over a network through GA optimization

3.3.3. Further Algorithm Details

- The GA will select parents in each generation using a simplistic deterministic tournament. A static tournament pressure is set (20 individuals). This is a relatively high pressure, particularly for smaller populations — the populations are normally set to $4 * N_g$, and most networks studied were of 50 nodes.
- We generate children out of random pair of candidate parents, and to do so we cut the genome of both parents at one random point and cross the two parts, thereby generating two children that each inherit one part of each of their parents' genome ("one-cut crossover").
- A special modification to the cross-over was that of ensuring the budget was still respected in the children. If after extracting parts of the genome of each parent, one child was "over budget", we exchange single genes until both children have the same exact (maximum legal) budget. For 200 parents, we generate *only* 180 children (90%), keeping the top 20 (10%) best parents as survivors for the next generation.
- As will be seen later in this document, and as described by the works of John Holland, we will then add some improvements by introducing randomized mutation, with a given "mutation pressure" of 10%. Once again, respecting budget was key and the last version of the code allows to change "one gene" for each of the 10% randomly selected individuals, but in fact *two* genes (one 1, one 0) are exchanged, thereby ensuring the configured budget for a given configuration is respected at all times.
- We finally generate the new generation with the best parents (10% best of former generation) and the new children, and iterate the evaluation over the SIS simulator for each individual, calculating a new population and its individuals' fitness.

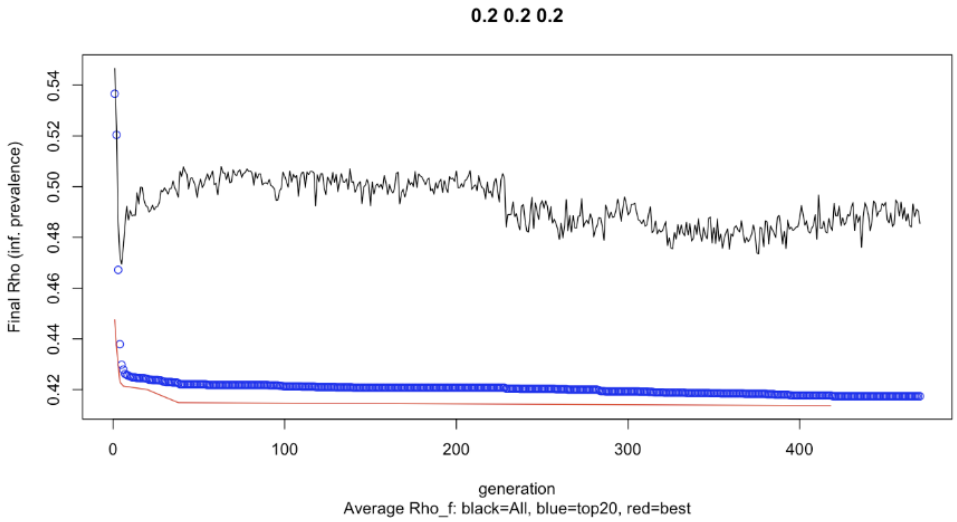


Figure 6: Final version, months later: confirmation of the progress for a BA Network, 50 nodes, budget 0.2 (10% of total possible), Base Beta 0.2, Base Mu 0.2

3.3.4. Leveraging Theoretical results

Why keep the top 10% individuals from one generation to the next?

This is inspired by theory, whereby the initial generation is "salted" with a few "good candidates": we use *centrality measures* to assign their genes, instead of random initialization — as per the rest of the first generation. For instance, in our approach, we can modify two variables for a fixed budget, per node: β and μ. Should we then assign *both* better β and better μ to nodes by centrality, in decreasing order? We confirmed already in the earliest versions of the program that, at least for our simple Scale-Free networks, Page-Rank and Eigen centrality are *generally* most correlated with best solutions in observations. It all depends on how much better we set the improved β_i (e.g. 0.05) compared to base-value β_b (> 0.2), and similar considerations apply for choice of μ ($\mu_i = 0.95$ for instance will soon be fixed in configuration).

We also knew by then that the budget is better spread than concentrated: we have observed how the algorithm has consistently preferred *not* assigning both improvements to one node (exception made of key nodes, and depending on budget) — See Results Analysis for further details. So we could in principle kick-start our initial population with, among other random individuals, a few *good theoretical choices*:

- 50% of budget assigned for improved β_i to best nodes by "Page-Rank decreasing sort", and then 50% of budget assigned for the next most central nodes for improved μ_i.
- Inversely, 50% first assigned for μ_i, and 50% then for β_i.
- 100% of budget assigned to top-Eigen-Centrality (similar to Page-Rank, indeed) nodes, split for both improved β_i, μ_i.
- etc.

This might conceivably make for a few theoretically top candidate individuals at the beginning of the selection process, as well as ensure reasonably that *even for small populations, the most central nodes are given a chance to participate*. Results soon confirmed our approach, "tricking the GA", was valuable.

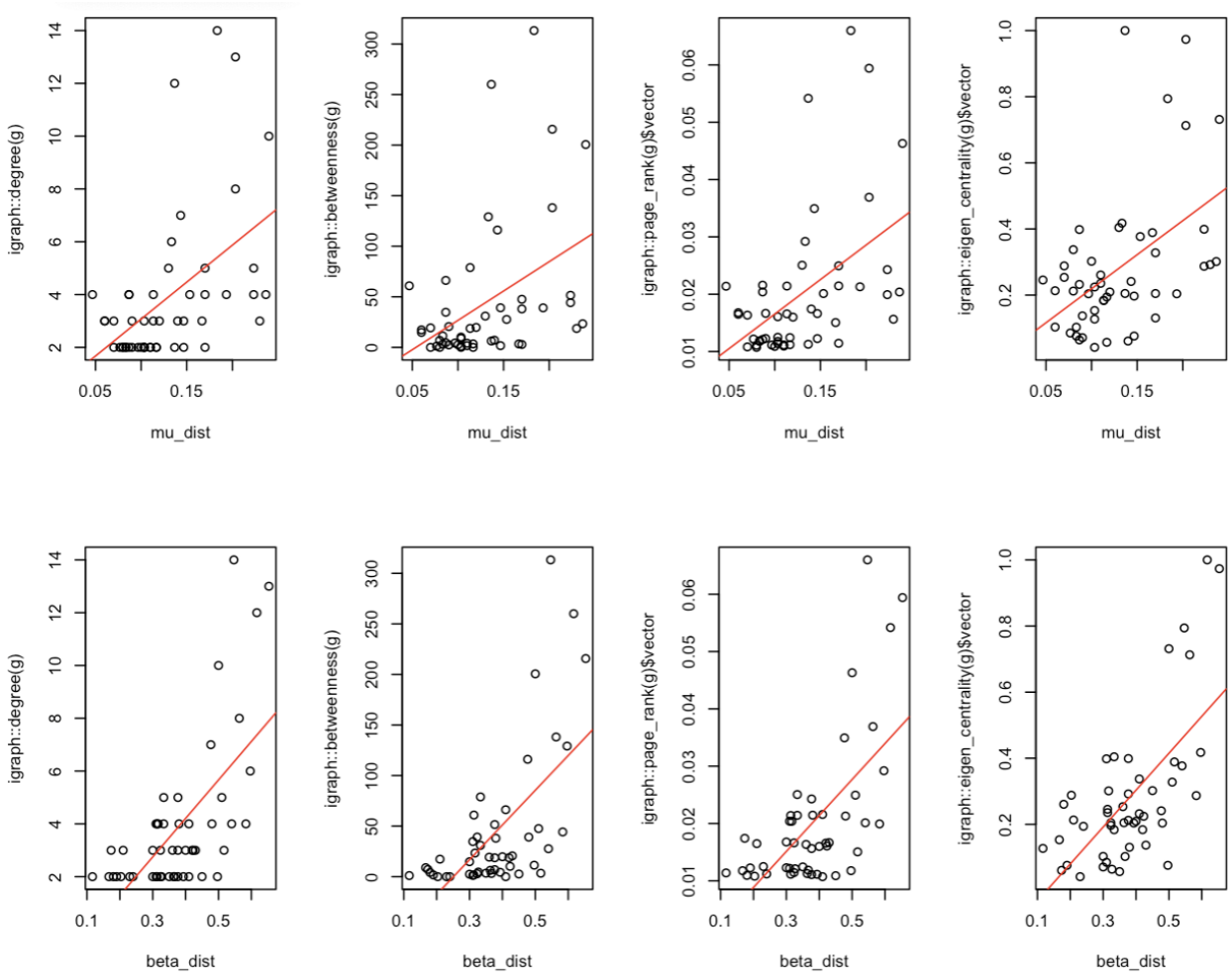


Figure 7: November 2023: Studying Centrality Measures and relationship with chosen nodes by a preliminary GA

3.4. Other applications: Changing fitness (optimization objectives)

For the sake of argument, we could suppose that we have a different objective in mind for protecting our network. Let us suppose that, as a Cybersecurity practitioner, one might focus on protecting *certain nodes* (computers) in particular, instead of the overall network. We could imagine some subset of the computers can be considered critical for some reason (highly confidential data, etc.). Let us call them the *Crown Jewels* of our theoretical network. Once the components described, up to this point, are ready, it is rather trivial to adapt our product to focus on such an objective (different, albeit very similar). In this case, we keep the minimizing objective, but instead of the final *overall* infection prevalence ρ_f of the whole network, we could assign the fitness measure for the GA, to optimize (minimize) the infection of *a subset of nodes* only. This was implemented as an exercise (see figures 8, 9). The program made what appear to be valid decisions (albeit maybe not global optima) for assigning a given budget to protect the Crown Jewels. In the attached figures, we select 3 nodes and consider them "special" (they appear bigger than the rest of the nodes).

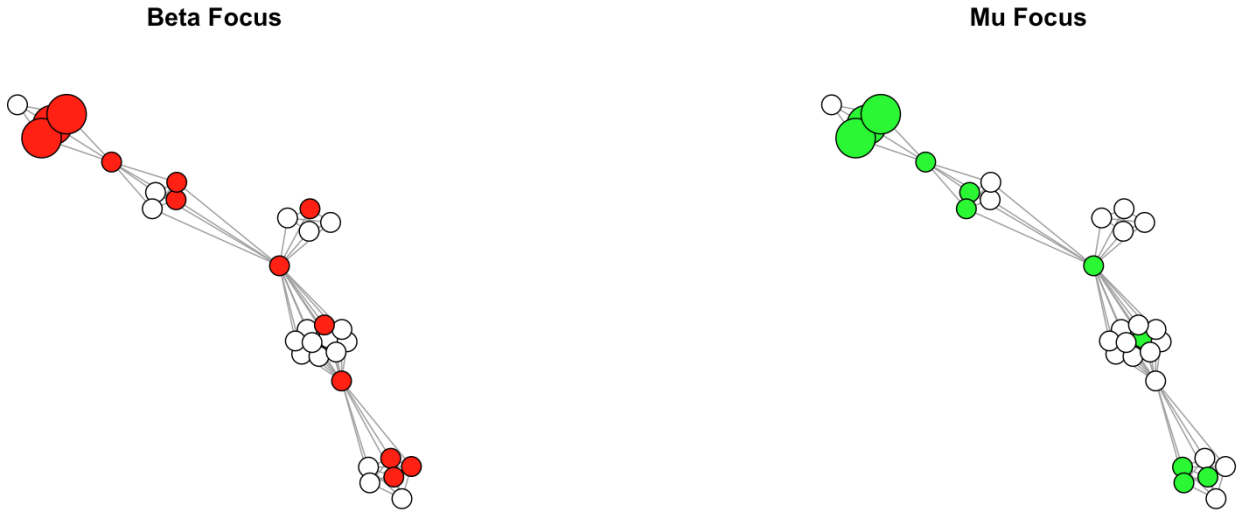


Figure 8: Focusing on certain nodes, the program looks for alternatives to protect such nodes in particular. With enough budget, aside from key nodes, the rest of assignations has little to no overlap

Note: Such a simple network, with its "segments", facilitates interpretation of results and concepts explanation to other IT practitioners. See https://www.kaizen-r.com/2023/10/Project-log-day-17-crown-jewels-version/ for a short discussion on this sub-topic.

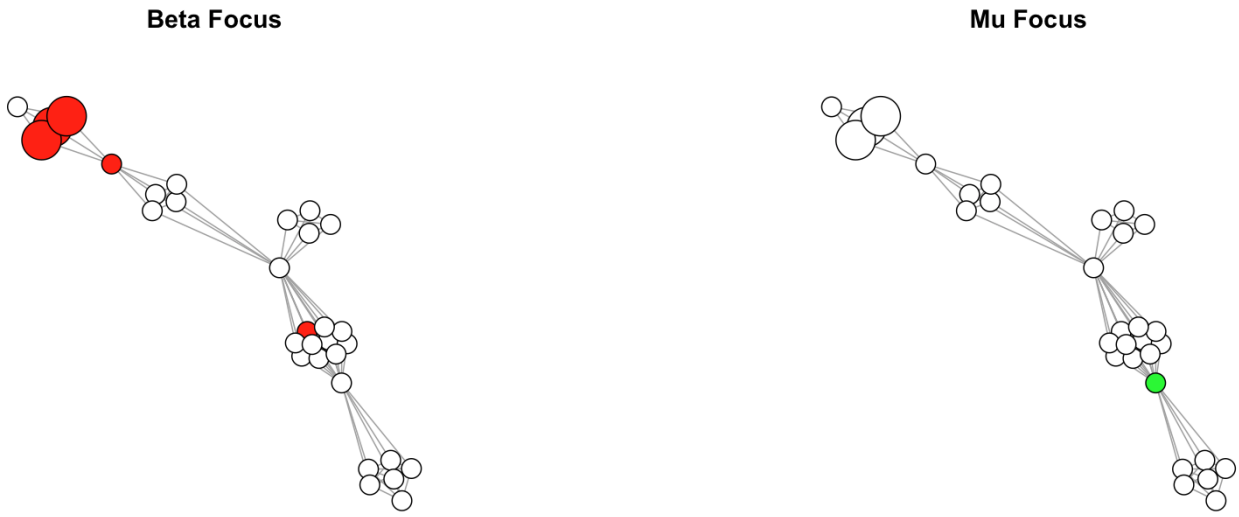


Figure 9: With little budget but more power of improvement by acting on Beta

3.5. Full Runs and HPC

One relevant aspect of the Project is when we proceeded to running "full sets" of simulations. For any given budget, we prepared our product to be able to propose results for any combination of β_b, μ_b. To demonstrate that and explore the optimisation across many such pairs of values, we had to run *many* simulations, each with many GA generations.

The reason for such a "brute-force" approach was motivated by a possible critic to our approach: There is no objective way to say "protecting a node (i.e., acting on the node's β) moves the node protection from 0.5 to 0.05". It is hardly realistic to assume we can set an actual exact number to the actions, even supposing we are capable of somehow encoding the value of our security measures and their impact. Therefore, we deemed it necessary to analyze what the value could be regardless — or rather, *for any* — of the corresponding actions impacts. Whether a machine without an anti-malware software is considered to be "half protected ($\beta = 0.5$)" or "little protected ($\beta = 0.9$)", we consider that actually deploying an anti-malware software on said machine brings its risk of getting infected to a very low value, "very protected ($\beta = 0.05$)", yet another arbitrary value. The important point being that we look at the *difference* of acting versus not acting, and we aim at doing so without having to worry of *how much* relevant are these differences.

Observing and discussing the value of the recommendations of our product across *all possible* such differences (and as can be seen later on, with rather smooth surfaces), for a given budget on a given network, will demonstrate when it can make sense to use our product (and when not), if not exactly quantitatively, then maybe with qualitative ranking of the value of different measures for protecting a node (e.g. "Putting an anti-malware software is more important than patching the Tomcat on the Application Server").

Let's then tackle the needs for our software product to be capable of running *many* of our simulations. The code was first implemented locally on a machine with a mix of R (at first) and C++ (soon, for obvious speed gains). For each generation, R would launch different processes (one per individual of the GA at that generation) in parallel (using R packages for parallelizing such as "doPar"), taking advantage of the available CPU cores/threads (depending on underlying CPU architecture), following Amdahl's law (i.e., portion of the process that can run in parallel). Then the whole SIS process (as described earlier), iterated a certain amount of times (for MonteCarlo purposes and ensuring evaluation for different random initial ρ_0), was coded in C++ (Gustafson's law, i.e., speed up the serial part of the code).

For a network of order N, we need to choose between two options per node per variable (assigning or not budget to improve β, and similarly for μ), hence the complete search space can be up to (upper limit, "full budget available"): 2^{2N}. Although most of the above options are discarded by the limited budget, so for a budget of 0.6 (i.e., 30% of the max budget of 2), we're talking about choosing $.3 * 2N$. For a network of 100 nodes, that's about 60 positions out of 200, $\binom{200}{60} = 7.04 * 10^{51}$, as lower spending would be pointless — 1 million tests seems like a fair compromise, even for an imperfect solution. Moreover, for each of these possible configurations, evaluating the fitness (ρ_f as described so far) requires running T_{max} time steps, *per individual*, and each time-step has a number of algorithm processing steps roughly related to the average degree of the nodes of the network.

With all the above, and simplifying a lot the calculations, *the processing time grows exponentially with the size of the networks being studied.* (An incorrect but interesting exercise towards "guesstimating" the actual complexity was done here: https://www.kaizen-r.com/2023/10/project-log-day-18-some-numbers/).

In summary, given the above, and consistent with the Master topics, we decided to implement the Project in a "highly" distributed manner (see figure 10). With the separation of the "worker" code, then set to run from a Docker container, one can centrally control the workloads distribution across different machines. A Dockerfile is created that prepares a container base image to serve a plumbeR-based API server to listen for requests to process certain configurations (the Dockerfile used for this is provided in folder: /docker_imgs/r_tfm/). This implementation provides a one-line command to instantiate new workers, which shows how this setup, the Project code, was evolved towards being useful for realistic productive setups, at scale. And indeed it worked as expected. The image for the server can then easily be created using a command such as:

```
docker build -t r_tfm_plumber_server1 .
```

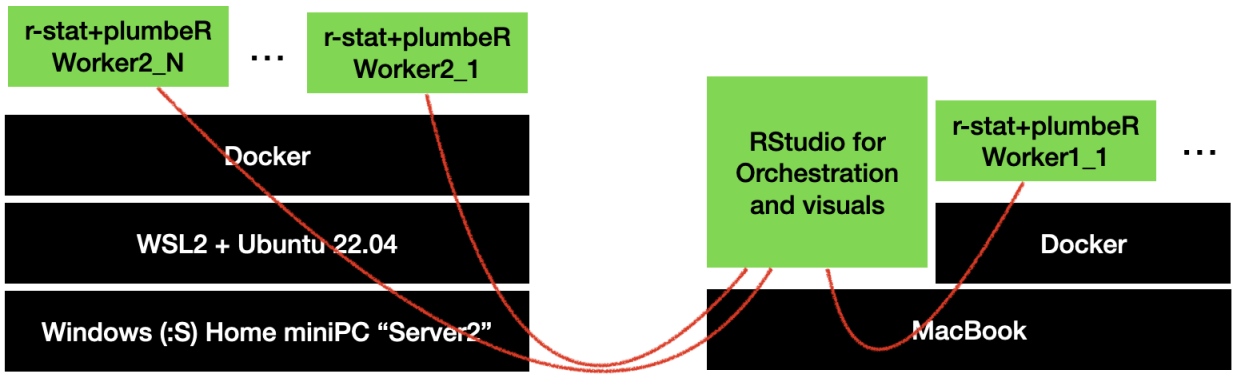


Figure 10: Distributing Workload on different physical machines, an example

The above was important to be able to run the many different configurations we wanted to run in a reasonable amount of time, and it is a direct application of the HPC course concepts.

Note: One difficulty faced while implementing this variation of the code was that plumbeR (an R Package to expose APIs) by default keeps the R interpreted code occupied and didn't by default created a separate process, making the API unresponsive. This was solved by adding the use of "futures" in R, which however forced to respect one thread/core of the CPU to receive API queries. Another difficulty faced was that of debugging evolutions of the code in such a distributed setup. We eventually settled to have two similar versions, one local for testing and validation, and one for distributed processing.

```
[1] "Worker localhost still busy with job 1 ..."
[1] "Worker 2 done with config 2"
[1] "Worker localhost still busy with job 1 ..."
[1] "Sending job: 3 to worker: 2"
[1] "Worker 10.0.1.7 still busy with job 3 ..."
[1] "Worker 1 done with config 1"
[1] "Worker 10.0.1.7 still busy with job 3 ..."
[1] "Sending job: 4 to worker: 1"
[1] "Worker localhost still busy with job 4 ..."
[1] "Worker 10.0.1.7 still busy with job 3 ..."
```

Figure 11: Distributing workload with more than one machine

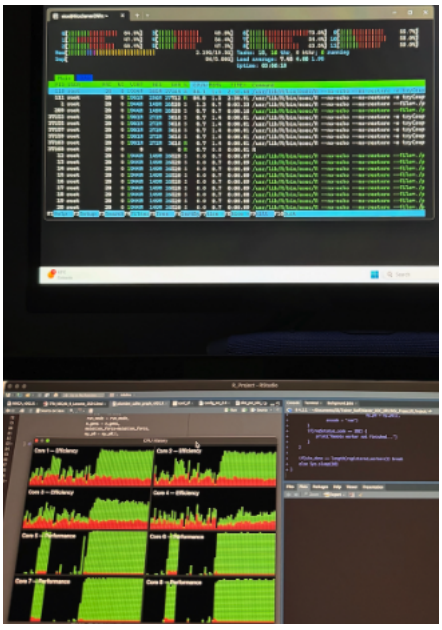


Figure 12: Running simulations across different machines and with multi-core/multi-threading

3.6. MMCA

3.6.1. Implementation

The implementation of a Mean-Field approach inspired by our Advisor's work (see (Gómez 2010)) is meant to provide an alternative to the actual Monte-Carlo step-by-step SIS simulation component we have used thus far. The only modification we make to the approach proposed in the paper is that of using different β, μ parameters for each node across the network. Here to avoid confusion with the reference paper's nomenclature, with the subscript "i" we refer to a node, for the next two equations only. The first term on the right-hand side of the next equation is the probability that node i is susceptible $(1-p_i(t))$ and is infected $(1-q_i(t))$ by at least a neighbor. The second term stands for the probability that node i is infected at time t and does not recover.

$$p_i(t+1) = (1-q_i(t))(1-p_i(t)) + (1-\mu_i)p_i(t)$$

$$q_i(t) = \prod_{j=1}^{N}(1-\beta_i a_{ij} p_j(t))$$

Where $q_i(t)$ is the probability of node i not being infected by any neighbor.

Note that in our implementation of the MC, we have considered the "reactive process" implementation (RP) and thus we have used the simplification $r_{ij} = a_{ij}$, which in our (once again simplified) approach with un-weighted un-directed networks, is always 1 for any existing edge as designated by our adjacency matrix. We also discarded same-step re-infection in the MC and thus needed to remove the component of the equations. Notice as the single modification above how we reference (different from the original paper) the node's specific values for β, μ, instead of the overall network's default parameters. Such implementation proves immensely faster (in processing time) than the basic SIS simulator used above. A key reason for the difference is that for the Monte-Carlo approach, we are forced to consider multiple executions with multiple initial subsets of infected nodes, always keeping $\rho_0 = 0.2$. A second obvious reason is that the MMCA approach needs not iterate over each node and consider their infection status, and eventually several or all of its neighbours, to calculate each time-step value of ρ_t.

```
>
> microbenchmark(iter_for_beta_vMMCA_cpp(Beta_v, mu_v, Nrep, Tmax, t_adj, my_p0),iter_for_beta_vMMCA(Beta_v, mu_v, Nrep, Tmax, t_adj, my_p0), iter_for_beta_v5(Beta_v,
mu_v, Nrep = 50, Tmax, Ttrans=90, t_adj, my_p0), times = 100L)
Unit: microseconds
                                                          expr      min        lq      mean    median        uq       max neval
              iter_for_beta_vMMCA_cpp(Beta_v, mu_v, Nrep, Tmax, t_adj, my_p0)   833.202   843.5135   853.2572   850.627   860.5285   901.672   100
                  iter_for_beta_vMMCA(Beta_v, mu_v, Nrep, Tmax, t_adj, my_p0)  3584.384  3656.6670  3849.1542  3710.561  3750.8030 17342.016   100
 iter_for_beta_v5(Beta_v, mu_v, Nrep = 50, Tmax, Ttrans = 90,      t_adj, my_p0) 12203.978 12378.2280 14547.8836 12701.759 12791.6720 28281.636   100
>
```

Figure 13: Comparing running times of the new MMCA approach vs the former Monte-Carlo approach

3.6.2. MMCA vs MC: limits to GA results comparison

We now need to consider how our modifications to the original SIS algorithm estimation of ρ_f would affect the comparison of the MC approach with the MMCA approach. The original MMCA was proposed as an alternative to MC but with single values for β, μ. We have implemented a version that accepts variations per node of these parameters. Namely, our objectives will try to assign "improved" β_i, μ_i, which in turn should, for each chosen/given node, reduce drastically its contribution to the overall ρ_f.

To make things comparable, and as the MC approach is stochastic in nature and so is not deterministic, we compare the MMCA to the average of a few (20 to 50) iterations of the MC approach. We do that to compare the (single) MMCA results with the corresponding (reasonable) range of the possible MC results. We can definitely observe that the MMCA (lines) "overshoots" for the lower values of β, in comparison with the MC results, although it otherwise approximates nicely the MC ranges.

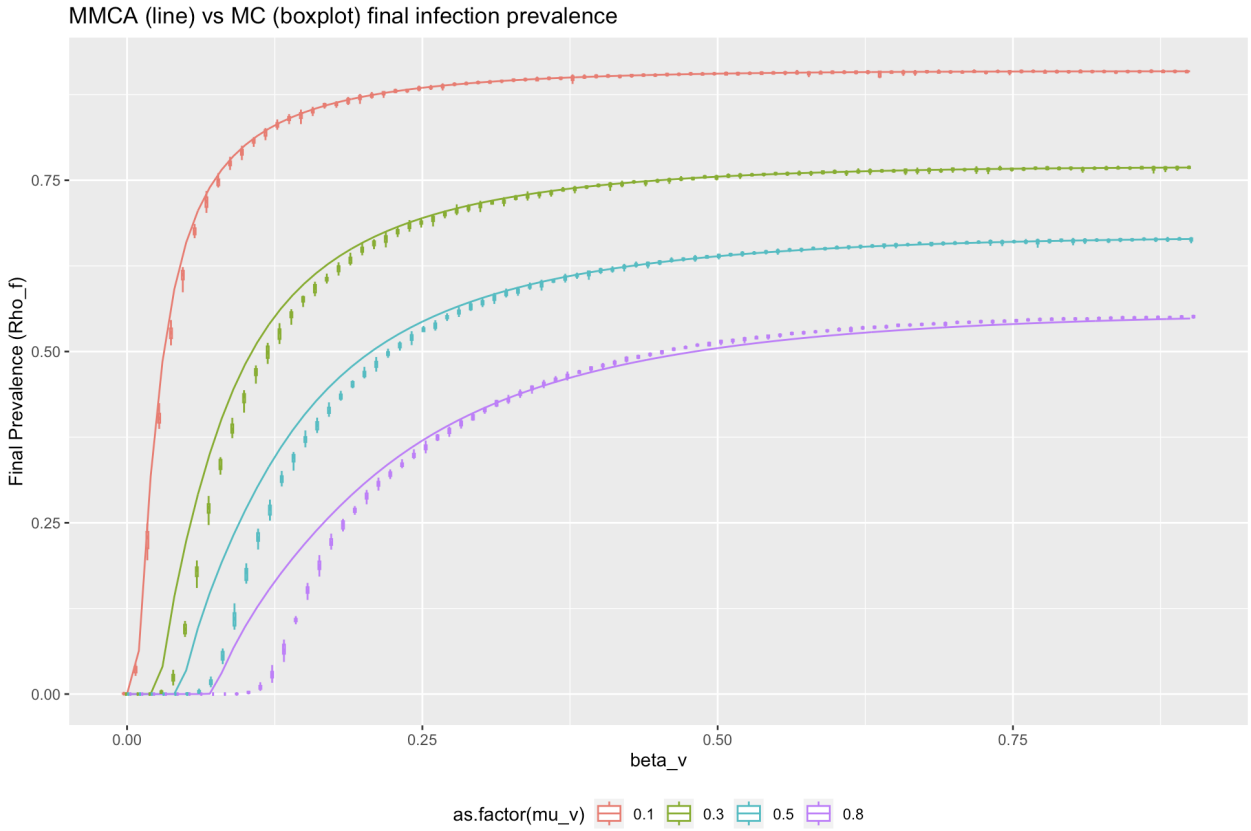


Figure 14: MMCA overshoots MC for lower β, particularly for higher μ, which is the whole concept of the GA choices

It does so particularly for higher values of μ. This is *relevant* for future comparisons of MC vs MMCA, as these are precisely the things we are changing, per node, and assigning differently with the GA. We can in fact visually analyze such differences by generating a set of 2500 random "GA individuals" (regardless of budget, just choosing random assignations of β_i, μ_i), see Figure 15: in roughly 80% of cases, the MMCA calculation for the corresponding ρ is higher than the calculated value by the MC (shown in red). In most of the resting 20% the results are almost identical (green), and in *exceptional* (43/2500) cases the MMCA actually "under-evaluates" compared to the MC.

In other words, we can expect that when comparing proposed "best configurations" (individuals) by the GA, if we evaluate them with the MC approach and with the MMCA approach, the MMCA will obtain generally higher values. The other issue with comparing results of running the GA with MMCA and with MC is that it is complicated, for two key reasons:

- First, they are not the same values, as shown so far.
- Second, as we have explained earlier, *neither the SIS simulator, nor the GA, is deterministic.* Any two executions can provide different results.

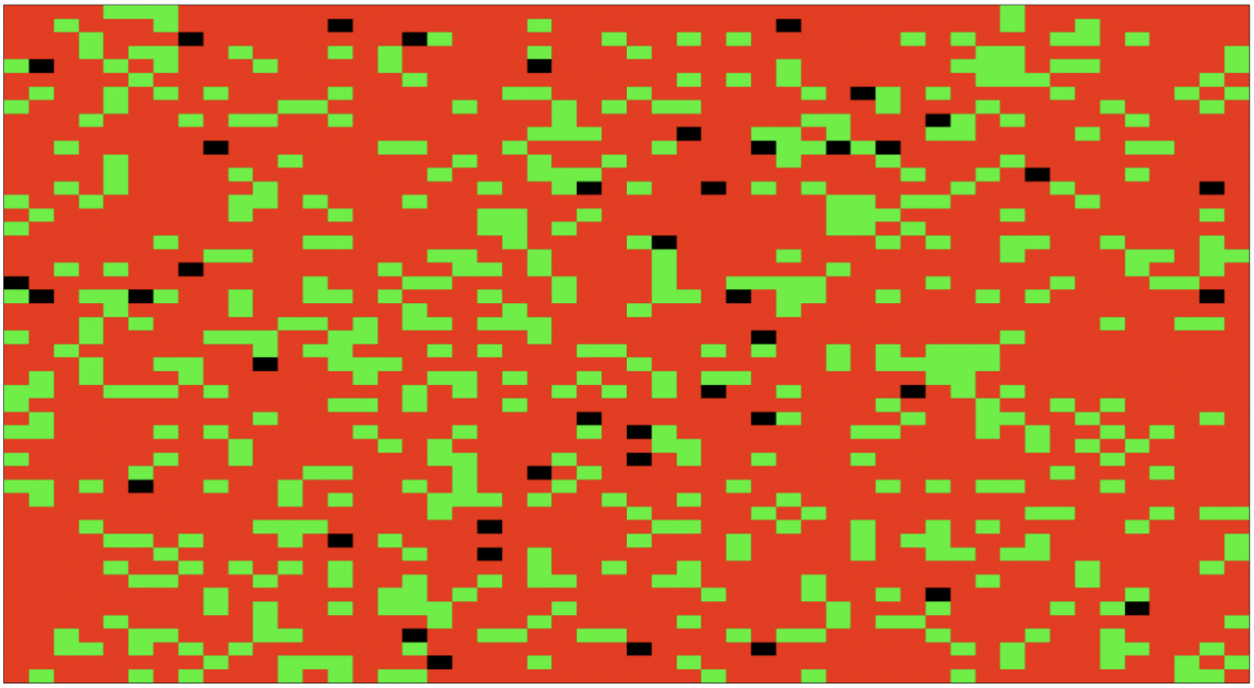


Figure 15: Calculating ρ_f for random individuals: MMCA overshoots (red) most of the time compared to MC

Finally, we could expect that for a random set of 100 legal individuals, both approaches (MC and MMCA) would agree on the order of their resulting ρ_f. And indeed, there is a *clearly visible correlation in the order* (Figure 16).

In other words, although imperfect, the GA with MMCA should obtain gains correlated to those found with MC for the same configuration(s). The recommendations with both algorithms are indeed found to be generally consistent with both approaches, as will be seen later.

3.7. Visualizations and Dashboard Tool

As a note, the "Project", once the product was implemented, consisted of a rather "brute-force" approach as discussed and justified earlier, whereby many hours were spent running simulations, for different networks, budgets, β_b, μ_b. However, this resulted in many populations of genes, infection prevalence, networks, etc. We consider it part of the outcome of our Project the corresponding visualization designs and tools (inc. a Shiny Dashboard, see Figure 17), that were used repeatedly for the preparation of this report.

Note: Such tools shall be leveraged as well in front of a Jury for the defense of the Project.

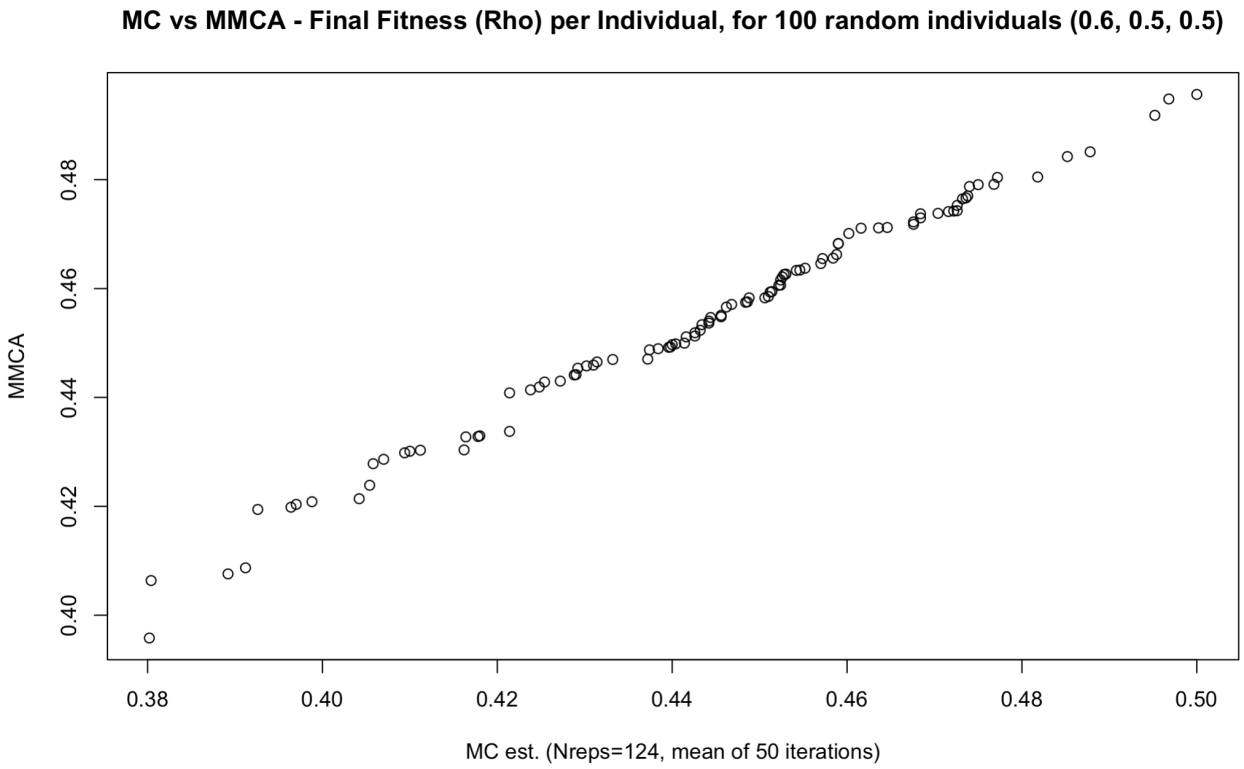


Figure 16: Correlation of ρ_{final} for MMCA vs MC

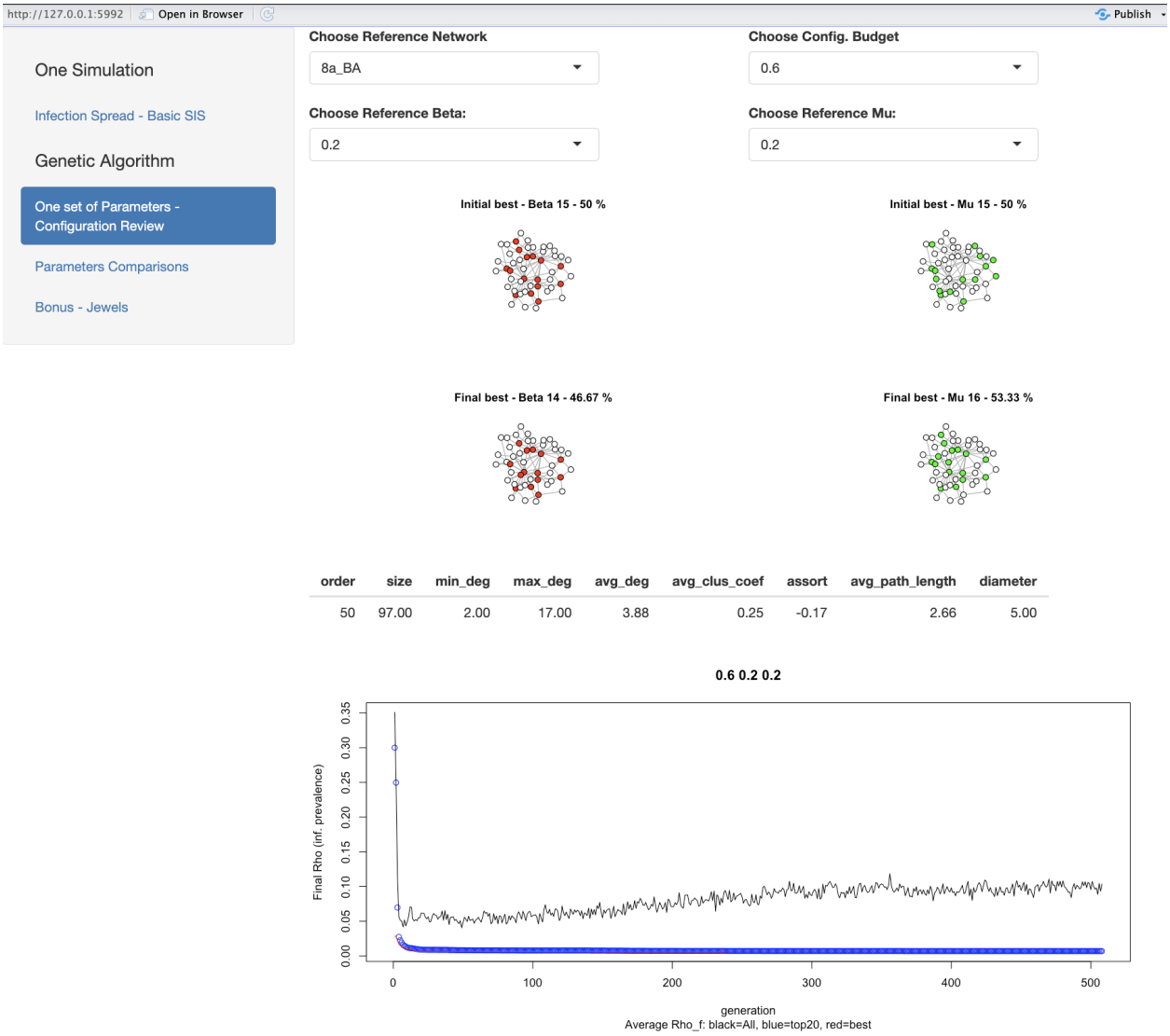


Figure 17: A Shiny Dashboard for interactive Project Demonstration

4. Results Analysis

As will be shown, the proposed approach can quite *consistently* decrease the final infection prevalence after a simulation of the spread of a virus, across varying parameters $(\beta_b, \beta_i, \mu_b, \mu_i)$, for a number of tested values and on different graphs/networks, provided sufficient generations and a few other running parameters are respected. That is, compared with a default budget assignation, i.e., random choosing of vertices for applying better β_i or μ_i. Which is of course the expected behavior of our program.

We shall also note that for certain parameters, particularly when sufficient budget is allocated, e.g. $Budget \geq 1 * N_g$, the simulator sometimes misbehaves as it *fails to optimize* (minimize) the final prevalence ρ_f. This is due to the *lack of improvement compared to random assignations* and values of final prevalence $\rho_f \approx 0$, making it hard or impossible for a GA to find any improvement — the exercise is then trying to improve on an initial random situation that is already almost perfect as per the objective. However as the goal is to find improvements under situations of limited budget (i.e., not perfect situations where all improvements can be applied), this issue with the simulation is *not relevant* for our goals.

Note: The consistent results were even observed in an earlier version of the program too, although with poorer gains, with a rather bad implementation of the Genetic Algorithm.

4.1. Teleological Results Review: Simple Network

Now that our GA can propose a better selection of subsets of vertices to modify given a limited budget, can we understand why such choices?

Let us consider results for our simplistic "5 Layers IT Network" (see Figure 18). *This network will be used again at later stages.* Starting from random improvement assignations, the GA proceeds to improve on it to reduce final infection prevalence ρ_f, with limited resources. By reviewing the chosen nodes, one can in this instance potentially interpret the recommendation (lower part of the illustration):

- Focus on network choke-points (in other words, possibly those with *"high betweenness centrality"*)
- mostly avoids spending on protection *and* detection for the same node. This seems like a key conclusion that might not be obvious to get to from a simple thought experiment, i.e., without such simulations. After multiple runs on multiple networks, we seem to confirm that the algorithm rather consistently chooses to *either* protect (improve to β_i) *or* detect (improve to μ_i) for a given node (at least for certain configurations).

We have shown and validated repeatedly how the Genetic Algorithm is effective in reducing infection prevalence by choosing where and how to allocate the budget, in almost all scenarios. Overall, when there IS improvement by the GA, the curve of reduction of prevalence over generations presents the form of an "elbow", with rapid improvements along the first few generations. Upfront that makes sense, as a purely

random budget allocation vs one that is focusing on actually reducing the final prevalence ρ_f, at the beginning will suppose most improvement, while after a while moving resources allocations might have more and more marginal impact. It might however make sense to give the computer enough time to process as the improvement do *not* always plateau *until later* (in numbers of simulations, that is, the number of generations for the GA). We can also observe that the proportion or *amount of improvement* is not static at all across configurations, as we discuss next.

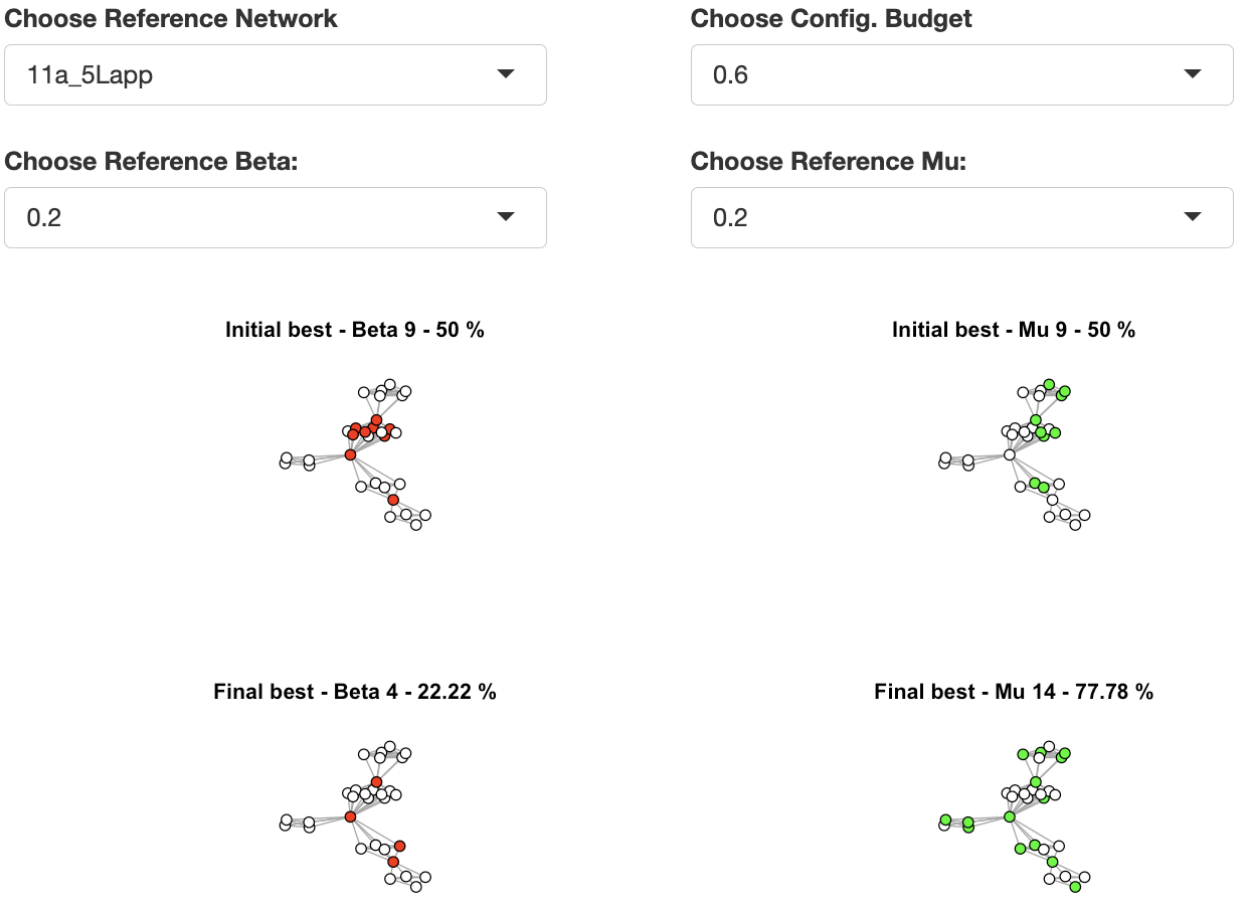


Figure 18: A GA algorithm proposes a better distribution of the improvement capacity on a simple network, with focus on choke-points and more budget spent on Mu when it makes more sense (higher Δ_μ)

4.2. Varying efficacy for different parameters

Let us recall the concepts of β, μ:

- Assigning a "better resistance to infection" to a node shall represent in the simulator the real-world act of e.g., "patching" a given software component, deploying new rules to an antivirus, etc. We therefore can decide to "increase the resistance", or in other words *decrease the probability of infection* — which will be achieved by applying an improved, lower, variable β, referred to as β_i, instead of a default or "base" beta β_b —, of a node or set of nodes, by spending effort towards the protection of said set of nodes. In human infectious processes, we would be talking in terms of *vaccination of a node*.

- Similarly, we could increase the probability of detecting early (and then fixing) an infection on a given node of the network. This would correspond to such acts of ensuring faster reaction times (augmenting the probability of cleanup from a base μ_b to a better/improved μ_i) through better monitoring or any similar measure. The comparison, for human infectious processes, would be that of a "cure" that eliminates the infection of a node and the risk of contagion by said cured node. In SIS simulations, this does *not* imply vaccination though, as a node (computer/person) can get reinfected.

Although only as a guideline, one shall realize that modifying the μ of a certain node *by a certain proportion* is not equivalent to modifying the β for that same node *in the same proportion*. That is, the S->I transition (that affected by β) is calculated considering all potentially infected neighbors of a given node, while the I->S transition is a probability affected only by the parameter μ locally to a node itself. Varying values of β and μ are both related to the final infection prevalence ρ_f that we intend to modify, but not in the same way. "Improved" μ_i (i.e., acting to clean faster a given node) of course has a local impact, but it is partial compared to "improved" β_i that can reduce the chance of infecting *a number of neighbours* (related to the degree of a chosen node). Suffice to say that to expect the exact same behavior out of tweaking in the same proportion both aspects is not realistic, and that will be reflected in later visualizations that will show how the infection prevalence surfaces are not symmetric (e.g. for instance, higher ρ_f for higher β_b). Starting with an initial prevalence of infection ρ on 20% of the nodes (fixed value), for a given network, we will ask the GA to find a better than random distribution of limited resources to ensure a minimal (local optimum) prevalence ρ_f. This is achieved after running the SIS simulation for several steps and several times (Monte-Carlo setup), for several possible distributions (the search space).

As per why consider a simulation instead of real-world testing, the Project overview above will have underlined the ideas of testing the effects of different budget spending towards protecting a network from infection, something that can hardly be done in the real world, at scale and with all the (unacceptable) risks and costs of such an approach.

4.3. How the GA makes its choices

In order to explain the recommendations of our product, we here proceed to analyse the GA choices. How to read the following visualizations:

- We present in red the nodes/genes which, in average across the top 20 individuals of a given generation, have most been chosen to be assigned an improvement.
- In white, the genes/nodes that have been *less* chosen to be assigned an improvement.
- To facilitate coherent visualizations across different parameters for a given network, we choose to sort the nodes *from left to right by decreasing page-rank centrality* score.

We separate the assignation of β in half of the visualizations (first), and μ in the other half (second), presenting them separately in two similar but independent visualizations. We then focus on the GA choices *per generation*. To represent the progress of the GA per generation, with the average assignation per node across the population of a given generation color coded as explained above, we order the generations from top (first generation) to bottom (last generation). E.g., as in the first generation the budget distribution is made at random, the first line (visually difficult to distinguish if too many generations have been needed) is orange-yellow across all nodes/genes. Both β and μ improvement assignations over time are to be studied together, and in fact both visuals are vertically aligned to facilitate comparison of nodes choices for both "actions".

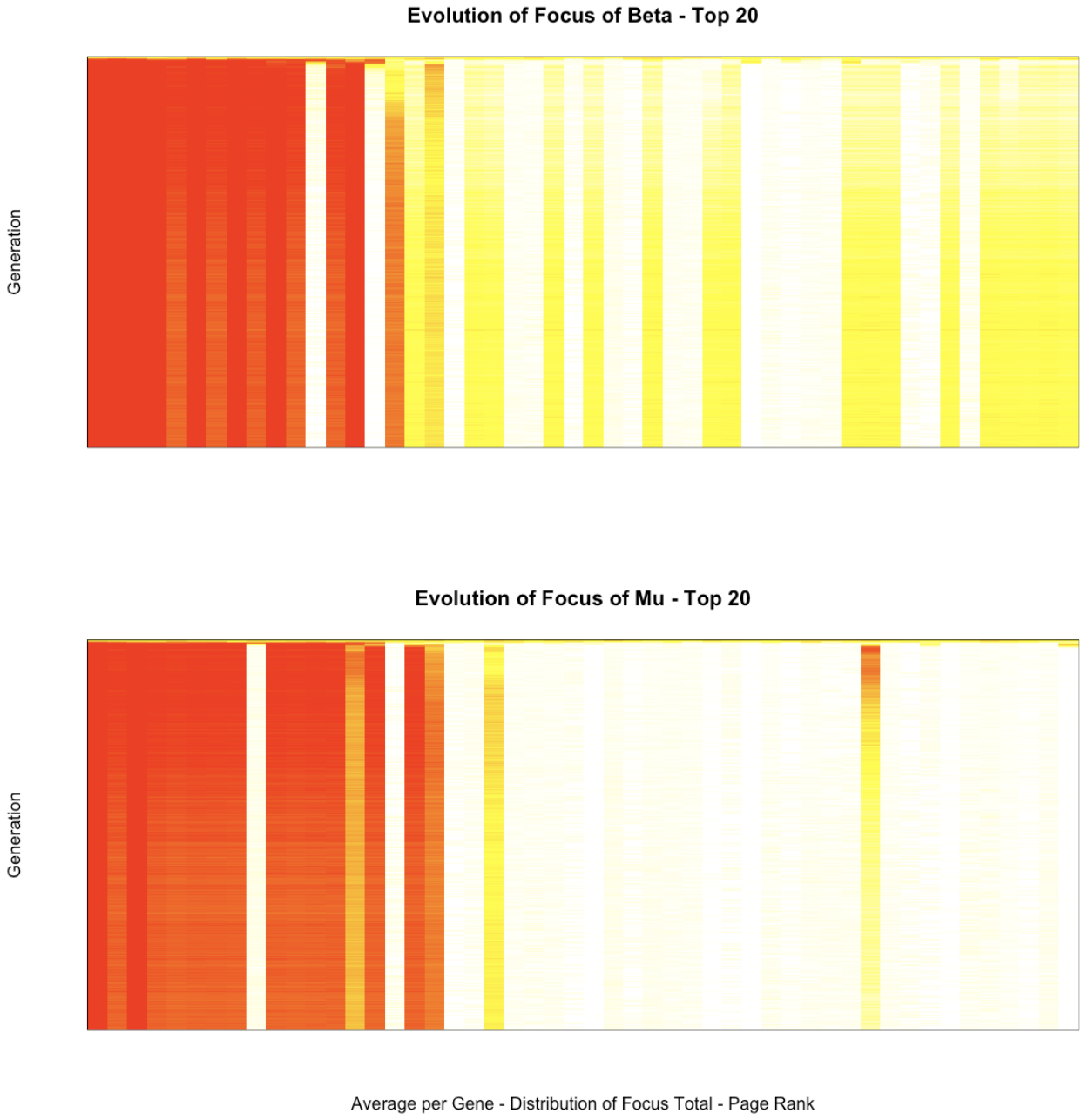


Figure 19: GA choices — BA network, 50 nodes — Progress for a given configuration: Budget 0.6, Base Beta 0.2, Base Mu 0.2 — Observe focus on central nodes (left) and repartition of budget across Mu and Beta

Let's now try to understand the results on such an example.

The visuals (e.g., Figure 19) appear red mostly on the left side, indication that the budget was spent mostly on central nodes (as per their Page-rank). In this setup, we have a low infection and low detection configuration, $\beta_b = \mu_b = 0.2$, for a relevant budget of 0.6. We can therefore conclude that there is value in recurring to μ_i, as $0.15 = \beta_b - \beta_i = \Delta_\beta < \Delta_\mu = \mu_i - \mu_b = 0.75$. Acting on β provides less value (but as will be discussed later on, these parameters are not symmetric).

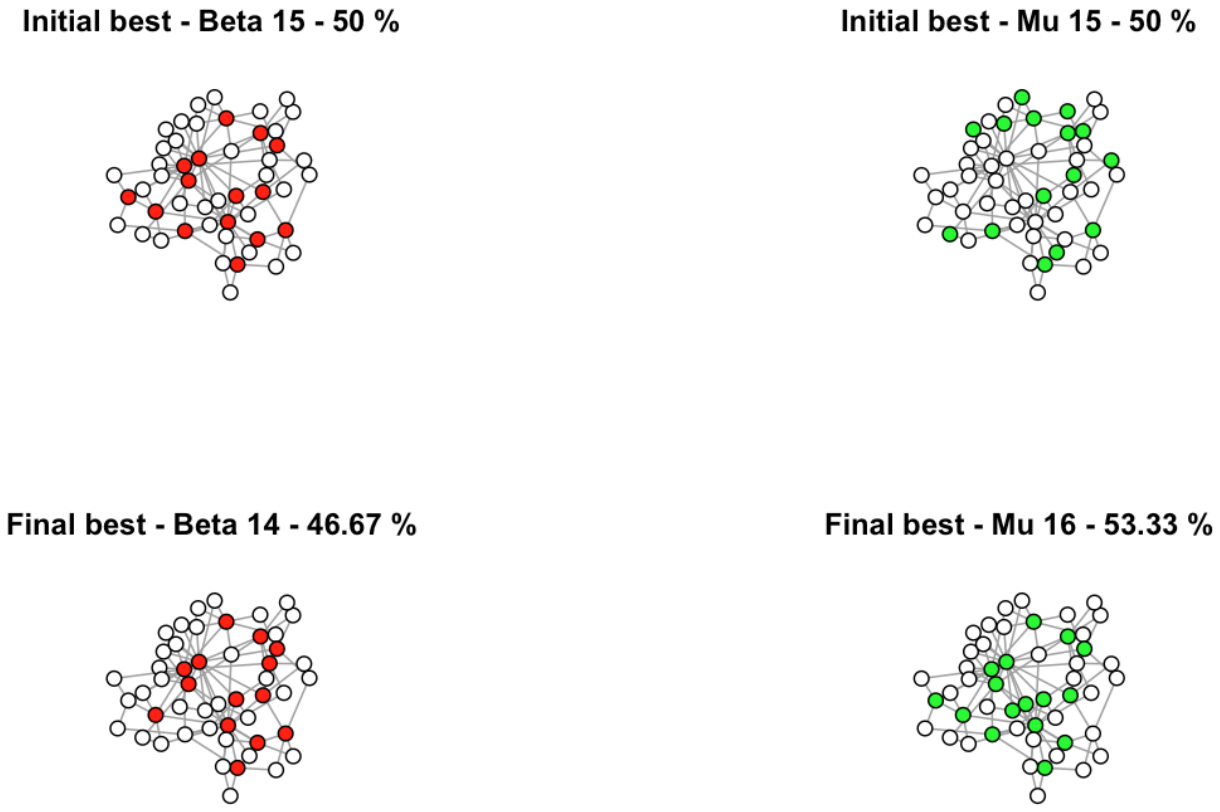


Figure 20: GA choices — BA network, 50 nodes — Repartition of budget across Mu and Beta

And with such a budget, for a BA network, we can obtain great recommendations (Figure 21).

Let us compare now with an equivalent configuration, but this time with *less* impact of assigning more of the budget to detection (see Figure 22):

$$0.85 = \beta_b - \beta_i = \Delta_\beta > \Delta_\mu = \mu_i - \mu_b = 0.05$$

Observe in Figure 23 how the algorithm indeed dedicates most of the available budget to improved β_i. As a reminder, the GA does not necessarily choose the best global optima, so the *strict* discussion from a purely theoretical stand-point is not warranted.

As a supplementary note, in this case the progress is equivalent, however the number of generation required to provide the recommendation by the GA is much lower, as little progress was found after a few initial generations. Please remember that the whole process is stochastic in nature on so many levels (i.e., involves random numbers generation and probabilities, and hence is not deterministic). This is to say that results across different tests could differ for the same configurations.

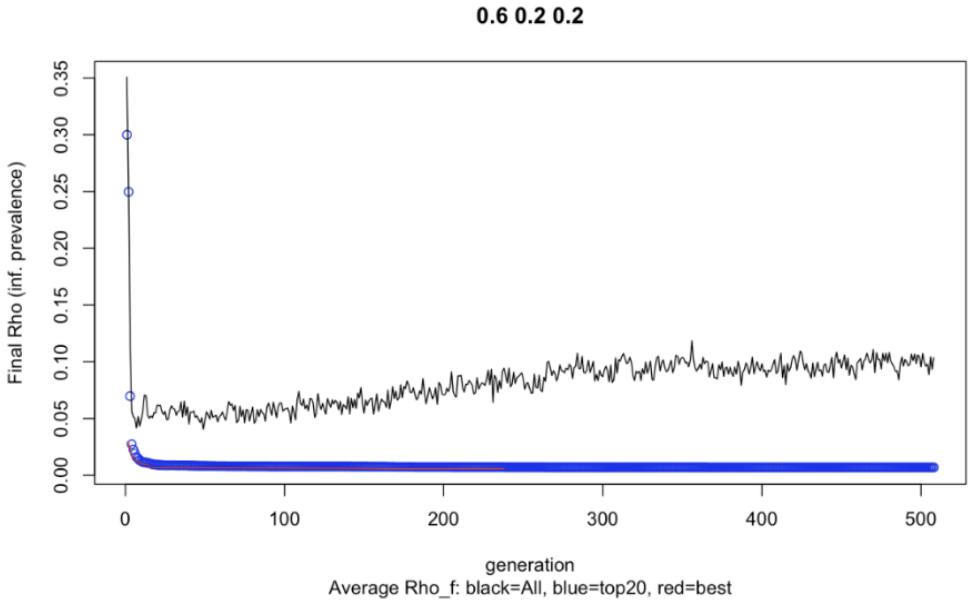


Figure 21: GA progress — BA network, 50 nodes — value compared to random budget assignation

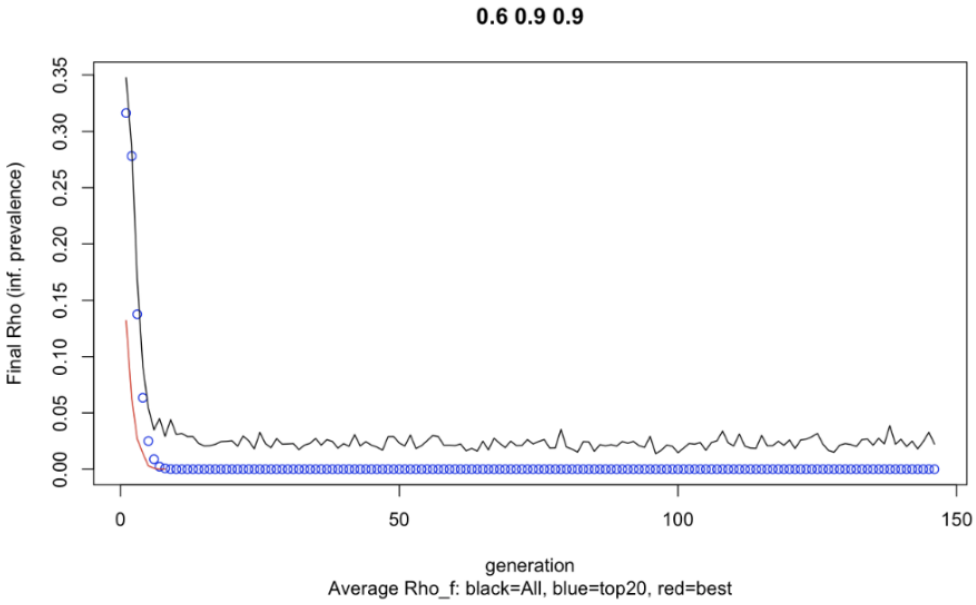


Figure 22: GA progress — BA network, 50 nodes — Per chance, fewer iterations, comparable results

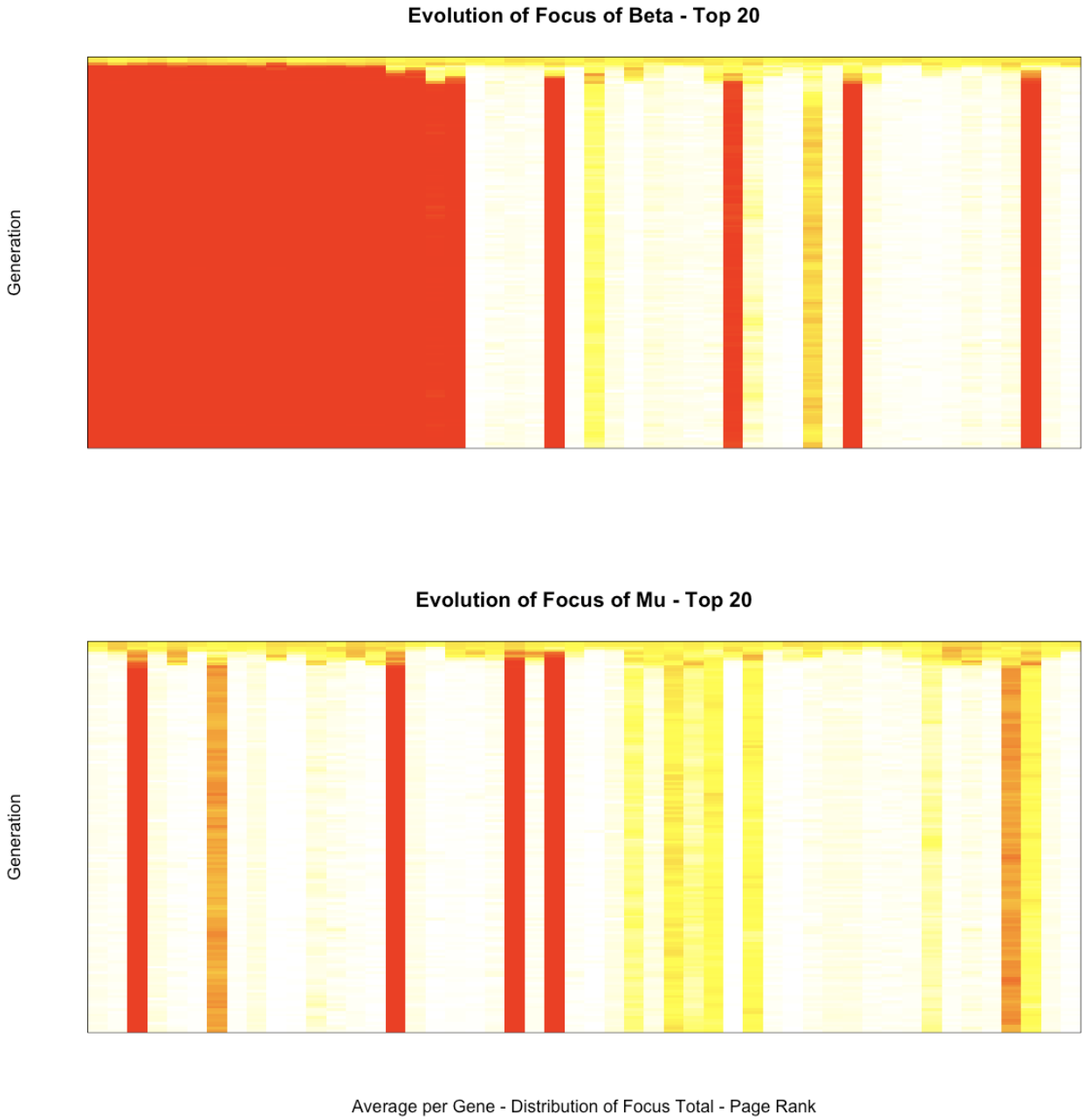


Figure 23: Lower value of detection (smaller Δ_μ), hence more focus on protection than detection

4.4. Impact of budget

Let's first understand that with a low budget, i.e., being capable of protecting or "cleaning faster" very few nodes, *there is only so much improvement one can achieve* in some cases.

It stands to reason that with higher budget, the overall prevalence of infection (post simulations), ρ_f should be lower (but this applies pre-simulation as well), which (thankfully) was very well confirmed, in various settings and for various networks.

Let's observe an example.

In this section and the next we use specifically created visualizations, in order to present the results in a comprehensive way. We compare:

- The random assignation final infection prevalence ρ_f (red-left below).
- To the same prevalence after using the GA to choose a better distribution of the budget (where to assign β_i, μ_i) (blue-center).
- Finally, we show the corresponding percentage of improvement (green-right).

This is shown from left to right, for a given network and a given budget, for all chosen combinations of base β_b, μ_b. Observe the effect on the efficacy of reducing ρ_f between figures 24 and 25.

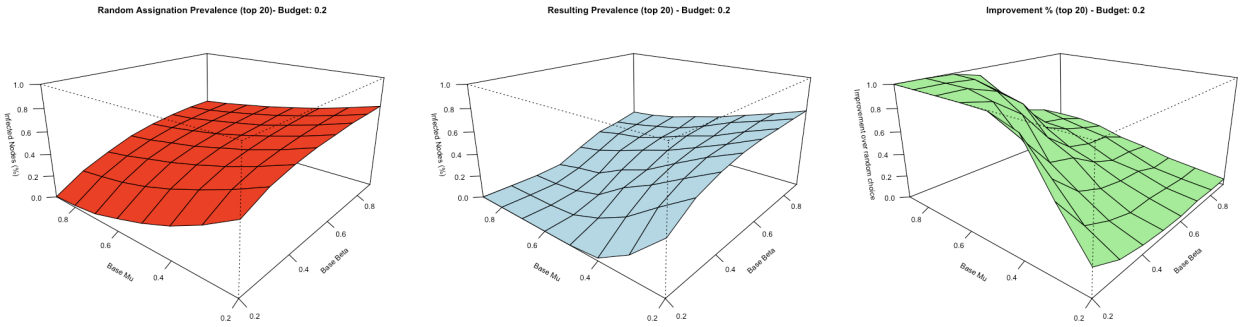


Figure 24: GA Progress: Barabási-Albert Network simulation with preset budget of 0.2

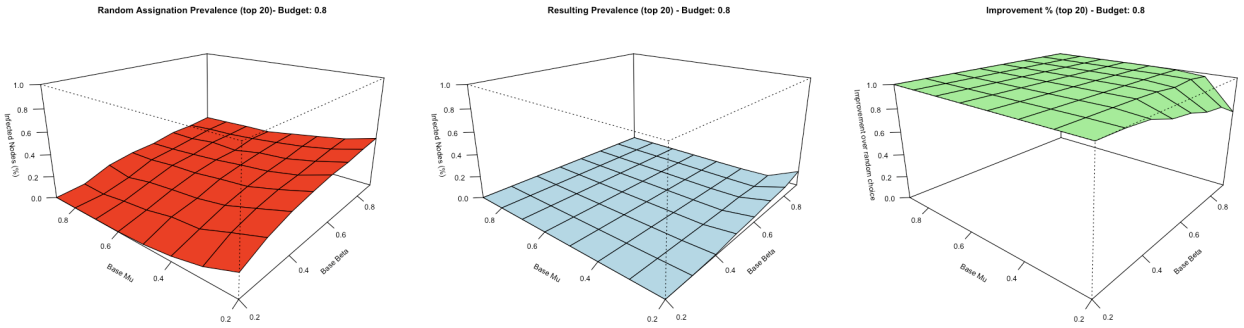


Figure 25: GA Progress: Barabási-Albert Network simulation with preset budget of 0.8

4.5. Impact of Topology

Note that in certain configurations of variables, and so far observed in particular for trees, rings and lattices, our GA *fails to improve* on a random proposal. It only makes sense that for a full network (K_n), as for any regular graph (i.e., where all nodes have the same degree k), it makes no difference whether to choose one node or another to protect it from infection, the overall prevalence will not change.

In the tests ran so far, a tree (and star) distribution tends to have an easier time reaching $\rho_f \approx 0$ after the transition phase, and hence little can be done by the GA to improve on the random assignation results.

Things appear to work better for manually created "segmented networks", as well as scale-free inspired networks. We also observe, as expected, better improvement for Scale-Free (BA) than Random (ER) networks.

As the origin of the idea was to work towards a better control (and hence better understanding) of infection spreading on "Cybersecurity-appropriate" IT Networks, it is to be noted that such networks with "subnets" of a few (or many, varying) nodes, i.e., similar maybe to the concept of "communities" or "clustering", might still very much benefit from this simulator, as they are *not* (a priori) "regular".

It appears that negative assortative mixing plays a role in ensuring good results, which is relevant for our context use-case. In fact, this is a key theoretical validation of our project here, as layed out here:

"On the other hand, technological and biological networks typically show disassortative mixing, or disassortativity, as high degree nodes tend to attach to low degree nodes [...] The properties of assortativity are useful in the field of epidemiology, since they can help understand the spread of disease or cures. For instance, the removal of a portion of a network's vertices may correspond to curing, vaccinating, or quarantining individuals or cells. Since social networks demonstrate assortative mixing, diseases targeting high degree individuals are likely to spread to other high degree nodes. Alternatively, within the cellular network—which, as a biological network is likely dissortative—vaccination strategies that specifically target the high degree vertices may quickly destroy the epidemic network." ("Wikipedia: Assortativity," n.d.)

However the relationship is not that simple. (Hird, Koelle, and Kolev 2013) In our particular tests, with high assortativity and high clustering, we also seem to obtain good results.

4.5.1. Watts-Strogatz

This small network implementation presents very "poor results" overall, as will be shown by comparison with the results on other networks.

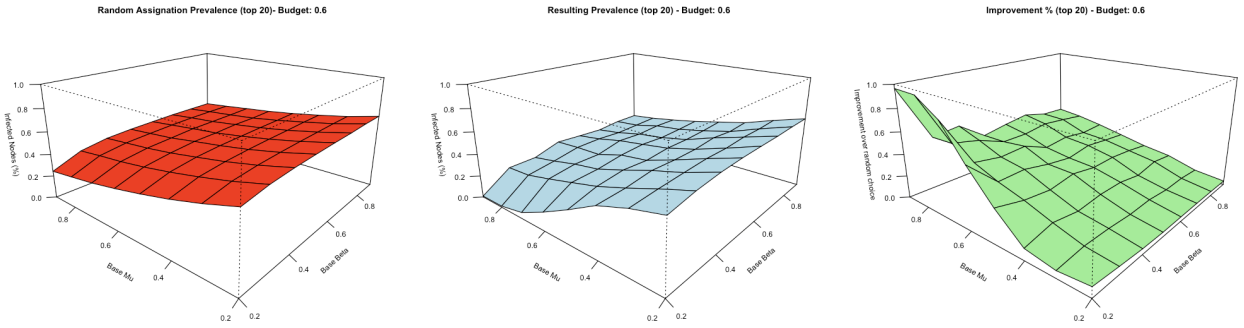


Figure 26: GA Progress: Watts-Strogatz Network simulation with preset budget of 0.6

Please note here how the improvement of a better budget allocation is almost irrelevant, *except* — in proportion — for very low β_b and high μ_b, i.e., in a situation that is already pretty much the best possible by

default (see already low prevalence overall in the left corner on the red and blue diagrams). This "lower-left" corner of the green visualization above will be observed across essentially all the configurations.

4.5.2. Erdős-Rényi

Although the ER model is not in fact as easily found in real World as a scale free network (Barabási 2014), it is a useful baseline for validation of the approach. Improvements observed with an ER network are not anywhere near as dramatic as can be expected, given our understanding: The importance of the *choice of the vertices* for improvement, if *no vertex is different from the others*, makes it harder to find any improvement at all.

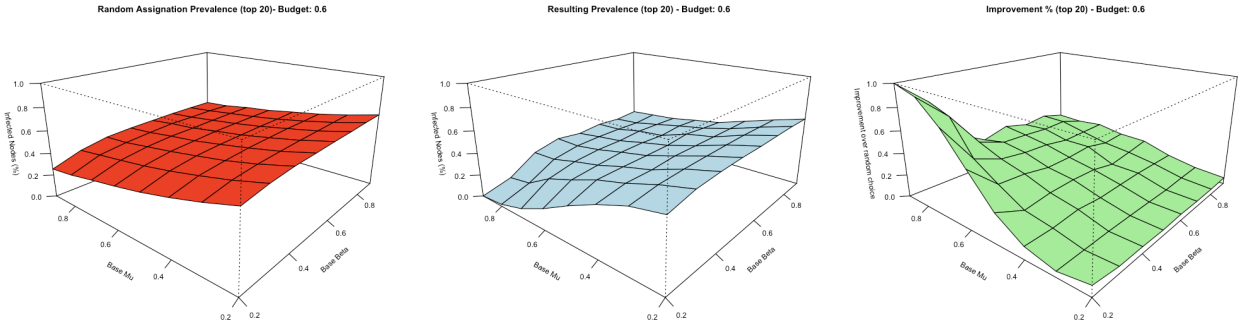


Figure 27: GA Progress: Erdős-Rényi Network simulation with preset budget of 0.6

These results are almost identical as those found for the "Small-World Network" above.

4.5.3. Barabási-Albert

This network shows good results compared to the above two, and we use it as a reference (alongside our simplistic "IT Network"), during this Project. With the same budget, almost all configurations (but the worst with too high β), bring the final infection prevalence to 0, containing completely the virus.

A perfect budget corresponds to the ability to deploy (effective) protection measures on all nodes (e.g., almost perfect vaccines), and ensuring a prompt detection and cure for *all* nodes. Here we observe almost similar results, with only 30% of the "perfect budget", for most configurations of β_b, μ_b.

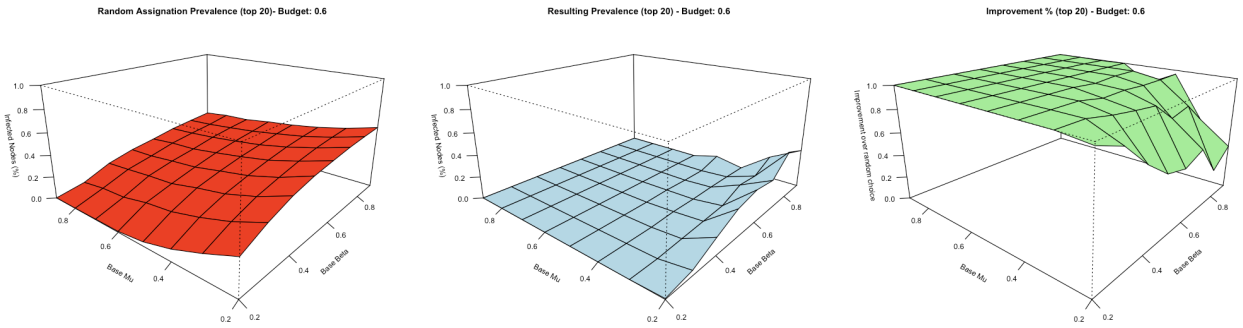


Figure 28: GA Progress: Barabási-Albert Network simulation with preset budget of 0.6

4.5.4. Particularly Disassortative Network

The results above, when considering the different statistics and results observed for different graphs, pointed us towards the assortativity as a potentially key factor for the efficacy of our product. For validation of the intuitions and the theoretical results, we hence proceeded to generate a network specifically chosen to have a rather *highly negative assortativity*. Results turned out to be the best found so far.

Note: To obtain such a network, we asked the igraph library for a few samples until it generated one network with such a highly disassortative network. The code for this is along these lines:

```
g <- sample_pa(50, power=2, directed = FALSE)
assortativity(g, types1 = V(g))
```

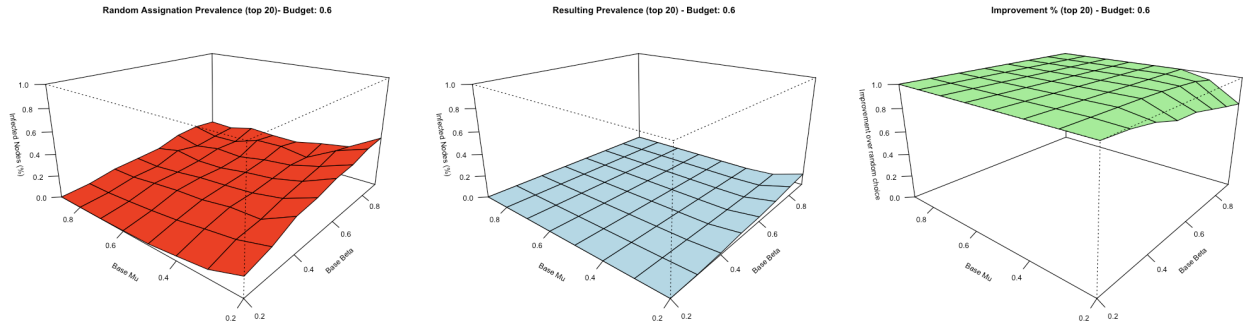


Figure 29: GA Progress: Disassortative Network simulation with preset budget of 0.6

5. Conclusions

Through the use of tools taken from Simulation, Complex Networks and Meta-Heuristics, we implemented a product that can give us a recommendation on how to *better use a limited budget to protect an IT Network*. Such recommendations could be used, alongside other decision support methods, to improve for example the protection of corporate IT networks.

Given the high number of operations required for our computer to reach its conclusions, we have chosen for this present Master's Thesis rather *small* networks, as imposed by processing times. Even so, we have prepared an implementation that scales horizontally, and is locally optimized through a mix of interpreted and compiled functions in our code. An "MMCA" alternative was implemented as well, and proved valuable for our Project, as it reaches comparable recommendations, for a fraction of the processing steps required by the original MC approach.

Even acknowledging that the GA is a meta-heuristic algorithm bound to provide "local optima" instead of a global optimum, the results were shown to be consistent with theoretical results from literature, and logic. Our product proves only useful for networks with irregularities and a minimal structure: it does not help for regular networks, and indeed is next to irrelevant for "small world" networks, which could be deemed a limitation. But this is expected: For instance, nodes centrality measures and network assortativity play a role in the GA recommendations, aligned with pre-existing work. This also validates the functionality of our product.

Analyzing relationships of nodes in a simulated computer networks *can* help propose better-than-random choices of which servers to patch more urgently, or which to monitor more closely, for example. Different recommendations, e.g., sometimes preferring to spend more effort towards monitoring than towards protecting certain nodes, are overall consistent with the expected value of each of these actions.

Finally, an interesting conclusion — that might not necessarily seem obvious without the use of our code which rather uniquely considers *two possible actions per node* — a conclusion observed time and again, was that with limited resources, it is apparently best, after taking care of key "choke-points", to *spread the efforts* and so not concentrate all possible efforts on fewer nodes. We wouldn't have necessarily come to this conclusion if it weren't for this project.

6. Assumptions and Limitations

The overarching concept of the Project is that of simulating IT networks and methods of protecting and/or improving detection of attacks on them. Although not to be taken at all seriously, a simile easy to understand for a Cybersecurity Employee would be that of the "WOPR machine" (https://youtu.be/iRsycWRQrc8?feature=shared&t=20).

A first difficulty in the definition of the Project was to *limit its scope*. The conceptual gap from human epidemiology to IT networks worm-like infections is great. For one reason, things in computer are mostly binary, either a system is vulnerable or it is not (i.e., either it runs a given Software package in a given version and configuration, or it doesn't, either its logs are monitored, or they are not...), while a simulation for spreading of a virus in the animal/human kingdom might consider the possibility to infect individuals with a certain probability. Similarly, the chosen implementation is a stretch from real-world IT Networks infections in that, if nothing else, it could be surprising to consider that if a given computer is cleaned, it could immediately get re-infected (which is indeed one assumption of the chosen SIS simulation). All the above — and much more — is valid criticism towards the Project described in this report, indeed. As it turns out, a common recourse to these discussions could be George Box's aphorism: "All simulations are wrong. Some are useful." (https://en.wikipedia.org/wiki/All_models_are_wrong). Many of such considerations were discussed at some length in the course of this report.

Second to the above critics might be that of the creation of a "representative network", with "applications" or software components, on different distributions of servers, interacting with one another. This is easier to answer, as one could in principle design such a graph that perfectly represents the IT communications networks, each edge representing a possible connection, available to a given server component, to other servers' components. Whether or not the networks used for this Project are representative of such networks is not a relevant matter, as per the main objectives, analysis and conclusions of the Project.

Given the above, we consider it acceptable to use a simple, "traditional" epidemiology simulator (SIS), although a SIR (or SIRS) model would probably have been more "realistic". Note that the choices of algorithms and implementations are not based necessarily on the "best possible simulation", or the "best possible optimization", but rather are somewhat simplistic, for two reasons:

- Simplicity will help better present and understand the overall Project concepts, its analysis and results.
- No simulation is perfect, and the goal of the Project is only to show that one *can* make better decisions, with certain constraints, using such concepts as simulation and Operational Research, as decision *support* tools, to be *added* to whatever toolbox is already in use by the Cybersecurity Operations Management team. Therefore the objective is absolutely *not* perfection.

7. Future Work

Our product generates simulations that are currently purposefully rather simplistic, so as to reach conclusions that could ideally be meaningfully analyzed while hoping to provide valuable conclusions. However, we made in the process many assumptions in simplifying our approach as was described early on, which might not hold true in a real-world scenario.

Several ideas could be studied to improve on the proposed work. We here present a few options.

SIRS: The SIS model is simplistic, and was used for instance to simplify calculations for the MMCA approximations. However we have assumed that infection detection would not entail any improvement to the protection of the infected nodes. This is generally not true in reality and a certain amount of immunity against re-infection of a cleaned node would reasonably be assumed. A SIRS ("Susceptible-Infected-Recovered-Susceptible"), whereby cleaned nodes are temporarily immune to re-infection could be instead implemented. Similarly, when an infection is detected in one node, a normal response in IT would be to review all neighbor nodes for infection, which could also be encoded in our product.

Varying default detection capability: Another important parameter that directly affects the μ_b parameter is the concept of IT inventory. In complex IT environment, it is sometimes realistic to assume that *not all* nodes are in fact known, and hence the capacity for detection on unknown nodes would be greatly decreased, compared to the detection capacity for well-known nodes. This could be encoded in our product by creating additional nodes with very low μ_b and for which our GA implementation would *not* be able to assign an improved detection (μ_i). Simulations of such scenarios would clarify the importance and role of what is commonly knonw correct/complete "Configuration Management Databases".

Varying default susceptibility depending on node types: In approximating an IT network of interconnected components, not all "nodes" would necessarily have the same probability of getting infected: A "firewall node" could be expected to have less risk of getting infected, whereas a "Web Application Server Node" is expressely exposed and complex applications inherently present a higher attack surface. This could be encoded with varying β_b for instance. Clients and servers would also be modelled differently, for instance. And instead of the assimilation of a node to a "computer", we could assume different Software components, each of which have varying amount of vulnerabilities discovered over time, which could in turn be modeled as an input. The interactions of the varying vulnerability exposure would make for a greater level of details and added complexity. However such an approach would render the analysis of results much more complex than was reasonable for this Master's Thesis.

Varying parameters based on Exposure: We assumed all nodes equal. A key aspect that is commonly considered when making decisions towards protecting certain nodes over others are their "exposure": We do not consider an Internet-facing server to be equally at risk as a server that can only be accessed by authorized employees from within a network perimeter. In that sense, our simulations *could* in principle assign a higher β_b for "Internet Exposed" nodes, and lower for internal-only nodes.

Attackers perspective enhancement: We assumed a 20% initial infection prevalence ($\rho_0 = 0.2$), and simple dynamics of infection spreading. This meant to cover for the figure of our "attackers". We then focused

our work solely towards defensive decisions (using for inspiration the NIST Framework, as noted early on). Other works have used for example reinforcement-learning to simulate the interactions of a more elaborate attacker and the corresponding response from a defensive perspective. There are frameworks oriented towards describing observed attackers behaviours (e.g. the MITRE ATT&CK framework). We could also model an attacker which could prefer a "directed attack" rather than a "spreading attack" (which does not usually apply to biological epidemiology). Other works indeed have taken the attacker into account (Michael D. Adams 2013).

Further improvement to our Meta-Heuristic: Our advisor suggested that a "Local Search Optimization" with faster algorithms could reduce processing times. Our GA in our (small) simulations usually converges fast towards a local optimum, and then keeps looking in the corresponding area. Once the "area" is defined however, whereby a few nodes are chosen to be assigned an improved β_i and others a μ_i, variations with a fixed subset of these optimizations might prove to converge faster.

8. References

Adler, Joseph. 2009. *R in a Nutshell*. O'Reilly.

Alex Andrew, Joshua Collyer, Sam Spillard. 2021. "Developing Optimal Causal Cyber-Defence Agents via Cyber Security Simulation." 2021. https://arxiv.org/pdf/2207.12355.pdf.

Ashby, W. Ross. 2015. *An Introduction to Cybernetics*. Martino Publishing.

Badham J, Stocker R. 2010. "The Impact of Network Clustering and Assortativity on Epidemic Behaviour." Theor Popul Biol. 2010. 2010. https://pubmed.ncbi.nlm.nih.gov/19948179/.

Barabási, Albert-Lázló. 2014. *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Basic Books.

Cortez, Paulo. 2021. *Modern Optimization with r (Second Edition)*. Springer. https://doi.org/10.1007/978-3-030-72819-9.

"Cybersecurity Game-Change Research & Development Recommendations." 2010. 2010. https://www.nitrd.gov/documents/cybersecurity/documents/CSIAIWGCybersecurityGameChangeRDRecommendations20100513.pdf.

Enrique J. Carmona Suárez, Severino Fernández Galán. 2020. *Fundamentos de La Computación Evolutiva*. Marcombo, S.L.

Erik V. Cuevas, Diego A. Olvida, José V. Osuna. 2017. *Optimización - Algoritmos Programados Con Matlab*. Marcombo, S.L.

Gómez, Sergio & al. 2010. "Discrete-Time Markov Chain Approach to Contact-Based Disease Spreading in Complex Networks." EPL Journal. 2010. https://webs-deim.urv.cat/~sergio.gomez/papers/Gomez-Discrete-time_Markov_chain_approach_to_disease_spreading_in_CN.pdf.

Hird, John, Rainer Koelle, and Denis Kolev. 2013. *Towards Mathematical Modelling in Security Risk Management in System Engineering*. https://doi.org/10.1109/ICNSurv.2013.6548565.

J. García, A. Berlanga, J. M. Molina. 2018. *Ciencia de Datos - Técnicas Analíticas y Aprendizaje Estadístico*. Altaria.

Kim, Jonghyun, Sridhar Radhakrishnan, and Jongsoo Jang. 2006. "Cost Optimization in SIS Model of Worm Infection." ETRI Journal. 2006. https://doi.org/10.4218/etrij.06.0206.0026.

La, Richard J. 2016. "Influence of Network Mixing on Interdependent Security: Local Analysis." IEEE 56th Annual Conference on Decision and Control (CDC). 2016. https://ieeexplore.ieee.org/document/7841901.

Leyffer, Sven, and Ilya Safro. 2013. "Fast response to infection spread and cyber attacks on large-scale networks." Journal of Complex Networks. July 2013. https://doi.org/10.1093/comnet/cnt009.

Michael D. Adams, Bruce Hoy, Seth D. Hitefield. 2013. "Application of Cybernetics and Control Theory for a New Paradigm in Cybersecurity." 2013. https://arxiv.org/pdf/1311.0257.pdf.

Mitchell, Melanie. 2011. *Complexity a Guided Tour*. Oxford University Press.

Neil Dhir, Niall Adams, Henrique Hoeltgebaumb. 2021. "Prospective Artificial Intelligence Approaches for Active Cyber Defence." 2021. https://arxiv.org/pdf/2104.09981.pdf.

Newman, Mark. 2018. *Networks (Second Edition)*. Oxford University Press. https://doi.org/10.1093/oso/9780198805090.001.0001.

Noam Goldberg, Ilya Safro, Sven Leyffer. 2012. "Optimal Response to Epidemics and Cyber Attacks

in Networks." 2012. https://www.researchgate.net/publication/266874694_Optimal_Response_to_Epidemics_and_Cyber_Attacks_in_Networks.

Peter Grandits, Vladimir M. Veliov, Raimund M. Kovacevic. 2019. "Optimal Control and the Value of Information for a Stochastic Epidemiological SIS-Model." 2019. https://www.sciencedirect.com/science/article/pii/S0022247X19303129.

Sayama, Hiroki. 2015. *Introduction to the Modeling and Analysis of Complex Systems*. Open SUNY Textbooks, Milne Library.

Shaffer, Stephen. 2023. "Flipping the Vulnerability Management Model: CVSS to SSVC." 2023. https://stephenshaffer.io/flipping-the-vulnerability-management-model-cvss-ssvc-aaa78f1426e1.

Sylvain P. Leblanc, Ian Chapman, Andrew Partington. 2011. "An Overview of Cyber Attack and Computer Network Operations Simulation." MMS '11: Proceedings of the 2011 Military Modeling & Simulation Symposium. 2011. https://www.researchgate.net/publication/220953920_An_overview_of_cyber_attack_and_computer_network_operations_simulation.

Venables, Phil. 2022. "Defense in Depth." 2022. https://www.philvenables.com/post/defense-in-depth.

Wagner, Neal, Cem Ş. Şahin, Michael Winterrose, James Riordan, Jaime Pena, Diana Hanson, and William W. Streilein. 2016. "Towards Automated Cyber Decision Support: A Case Study on Network Segmentation for Security." In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–10. https://doi.org/10.1109/SSCI.2016.7849908.

"Wikipedia: Assortativity." n.d. https://en.wikipedia.org/wiki/Assortativity.

Annex I: GA: Code improvements

Our initial version of the code had a very poor tournament implementation, without possibility to modify the tournament pressure for parents selection. Many low value parents would participate in the reproduction process: random pairs without replacement were generated from the complete parents pool, and the best of each pair would become parents. That left only half of the parents to participate of the reproduction, but also, for any given pair, two "bad" parents could compete, and the less bad would be selected.

Although not technically incorrect, it was a rather poor way to implement the GA's parents-selection step.

Moreover, in said former version, some children were generated above budget and hence had to be "pruned" out of the list of candidate individuals, greatly reducing the efficacy of the GA. This has been fixed in the latest versions of the product.

Annex II: The concept of "Operational Cybersecurity Policies"

We call them policies for short. To keep things simple, this is in essence what we can act upon in our simulator, per node: How much "protection" do we assign to each node of a given network (or subset thereof), and how much "detection and cleanup". As an illustration of what would happen on a somewhat more realistic network, we assign different protection and detection capacity to each node based on its "type", simply assigning fixed values per node type. In the actual implementation, a first parameter set encoding could (did) look like so (simplistic example):

```
## ====================
## Monitoring policy: How "fast" do I detect infection of a given node?
## ====================
nodes_mus[sec_nodes] <- 0.9
nodes_mus[app_frontend_nodes] <- 0.95
nodes_mus[app_db_nodes] <- 0.8

## ====================
## Protection policy: How "resistent" is a given node to infection?
## ====================
nodes_betas[sec_nodes] <- base_beta / 40 # Super Low infection for Security nodes
nodes_betas[app_backend_nodes] <- base_beta / 3 # Low infection for backend
nodes_betas[app_db_nodes] <- base_beta / 10 # Very Low infection for DB

# ## Initial population infected at 20%, but NOT the security nodes.
# candidates <- c(app_frontend_nodes, app_backend_nodes)
## Initial population of Front-End Servers infected at 20%.
candidates <- app_frontend_nodes
```

Figure 30: Simple Operational Cybersecurity Policy

See a summary of the progress at this stage here: https://www.kaizen-r.com/2023/10/Project-log-day-11-simplistic-policies/. A more advanced concept that inspire our simplistic "Policies" could be that of a "Security Posture", which could be somewhat partially derived from such "Policies": https://securityscorecard.com/blog/what-is-a-cybersecurity-posture/, https://www.bitsight.com/glossary/cybersecurity-posture. In concept, with the above we also show that, *node type aside*, one could assign, with the generated code so far, manually to each node its own β and μ value. *From this point on in time, we simplified further the approach by assuming all nodes to be of the same type.*

Annex III: Technology stack

Running environment

- Laptop: Macbook Air M1 (aarch64-apple-darwin20 (64-bit)).
- From December 2023 approximately, a MiniPC (AMD Ryzen7) with 8 Cores (2 threads per core) was added to our setup to run more simulations in parallel.
- IDE: RStudio Version 2023.06.0+421. Later updated to RStudio Version 2023.12.1.

Report Generation

Testing Quarto (instead of our traditional .Rmd) from RStudio was finally discarded, and the more traditional RMarkdown was used in the end.

Notes: Also learning at the same time how to use Pandoc/bib files for references... Which of course requires more complete LaTeX components — more than we ever needed in the past that is —, and so we have to search and get... MacTeX (8.6GB of Disk space use, *for a 5.4GB pkg file*): https://tug.org/mactex/mactex-download.html. Then of course we face plenty of problems... For instance wrong Latex package is used (still 2022 when we just installed 2023), and so we follow this: https://stackoverflow.com/questions/33650869/how-can-i-set-the-latex-path-for-sweave-in-r. Nothing worked, and after a few hours we switched back to .Rmd (Markdown) to at least get started.

Programming Languages

- R version 4.3.1 (2023-06-16) – "Beagle Scouts" (Later was updated to 4.3.2)
- XCode for C++ components compilation, but mostly Rcpp to call on C++ auxiliary functions.

Annex IV: Timeline of the Project through the blog

Following the recommendations of our thesis advisor, we have made regular notes of our progress (and sometimes lack thereof). We have also published some of these notes on our personal blog explain some more, see https://www.kaizen-r.com/category/master-thesis/.