Universitat Autònoma de Barcelona

SCALABLE AGENT-BASED MODEL SIMULATION USING DISTRIBUTED COMPUTING ON SYSTEM BIOLOGY

Ghazal Tashakor

SCALABLE AGENT-BASED MODEL SIMULATION USING DISTRIBUTED COMPUTING ON SYSTEM BIOLOGY

Universitat Autònoma de Barcelona

Presented to
The Academic Faculty

By

Ghazal Tashakor

In Partial Fulfillment
of the Requirements for the Degree
Doctorate in Computer Science in the
School of Engineering
Department of Architecture of Computers and Operating Systems

The Autonomous University of Barcelona

September 2020

SCALABLE AGENT-BASED MODEL SIMULATION USING DISTRIBUTED COMPUTING ON SYSTEM BIOLOGY

Approved by:

Dr.Remo Suppi, Advisor
Department of Computer Architecture and Operating Systems
The Autonomous University of Barcelona

A behavioral approach is critical when developing practical AI systems in general. The result might be a system that has the desired behavior but is hard to explain. In that case it would be fair to say that the software system is similar to a human in that both cannot be explained by a simple model.

Noam Chomsky

Dedicated to her memory

ACKNOWLEDGEMENTS

I would like to express my special appreciation and thanks to my advisor and my parents

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

SUMMARY

Agent-based modeling is a very useful computational tool to simulate complex behavior using rules at micro and macro scales. This type of modeling's complexity is in defining the rules that the agents will have to define the structural elements or the static and dynamic behavior patterns.

This thesis considers the definition of complex models of biological networks that represent cancer cells obtain behaviors on different scenarios by means of simulation and to know the evolution of the metastatic process for non-expert users of computer systems. Besides, a proof of concept has been developed to incorporate dynamic network analysis techniques and machine learning in agent-based models based on developing a federated simulation system to improve the decision-making process.

For this thesis's development, the representation of complex biological networks based on graphs has been analyzed, from the simulation point of view, to investigate how to integrate the topology and functions of this type of networks interacting with an agent-based model. For this purpose, the ABM model has been used as a basis for the construction, grouping, and classification of the network elements representing the structure of a complex and scalable biological network.

The simulation of complex models with multiple scales and multiple agents provides a useful tool for a scientist, non-computer expert to execute a complex parametric model and use it to analyze scenarios or predict variations according to the different patient's profiles.

The development has focused on an agent-based tumor model that has evolved from a simple and well-known ABM model. The variables and dynamics referenced by the Hallmarks of Cancer have been incorporated into a complex model based on graphs. Based on graphs, this model is used to represent different levels of interaction and dynamics within cells in the evolution of a tumor with different degrees of representations (at the molecular/cellular level).

A simulation environment and workflow have been created to build a complex, scalable network based on a tumor growth scenario. In this environment, dynamic techniques are applied to know the tumor network's growth using different patterns.

The experimentation has been carried out using the simulation environment developed considering the execution of models for different patient profiles, as a sample of its functionality, to calculate parameters of interest for the non-computer expert, such as the evolution of the tumor volume.

The environment has been designed to discover and classify subgraphs of the agent-based tumor model to execute these models in a high-performance computer system. These executions will allow us to analyze complex scenarios and different profiles of patients with tumor patterns with a high number of cancer cells in a short time.

RESUM

El modelat basat en agents és una eina informàtica molt útil que permet simular un comportament complex utilitzant regles tant a escales micro com macro. La complexitat d'aquest tipus de modelat està en la definició de les regles que tendran els agents per definir elements estructurals o els patrons de comportament estàtics i/o dinàmics.

La present tesis aborda la definició de models complexos de xarxes biològiques que representen cèl·lules canceroses per obtenir comportaments sobre diferents escenaris mitjançant simulació i conèixer l'evolució del procés de metàstasi per a usuaris no-experts en sistemes de còmput.

A més es desenvolupa una prova de concepte de com incorporar tècniques d'anàlisi de xarxes dinàmiques i d'aprenentatge automàtic en els models basats en agents a partir del desenvolupament d'un sistema de simulació federat per millorar el procés de presa de decisions. Per al desenvolupament d'aquesta tesi s'ha tingut que abordar, des del punt de vista de la simulació, la representació de xarxes biològiques complexes basades en grafs i investigar com integrar la topologia i funcions d'aquest tipus de xarxes interactuant amb un model basat en agents.

En aquest objectiu, s'ha utilitzat el model ABM com a base per a la construcció, agrupament i classificació dels elements de la xarxa i que representen l'estructura d'una xarxa biològica complexa i escalable. La simulació d'un model complex de múltiples escales i múltiples agents, proporciona una eina útil per a que un científic, no-expert en computació, pugui executar un model complex i paramètric i utilitzar-ho com a eina d'anàlisi d'escenaris o predicció de variacions segons els diferents perfils de pacients considerats.

El desenvolupament s'ha centrat en un model de tumor basat en agents que ha evolucionat des d'un model ABM simple i bé conegut, al qual se li han incorporat les variables i dinàmiques referenciades per l'Hallmarks of Cancer, fins a un models basat en grafs. Aquest model, basat en grafs, permet representar a diferents nivells d'interacció i dinàmiques dins de les cèl·lules en l'evolució d'un tumor que permet diferents graus de representacions (a nivell molecular/cel·lular).

Tot això s'ha posat en funcionament en un entorn de simulació i ha creat un flux de treball (workflow) per construir una xarxa escalable complexa basada en un escenari de creixement tumoral i on s'apliquen tècniques dinàmiques per conèixer el creixement de la xarxa tumoral sobre diferents patrons.

L'experimentació s'ha realitzat utilitzant l'entorn de simulació desenvolupat considerat l'execució de models per a diferents perfils de pacients, com a mostra de la seva funcionalitat, per a paràmetres d'interès per a l'expert no-informàtic com per exemple l'evolució del volum del tumor.

L'entorn ha estat dissenyat per descobrir i classificar subgrafs del model de tumor basat en agents, que permetran distribuir els models en un sistema de còmput d'altes prestacions per poder analitzar escenaris complexos i/o diferents perfils de pacients amb patrons tumorals amb un alt nombre de cèl·lules canceroses en un temps reduït.

RESUMEN

El modelado basado en agentes es una herramienta computacional muy útil que permite simular un comportamiento complejo utilizando reglas tanto en escalas micro como macro. La complejidad de este tipo de modelado radica en la definición de las reglas que tendrán los agentes para definir los elementos estructurales o los patrones de comportamiento estáticos y/o dinámicos.

La presente tesis aborda la definición de modelos complejos de redes biológicas que representan células cancerosas para obtener comportamientos sobre diferentes escenarios mediante simulación y conocer la evolución del proceso de metástasis para usuarios no expertos en sistemas de cómputo. Además se desarrolla una prueba de concepto de cómo incorporar técnicas de análisis de redes dinámicas y de aprendizaje automático en los modelos basados en agentes a partir del desarrollo de un sistema de simulación federado para mejorar el proceso de toma de decisiones.

Para el desarrollo de esta tesis se han tenido que abordar, desde el punto de vista de la simulación, la representación de redes biológicas complejas basadas en grafos e investigar como integrar la topología y funciones de este tipo de redes interactuando un modelo basado en agentes. En este objetivo, se ha utilizado el modelo ABM como base para la construcción, agrupamiento y clasificación de los elementos de la red y que representan la estructura de una red biológica compleja y escalable.

La simulación de un modelo complejo de múltiples escalas y múltiples agentes, proporciona una herramienta útil para que un científico, no-experto en computación, pueda ejecutar un modelo complejo paramétrico y utilizarlo como herramienta de análisis de escenarios o predicción de variaciones según los diferentes perfiles de pacientes considerados.

El desarrollo se ha centrado en un modelo de tumor basado en agentes que ha evolucionado desde un modelo ABM simple y bien conocido, al cual se le han incorporado las variables y dinámicas referenciadas por el Hallmarks of Cancer, a un modelo complejo basado en grafos. Este modelo, basado en grafos, se utiliza para representar a diferentes niveles de interacción y dinámicas dentro de las células en la evolución de un tumor que permite diferentes grado de representaciones (a nivel molecular/celular). Todo ello se ha puesto en funcionamiento en un entorno de simulación y se ha creado un flujo de trabajo (workflow) para construir una red escalable compleja basada en un escenario de crecimiento tumoral y donde se aplican técnicas dinámicas para conocer el crecimiento de la red tumoral sobre diferentes patrones.

La experimentación se ha realizado utilizando el entorno de simulación desarrollado considerado la ejecución de modelos para diferentes perfiles de pacientes, como muestra de su funcionalidad, para calcular parámetros de interés para el experto no-informático como por ejemplo la evolución del volumen del tumor.

El entorno ha sido diseñado para descubrir y clasificar subgrafos del modelo de tumor basado en agentes, que permitirá distribuir los modelos en un sistema de cómputo de altas prestaciones y así poder analizar escenarios complejos y/o diferentes perfiles de pacientes con patrones tumorales con un alto número de células cancerosas en un tiempo reducido.

CHAPTER 1
INTRODUCTION AND BACKGROUND

Systems Biology (SB), mainly at the cellular level, is a complex natural phenomenon today for research in drug development and biotechnological productions and applications. Mathematical models of cellular level are built-in bio repeated cycles at a portion of time to arrive at a decision.

The bio iterative cycles' primary idea is to develop systems by allowing software developers to redesign and translate cellular metabolism.

The objective is to bring the desired decision or results closer to discovery in each iteration. These experiments on a multiscale iterative cycle of computational modeling besides experimental validation and data analysis would produce incremental samples for the high-throughput technologies. Also, the cell's behavior emerges at the network level. It requires much integrative analysis, and due to the size and complexity of intercellular biological networks, the computational model should be an essential part of the production or application.

Systems Biology (SB) would not be needless of developing integrated frameworks for analysis and data management. As well as the intercellular level that many kinds of research in Systems Biology (SB) address, the cellular population matter too [48].

1.1 Agent-based modeling and Simulation

Agent-based modeling and simulation will be consummated by bringing up novelty in computing clusters or clusters on computational grids. This novelty is based on the number of interacting agents or the complexity of the model.

A novel complex system that has developed in this case needs to be understandable and adaptive for both novice or veteran, so it must be a package that does something beyond computing, such as a versatile framework that has APIs, GUIs, or IDEs and has its intuitive terminology [1].

Agent-based modeling allows developers to synthesize analyzed results between the human environment and computing system environment. This synthesizing help modelers to investigate and design more persistent patterns at different levels through time and resources space. ABM also prepares the dynamic of the system due to the interactions between agents and output data. ABMs are recently becoming more complicated and involved with several challenges.

Initializing and parameterizing get more and more difficult. Evaluation becomes hard, and finally, analyzing the multidimensional output data will not be accessible at all [2]. Quantifying the level of complicatedness about the models is a fundamental investigation for scientific researchers and developers to define the system's flexibility dynamically for future evaluation, especially when the final decision will be about human environments and their behavior space. Besides, modelers have increased their models' complications as fast as the data availability and computation power improvements. They never could justify the level of complicatedness that they have constructed [2]. The system, which has been called complex, shows particular characteristics ordered as follows [4]:

1. Interacting agents

2. Modifying agents' behavior space

3. Being open

4. Having emergent properties which appear without a central control

5. Showing both ordered and disordered behavior patterns

The behavior space pulls out from the simple relations between the agents in the bottom-up agent-based model approach. Each agent analyses its current situation and operates based on some algorithms, rules, and equations.

Large-scale behavior space in ABM depends on the number of agents in simulation and its effects on the system's reality level, constructed as a model. It may put the model validation in danger. In the fields such as biology, ecology, and social modeling, simulating realistic models contains agents being processed per time unit. This kind of large-scale simulations needs a high demand for computing power and resources [3].

For complex systems, modeling must start with specific questions to formulate a conceptual model that simulates the entire system's elements and processes. Using observed patterns for designing a model occurs at different spatial and temporal scales and different hierarchical levels because multiple patterns were vital to modeling spatiotemporal dynamics. This pattern-oriented theory development approach is increasingly used in agent-based complex systems [3].

Agent-based modeling (ABM) focuses on the rules and interactions among the individual components of the system in a virtual world, and it should be noted that a mathematical and computational model which has created based on agent-based modeling, concerning the mechanisms of the real-world example, lead us to the patterns of data. Therefore, ABMs start with the rules, mechanisms reconstruct through the mathematical or computational form, and observe data patterns. If the goal would be only finding patterns in an existing dataset, then agent-based modeling cannot be useful.

ABMs, in comparison to the other object-oriented rule-based modeling systems, seem more appropriate for grid-based and spatial nature models. Processing an individual agent's heterogeneous behavior within a dynamic population of agents that cannot be controlled by an overall controller needs higher-level system parallelism that ABMs supports. There are many available agent-based simulation frameworks, applications, and software like MASON [1], REPAST [2], FLAME [3], Swarm [5], JADE [6], NetLogo [4], Meta ABM, and a long list of others which some are highly specialized or labeled by specific general fields such as business and organization, economics, infrastructure, crowds, military and biology[1].

A typical ABM framework that executes serially on the CPU needs new strategies to evaluate the complex models. ABM mainly requires broad various input parameters that each can have a range of different values. The system requires large-scale computational experiments to explore the parameter variation space of such a complex model. The simplest solution consists of decomposing models. Each component could be processed independently by a processor to reach good scalability in ABMs; this could be an issue during parallelizing ABMs [3].

From the modelers' perspective, after implementing the model and scaling the parameters to produce a good pattern or values, the next step will be checking the reliability of the model's implementation. Simple models with fewer parameters and work with simple processes are more accessible to get verified and validated. The major challenge for modelers happens in more complicated models with large parameter spaces and many interconnected sub-processes [1].

Verification and validation are affected by replication in computational models. Replication effects model verification while the modeler is comparing the implemented model with the conceptual model. On the other hand, it affects model validation during the correspondence between the model and the real world. Computation replication often happens both in parameters space and time. Data replication is not the purpose of the modelers' perspective, but it could be the developer's issue [4].

Regards to the above perspective, many challenges are identified. The first challenge is the modeler's and a researcher's challenge to optimize their use cases to set the large parameter space and cover them by submitting millions of simulations as the job scheduling. To achieve this goal, they put an extreme load on the system in many HPC sites [6] [7].

Monitoring and checking the results of these jobs has also been a very time-consuming process. The second is about restrictions and limits for the number of jobs that a modeler can submit to the local queue. Providing many simulations to run per job submission is fatal for some local HPC sites. Furthermore, there are intervals in the execution times, such as early termination of some jobs, which should be detected by the researchers to save the remaining time left of the CPU and fulfill the run time requirement for the other simulation jobs manually.

The third is sharing dynamic libraries that are required for the run-time on the execution machine. There are many restrictions for installing libraries on the local hosts, which means there is no guarantee that researchers can install the required ones if they could not find them on the machine, so the role of the tuned frameworks is linking shared libraries to certify that the required libraries for the run-time will be available. The fourth challenge is analyzing the output files during the simulation to cut the computation loops and loss [6] [8].

1.1.1 Agent-based models in Systems Biology

Biological systems include random behaviors, and ABMs accommodate this via population agents' generation into the agent's rules. ABMs have a level of abstraction to create a new cellular state or environmental variable without changing the simulation's core aspects. To aggregate the paradoxical nature of emergent behavior, which could observe from any agent in contrast to the model's conceptual rules, ABMs reproduces emergent behavior. Emergent behavior has a range of stochasticity like real-world Systems Biology. In summation, ABMs lead to the most robust relevance to the real-world [9].

Finding software platforms for scientific agent-based models requires comparing specific software issues such as emulating parallelism and developing schedulers for multiple iterations that manage ABM runs. Agent-based models based on complex systems biology are dynamic networks of many interacting agents. Since there has not yet been a general framework for designing, testing, and analyzing bottom-up models of cellular automata or agent-based models, recent modeling advances have come together in a broad strategy called pattern-oriented modeling. This strategy provides a unifying framework for discovering agent-based complex systems' organization and may lead to merging algorithmic theories of the relation between adaptive functioning and behavior in complex systems [10].

Pattern-oriented modeling can reduce uncertainty in model parameters in two ways. First, it helps make models structurally realistic, making them less sensitive to parameter uncertainty [10]. For instance, the model reproduces an array of probable patterns without tuning parameter values taken from the user. Then, passing through the analysis workflow and changing parameters as feature subsets shows relatively independent effects on different outputs from the patterns. So, the model could be calibrated manually and independently. Second is the realism of the structure and mechanism of pattern-oriented models help parameters interact in ways like interactions of the real mechanisms [10]. In this case, using a technique which is known as ”inverse modeling” helped us scale the parameters by finding values that reproduce multiple patterns simultaneously. For example, in the tumor growth analysis scenario, this inverse modeling can help scientists find essential values and profiles.

Pattern-oriented modeling explicitly considers mitosis is the primary idea for studying avascular tumor growth in using agent-based dynamic networks of many interacting agents within the tumor.

In evolutionary versions of these scenarios, modeling the interactions of these types of models across large multiscale needs agent-based modeling. Each agent has a Boolean network for itself expression, such as a gene. A proper production or application of systems biology (SB) must support a compatible simulation method and suitable methods for model parameter estimations that represent experimental data [10].

Graph-based Topology for Agent-based modeling

The representation of complex cellular networks as graphs has made it possible to systematically investigate these networks' topology and function using well-understood graph-theoretical concepts. Graph theory can be used to predict the structural and dynamical properties of the underlying network. Such predictions can suggest new biological hypotheses regarding, for instance, unexplored new interactions of the global network or the function of individual cellular components that are testable with subsequent experimentation.

The graph structure could be a simplistic dynamical system originating from small Boolean network models, where nodes represent discrete biological entities such as mRNA or protein. The nodes can be thought to be either on or off and edges their Boolean relationships as genotypes, can give rise to a multitude of designable dynamical phenotypes outputs. On the other hand, the Gene expression data are collected from the protein-protein interaction (PPI) networks. In this system, genes are mapped to the corresponding proteins for representing a network graph of (PPI).

Mathematical modeling also enables an iterative process of network construction, where model simulations and predictions are closely coupled with new experiments chosen systematically to maximize their information content for subsequent model adjustments, providing increasingly more accurate descriptions of the network properties.

The topological relations underlying graph-based methods can also convey structure to putative pathways that avoid approaches that test many known sets of molecules without causal interactions. Furthermore, graph formalisms may provide powerful tools for omics data integration to address fundamental biological questions at the systems level [11].

Analysis of Interaction Patterns

Graph algorithms have been used to characterize inter-connectivity and more complex relationships between nodes. This method can facilitate biological network modeling and its fundamental biological concepts, such as cellular pathways and gene expression profiles gathering from the PPI network [11]. Once a biological network has been represented as a graph, the conventional graph-driven analysis workflow involves [11]:

- Evaluating the specificity of the model predictions using graph evolving behavior patterns.

- The shortest path length of indirectly connected nodes.
- Computing the local graph properties such as the number and complexity of clustering sub-graphs.
- Centrality measures and statistics.

Mapping agents to the nodes in the graph-structure model coordinates the values' assignments to their variables to maximize their aggregation. Agents work as states, locations, or even sometimes as controls of all the variables that map to the nodes.

Subgraphs and Centrality Discovery

Discovering a complex network community structure is a significant challenge. Many advanced algorithms have been proposed to detect community structures in complex networks, but most have limitations. The limitations include supporting the large-scale network discovery, overlapping communities, large multiple parametric dependencies, specific structures. Therefore still cannot generate stable partitions [127].

Some methods proposed to detect the community utilizing one kind of network representation like topological measures. For example, spectral clustering (SC) for discovering the community in the graph network can effectively cluster networks, but finding the critical factor to affect the graph clustering in biological networks is very difficult by using this method because the algorithm for this task needs to overcome the problem of the data representation of heterogeneous information in multiscale biological models.

One of the most critical tasks in the analysis of protein-protein interaction (PPI) is to predict a group or cluster of transiently interacting proteins that together can accomplish a biological function. These groups can be mapped to specific subgraphs in the network [115].

Characterizing nodes in a network is according to the number of closed walks starting and ending at the node. Each closed walk is associated with a connected subgraph, and the measure counts the times that a node takes part indifferently connected subgraphs of the network. The node behaves like a hub. They have called this measure the ”sub-graph centrality” (SC) for nodes in a network [115].

Since molecular sub-typing could be done based on gene expression patterns, and it helps in tumor-derived cell classification [13], then a subgraph represents a subset of nodes with a specific set of edges connecting them. As the number of distinct subgraphs grows exponentially with the number of nodes, efficient and scalable heuristics have been developed and applied for detecting the given subgraphs and their frequencies in large networks.

Under the random graph model, it is also possible to calculate the estimated distribution of different subgraphs with a given number of nodes, edges, and their specific global properties like degree distribution and clustering coefficient [12] analytically. Such approximate analytical expressions will save substantial amounts of computing time when analyzing, e.g., lists of proteins with large undirected graphs representing their known functional relationships [13].

Centrality is a local quantitative measure of a node’s position relative to the other nodes. It can be used to estimate its relative importance or role in the global network organization. Different computations of centrality are based on the node’s connectivity (degree centrality), its shortest paths to other nodes (closeness centrality), or the number of shortest paths going through the node (betweenness centrality) [11].

In particular, the best performance in identifying essential proteins was obtained with a novel measure introduced to account for a given node’s participation in all subgraphs of the network (subgraph centrality), which gives more weight to smaller subgraphs. It was proposed that those ranking proteins, according to their centrality measures, could offer a means of selecting possible targets for drug discovery [14].

Paths and Pathways

In the directed graphs theory, a path is a chain of distinct nodes connected by directed edges, without branches or cycles. Such pathways in cellular network graphs can represent. Pathway redundancy (the presence of multiple paths between the same pair of nodes) is an essential local property that is thought to be one reason for many cellular networks' robustness.

Betweenness centrality can measure the effect of node perturbations on pathway redundancy, whereas path lengths characterize the response times under perturbations. Besides the various commercial software packages for pathway analysis, there exist also freely available tools for any specific graph queries, such as finding the shortest paths between two specified seed nodes on degree-weighted metabolic networks [15] or searching for linear paths that are similar to query pathways in terms of their composition and interaction patterns of a given protein-protein interaction (PPI) network [16].

1.1.2 Towards Scalable HPC Simulation

Constructing a scalable and dynamic simulation tool capable of running HPC (High-Performance Computing) is already a mature solution for dealing with extensive and multiscale modeling and simulation problems. Many simulation frameworks can be run in HPC environments, especially agent-based [86]; however, only a few of these are suited for large-scale in parallel and distributed infrastructures [87].

A good example is REPAST-HPC [88], a framework based on the Message Passing Interface (MPI) standard [93]. Large-scale social science simulations could also be supported by the PDES-MAS framework, based on the Parallel Discrete Event Simulation (PDES) paradigm. In PDES-MAS, a simulation model is divided into a network of concurrent logical processes where the state shared between the workers is managed by a tree-like network in a space-time Distributed Shared Memory (DSM) model [90]. The Pandora framework is another example of an MPI-based tool with the possibility of using a Cloud infrastructure

[91] alongside the traditional HPC resources.

As mentioned earlier, all the platforms use relatively standard technologies, like the message passing (MPI) protocol, to create processes for parallel and distributed execution and communication and synchronization mechanisms. This well-established standard has been used in simulation for many years. The standard approaches employing MPI are implemented in C++, which can offer good performance but is inefficient in rapid experiment development and relatively inflexible as a general implementation technique [92].

The degree of distribution introduced by high-performance computing provides a greater magnitude of a performance boost and exhibits all the disadvantages of distributed computation [92]. A distributed solution's performance is prone to decrease due to data synchronization, inter-node communication, inefficient serialization, and hardware failures. Maintaining a consistent state in a large-scale cluster is a challenge of its own and building a solution on its basis requires much effort. This must be custom-tailored to ensure that the overhead does not outweigh the advantages [92].

Usually, consistency and scalability make troubles. Consistency requires that nodes in the cluster share information among themselves, which in turn reduces scalability. Cluster sharding is a way to bridge these two qualities within a single concept. Cluster sharding solves the problem by partitioning the Grid using the consistent hashing of aggregate identifiers. These partitions are often called shards sets of actors. Each shard can only be present on a single node at any given time [92].

Since this is an HPC hardware solution, there will be much inconsistency in resource allocation. If there were too few shards, some nodes would not have any work to do. The load-balancing mechanism will introduce unnecessary overhead, and it could be new challenges for simulation complexity or simulation timeline.

1.1.3 High-Performance Computing for Agent-Based Modeling

Different tools and environments can be found in ABM execution models using different simulation methods and infrastructure kernels, as viewed in [83,84]. If it is considered selection criteria as environments which supports parallel and distributed ABM simulation then, it can be listed as D-Mason [86], EcoLab [94], FLAME [95], MobiDyc [96], Pandora [97], and Repast HPC [98] and, as concurrent execution, Netlogo [4] but only for the Behavior Space.

D-Mason is a parallel extension of the MASON library [99] that allows writing and executing simulations of agent-based models.

EcoLab was initially designed to support an abstract ecology model and allows the agent partitioning over processors and fault tolerance support.

FLAME (FLexible Agent-based Modeling Environment) is a code-generated tool that allows the user to define their agent-based model and automatically generates an optimized C code for efficient parallel processing [100]. Mobidyc aims to develop agent-based models in the fields of ecology and biology.

Pandora is an ABMS tool from the Barcelona Supercomputing Center used in various projects related to archeology. Repast HPC is an agent-based modeling and simulation environment that implements the main concepts of Repast Simphony with support for parallel-distributed environments. The objective of these designs is to obtain maximum scalability in supercomputers for extensive project simulations.

Unlike the previous ones, Netlogo was designed as an academic project with teaching/education objectives. However, its usability characteristics (including editing/ simulation/visualization in the same environment) and a friendly interface, and a simple programming API have become a reference in different areas where the simulation adds strategic value. However, its users are distant from technological environments. Netlogo uses the Java RTE and can run without difficulties on any computer/OS or in a browser with a Java plugin (applet mode).

The main Netlogo limitation is that it only supports multi-threading in the Behavior Space tool, so its performance is limited to the number of cores/ threads in the local infrastructure. Different initiatives have emerged, such as Netlogo-cluster and GNU GLP 3, that allow the parametric execution of Netlogo scenarios on an HPC cluster using script files. It helps the execution and distribution of work on the cluster nodes.

New environments have emerged based on workflow formalisms such as OpenMOLE [101] and SCOOP [102]. OpenMOLE is a workflow engine for exploring simulation models that use high-performance computing, where they transparently delegate their concurrent executions to the remote execution environment. SCOOP is a distributed task environment that allows concurrent parallel programming in multiple environments.

Although these environments provide extended functionality and many performances in different languages/frameworks, none of them has turned out to have a simple interface for using the HPC environment as an elastic environment for the ABM model simulation. Except for a few exceptions, most of them require a series of technological knowledge to interact with the environment, to re-code the model, to integrate it with the simulator/framework, to configure the environment, or to generate the workflow that makes them unviable for a community of scientists, which only need the ABM simulation as an instrument.

Nevertheless, we have always approached by NetLogo modelers, especially biologists, economists, and epidemiologists, about their problem while they are designing a model that is executing many repetitions to collect a comprehensive data set.

The process is computationally costly and puts a hefty load on discrete nodes. Most of the NetLogo modelers want to farm out the experiment runs into different threads to save time. Modelers are also non-expert users in confronting software described above and clustering techniques. The only possible solution is to execute the model on separate NetLogo instances running on discrete machines for them, which means data splitting and regression parameters analysis.

NetLogo is widely recognized as a relatively easy platform for agent-based simulation, in the sense that it allows modelers, including both beginners and experienced ones, to move rapidly through the design, programming, and testing stages and on to using models for analysis and developing scientific understanding. However, NetLogo is not widely recognized as an easy, efficient platform in the sense of providing low (or at least reasonable) execution times for working models. Potential users who hear NetLogo's reputation as too slow or limited for bigger models, or who perhaps look only at the elementary examples that (understandably) dominate NetLogo's built-in model's library, may be discouraged from selecting NetLogo as a platform for large scientific models.

Choosing another platform, especially those requiring programming in a base language, such as C++ or Java, comes at a high cost: Programming will take much longer, mistakes will be harder to find and hence more expensive, the tools necessary for testing and understanding models (graphically and interactive displays, experiment managers like Behavior Space) must be developed, and unless the code is designed as cleverly as NetLogo's primitives appear to be the result may turn out to be slower than NetLogo.

There is now sufficient evidence that NetLogo is neither inherently slow nor incapable of handling large and complex models. In previous experience comparing ABM platforms [103] [104], while confirming the general understanding in computer science that it is not simple or straightforward to say which platform or programming languages are faster or slower, indicates that NetLogo is not dramatically slower at executing models than other popular platforms are.

Perhaps the best evidence that NetLogo is suitable for large scientific models is that many such models have now been successfully implemented and analyzed extensively in NetLogo [104]. The use of a High-Performance Computing (HPC) cluster, typically an extensive massively parallel computing system with many processor cores and a shared storage system connected via a fast network, allows Behavior Space to execute large numbers model runs simultaneously and makes extensive simulation experiments feasible even for prolonged models. In one example, Ayllón [103] reports parameterization and sensitivity analysis experiments that each required thousand of runs of a model that can take hours to days per run, made possible via HPC and Behavior Space. In another example, Sheppard [43] used HPC to run their vehicle model millions of times to optimize electric vehicle charging station locations in Delhi. Many universities and research laboratories have HPC clusters, and some commercial ”cloud computing” services offer free trial access to clusters.

Providing a job submission interface for many applications is a common strategy which some famous tools such as Globus [105] or UNICORE [106] [107] have known in this regard. Both software provides a scalable environment to compute resources for research projects and submits the jobs through a gateway nod at each engaged site. This could be a vulnerable point. At the same time, many simultaneous users require a thousand submissions and cause overloaded gateway resource nodes.

Due to this occurred weakness, some other tools which suggested an alternative solution came up. Pegasus/Condor-G, Nimrod, and APST [85] are all three other systems that have scheduled submitting jobs through some smart central agents across the Grid. Their solution indeed worked significantly better than overloading each gateway node by submitting as many jobs as possible [108].

Based on the overview of job creation, submission, and execution in UNICORE middleware, which has been addressed in a study of four grid middleware technologies published, a user creates his abstract job as an object in a serialized form java object or XML format. The abstract job object (AJO) is how a UNICORE client specifies the UNICORE server’s work. The specification of jobs can be retargeted to the different server addresses, but the work specification remains the same. The abstract jobs’ executing order and their child’s abstract actions are kept in a directed acyclic graph (DAG), a hierarchical data structure. UNICOR’s communication methods based on the AJO model does not support synchronous message passing [8].

Another one was introduced as an open-source project led by the University of Melbourne called Gridbus [106]. The project runs the design and development goal through service-oriented clusters and middleware technologies to support science applications. It focuses on scaling the efficiency of computing models from the clusters to the Grid. The Gridbus components provide software technologies such as a GridSim toolkit for simulation and modeling of resources. GridSim can simulate parallel and distributed scheduling systems such as brokers or grid schedulers for evaluating the scheduling algorithms' performance [106] [109].

These environments provide resource allocation and management by the web services interface and services to collect data from any information source such as XML-based. However, to do anything more complicated than submitting jobs, a user needs to use another tool in conjunction or create his services that are not easy for most non-expert users.

Data splitting has been used for various purposes, such as eliminating the data attributes that have unfavorable effects on the clustering performance and its time complexity. The data splitting role is selecting and reconstructing the input data features. It could be the fundamental step of the clustering analyses and execution process [110].

Many papers present data decomposition as one of the most prevalent tasks of parallelizing applications for multi-core architectures. According to the bibliographic analysis carried out, there are up to ten critical requirements for tool support to help HPC developers in this area [111]. Interpreted languages are ideal for building control and administration tools and providing a framework for non-time-critical components of a broader application (e.g., GUI and file handling). A parallel scripting language is needed to provide the same functionality [112].

Python language enables programmers to write truly distributed, parallel scripts that allow developers to write parallel extension modules. Python has built-in and external modules that simplify implementation [92].

1.2 Complex Networks

The study of complex networks involves physics, mathematics, chemistry, biology, social sciences, and information sciences. These networks are commonly represented by directed or undirected graphs, including sets of nodes representing objects such as people or groups of people, cellular and molecular entities, computers, or any other thing. These objects joined together in pairs by edges in linking and presenting the type of relationship.

The network could be the Internet, the World Wide Web, social networks, information networks, neural networks, metabolic networks, and protein-protein interaction networks [26]. Network Informatics is an interdisciplinary science based on informatics, network science, and other related scientific disciplines [31]. Network Informatics aims to understand and investigate the structure, properties, and organization of information in the network. The scope of network informatics covers theories, algorithms, and software of network informatics; mechanisms and rules of flow and organization of information in the network; theory and Methodology of dynamics, optimization, and control of information networks; network analysis of information networks; factors that affect organization and communication of information, Etc. [34].

However, the representation of complex systems as a network is not enough to study a particular problem because it gives minimal information about the system's structure in the real world. This limitation guides the study by expanding the network scale and levels. The decomposition of large networks into distinct components, events, or modules, has come to be regarded as a practical approach to deal with large-scale networks' complexity [27–29]. In summary, the Erdös-Rényi [37] model, Watts and Strogatz model, and the Barabasi-Albert [38] model are three famous network models for random graphs.

Erdös-Rényi model takes a few vertices N and connects nodes by randomly selecting edges from the $N\,(N\text{-}1)\,/2$ possible edges. The degree distribution for this model is given by a binomial distribution [37][38]. Watts and Strogatz model use for small-world topology, and the Barabasi-Albert model describes scale-free networks. It is one of the most basic models since it describes most biological networks by revealing information about the network dynamics, especially from evolutionary models [36].

In the physics literature, networks with high clustering are commonly modeled by the small-world network model of Watts and Strogatz(WS), while networks with the Power-Law degree distribution modeled by the scale-free network model of Barabasi-Albert model[135]. Although both models have a logarithmically increasing with the network size, each model lacks the property of the other model: the Watts and Strogatz (WS) model show a high clustering, but without the Power-Law degree distribution, while the Barabasi-Albert model (BA) with the scale-free nature does not possess the high clustering [135].

The ideal model is a network model with both the perfect Power-Law probability distribution and high clustering. However, for evolving models, the integration of Holme and Kim's algorithms as a generator of growing graphs with a Power-Law probability distribution and an approximate average clustering makes it possible to have a disconnected graph. It is essentially the Barabasi-Albert (BA) growth model with an extra step that each random edge is followed by a chance of making an edge to one of its neighbors, too (and thus a triangle).

Holme and Kim's algorithm improve on Barabasi-Albert (BA) growth model because it enables a higher average clustering to be attained if desired. The initial m nodes may not be all linked to a new node on the first iteration, like the Barabasi-Albert growth model.

Power-Law probability distribution and approximate average clustering methods to complete the clustering and classifying helped cut off a large multiscale model. Agent-based modeling of the complex network in some way drives the emergence of the agent mining field. Agents can support and enhance the knowledge discovery process in many ways [31]. For instance, agents can contribute to data selection, extraction, preprocessing, and integration, and they are an excellent choice for peer-to-peer parallel, distributed, or multi-source mining. In cases where precise contact network data are unavailable, an alternative is to mine [32].

1.2.1 Network Complexity of Biological Models

Networks follow patterns and rules and have a specific topology that allows scientists to conduct a more in-depth investigation of biology information extraction. Within the fields of biology and medicine, Protein-protein interaction (PPI) networks, biochemical networks, transcriptional regulation networks, signal transduction, or metabolic networks are the highlighted network categories in systems biology which could detect early diagnosis [34][35].

All these networks need consistent data to be produced experimentally or retrieved from various databases for each network type. However, besides analyzing data structures for computational analysis, several topological models have been built to describe the global structure [34].

The network models can be constructed based on existing knowledge of molecular interactions, the relationship between data profiles, or the mapping of data onto knowledge-based networks. They have been developed to assist with a variety of decisions [33]. In this construction method, once the model architecture has been defined, the network structure (i.e., the interactions between the components) and the model parameters (e.g., type-/strengths of these interactions) need to be learned the data. Several different model architectures for reverse engineering GRNs from gene expression data have been proposed over the last years. They cover varying degrees of simplification and reflect different assumptions of the underlying molecular mechanisms.

Generally, the network nodes represent compounds of interest, e.g., genes, proteins, or even modules (sets of compounds). Knowledge-based modeling is tackling complex data for studying complex metabolism. The aim is to provide support for laboratory test ordering or designing a scientist's decision support system. The knowledge-based modeling contains the rules and associations of compiled data, mostly mined from the rules.

Unlike non-knowledge-based modeling, knowledge-based models use a form of artificial intelligence to allow the computer to learn from past experiences or to recognize patterns in the clinical data. Two types of non-knowledge-based systems are neural networks or genetic algorithms. There is no need for writing rules or configuring expert parametric inputs, but since the system cannot explain why it uses the data the way it does, mostly clinical does not like to use them because of no reliability. This method always relies on existing data and therefore encounters data security, and big data deficiencies will be the consequences.

Building a Knowledge-based model that will cover multiple scales from the genotype and various biochemical reactions to the details on cell morphology and the probable behavior pattern of millions of individual cells is interacting with the other cells is feasible. The integrated network-based analysis aims at identifying coordinated changes in molecular processes. In this sense, network-based analysis of high-throughput data provides the means for generating biologically meaningful hypotheses and for extracting behavior patterns from experiments to unveil the underlying regulatory mechanisms [33]. Tools and measures have been developed to identify whether a given network is modular or not and detect the modules and their relationships in the network to investigate the modularity of interaction networks. By subsequently contrast the found interaction patterns with other large-scale functional genomics data, it is possible to generate concrete hypotheses for the underlying mechanisms governing, e.g., the signaling and regulatory pathways in a systematic and integrative fashion. For instance, interaction data together with mRNA expression data can be used to identify active subgraphs, that is, connected regions of the network that show significant changes in expression over subsets of experimental conditions [30].

A model which covers multiple scales from the genotype and various biochemical reactions to the details on cell morphology also shows the probable behavior pattern of millions of individual cells interacting with the other cells to form the whole tumor tissue, is a mega-model with the structural complexity and it could be comparable with the biological network. Inevitably, this model consumes more computational time [33]. Therefore, it is more desirable to find a way to integrate and bridge independent sub-models rather than build a single mega-model that encompasses all the development complexity. This integration may be separate models that consider distinct parts of the evolving process or the same process but on different scales.

Van Someren has described in [161] that model architectures can be distinguished by representing the network components' activity level.

The concentration or activity of a compound can be represented by Boolean ('on', 'off') or other logic values (e.g. 'present', 'absent', 'marginal'), discrete (e.g. cluster labels), fuzzy (e.g. 'low', 'medium', 'high') or continuous (real) values. Moreover, network model architectures can be distinguished by the type of model (stochastic or deterministic, static or dynamic) and the type of relationships between the variables (directed or undirected; linear or non-linear function or relation table).

A metabolic network reaction can be described as a weighted directed edge in a directed graph where nodes are the chemicals and edges are the reactions. There is a lack of a well-developed theory for the structural analysis of directed graphs, so two alternative representations of a metabolic reaction are usually utilized [39]. One is a bipartite graph, and the other is a substrate graph.

The bipartite graph is used when modeling relations divided into two different objects, such as parents and children. The substrate graph is more useful to study chemical reactions while two nodes are connected if the corresponding chemical compounds take part in the same reaction. Global characterization of the metabolic network can be carried out by obtaining the average sub-graph centrality. Small subgraphs capture specific patterns of interconnection characterizing the biological networks at the local level [40].

Tumor growth Network Complexity

Tumor evolution is a complex multiscale process that depends on molecular growth factors such as genetic mutation, gene expression, cell adaptability, robustness, and cellular growth factors significant from the cell micro-environment such as multiple metabolites nutrient gradients.

On the other hand, tumor cells can mechanically interact with other tumor cells and various other stromal cells, such as fibroblasts, macrophages, and immune cells. Primarily, analyzing tumor evolution and metastasis was only taken into consideration at the gene or protein scale. However, recently the impact of this evolution at the cellular scale and level has also been considered since most tumors' progression depends strongly upon the interaction of the cells and the host tissue's cellular architecture.

Cell evolution is often modeled using a specific cellular process simulation such as cell growth, division, death, or movement. These simulations are determined sequentially by comparing cell status, cell age, nutrient level, the number of cell neighbors, or the configuration of cell membrane receptors [41]. Stem cells exist in many different types, as they have been identified in various tissues and organs. Each stem cell type is classified by their origin in the body and their potential to differentiate into other cell types. This potential varies among stem cell types.

Cancer Stem Cells (CSCs) or tumor-initiating cells were identified and characterized as a unique subpopulation with stem cell features in many cancer types. Current studies about CSCs provide novel insights regarding tumor initiation, progression, angiogenesis, resistance to therapy, and interplay with the tumor microenvironment. A cancer stem cell niche has been proposed based on these findings.

The niche provides the soil for CSC self-renewal and maintenance, stimulating essential signaling pathways in CSCs and reducing factors that promote angiogenesis and long-term growth of CSCs. The tumor growth rate and angiogenesis of each transplantation were then compared. Angiogenesis involves the stimulation of Endothelial Cells (EC) by growth factors. Gene Ontology (GO) provides a rich resource of gene functions and locations in many different species, positive regulation of angiogenesis, and negative angiogenesis regulation.

Another approach involves cell interactions with external factors that are taken a concentration of metabolites using the neural networks, signaling pathways, or protein networks [42]. Combining dynamic gene expression time-course data for stimulated EC with protein-protein interactions associated with angiogenesis could reveal how different stimuli result in different network activation patterns and could implicate signaling intermediates as points for control or intervention.

A primary tumor model likes Wilensky's tumor model addressing the avascular growth state depends on differential equations. They are classified as "lumped models" to predict the temporal evolution of overall tumor size [17] [18]. Since lumped models just provide a quantitative prediction of tumor size over time with only a few parameters and very low computational results, they would not be enough for a detailed investigation of many other events such as spatiotemporal dynamics oxygen and nutrients or cell to cell interactions. Also, considering stromal cells, which play a significant role in cancer growth and progression in tumor cells' interaction. This means disregarding the mutations in the tumor microenvironment and metastasis [18].

These shortcomings lead us to *In Silico models* of the tumor microenvironment. In silico [19] refers to computational models of biology, and it has many applications. It is an expression used to mean performed on a computer simulation. In silico models are divided into three main categories [19]:

- Continuum based models that solve the spatiotemporal evolution problems in density and concentration of cellular population in the tumor microenvironment.
- Discrete based or agent-based on a set of rules to change the cells states and manage the tumor microenvironment's cell interactions.

- Adaptive Hybrid models that integrate both the above solutions.

Spontaneous tumor, which progresses from the initial lesion to highly metastatic forms, is generally profiled by molecular parameters such as prognosis response, morphology, and pathohistological characteristics. Tumors can induce angiogenesis and lymph-angiogenesis, which plays an important role in promoting cancer spread. The previous studies have shown that the cancer stem cell (CSC) theory could become a hypothesis for tumor development and progression.

These CSCs have the capability of both self-renewal and differentiation into diverse cancer cells. So one small subset of cancer cells has the characteristics of stem cells as their parents. Hereditary characteristics play a particular role in malignant proliferation, invasion, metastasis, and tumor recurrence [20].

Recent researches show the possible relationship of cancer stem cells, angiogenesis, lymph-angiogenesis, and tumor metastasis is becoming a challenge. Due to many pieces of evidence and reviews such as [21] [22], metastasis is defined as the spread of cancer cells from the site of an original malignant primary tumor to one or more other places in the body, and more than 90 percentages of cancer suffering and death are associated with metastatic spread. Folkman proposed in 19971 that tumor growth and metastasis are angiogenesis-dependent, and hence, blocking angiogenesis could be a strategy to intercept tumor growth [22]. Genetic approaches later confirmed his hypothesis. Angiogenesis occurs by migration and proliferation of endothelial cells of original blood vessels [21].

Since tumor angiogenesis is critical for tumor growth, monitoring it validates biomarkers such as VEGF, CD45, EPCs [21] helps develop drug resistance. Accordingly, translational cancer research has contributed to the understanding of the molecular and cellular mechanisms occurring in the tumor and in its microenvironment that occurs metastasis, which could present relatively a model similar to the physiology of human or at least have the capability for going through genetic manipulations that bring them closer to human.

Hence tumor modeling with a high spatiotemporal resolution combined with parametric opportunities has been rapidly applied in technology [23]. Mathematical and computational models of the tumor should have covered different aspects of tumor interactions in its microenvironment. Most of the avascular tumor models address the tumor growth dynamics, not considering the angiogenesis setting. However, computational models have started research on tumor-induced angiogenesis since cancer research, so the angiogenesis models have a solid background in experimental observations. The most established research explicitly focused on stromal cells to achieve the closest computational model of angiogenesis based on Monte-Carlo simulation[17].

According to Monte Carlo simulations and energy minimization, cellular models are expanded and elaborated on cellular automata to allocate more than one lattice site to each cell and describe cell to cell and tumor-stroma interactions. Nevertheless, building a bulky model over a range of matrix densities, which covers numerous factors in this way for large domain sizes and 3D simulations, is restricted by computational and application costs[17].

Based on paper [17], a minimal coupling of vascular tumor dynamics to tumor angiogenic factors through agent-based modeling could progress the experimental studies during recent years. They have mentioned in both reviews that it is challenging to simulate all the processes of a complex system such as tumor growth, metastasis, and tumor response treatments mathematically because mathematical modeling is still a simplification of the systems biology; the results need validation. Also, to build a predictive experimental plus theoretical application without clinical data required to parametrizing and validating again.

1.2.2 Network Analysis Tools

Several such software packages and tools are developed for these above challenging tasks along with their specific functionalities. Publicly available software systems that use graph-based data, integrating visual frameworks for networks include e.g. Cytoscape together with its recent plug-ins [43–45], Osprey [46], GiGA [47], megNet [48], VisANT [49], BioPIXIE [50], Pointillist [51, 52], PIANA [53] and PathSys [54].

An essential component of such systems is the possibility to visualize the graphs under analysis. This can be regarded as a fundamental tool in the explorative network analysis; even if one wants to address only a particular question within the given network graph, it may help visualize the result to discern possible flaws or follow-up questions. Recently introduced graph drawing tools include, e.g., WebInterViewer [55] and PATIKAweb [56].

By meeting the challenges of automated construction and simultaneous visualization of multiple pathways, such software tools can help relate the selected node sets and their interconnections to the underlying biological significance. These advanced graph algorithms and post-processing tools can also be used in conjunction with more specific, freely available software packages.

One major challenge of computational network analysis deals with selecting appropriate model types for analyzing data from different experimental approaches. Accurate modeling and integration of protein interactions measured with the yeast two-hybrid and affinity purification/mass spectrometry can be very critical, e.g., for understanding the physical properties and functional operation of local protein complexes [57].

A few computational approaches have been tried to reconstruct the underlying global network structure or even the causal regulatory relationships between the nodes from the experimental data sets. This challenging problem is often referred to as network inference or reverse engineering [85, 86]. For instance, several works have dealt with gene regulatory network inference from gene expression microarray data alone. In such a hypothetical network, the nodes conventionally correspond to both the gene and the protein it encodes, and the edges to the statistical relations between the genes.

The Bayesian network offers a convenient probabilistic model, where nodes represent gene expression levels as random variables. The edges represent their conditional dependence relations and the corresponding DAG the joint probability distributions of the observed expression patterns. However, it has been recognized that these data are enough to reconstruct only relatively small networks. Even in idealized situations, the estimated models contain many false edges because the expression data alone cannot unambiguously distinguish the underlying target network [58].

Further challenges are faced when applying the inference algorithms to limited quantities of experimentally collected noisy data from real biological systems [59]. Suggested solutions to tackle these problems include the usage of gene network motifs [60], network pruning methods [63], or reduced network models [61]. Such reverse-engineered gene networks could be of great medical significance, for instance, in the identification of drug targets [62].

Egocentric Network analysis

Egocentric social network analysis (SNA) is a methodological tool used to understand the structure, function, and composition of network ties around an individual [136]. Both socio-centric network analysis and egocentric network analysis share the basic assumption that individuals' behaviors, beliefs, attitudes, and values are shaped through contact and communication with others [136]. However, these two methods are distinct in several important ways [137]:

1. Unbounded versus bounded networks: Socio-centric network analysis collects data on ties between all members of a socially or geographically bounded group and has limited inference beyond that group, but egocentric network analysis recognizes individuals' particular community networks across any number of social settings using profile generators and is, thus, less limited in theoretical and substantive scopes.

2. Focus on individual rather than group outcomes: Socio-centric network analysis often focuses on network structures of groups as predictors of group-level outcomes (e.g., the concentration of power, resource distribution, information propagation). In contrast, the egocentric network analysis is concerned with how the population's interaction patterns shape their individual-level outcomes (e.g., security, behavior, probabilities).

3. flexibility in data collection: Because the socio-centric network analysis must use as its sampling frame a census of a particular bounded group, data collection is very time-consuming, expensive, and targeted to a specific set of research questions. In contrast, because egocentric network analysis uses individuals as cases, potential sampling frames and data collection strategies are virtually limitless.

Egocentric data collection tools can easily be incorporated into large-scale or nationally representative surveys fielded for various other purposes [129].

Many computational methods have been developed to extract genome profiles or pathway information from biological or clinical networks to identify subnetworks and hub nodes, which have an essential role among other nodes and neighborhoods. For example, analysis of protein-protein interaction (PPI) [16] networks helped to understand better the complex biology of specific disease complex systems, including cancer.

Gene expression data have been collected from PPI networks on cancer tumors for developing algorithms. These algorithms functionally express gene profiles into some standard modules, shared in a subset of different cancers and tumors. Genes are mapped to the corresponding proteins for representing a network graph of PPI to achieve the best-translated modules.

There is a need to estimate many analysis techniques and modules that could translate PPI networks' most relevant expression to model a sizeable multiscale tumor network. Egocentric network analysis techniques and modules frequently use in the social network. However, some proposed new methods in Bioinformatics such as EgoNet [23] present their method based on egocentric network-analysis techniques to search and prioritize disease subnetworks and gene markers from large-scale biological networks. They have developed an algorithm to identify significant sub-networks that are functionally associated with diseases and predict clinical outcomes.

1.3 A Data-Science Approach for Agent-Based Modeling

Data-science uses scientific methods, processes, algorithms, and systems to extract knowledge from structured and unstructured data. Data science is related to data mining, machine learning, and big data. Data science has been applied in solving involved practical related complications in various applications, becoming more popular and acceptable every day [124]. Artificial Intelligence (AI) has been used in many applications to solve classification, diagnosis, selection, and prediction problems. The techniques, models, capture the uncertainty between real-life cause and effect scenarios, hence incorporating available episteme with probabilities and probability inference computations [125]. The most widely used techniques include artificial neural networks, fuzzy logic, expert systems, and generic algorithms with interesting hybrid developments. Other existing techniques include support vector machines, functional network, cased based reasoning, and expert systems. Many artificial intelligence techniques have shown great potential in generating accurate analysis and results in different areas of applications such as engineering, banking, medicine, economics, military, marine, Etc. They have also been applied to identify, select, optimize, predict, forecast, and control complex systems [126]. According to the Artificial Intelligence Applications Institute (AIAI) [124], AI technology transfer's areas of application are as follows:

a) case-based reasoning: a methodology adaptation based on past evidence and existing corporate resources such as databases to pilot diagnosis and fault finding;

b) genetic algorithms: a search technique adaption with extensive applicability in scheduling, optimization, and model adaptation;

c) planning and workflow: the modeling, task setting, planning, execution, monitoring and coordination of different endeavors;

d) intelligent systems: an approach to building knowledge-based systems.

Data-Science is an in-depth process that involves preprocessing, analysis, visualization, and prediction. Besides, AI is the implementation of a predictive model for anticipating future events. Data Science includes several statistical techniques, whereas AI makes use of computer algorithms.

As a subset of AI, machine learning algorithms build a model based on sample data. Both machine learning and data science depend on each other for various applications that need data analysis. Accordingly, Agent-based modeling (ABM) and Machine learning (ML) can be combined in a variety of ways and has been examined in the past [65]. Genetic algorithms (GAs), neural nets (NNs), and Bayesian Classifiers can easily be incorporated into many agent-based models [64]. The machine learning algorithm can use the ABM as an environment and a rewarding generator, while the ABM uses the machine learning algorithm to maintain the agents' internal models. Many practical details must be addressed when deciding how to integrate ABM and data-science using ML.

Many specific algorithms are more or less useful and must be carefully considered. Neural networks, for instance, are good at classifying large amounts of data relatively quickly, but in the end, it is tough to determine how they are making their decisions. Decision trees, on the other hand, not very good at classifying continuous data. For the parametric execution, many parameters need to be set and tuned to work appropriately within the ABM environment.

Much of this is a matter of art to get the results one desires, but some ML algorithms have more advanced to tuning the parameters. It is possible to declare combining theoretical disciplines such as graph theory, machine learning, and statistical data analysis to arrive at a new field to explore complex networks by using data-science methods in an interdisciplinary manner considering a detailed analysis. Once the system is represented by a network, network analysis methods can be applied to extract useful information regarding essential system properties and investigate its structure and function. Various statistical and data-science methods have been developed for this purpose and have already been applied to networks [67].

For instance, the partitioning of a complex mega-network into different clusters or communities is of paramount importance, in particular, because the results of most methods are highly sensitive to their parameters and data quality. On the other hand, the predicted clusters can vary from one method to another, especially when the boundaries and connections between the modules are not clear-cut.

1.3.1 Data Distribution with the help of Data-Science Techniques

Data-Science algorithms are increasingly popular for handling large-scale data analytics. With the prevailing era of tremendous data sizes and complex algorithms, it is necessary to use distributed parallel computing to compute these algorithms quickly.

Many frameworks are available to run such algorithms parallel from the High-performance computing (HPC) and Big Data communities. Today, there is much focus on comparing HPC technology with Big Data technology to understand its pros and cons. These comparisons contribute to the HPC/Big Data convergence by identifying areas that can be improved in each technology domain [142].

From the modeling, multidimensional scaling (MDS) [142] is a popular, well-established machine learning technique for projecting high dimensional data into a lower dimension to be analyzed. It has been extensively used to visualize high dimensional data by projecting into 3D or 2D models. Multidimensional scaling is a computationally expensive algorithm. The best algorithms are in the range of O(N2). When applied to a more extensive data set, the computation time increases exponentially. The algorithm can be made to run efficiently in parallel to reduce the computation time requirements. Numerous frameworks exist in the Big Data community for doing large-scale parallel computations for ML algorithms, including Hadoop [143], Spark [144], Flink [145], [146], Tez [147] and Google Dataflow [148]. It is worth noting that no single technology stands out, among others, as the best [142].

MDS is the most complex algorithm among the other algorithms. It is encountered in many inefficiencies of Spark and Flink while implementing the algorithm. For Flink, the most significant inefficiency is its inability to support nested loops. This leads to a very laborious implementation where saving the intermediate data to the file system at each iteration in the outer two loops is the matter [142].

Flink is designed to read the input files each time it does the iterations, adding to the overhead. The Spark MDS implementation's main inefficiency is caused due to the lack of all-to-all collective operations. Using reduces operations followed by a broadcast operation added a couple of overheads to the algorithm [142]. K-Means showed some interesting characteristics in these frameworks. In parallel, K-Means communication cost is a direct function of the number of centroids involved, and it does not depend on the number of points. With an increased number of points, the computation time increases, but the communication time remains the same [142].

Most of the practical clustering problems do not have hundreds of thousands of clusters. This means K-Means can perform equally in Flink or spark for practical data analytics task transfers since they effectively use the networks in all the nodes[142]. The experiments showed that big data frameworks usually do not scale well for algorithms with a high frequency of communicating and compute regions. Also, it was evident that Spark performed better than Flink when there was a high frequency of communication and computation regions. The algorithms with more extended computation and communication times performed well in Flink than Spark. However, MPI has implemented very efficient collective communication algorithms, while Flink and Spark rely on point to point connections. Even though we mainly talked about Spark and Flink, there are many frameworks utilized in the Big Data community, each having pros and cons over each other.

Pregal [151], Apache Hama [152], and Apache Giraph [153] are frameworks that were developed around the BSP model. There are also numerous distributed graphs processing frameworks, including GraphX [154]. There are many comparisons in many distributed graph processing frameworks, and much research has also been done on integrating HPC technology into big data frameworks to improve performance [149] [150]. Hpc-abds [156] discusses and summarizes HPC and Big data convergence.

Mlib is built on top of Spark and offers a wide variety of machine learning algorithms. FlinkML is the requisite library for Flink. Intel Data Analytics Acceleration Library (DAAL) [157] from Intel has been tuned for Intel architecture; it provides functions for deep learning, classical machine learning, Etc.

Tensorflow [158] is a library developed at Google. Algorithms developed using TensorFlow can be executed on various heterogeneous systems that range from mobile devices to supercomputers. H2o [159] is a machine learning framework that supports Spark and Hadoop with simplified APIs for ease of use. Apache Mahout [160] remains another popular framework developed initially to support machine learning on top of Hadoop and later expanded to support other frameworks. This thesis's concern is how a user of multidimensional scaling (MDS) agent-based models can simulate the complexity and adapt the machine learning techniques and frameworks for HPC integration.

1.3.2 Graph Clustering

An approach to identifying functional modules in biological complex networks is discovering similarly or densely connected subgraphs of nodes (clusters), potentially involved in standard cellular functions or protein complexes [68]. The challenges of clustering network graphs are like those in the cluster analysis of gene expression data [68].

In particular, most methods are highly sensitive to their parameters and data quality. The predicted clusters can vary from one method to another, especially when the boundaries and connections between the modules are not clear-cut [69]. It is also important to note that the modules are generally not isolated components of the networks, but they share nodes, links, and even functions with other modules. The best perspective is partitioning a given graph's nodes into distinct clusters, depending on their neighboring interactions. Other local properties, such as centrality measures, can be used for clustering purposes as well.

A recent algorithm by Dunn et al. [70], for example, divides the network into clusters by removing the edges with the highest betweenness centralities, then recalculating the betweenness and repeating until a fixed number of edges have been removed. Another approach to decompose biological networks into modules applies standard clustering algorithms on vectors of nodes' attributes, such as their shortest path distances to other nodes [71]. Regarding the importance of the prediction, it can be argued that supervised clustering (or classification), rather than unsupervised clustering methods, should be employed. In the context of cellular networks, classification aims at constructing a discriminant rule (classifier) that can accurately predict the functional class of an unknown node based on the annotation of neighboring nodes and connections between them [72].

There are two classes of algorithms typically using for network partitioning. The first class of algorithms is graph clustering algorithms, and the second class of algorithms is community detection algorithms. The relevant applications include very-large-scale integration (VLSI) and distributing jobs on a parallel machine for graph clustering algorithms. The most famous algorithm in this domain is the Kernighan–Lin algorithm [73], which still finds use as a subroutine for various other algorithms. The other graph clustering algorithms include techniques based on spectral clustering [74].

Initially, community detection algorithms focused on social networks in sociology. They now cover networks of interest to biologists, mathematicians, and physicists. Some popular community detection algorithms include Girvan–Newman algorithm [34], Newman's eigenvector method [75,76], clique percolation algorithm [77], and Infomap [78]. Additional community detection algorithms include methods based on spin models [79,80], mixture models [81], and label propagation [31]. Since Graph clustering algorithms apply to very large-scale integration, distributing jobs on a parallel machine, and other applications found in computer science, it can be an exciting solution for integrating high-performance computing (HPC).

1.4 Thesis Overview

The thesis's general objective is to develop new methodologies, techniques, and policies to execute large scale, complex ABM models oriented to immune system dynamics or epidemiological propagation using HPC resources.

For this purpose, the thesis focused on two specific approaches. The first approach provides an environment and Methodology for scientists to use HPC resources to execute large ABM models in distributed form. The other is analyzing the execution data to obtain efficient model data distribution (agents) in the underlying infrastructure.

The best improvement is expected to achieve predictive modeling in both preclinical and clinical states while a scientist studies different patterns of the model.

Wilensky's tumor model [18] in the Netlogo library is used from the beginning as the base for the first approach in this thesis, explained in chapter two. This model is designed as a self-organized ABM model that illustrates a tumor's growth and how it resists chemical treatment.

In this case, the model consists of two kinds of cells: stem cells and transitory cells and includes controls to kill transitory cells that are younger, kill a stem cell, move cells, or visual information about the total number of living cells their trajectory [25].

In chapter three of the thesis, the research goes to the next step of the simulation of a multiscale graph agent-based model, which shows the dynamic behavior of tumor-growth based on analytics computing. Designing a graph workflow simulation system for modeling, growing behavior, and subgraph analysis leads the research deeper into dynamic modeling by creating a discretion model from a growing multiscale agent-directed simulation.

Expanding the research to the incorporated network analysis and ML clustering algorithm into the agent-based tumor model is the functional implementation to achieve the subgraphs. Chapter four's open lining development is about Cloud simulation joined with

High-performance simulation (HPS) to design an end-to-end system design for automation using large relational databases.

Contributions of the thesis are:

a) Development of a methodology for parametric analysis of models based on ABM on HPC for non-expert users.

b) implementation and deployment of a parametric simulation environment on HPC.

c) modeling and experimentation with a tumor growth model based on Hallmarks of Cancer on the Netlogo platform.

d) Model-based on graphs and ABM for multilevel analysis of tumor growth.

e) Methodology, workflow, and development of an environment based on the Python + Mesa Platform for graph-based tumor growth models.

f) Application of ML techniques for the subdivision of graphs and their subsequent execution in HPC environments.

CHAPTER 2
HIGH-PERFORMANCE COMPUTING (HPC) AS A SERVICE

High-performance computing as a service (HPCaaS) is the provision of high-level processing capacity through the local high-performance computing (HPC) systems or the cloud for scientific computing and big data analysis. One of the common problems with using High-Performance Computing (HPC) environments for non-technological users, such as economists, biologists, epidemiologists, Etc., is the complexity and difficulties involved with the configuration of the agent-based simulations(ABM).

It is usual for the ABM models to have data distributions and scenarios representing a large number of simulations to obtain statistic stability and explore a more prominent parameter data-space (generally tens of thousands of simulations). The ABM parametric simulation can benefit the power of these infrastructures and reduce the time for the analysis of parameter combinations. However, in order to use parametric simulations on HPC clusters, the configuration, and adaptation of simulation parameters and passing them to an HPC environment is too complicated for users who are not familiar with this methodology and environments, even if they use a queuing system in an HPC environment where the user should only access and execute a command.

From the modelers' perspective, after implementing the model and scaling the parameters to produce a good pattern or values, the next step will be checking the reliability of the model's implementation. Simple models with fewer parameters and work with simple processes are more accessible to get verified and validated. The major challenge for modelers happens in more complicated models with large parameter spaces and many bound sub-processes [82].

Verification and validation are affected by replication in computational models, especially when they become complicated. Replication effects model verification while the modeler is comparing the implemented model with the conceptual model. On the other hand, it affects model validation during the correspondence between the model and the real world. Computation replication often happens both in parameters space and time. Data replication is not the purpose of the modelers' perspective, but it could be the developer's issue [83].

Regards to the above perspective, many challenges are identified. The first challenge is modelers. Researchers challenge to optimize their use cases to set the large parameter space and cover them by submitting tens of thousands of simulations, although job scheduling. To achieve this goal, they put an extreme load on the system in many HPC sites [84] [85]. Monitoring and checking the results of these jobs has also been a very time-consuming process.

The second is about restrictions and limits for the number of jobs that a modeler can submit to the local queue. Providing many simulations to run per job submission is fatal for some local HPC sites [84]. Furthermore, there are intervals in the execution times, such as early termination of some jobs, which should be detected by the researchers to save the remaining time left on the CPU and fulfill the run time requirement for the other simulation jobs.

The third is sharing dynamic libraries that are required for the run-time on the execution machine. There are many restrictions for installing libraries on the infrastructure nodes, which means there is no guarantee that researchers can run the simulation experiments using all available nodes. The last challenge is analyzing the output files during the simulation to cut the computation loops and loss [84].

2.1 NetLogo Tumor ABM Model 1st approach

The first design proposed in this thesis was the definition of the agent-based tumor model of the present work is based on the Wilensky model [1] and considers the premises stated by Hallmarks of Cancer [5].

Some considerations about hallmarks are listed in [4] with the acquired capabilities of cancer:

a) Self-sufficiency in growth signals

b) Insensitivity to antigrowth signals

c) Evading apoptosis

d) Enabling replicative immortality

e) Sustained angiogenesis

f) Cells invasion and metastasis

Wilensky's tumor model [1] permits us to change the parameters that affect tumor progression, immune system response, and vascularization. Outputs included the number of living tumor cells and the immune system's strength, which control cells.

In this model, two kinds of cells are considered: stem cells and transitory cells. An agent represents each cell, and tumor cells can breed, move, or die where the time is measured in simulation steps (ticks). Initial tumor cells are blue, and the cells change their color depending on their age (different colors from the red palette) with the age of fewer than 16 ticks.

A stem cell can divide either asymmetrically or symmetrically during the mitosis. In this case study, it is considered malignant tumors where asymmetric mitosis is followed. In this process, the stem cell divides symmetrically into two stem cells, and the first child of this division remains static, but the second cell moves to distant organs to generate a metastasis.

The simulation presented cell control with different and constant immune responses by killing transitory cells, moving stem cells, and original cells. The tumor grows and starts to grow exponentially in the world size, with 60 immune cells present. The tumor grew linearly over time. Linear growth over the periods will be observed when many initial

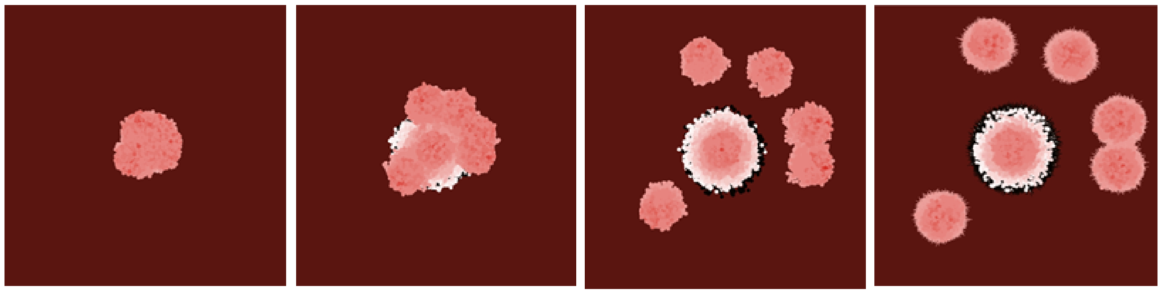


Figure 2.1: Stem cells in the context of evolution and metastasis visualization (from 6 original stem cells to23, 000-29,000 in a steady-state).

immune cells were used in the model. So, increasing immune cell numbers is associated with an impressive decrease in the tumor burden.

In this model, only metastasis (and not an invasion) is the mechanism of spreading cancer cells, reaching it to an organ different from the original tumor using blood or lymphatic paths. In the model, the metastasis is red and made of cells that die young. As the tumor propagate and get larger, stem cells reproduce and die younger.

To represent this behavior, the model includes parameters for modeling the number of initial stem cells, the growth factor to model the self-sufficiency of growth signals and insensibility to antigrowth signals, three values to represent the apoptosis grade (normal, low, very low), and a replication factor to model the replicative immortality and angiogenesis capacity of the tumor.

The simulated model represents a tumor outgrowth caused by symmetric mitosis of the stem cell. The outgrowth will turn into metastasis and grow in remote regions. By exploring the model, reveals a new dimension, and the preliminary simulation grasps tumor behavior, which is shown in Figure 2.1 and Figure 2.2.

Figure 2.1 shows the simulated evolution of six original stem cells (center) with a random position metastasis in new independent tumors with growth-factor=1.25, replication-factor=high, and apoptosis=normal. In this case, the six stem cells' evolution reaches, in a steady-state, a range of 23,000 to 29,000 cells (agents). Figure 2.2 shows the steady-state of

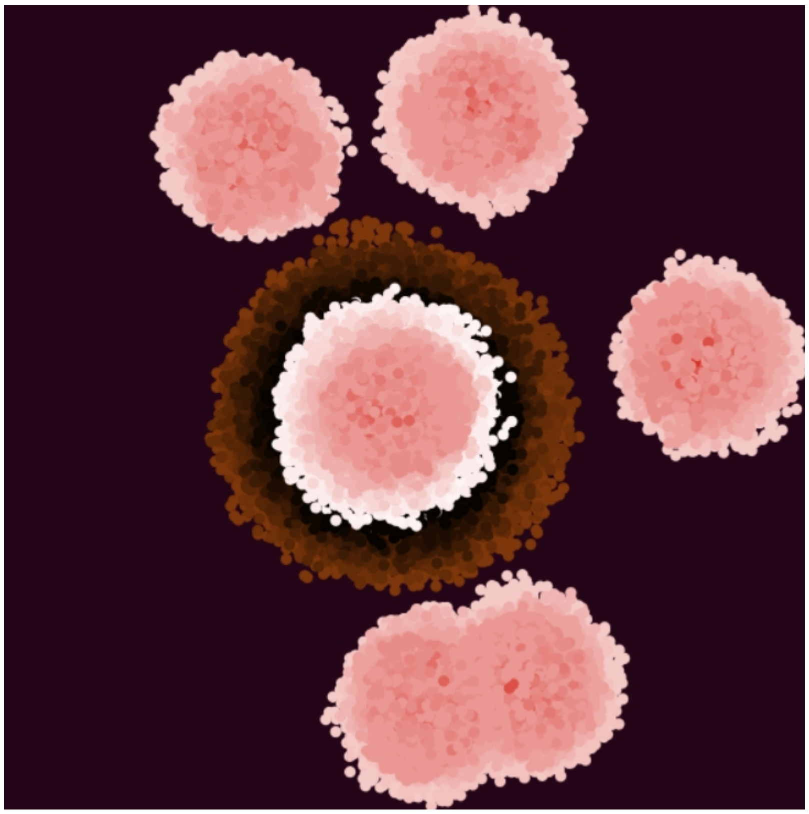


Figure 2.2: 6 stem cells, evolution and metastasis visualization with grow-factor=1.75, apoptosis=low and replication-factor=high (near 200,000 cells in steady state).

a tumor metastasis visualization with six stem cells and the grow-factor=1.75, replication-factor=high, and apoptosis=low. As can be seen, the growth of metastasis is more aggressive, and through reducing apoptosis, there is a more significant number of cells that do not die, amounting to near 200,000 cells (agents).

2.2 HPCNetLogo Framework

In order to provide an easy to use and user-friendly solution for non-technical users, we have designed and deployed a workflow using different tools, collaborating in the simulation of parametric ABM on an HPC cluster. The execution of this workflow is unassisted, and the initial user interaction is through an interface based on Web technology. In the local environment, the user configures his model, performs the local tests, sets up the Behavior Space, and, through a service, schedules his model and indicates where and how he wants

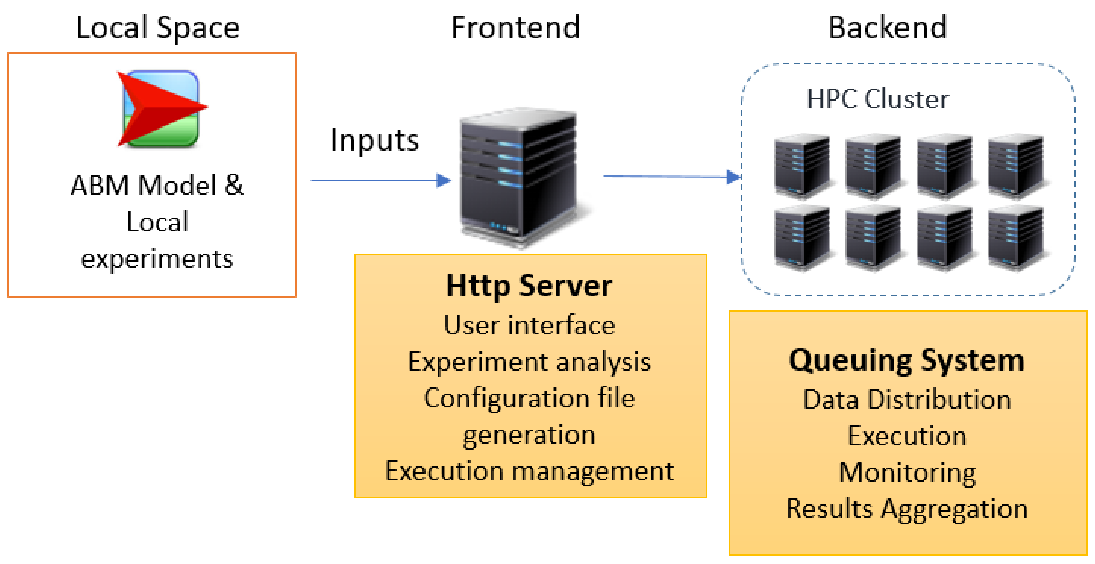


Figure 2.3: The architecture of the infrastructure environment.

to run. In the development environment, the Front-End service analyzes the user-specified model. This generates all configuration files required for the proposed simulation scenarios, creates the initialization files for the high-performance cluster queue system, and then sends them to Back-end. There the system executes the experiment, and when it finishes, the user can see the result through the same web interface. Figure 2.3 shows the proposed architecture.

The tool for Front-end infrastructure is implemented in Python. The front-end interface uses a splitter module to obtain the different simulation scenarios configured by the user within the Behavior Space [15]. This module (based on [16]) performs the separation of the different ABM simulation scenarios by generating a configuration XML file for each of the combinations of the variables defined in the experimental set indicated by the user.

These configuration files are a parametric simulation with a subset of this model's "data space" (scenario). Each simulation with its data set will run on an HPC node concurrently with other simulations with a different data set, reducing the user-defined data space's overall execution time. The implementation process in the proposed infrastructure consists of three stages:

a) Initialization and integration,

b) Deployment and

c) Execution.

In the first phase, the interface receives data files and the user's model. Then the interface back-end analyzes the experiments and generates configuration and initialization of XML files in the HPC queuing system automation.

In the second phase, the configuration and simulation data's deployment is made to cluster master node. Finally, in the third phase, the execution starts, and the back-end waits for the results to forward them to the user. In order to maintain the confidentiality of the data during the whole process, the interaction with the Front-end/Back-end is through secure protocols (HTTPS in the case of Fronted and SSH+PKI in the case of the cluster). Figure 2.4 shows the ABM simulation execution process. It explains the steps that are performed in the proposed infrastructure.

As shown in Figure 2.5, task 53 (jobid) is running (indicated by the red status icon), and task 54 has been deployed, but they have not been sent to execution as indicated by the green arrow in the state. To start the process, the user must click on this arrow, and this will start the execution of the model on the cluster.

Data splitting is used to be one of the phases of the parallelization of data-intensive applications in the scientific domain for executing distributed-memory architectures. Data splitting as a simple solution works where each necessary operation gets to the final result executed by a single worker, and each worker executed the same number of operations. There is an asymmetry of the relation between the number of workers and the quantity of data that is decomposed in the real world. This directly affects the performance of the solution [63] [64].

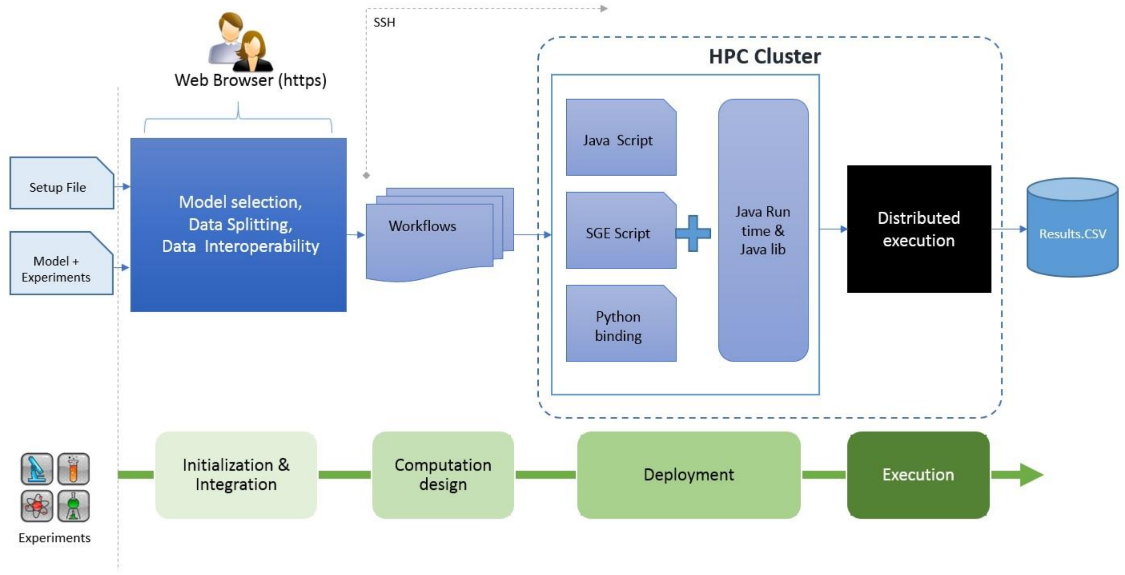
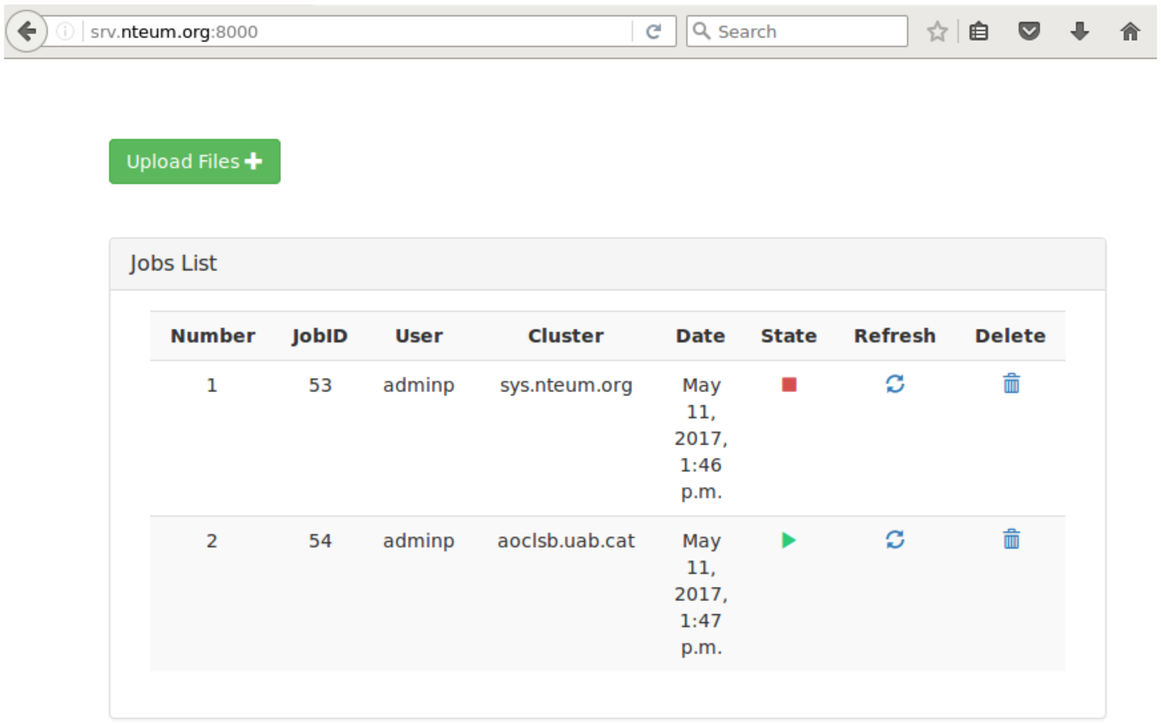


Figure 2.4: Stages of the ABM simulation execution process.



Number	JobID	User	Cluster	Date	State	Refresh	Delete
1	53	adminp	sys.nteum.org	May 11, 2017, 1:46 p.m.	■	↻	🗑
2	54	adminp	aoclsb.uab.cat	May 11, 2017, 1:47 p.m.	▶	↻	🗑

Figure 2.5: User interface.

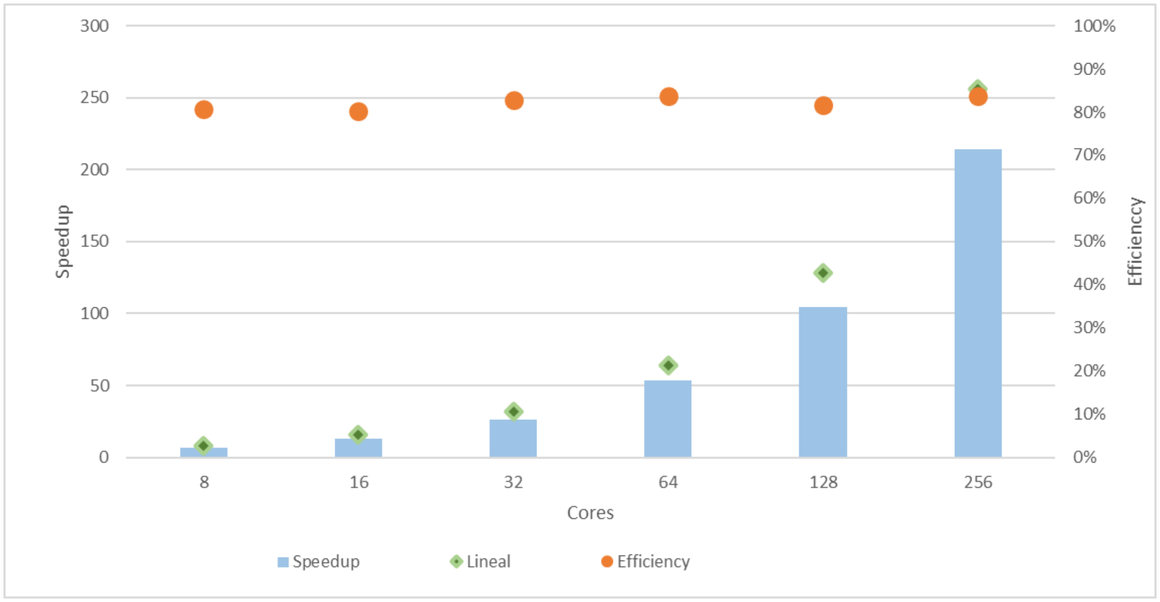


Figure 2.6: Speedup and Efficiency for 8 to 256 Cores.

2.2.1 Experiments and Results

A set of simulations with different parameters and values has been performed to explore, from the point of view of performance, the model's behavior (speedup, efficiency, and scalability) to analyze the developed model's behavior on an HPC cluster. These experiments are the proof of concept considering HPC performance where the scalability, number of agents, and performance of the model are analyzed but not the simulation result as a growth factor, apoptosis, angiogenesis, or other exciting values for an oncologist or researcher in tumor propagation (however these are available as simulation results). Figure 2.6 shows the speedup and efficiency of these executions. In this figure, the speedup is close to linear in the lower number of cores and gets reduced towards the higher number of cores. Although the speedup and efficiency of about 80 percentages seem promising, the main issue is the memory contention problems at the Java run-time (Netlogo) for higher core configurations and the I/O from the network file system (NFS).

Figure 2.7 shows how scalability effects of the model over the speedup and efficiency considering a grow-factor equal to 2.0 and the speedup compared with the values of the

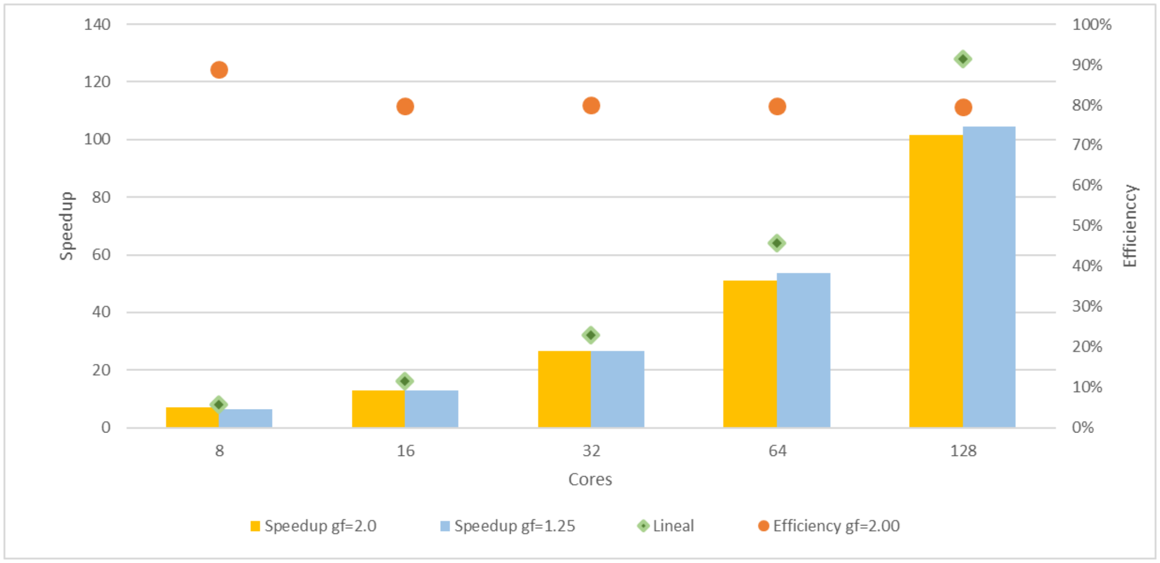


Figure 2.7: Speedup and Efficiency for 13,000 and 110,000 Agents

grow-factor equal to 1.25. This growth-factor, in this case, study, implies that for the value of 1.25, there are about 13,000 agents, and for the value of 2,00, there are about 110,000 agents. As can be observed, the speedup and efficiency are affected by the grow-factor equals 2.00, but it keeps on showing the value's performance.

Figure 2.8 shows the impact of the shared resources at the Java runtime (NFS, memory) during the model execution for the larger number of agents: 30 initial stem cells with a growth-factor equal to 2.00 that arrives at 361,000 agents (cells). In this case, the figure shows the distribution of the execution time in a box and whisker plot. As can be observed, the model is executed in a fixed number of time units (1,000) at the eight cores and runs about 2,234 seconds (average), but on 64 cores, it will be run about 4,417 seconds (average).

Figure 2.9 shows the model's scalability for different numbers of stem cells (from 6 to 60) and to values of grow-factor (1.25 and 1.75). As can be observed, the scalability of the model is linear. In this experiment, the simulation shows that in the steady-state (1.000 simulation steps) for the grow-factor of 1.25 and 6 initial stem cells, the tumor arrived 6. 6k agents (cells). For 60 initial stem cells, the tumor arrived at 122k agents (cells). For growth-

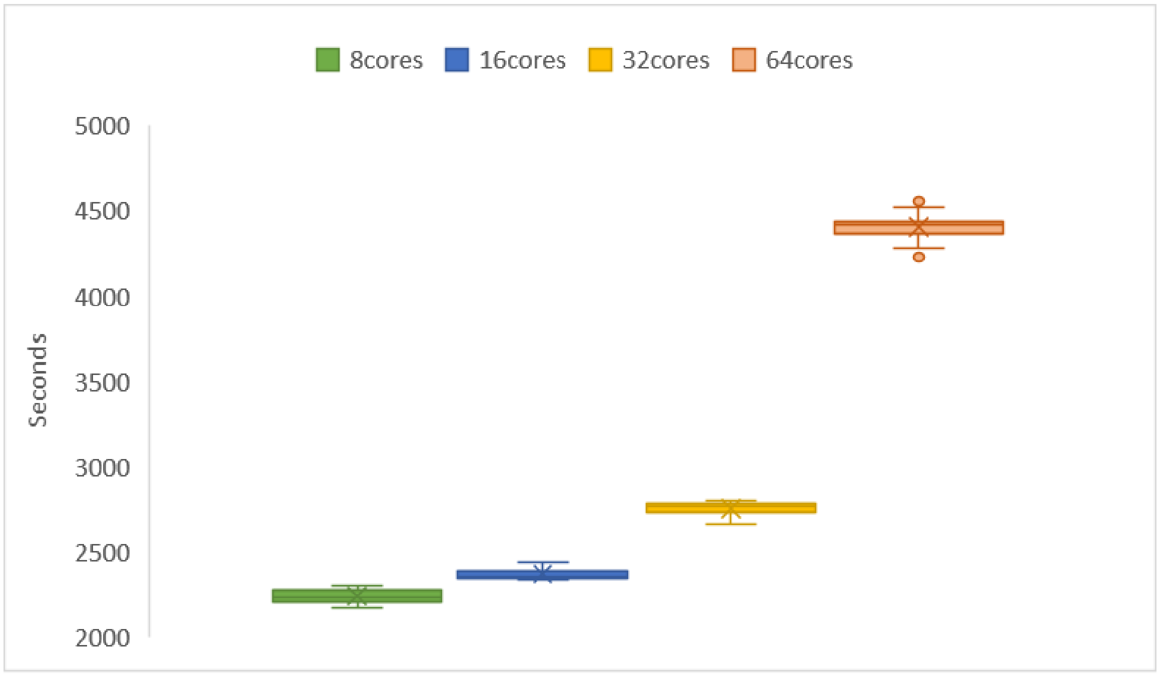


Figure 2.8: Box Diagram of Execution Time for 361,000 Agents.

factor of 1.75 and 6 initial stem cells, the number of agents (cells) was 76k, and for 60 initial stem cells, the number of agents (cells) augments to 404k. It is necessary to remark that employing HPC is extremely necessary because this type of analysis is time-consuming, as can be seen. At these proofs of concepts, only the minimum number of repetitions to obtain stable statistics data which have been executed, and only 1000 simulation steps have been performed. It is important to consider that in other configurations/parameters may be necessary to increase the number of simulation steps to justify employing HPC for this type of experimentation.

Implementation

The core of the proposed solution is sequential python scripts, which call commands, modules, and libraries that, without user intervention, generates all the necessary automation and adapts the user's requests to the HPC environment. It is a web development framework that handles user/modeler requests triggered by actions such as run their model from

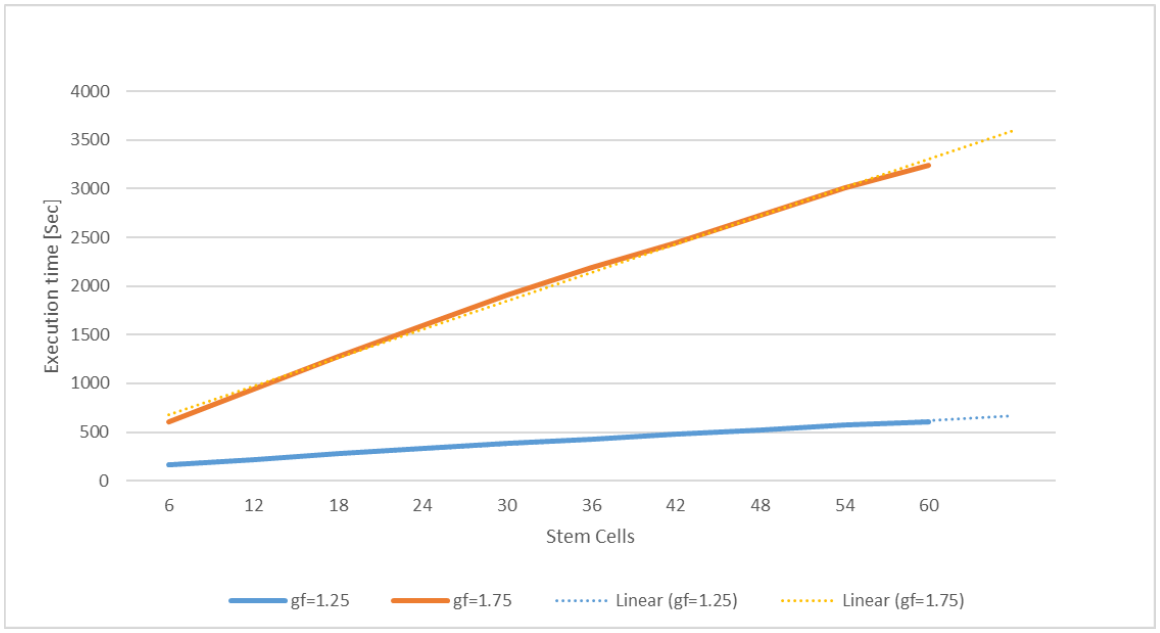


Figure 2.9: Simulation runtime (average) for growth factors 1.25, 1.75 and from 6 to 60 stem cells

a website. Those requests activate commands to run the shell for access to the clusters and modules which generate fragmented data set.

The development environment dynamically and automatically creates shell scripts for each pass and experiment. It is created based on the condition of the experiment. The scripts created on the master node would safely run the experiment and update the go's progress. The algorithm and its different stages of the above pipeline presented in algorithm 1.

Using NetLogo on an HPC cluster is not necessarily tricky, especially if administrative support is available. Doing so requires installing NetLogo and Java Runtime Environment, so they are accessible to all nodes; and a batch file that specifies the path to the NetLogo and Java directories, the path to and name of the NetLogo file, and the Behavior Space experiment to run and its options (e.g., output file name and format). It is essential to use the ”table” output format because the alternative ”spreadsheet” format stores all result in memory and consume all available memory.

In an HPC cluster, Behavior Space experiments are run in ”headless” mode, that is, without any graphical user interface. Documentation and tools for using NetLogo headless and on an HPC cluster are available in the NetLogo User Manual’s Behavior Space section [104].

Algorithm 1 NetLogo jobs to the cluster-nodes algorithm

```
Input: ($Model$, $Experiment$, $credentials$)
Output: (Simulation results as csv)
SPLIT $Model - Experiment$ pair by $n$
CREATE $n$ sets of $setup - files$
UPLOAD $Model$ and $n \times setup - files$
CREATE output directories in server
UPLOAD $n \times scripts$ for $n \times setup - file$
CALL submit $\{scripts\}$ = qsub NetLogo JOB to Cluster-MasterNode
CALL agents
while <busy> do
    wait!
end while
check stats agents
RECORD {stats}
DOWNLOAD {results}
DOWNLOAD {stats}
MERGE {results} to {stats}
EXPORT {stats}
EXPORT {results}
END
```

For the HPC environment, in this case, an SGE (Sun Grid Engine) Cluster has been used but can be easily adapted to other HPC environments such as Scrum, PBS, Etc. The modelers use their secure credential and individual identifications to get a sign in, and their SGE keys (cluster access) are encrypted. The SGE commands automatically run after receiving the output XML files, which have been produced by the splitting method from NetLogo behavior space experiments. At this level, user communication to the server is no more needed, and the SGE allocation process will run in the background.

The complete code is available in Appendix A and can be found in the following URL link: https://github.com/hpcnetlogo/hpcnetlogo.

2.3 Conclusion

Considering life science and systems biology(SB) data and network analysis, such as the simulation and analysis of NetLogo's tumor growth behavior, it is difficult for researchers. Because almost everything is about cell communication and dependencies in such a model, and it is not easy to identify insights. The processing could be prolonged and takes days to finish experiments that depend on the tumor's size. Graph-based representation of the biological networks makes large amounts of data dependencies more accessible and multi-dimensional. The next chapters describe how the model and the tools develop and translate into a more dynamic structure in a different framework. The accepted publications with the peer review for this chapter are P1 and P2, mentioned in section 4.3 of the thesis.

CHAPTER 3
STATISTICS, DATA ANALYSIS, SIMULATION, AND VISUALIZATION

The most novel revision on the ground of the mitosis scenario of tumor growth in this chapter of the thesis is modeling graph-based Python representation of a tumor network growth. The past significant changes simulate the growth behavior in multi-state and multi-scale components based on graph theory, state machine theory, and algorithmic state machine (ASM) method nested in the agent-based model [31].

Choosing Python would make a simulation of agent-based modeling easier for people in academia or who may not have experience coding. The speed, built-in functions for recording variables during simulations and exporting them easily to HTML, XML to view results, easy parallelism for running multiple simulations at once, simplifying object-oriented programming (OOP) elements having beginner-friendly syntax all are the reasons for this choice.

The initial design aspect comes from the acute inflammation based upon angiogenesis controlling rules and critical factors of the metastasis, which feeds a wider parametric variety of input into the system.

3.1 The Tumor Simulation using Python+Mesa

Analyzing a new model using the Mesa (ABM framework) in Python relies on simulating agents' interactions to evaluate their effects on each other and the system. It is often used for prediction and detection in complex phenomena. Mesa is the agent-based python project, which has started in recent years, and it has taken into consideration very fast among researchers. Mesa is modular, and the meaning is that it is modeling, analysis, and visualization components are kept separate but intended to work together.

Modules are model and agent classes, a scheduler to determine the sequence in which the agent acts and space for them to move around. The Analysis Tools collect data generated from the model by running it multiple times with different parameter values and the visualization classes for launching an interactive visual model, using a server with a JavaScript interface.

For studying each model's behavior under different conditions, the scientist needs to collect the relevant data of the model while it is running in which the ”Data Collector” class is defined for this task. Running the model with different string points in Mesa determines in a Batch Runner class. Data collection and batch running are implemented in the appropriately named analysis modules [31].

Mesa's adapting modules allow scientists to make changes to existing ABMs to conform to future framework requirements. Also, monitoring the data management issues while processing actions is happening in parallel seems facile in Mesa. Each module has a Python part, which runs on the server and turns a model state into JSON data [113]. Figure 3.1 shows the chart of the adapting modules of Python+Mesa in this Simulation.

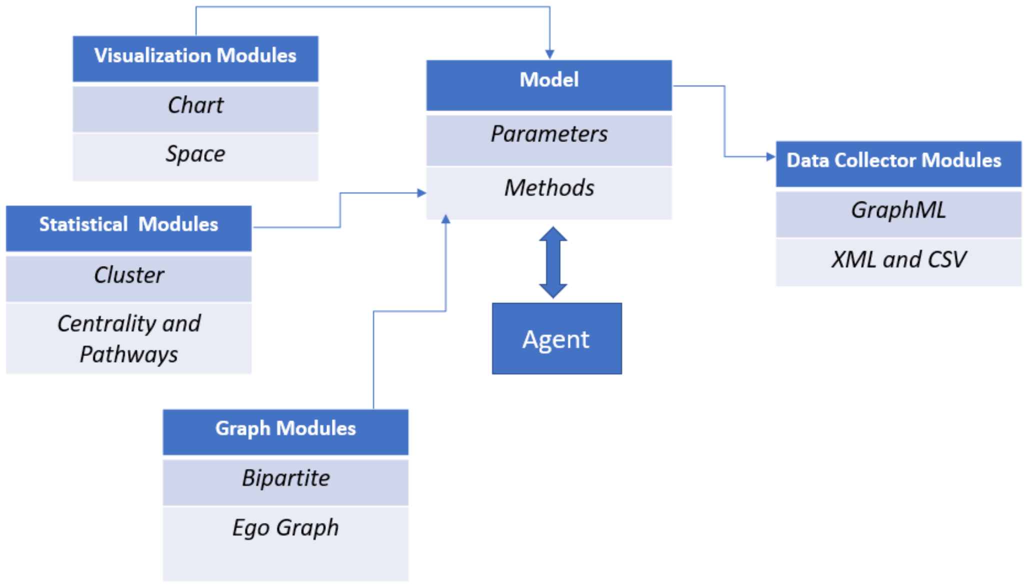


Figure 3.1: The chart of the adapting Python+Mesa modules in the new agent-based model.

The idea of re-implementing the tumor model is to be the Python 3-based counterpart to NetLogo.

Since Tumor progression is a complex multi-stage process in which several distinct properties have to be acquired by the tumor cells sequentially, or in parallel, the first problem using the NetLogo or another software framework for simulation is modeling a behavioral space which supports multithreading.

Considering the performance is limited to the number of cores at the local infrastructure, this limitation expands the local processing problem. The extensive parametric simulations need HPC clusters. It is revealed that an open framework is needed which does not have scalability issues to design an extensive network. The framework is to reduce the necessary time and explore a limited model data space and distribute the model for exploring the mega model by digging into the microenvironment scales.

The initial design aspect of a multi-state architecture of the tumor growth model comes from the acute inflammation based upon the key factors involved, such as angiogenesis [10].

Tumor angiogenesis is critical for tumor growth and maintenance. The model strategy must begin with an initial identification of a minor population of cells with the characteristics of ”tumor-initiating” cancer stem cells. These cancer stem cells in the assumed tumor reside close to blood vessels.

Therefore, choosing an angiogenetic switch for changing the state of the model must balance the dynamic. Angiogenetic and anti-angiogenetic factors are essential values that could be chosen for Data-Driven Decision Making (DDDM) involve collecting data based on measurable facts and calculations, analyzing patterns and facts from these insights. In the case of anti-angiogenetic, there will be the probability of a quiescent tumor. Inside the agent-based model, tumor cells could be affected, inflamed, and turn quiescent. Considering the angiogenetic and anti-angiogenetic, the simulation of the evolution of metastasis and measuring the tumor volume ratio and the dynamics of the tumor under the influence of variation of these factors are implemented.

As a foundation of the tumor model, a simple random graph agent-based on network topology is a simple solution but selecting the most suitable topology is the matter. Barabasi-Albert (BA) model [38] is a scale-free network, and it is one of the most basic models since it describes most of the biological networks, especially from evolutionary models. Scale-free networks are a network that includes few nodes highly connected to other nodes in the network. These nodes present a much higher degree than most other nodes in the network.

The Barabási–Albert (BA) model generates random scale-free networks using a serial degree distribution among the nodes. The random generator assigns N nodes, but first, it begins with a small set of connected nodes and then adds nodes, one at a time, until the model gets N nodes.

Adding each node at a time connects them to a small number of existing nodes with degrees. As a result, nodes with higher degrees (the primary nodes) tend to get an even higher degree. Since it cannot fulfill the cell interaction structure or helping the simulation of the behavior of the angiogenetic switch within the tumor and also creating connections are time-dependent during the simulation, then selecting the Erdös-Rényi [37] topology seems a better choice.

In comparison to the Barabasi-Albert (BA) model, The Erdos-Renyi network assigns N nodes at first. It then connects each pair with probability P. This means that no nodes will have a much higher degree than the others. The growth is more open to the probability rules of angiogenetic and anti-angiogenetic factors. The pseudo-code generating a random network based on the Erdos-Renyi network is described in Algorithm 2. Thus, for P as a Probability and edge creation in n number of nodes, if (Ps> log (n) /n)) almost all vertices are connected, and this function returns a directed graph. Erdös-Rényi model takes a number of vertices N and connects nodes by randomly selecting edges from the (N (N-1) /2) possible edges.

Algorithm 2 Pseudo-code generating a random network

```
Input: (n, p, p*)
Output: (True, False)
for n in enumerate(nodes) do
    for p in enumerate(Ps) do
        if $p<p^*$ then
            $p_{connected}(n, p) = 1$
        else
            return 0
        end if
    end for
end for
```

3.1.1 Visualized Results

As shown in Figure 3.2, The graph agent-based tumor model generated by Algorithm 2 in the Mesa framework, shows the cancer stem cells (red metastatic) connected on the black edges to the other stem cells (green) based on the probability of angiogenetic factor. The gray dead cells were also created based on a quiescent tumor's probability based on the anti-angiogenetic factor. The interactive visualization in Mesa helps to identify insights and generate value from connecting data.

The strength of using the browser user interface (BUI) of Mesa is an interactive data visualization which allows the user such as an oncologist to see the model running in the browser and second is setting up the data, parameters, figures, and plotting while exploring the model data space. Figures 3.2 and 3.3 show the samples.

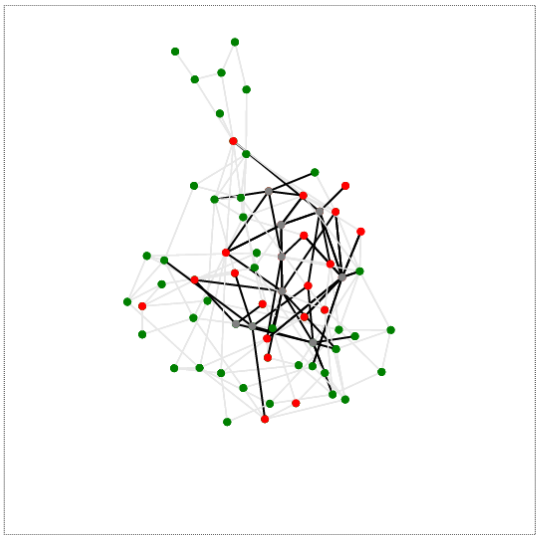


Figure 3.2: Graph visualization for three states (normal, dead and metastatic) which has shown in three colors.

Python data visualization provides strong support for integration with several technologies and higher programming productivity across the development lifecycle compared to just using agent-based modeling software.

The front-end integration is the strength of Python modeling based on packages such as an ABM package (Mesa), analysis, and visualization packages (Numpy, SciPy, Matplotlib). On the other hand, the simulation is getting more orchestrated. At the same time, it is united with machine learning packages (NetworkX) or distribution utility modules and third-parties (Pypi) and container facilities in deep at the back-end.

Figure 3.3, the generated tumor shows the changing state of a tumor metastasis with 1200 stem cells. From the biological perspective, the plot implemented in figure 3.3 shows the changing scale of inflammatory (red line) because of angioprevention interference. The gray line in the plot shows the increasing number of dead cells. Figure 3.3 shows the ratio of the dead cells to the inflamed cells. It has also demonstrated different tumor growth behavior upon the good laboratory condition from the angiogenic switch.

The transitory cells are dynamic intermediate states of the agent called inflammatory because of angioprevention. However, since the figure is the tumor's final image, they all changed to dead or metastasis.

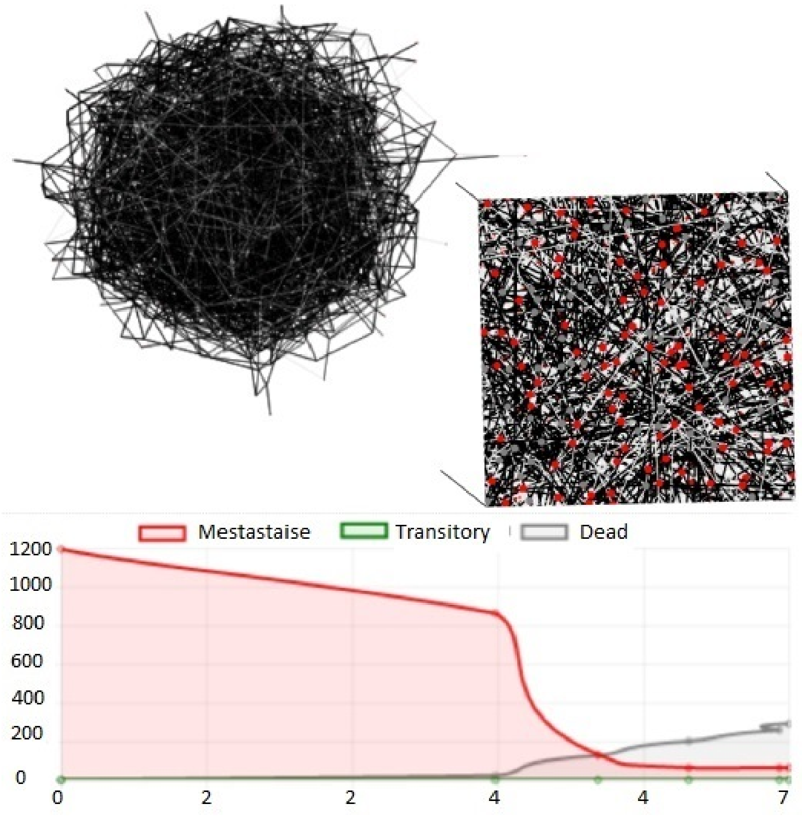


Figure 3.3: Graph agent-based visualization of tumor growth to 1200 cells.

3.2 Simulating Tumor Growth

The data exploration of a multi-scale large parametric model needs incorporating data science, graph analytic, and machine learning. Then, we proposed a simulating tumor model and evolving behavior by modeling based on graph theory and statistical analysis on different scales to create the cell life cycle. The following figure 3.4 shows the four steps (a, b, c, and d) for simulating the tumor growth model and evolving analysis.

The first step is simulating an initial tumor by setting up initial input features, including normal cells and cancer cells. Since two disjoint sets of input features have no common element, the initial pattern is better to be a bigraph.

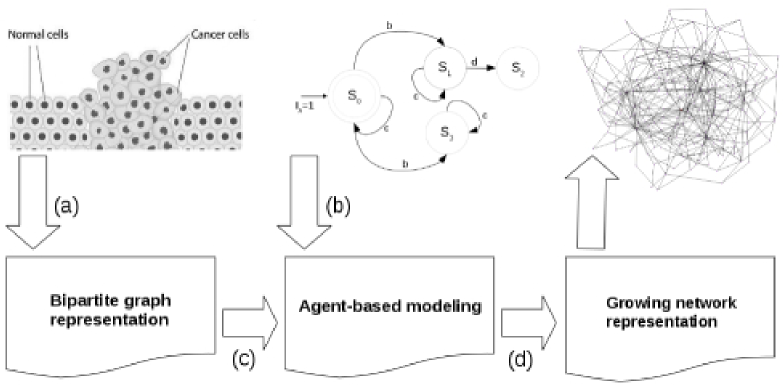
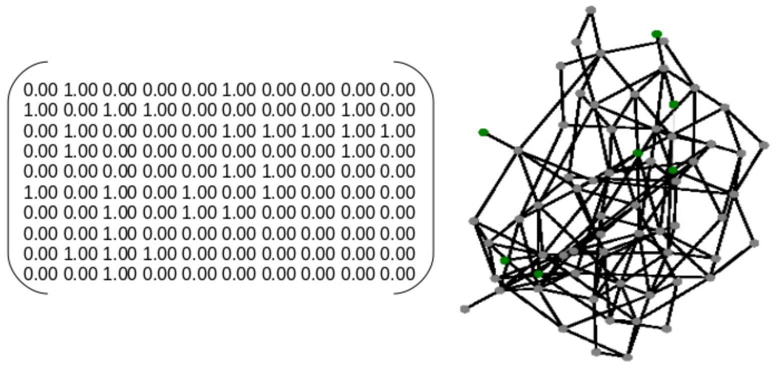


Figure 3.4: The scenario of simulating tumor growth model and its evolving analysis.

The second step is nesting the initial bi-graph in an agent-based model, which feeds from a probabilistic finite-state angiogenic switch model that acquires the acute inflammation.

Forwarding the initial bipartite graph agent-based model into the growing module, generating a directed graph with the given degree distribution is the final step of the workflow to represent a growing tumor. The following is a description of each step.

a) Bipartite graph representation: The bi-graph normally is used to simulate disease spread, especially detecting cancer in a behavior stated way except on a very large scale.



$$\begin{pmatrix}
0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
1.00 & 0.00 & 1.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
1.00 & 0.00 & 1.00 & 0.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.00 & 1.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00
\end{pmatrix}$$

Figure 3.5: The bipartite graph representation and its corresponding adjacency matrix.

An initial tumor is simulated by setting up the graph in two different data classes (Normal Cells, Cancer Cells) to create a bipartite graph model as the tumor's static scale. Figure 3.5 shows a sample of bipartite graph representation and its corresponding adjacency matrix.

b) Finite-state machine: The fundamental biological aspect of the probabilistic finite-state model of this tumor growth comes from the acute inflammation based upon the critical factors involved, such as an angiogenic switch. Tumor angiogenesis is essential for tumor growth and metastasis. The angioprevention factor threshold is compared with the assessment values of transition probability selected by oncologists interactively. They can verify and revise the parameters while the experimentation is ongoing.

   The comparison works as a trigger to change the cells' state from their current state to the proliferation state or the inflammatory state. Afterward, under the influence of evolving angiogenic switch values, the inflamed state may turn to the progression state, and metastasis happens. Since the stem cell quiescence is a way to control the inflammation in the tumor microenvironment, the simulation considers probable immune state at the state model. The way is targeting inflammation using angioprevention to stop cancer cells from moving to the proliferating state. Figure 3.6 shows the adopted finite-state machine for Angiogenic switch control in this tumor growth model.

   As shown in the figure, the angioprevention method causes proliferation by blocking angiogenesis in one state. For causing the other state, which is inflammation of the cells, it needs to switch to an angiogenic state. The inflammation state coming after that causes the local invasion and progressive metastasis through lymphatic and vascular.

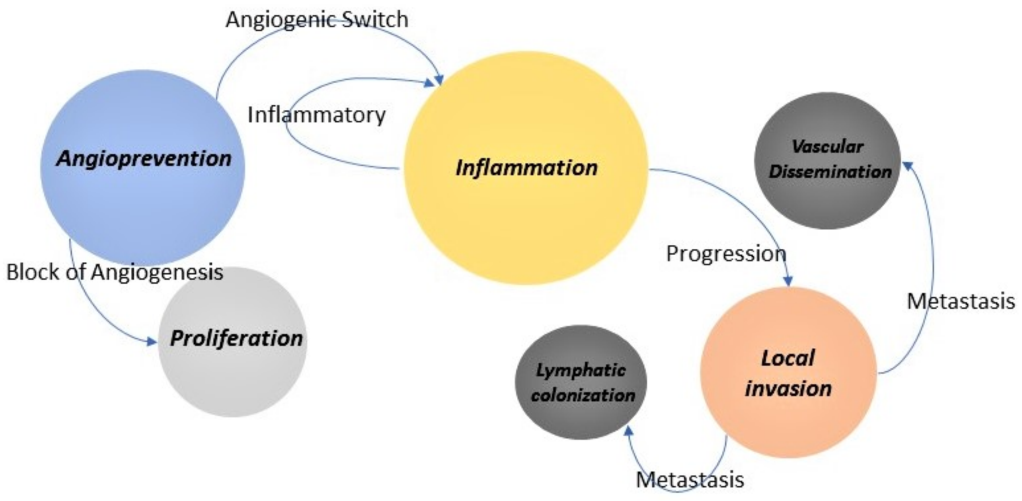
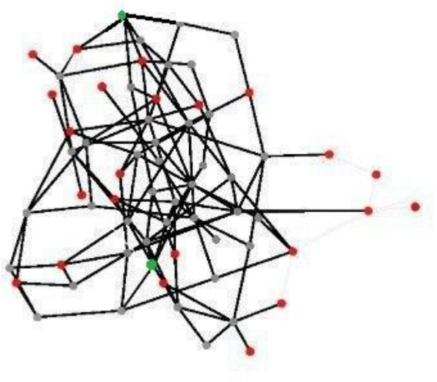


Figure 3.6: Finite-state machine for Angiogenic switch control in tumor growth model.

c) Agent-based modeling: The model has two core classes. The model has two core classes. One is the tumor model states, attributes, and scheduler component, and the other is the tumor age.Four states in the tumor model class are defined as metastasis cells, dead cells, normal cells, and inflammatory cells. The graph generator and the clustering method for sub-graphing are returned into the tumor model class.

   The inflammation of the cellular neighborhood based on Angiogenic switch rules and the help of the Finite-state machine probabilities, all selected and verified by oncologists interactively to simulate the tumor's growing scenario model is created at the tumor agents class.

   In Mesa framework, for studying the model's behavior under different conditions, it has needed to collect the model's relevant data while running in which the Data Collector class is defined for this task. The tumor model's data collection is in the tumor model class and extracts the CSV or XML files' data. Figure 3.7 shows the reduction of metastasis state of the cells affected by angiogenic control factors in the time. In this experiment, the normal cells, Dead cells, and metastasis cells are classified in three colors in green, gray, and red.



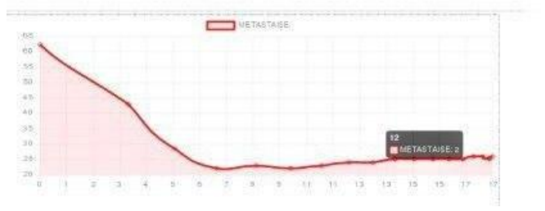


Figure 3.7: The reduction of metastasis state

d) Tumor network growing representation: Tumor growth curves are used to derive the Tumor Control Index (TCI), such as tumor progression, rejection, and stabilization. It is essential to combine both rejection and stabilization challenges for showing the potential of extensive parametric and multi-scale agent-based modeling under the influence of different control factors.

   For simulating the growing behavior of the simulated tumor model, it is needed to define some data analysis for computational statistics such as calculating the tumor volume using degree distribution. The degree distribution is the probability distribution randomly is considered in the tumor model for a cell reached to other cells in the community. The final model returns an evolved graph with the given degree sequences by generating a directed graph in the tumor model class based on the initial graph data.

3.2.1 Statistical Analysis and Results

There is a need for statistical analysis that leads to volume ratio measurements of the tumor model, as it could be seen in figure 3.8.

Traditionally, calculating the tumor volume (mm3) is based on the presented formula 3.1 using tumor width (W) ,and tumor length (L):

$$TumorVolume = W^2 \times L/2 \tag{3.1}$$

This calculation, which works pretty well for clinical issues, was made based upon the assumption that solid tumors are more or less spherical like the version of Netlogo Wilensky's tumor model, but not proper for the metastatic form due to the phase transition and spreading dynamics.

By generating a random network with a given average degree (K) and initial size of the tumor (N), we could construct a degree sequence (m) and it is presented in the following formula 3.2:

$$1/2\sum_{i=1}^{N} K(i) = m\binom{n}{2} = N, K = 3,4,5,6,7,8 \tag{3.2}$$

For volume calculation, it is assumed that every cell inside the tumor has three states (normal, dead, and metastatic) that correspond to the tumor by edges. With the given K degree from three to eight and N initial nodes whose degree sequence of (m) could lead the simulated model to calculate the tumor volume ratio and obtain the plot. Algorithm 3 is a pseudo-code to simulate the degree sequence dynamics under the influence of the mentioned factors producing tumor volume in a graph network.

Figure 3.8 shows the results of executing the simulated model with different angiogenesis control factor in (0.1 to 0.9) for 60, 360, 650, 100, and 1200 cancer stem cells (CSCs) to plot different tumor volume (mm3).

Algorithm 3 The pseudo-code producing tumor volume.

```
Input: (P, N, K=3)
Output: (Tumor Volume)
while p_{connected}(n, p)=1 do
    for N in nodes(G) do
        if $p\binom{n}{k}$ < Control Factor then
            Return m/n
        else
            return 0
        end if
    end for
end while
return G
```

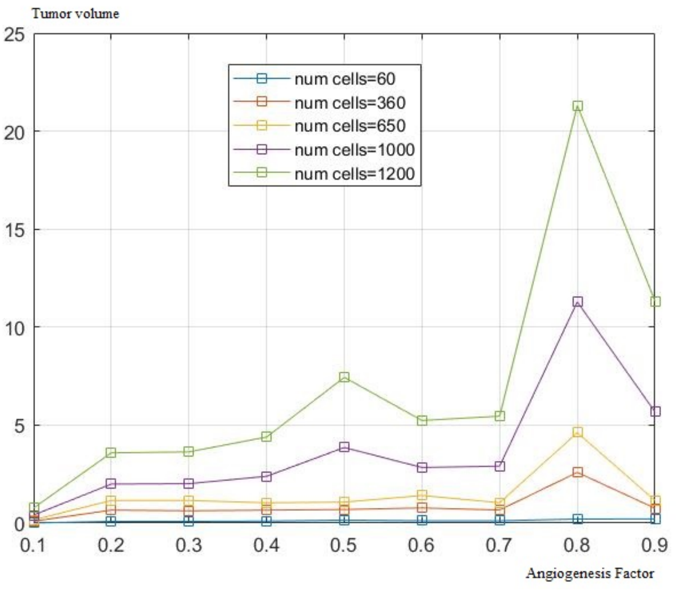


Figure 3.8: Tumor Angiogenesis volume ratio for 60 to 1200 cancers stem cells (CSC).

3.3 Simulation and Computational Analysis Multi-scale Tumor Growth

Following the dynamic implementation of tumor growth, network analysis algorithms are mainly concerned with network community identification [115]. The concept of community is common, and it is considered as the classification of objects in categories. A community is defined as a subset of nodes (as cells) within the graph. Such connections between the nodes are denser because of identity or dependencies than connections with the rest of the network.

The investigation of community structures in biological networks needs studying functional studies in metabolic networks [115]. Evolving the cellular biological network to the molecular level developing the tumor model to the other scale is like zooming in to get more profound insights. Moving from the tumor's cellular level to the molecular level needs to transform the growing bipartite graph to a scale-free molecular format, which is a better simulation of cellular interaction and molecular interconnections.

The simulation must illustrate an evolution of tumor growth in different patients and the analysis of their data. Let us assume that an oncologist needs a growing analysis of a patient's cellular interplay with a newly diagnosed cancer disease to develop the modeling workflow's evolutionary idea. Therefore, the tumor's basic level must be characterized for future prediction of the possible growth behaviors.

Figure 3.9 proposes a computational simulation system's workflow besides the modeling workflow about a scenario of tumor growth. The idea is preparing data from different scales and stages of the tumor metastasis evolving behaviour.

As can be seen the part (ii) in Figure 3.9, it is defined as a three-step scenario for a workflow. As the scenario of simulating tumor growth model and its evolving analysis explained before in figure 3.4, the first step simulates an initial tumor by setting up initial features to create an initial graph model as the scenario of simulating tumor growth model and its evolving analysis. Then the second step is forwarding the initial graph to a growing module.

The third step is feeding the growing tumor by changing the subset features of the angiogenic switch. The last two steps of the scenario as mentioned earlier are implemented as a growing network module with redirection (GNR) [114] nested in the simulated tumor agent-based model.

The growing information network with redirection (GNR) is defined as nodes connecting new links through *n* redirection. The probability for a node to connect a link depends on the number of paths of length *n* arriving at this node. When a new node is added to the graph, it connects to a randomly selected node or the n ancestor of this selected node. The ancestor is found by following *n* links from the randomly selected node. The growing module with redirection is used to transition to a phase or scale where evolve is happening. Also, it is useful for analytical predictions for the behavior of the degree distribution in a graph.

The subset features of the angiogenic switch define as the state transition based on the probabilistic state machine. It is considered that the probability of the transient state adopted in the subset features is necessarily valued between 0 and 1. This fuzzifies all input values before executing the growing rules to get the different behavioral patterns as the fuzzy output. Finally, there is a designed data visualization tool for the oncologists in the back-end computational part of the workflow. The next chapter will explain how the workflow's computational part is incorporated with network-based machine learning techniques to make the back end of this workflow ready for distributing on the clusters.

It is deployed two complementary graphs-driven methods for analyzing and estimating probable growing network patterns. The amount of data which is extracted from the methods address some problems in cell biology. A fast binomial graph generator is chosen on Erdös-Rényi topology for initialization, so the static preliminary model is used as an initial input.

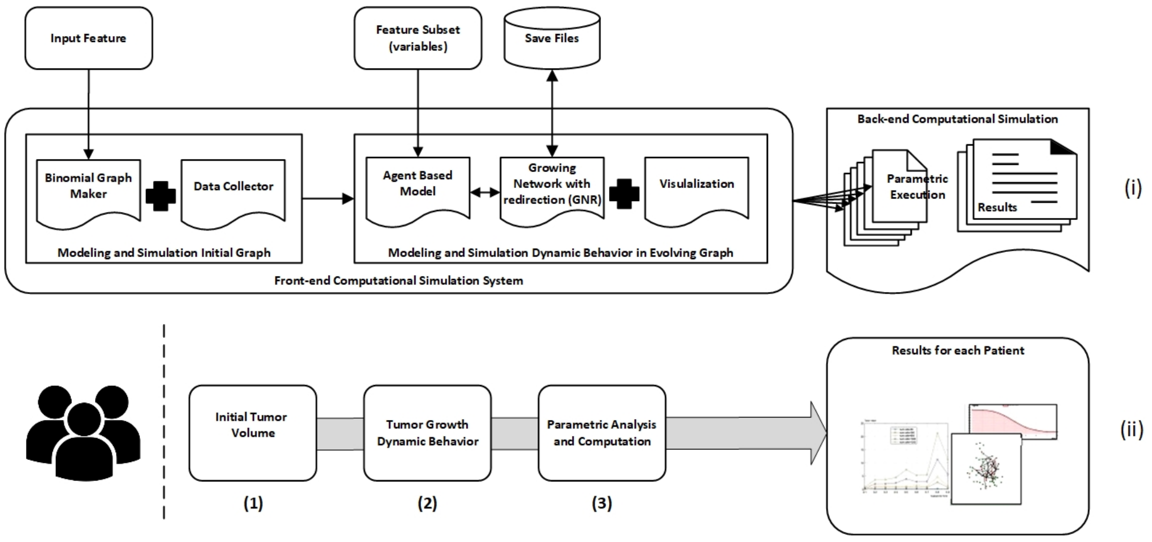

Input Feature
Feature Subset (variables)
Save Files
Back-end Computational Simulation
Binomial Graph Maker
Data Collector
Agent Based Model
Growing Network with redirection (GNR)
Visulalization
Parametric Execution
Results
(i)
Modeling and Simulation Initial Graph
Modeling and Simulation Dynamic Behavior in Evolving Graph
Front-end Computational Simulation System
Results for each Patient
Initial Tumor Volume
Tumor Growth Dynamic Behavior
Parametric Analysis and Computation
(1)
(2)
(3)
(ii)

Figure 3.9: A workflow of a computational simulation system for modeling a scenario of tumor growth.

To simulate the growing network nested in the agent-based model, a growing network with redirection (GNR) graph with probability (p) helps add nodes one at a time with a link to the initial nodes. In this graph, a target node is a node where a new link is attached. Target nodes are selected randomly following a uniform distribution. The redirection probability (p) is set, as shown in formula 3.3, which gives a new tumor growth pattern. Based on [116], the following equation in formula 3.3 uses to calculate the probability (p) of tumor growth. It contains five biological parameters, which are known as a cancer driver: the number of divisions (d), the number of stem cells (N), the number of critical rate-limiting pathway driver mutations (k), the mutation rate (u).

$$p = 1 - (1 - (1 - (1 - u)^d)^k)^N \tag{3.3}$$

All the subset features get in the computational simulation system through agents as state transition probabilities to change the state of the cells and affect the growing module.

3.3.1 Probabilistic State Machine Tumor Growth

Using probabilistic automaton (PA) in computational biology can be a useful aspect of tracking a natural problem. Figure 3.10 shows the metabolic flowchart of this aspect and how the probabilistic finite-state automata (PFA) for modeling state machine is created based on the flow chart.

The strategy begins with the initial identification of a minor population of cells with the characteristics of "tumor-initiating cancer stem cells," They will be assumed inflamed or dead under the influence of angiogenic switch factors. The inflammation triggers the tumor progression, and the dead cells introduced the proliferation stage in this cycle. All the transition states translated into the format of the PFA model.

The threshold of angioprevention K is compared with the assessment values of transition probability P_A, selected by oncologists interactively. The comparison works as a trigger to change the state of the cells from their current state S_0 to the proliferation state S_1 or the inflammation state S_3. Afterward, under the influence of changing angiogenic switch values Ω, the inflammation state S_1 may turn to the progression state S_2 and metastasis can happen.

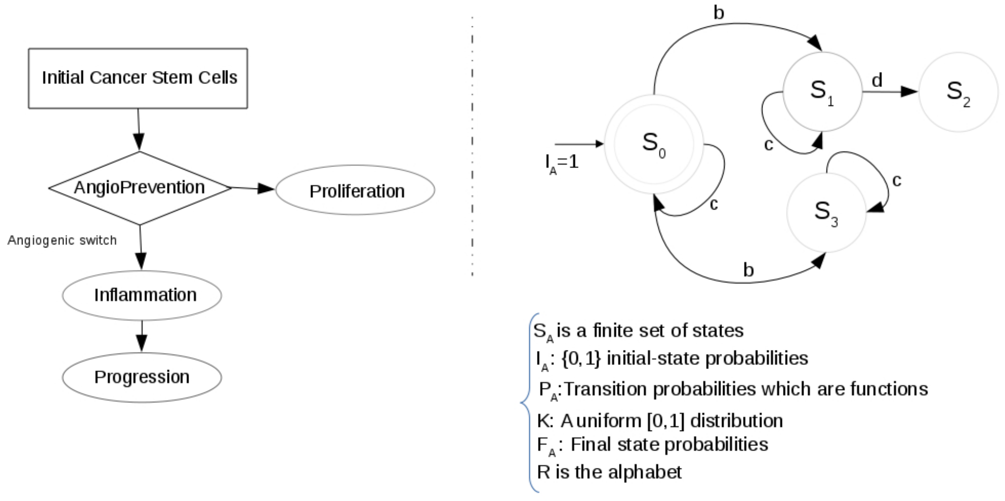

Initial Cancer Stem Cells
AngioPrevention
Proliferation
Angiogenic switch
Inflammation
Progression
b
d
S_0
S_1
S_2
S_3
I_A=1
c
c
c
b
S_A is a finite set of states
I_A: {0,1} initial-state probabilities
P_A:Transition probabilities which are functions
K: A uniform [0,1] distribution
F_A: Final state probabilities
R is the alphabet

Figure 3.10: Flowchart representation of the tumor behavior in a PFA model.

From the probabilistic point of view in equation 3.4 and 3.5, a finite generic state S_A is transitioned under the influence of angiogenic probabilities defined by alphabets Ω. K is a threshold drawn from a uniform [0,1] distribution. If the transition probability P_A is greater than K, the current state assumed to be extended with $P_A(s, b.s')$. In the End, initial-state probability I_A, final state probability F_A, and transition probabilities P_A are considered total, and *PFA* definition will be a tuple as it is shown in equation 3.5.

$$\sum_{s \in S_A} I_A(S) = 1 \tag{3.4}$$

$$F_A(s) + \sum_{s' \in S_A, b \in \Omega} P_A(s, b.s') = 1 \quad \text{where} \quad \forall\, P_s > K_i \tag{3.5}$$

3.3.2 Results of Parametric Execution

Human-tumor-derived cell lines contain familiar and different, transforming genomic profiles, essential for a comprehensive understanding of tumorigenesis and identifying tumor evolution's earliest events [121].

four different parametric baseline executions are assumed for monitoring tumor-growth model in four different patients, as can be seen in figure 3.11 to show the parametric execution. It is considered three repetitions as growth patterns to extract data from the growing network in different redirection patterns for each baseline.

The goal is to simulate the dynamic behavior patterns of tumor growth. The first baseline was set to an initial state of 200 stem cells and increase to 400 under the influence of 50 cancer stem cells. The second baseline was set to an initial state of 400 stem cells and increase to 800 under the influence of 200 cancer stem cells. The third baseline was set to an initial state of 600 stem cells and increase to 1200 under the influence of 400 cancer stem cells. Moreover, the fourth baseline set to an initial state of 1200 stem cells and increase to 2400 under the influence of 650 cancer stem cells. Figure 3.11 shows the visual format of these baselines.

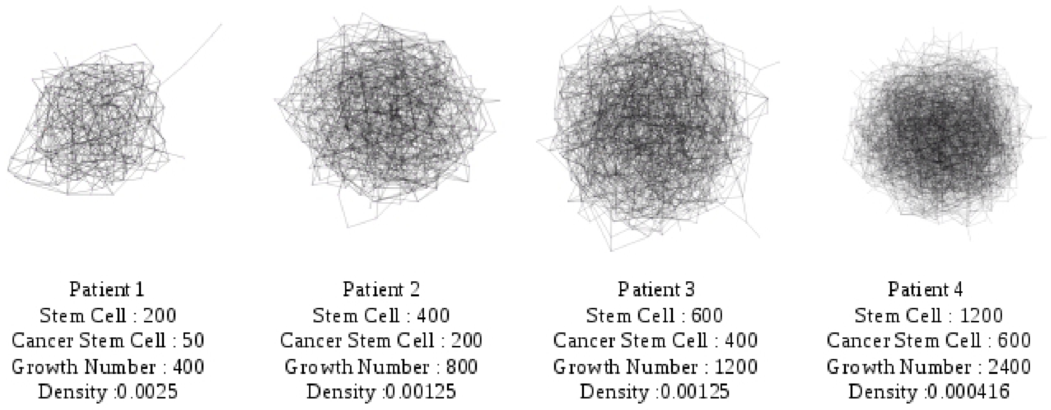


Figure 3.11: Initial tumor graph configuration of parameters to set up four baselines (Symbol of four patients).

The configuration management of these four baselines based on their dynamic parametric potential is established to analyze tumor density variations, tumor-derived cells into cancer redirection, and identify those essential cells' genomic profiles. Significant revision regarding graph pattern configuration is extracted from neighborhood analysis in graph theory and graph centrality methods such as centrality closeness or betweenness. Centrality closeness or betweenness quantifies the number of times a node acts as a bridge along the shortest path between two other nodes.

In the graph-agent-based conception, vertices that have a high probability of occurring on a randomly chosen shortest path between two randomly chosen vertices have a high betweenness. The graph-agent-based reconstruction of tumor-derived cell lines and producing the cellular profiles by extracting data from extensive parametric experiments helps compare the baselines' results in table 3.1 to Table 3.4. Each table shows three different tumor growth patterns for each patient. The important extracted data include the essential genomic profile and the identification number (ID) of the tumor-derived cell in each pattern.

As could be seen, the most aggressive growth pattern belongs to table 3.4 about patient number three for the average value of the essential genomic profile of tumor patterns. The number of tumor-derived cells is higher than the other tumors.

Cell lines serve as models to study cancer biology and to connect genomic variation to angiogenic responses. This modeling can aid in understanding different tumor behaviour.

The tumor-derived cell distribution results are significant for molecular and cell line studies.

Molecular sub-typing could be done based on gene expression patterns, and it helps in tumor-derived cell classification [123].

Table 3.1: Tumor-derived cells ID and their Genomic Profile for Patient1

initial Tumor (Patient1)	Growth Pattern 1	Growth Pattern 2	Growth Pattern 3
tumor-derived cell ID	10	4	18
Essential Genomic Profile	2.19E-03	4.70E-03	3.34E-03

Table 3.2: Tumor-derived cells ID and their Genomic Profile for Patient2

initial Tumor (Patient1)	Growth Pattern 1	Growth Pattern 2	Growth Pattern 3
tumor-derived cell ID	6	12	6
Essential Genomic Profile	1.34E-03	8.94E-04	1.50E-03

Table 3.3: Tumor-derived cells ID and their Genomic Profile for Patient3

initial Tumor (Patient1)	Growth Pattern 1	Growth Pattern 2	Growth Pattern 3
tumor-derived cell ID	1	17 and 10	5
Essential Genomic Profile	3.96E-04	7.07E-04	6.85E-04

Table 3.4: Tumor-derived cells ID and their Genomic Profile for Patient4

initial Tumor (Patient1)	Growth Pattern 1	Growth Pattern 2	Growth Pattern 3
tumor-derived cell ID	1	6	1
Essential Genomic Profile	2.70E-04	3.70E-04	2.13E-04

Figures 3.12 to 3.15 are visual representations of genome profiles variation of tumor-derived cell distribution in the Tumor of patient number three.

Tumor of patient number three is considered because it behaves more aggressively than the others, mainly because it grows very radical in its growth patterns numbers two and three. Figures 3.12 to 3.15 are line graphs representing how the genomic profiles have changed over the growing tumor-derived cells. It can show dependencies between these two parameters during a particular period of Tumor growing time.

Each pattern shows the diversity of dependencies between cell profiles and the redirection growth of tumor-derived cells. All these patterns for representing a tumor growth in different directions could be a predictive result of the most aggressive and probable metastasis growth by distributing the tumor-derived cells. All the patterns illustrated in figures 3.12 to 3.15 shows that each Tumor could grow into a different pattern based on the probability of tumor-derived cell distribution. Each pattern shows the analysis of the tumor-derived cell's variation by identifying the genomic profiles of those cells.

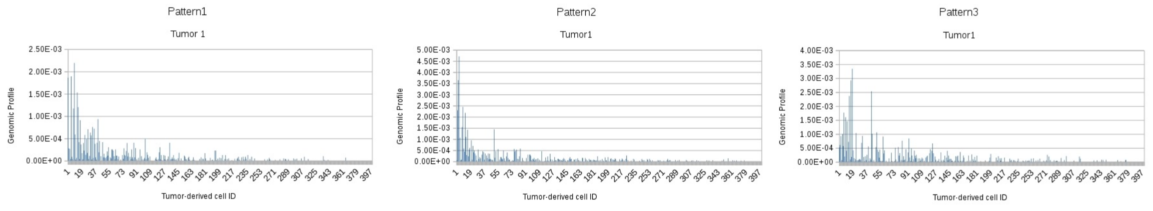


Figure 3.12: Three patterns of redirected growing tumor in patient1 model based on its tumor-derived cell distribution.

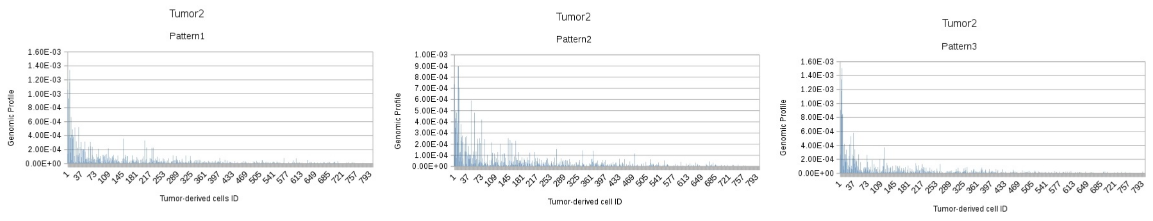


Figure 3.13: Three pattern of redirected growing tumor in patient2 model based on its tumor-derived cell distribution.

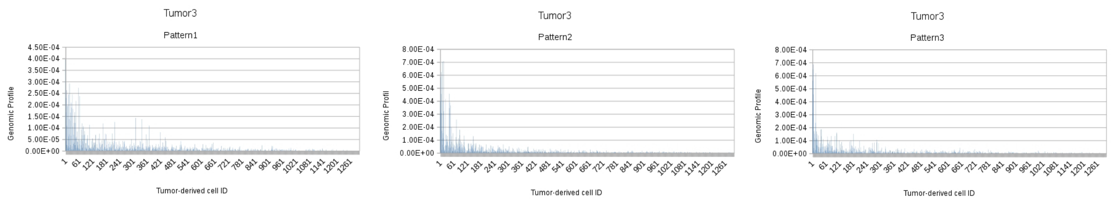


Figure 3.14: Three pattern of redirected growing tumor in patient3 model based on its tumor-derived cell distribution.

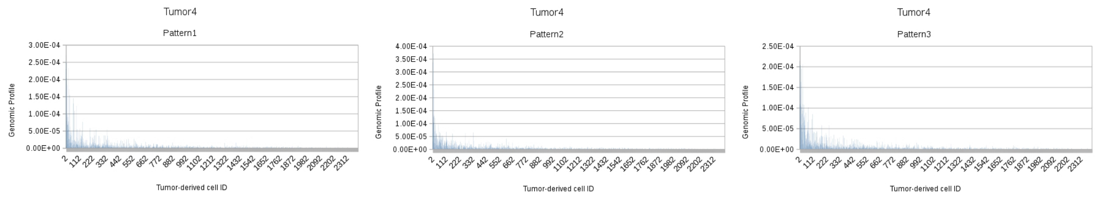


Figure 3.15: Three patterns of redirected growing tumor in patient4 model based on its tumor-derived cell distribution.

From another point of view, by computing the dead cells' ratio to the inflamed cells, we have demonstrated different tumor growth behavior upon the angiogenic switch's adequate laboratory condition.

The results of calculating the ratio of these two agents in the graph-agent-based model help analyzing the other important motive for tumor growth behavior. Stem cell quiescence is a way to control the inflammation in the tumor microenvironment, and increasing angiogenic values cause inflammatory reactions and raise the number of inflamed cells. Targeting inflammation by using angioprevention and stopping cancer cells from proliferating helps decrease the number of inflamed cells.

The new experiments on this concept are done based on table 3.5, showing parametric input. The table below shows three columns of angiogenic switches that each column considered three different probability values.

Table 3.5: Angiogenic switch key factors as transition probabilities

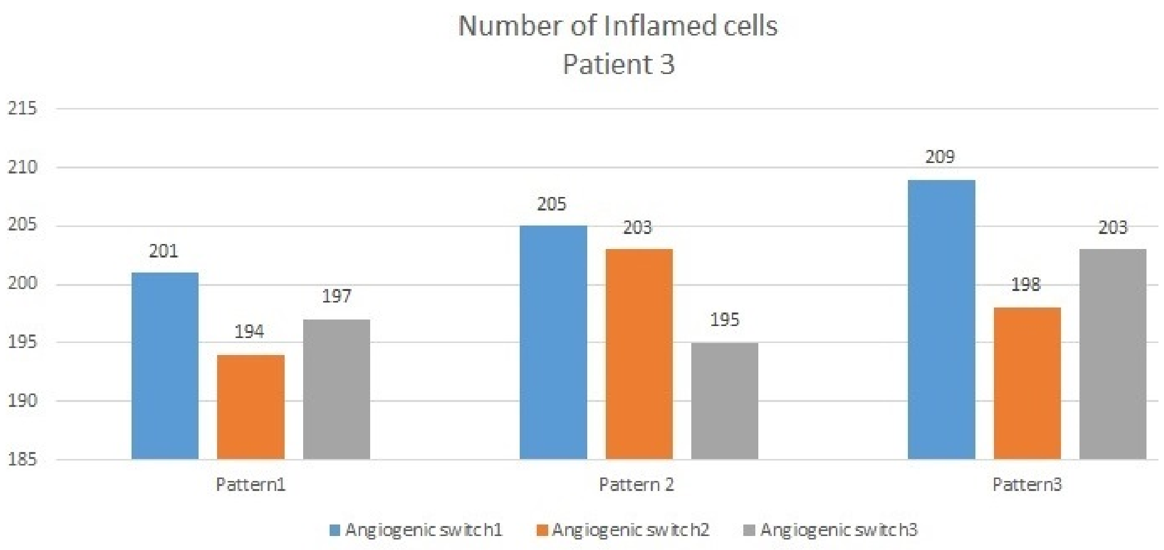
PA Values	Angiogenic switch1	Angiogenic switch2	Angiogenic switch3
AngioPrevention	0.4	0.6	0.4
Angiogenesis	0.6	0.4	0.6
Quiescent	0.2	0.2	0.8

As shown in the table, angiogenic switches classify into three categories called angiogenesis, angioprevention, and quiescence for one to three lists of input parameters called an angiogenic switch. The experiments were executed on tumor number three, which was the more aggressive one among the others. The results of these experiments are shown in figure 3.16.

Figure 3.16 illustrates the change scale of inflammatory in tumor number three based on different angiogenic key factor parameters determinate in table 3.5.

The X-axes show three different generated tumor growth patterns for patient number3, and Y-axes show the number of cells, which in the first bar chart is a number of inflamed cells, and in the second chart is the number of dead cells.

As shown in the charts, the number of inflamed cells is raised while the input angiogenic switch number one was considered and in versus the number of dead cells got reduced. This is evident in the chart that angiogenesis and inflammation are mutually dependent.



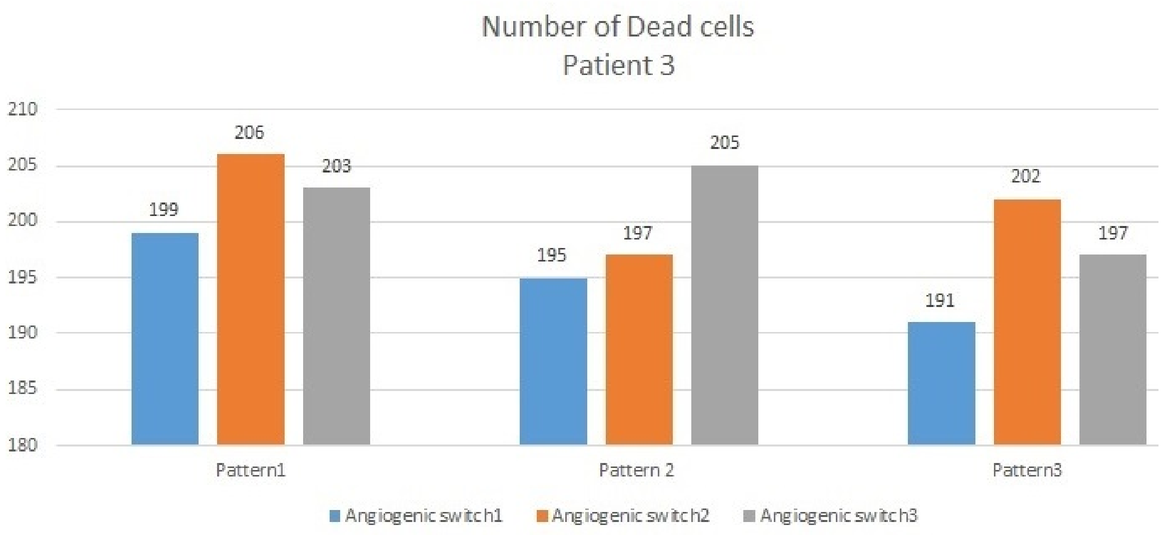


Figure 3.16: The result of the change scale of inflamed and dead cells of tumor number three.

3.4 Incorporated Network Clustering Techniques into Tumor Model

Using a cluster computing system to analyze and extract information allows us to scale the extensive, complex models and run many replicates of the large parametric simulation on the SGE, as demonstrated in chapter two of this thesis. Implementing a discrete model from a dynamic coherent agent-based model and distributing the subsets among the clusters needs to discover the agent-based model's complex network community structure and classify the total number of agents based on detecting scales.

The growing agent network model is one of the most complex models for distribution because since the agents behave in a stochastic, nonlinear manner over time, so there is a need to discover a class of the agents based on a kind of similarity such as their scale or state at their community. Therefore, there is a need to rely on a complex network similarity decomposition algorithm for the parallel executions, which able to support traditional high-performance computing environments.

In this chapter of the thesis, three main algorithms and techniques in network discovery, decomposition, and analysis, are selected. They are Power-Law distribution, Egocentric network analysis, and discovery of Subgraph Centrality [135] [137] [115]. Therefore, Discovering, Classifying, and Sub graphing an extensive and multi-scale network implemented and nested in the same scenario explained in the modeling and simulation chapter.

There is a methodological approach to the analysis of egocentric social networks and the structure's analysis, including the degree of the egocentric network, the strength of the ego-node ties, and many other parameters that are similarly considered in biological models. Power law distribution and Subgraph Centrality discovery methods as the ego network features could simulate a definition of cellular or molecular neighborhoods in the biological models.

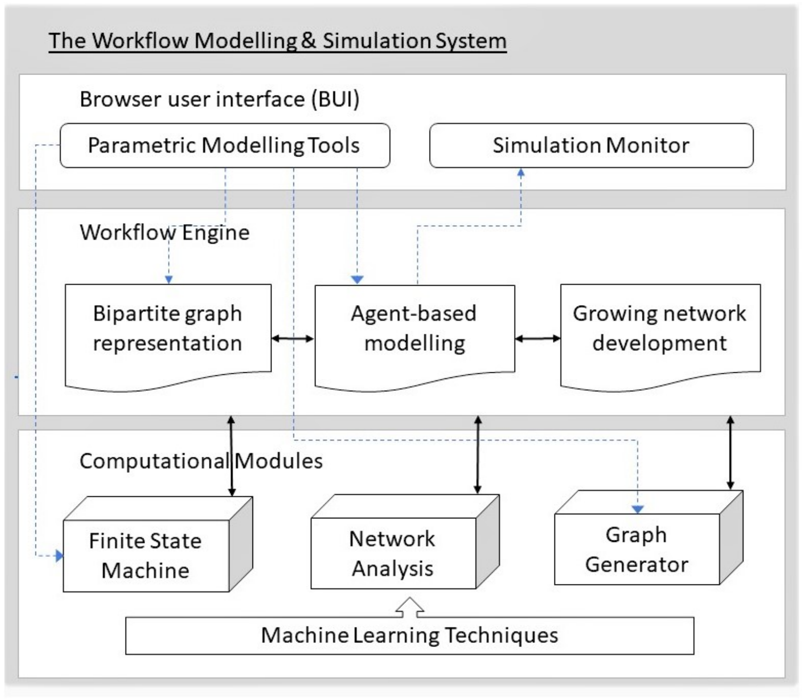

The Workflow Modelling & Simulation System
Browser user interface (BUI)
Parametric Modelling Tools
Simulation Monitor
Workflow Engine
Bipartite graph representation
Agent-based modelling
Growing network development
Computational Modules
Finite State Machine
Network Analysis
Graph Generator
Machine Learning Techniques

Figure 3.17: The proposed workflow for modeling and simulation system.

The workflow in Figure 3.17 shows that the modeling and simulation process built on different modules such as generating a graph, devising the network structure and analysis measures, machine-learning algorithms, and implementing state machine models for computation.

The workflow engine is the agent-based model that gets fed from the computational level. There is a browser user interface (BUI) for non-professional users to select and change the variables of the parameters and rules quickly on top of all layers. The workflow is used to construct a complex, scalable agent-based model on a tumor growth scenario integrated by incorporating network analysis techniques.

3.4.1 Power-Law Distribution

Statistically, a power-Law [135] is a functional relationship between two quantities. A relative change in one quantity results in the symmetric relative change in the other quantity, independent of those initial profiles (state, size, or weight). The distributions of a wide variety of physical, biological, and artificial phenomena almost follow a Power-Law over a wide range of values such as the size of activity patterns of the neuronal population in the network or the scale of negotiations between the nodes in the different types of cellular or molecular networks.

Considering the heterogeneous cell-agent population in the simulated tumor growth model in chapter three, implementing Power-Law for the growth of a heterogeneous population of the cells helps because the Power-Law is contrary to traditional Gaussian averages in that they demonstrate correlated phenomena.

The idea is that a Power-Law encodes the high frequency of single-cell identical profiles. With the same hypothesis, the Power-Law sequence algorithm, as shown in algorithm 4, using Power-Law exponent growing graphs that should return a sequence of length *n* from a Power-Law probable identical cell distribution is implemented in the model. The following algorithm returns a sample sequence of length *n* from graph *G* using Pareto distribution function based on the probability *P* that shows *n* is a portion of the graph that has power on the other nodes. The algorithm is in python implemented in the NetworkX package.

Algorithm 4 Power-Law exponent growing graph algorithm.

```
Input: (n, p, exponent=2.0, source =G)
Output: (sample sequence of length n from a power law distribution)
while $source > n$ do
  if [$(p < 1) \vee (p > 0)$ ] then
    for ($i$ in integer range) do
      $Output \leftarrow$ G.paretovariate(exponent-1)
    end for
  end if
end while
```

The graph generator of the tumor model using the Power-Law method works so that each random edge is followed by a chance of making an edge of one of its neighbors (and thus a triangle) and connecting these cells. The method is shown in algorithm 5.

Algorithm 5 The Power-Law Algorithm for creating the tumor Graph

```
Input: (number of nodes, number of random edges to add for each new node, probability of adding a triangle after adding a random edge)
Output: (Tumor Graph)
while $loopcondition$ do
    ADD $m$ initial nodes
    LIST the existing nodes
    ADD $m$ initial nodes
    LIST the existing nodes to sample from
    Now ADD the other $n - 1$ nodes
    ADD one node to list for each new link
    ADD $m - 1$ more new links
    clustering step: ADD triangle
    if there is a neighbor without a link $n$ then
        ADD triangle
    end if
end while
ADD source node to list $m$ times
RETURN the Graph
END
```

As shown in algorithm 4 and algorithm 5, the integration of a generator of growing graphs with the Power-Law probability distribution and an approximate average clustering makes it possible to have a disconnected graph.

Power-Law probability distribution and approximate average clustering methods to complete the clustering and classifying the identical cell profiles simulated for each cell/agent help to cut off the simulated large multi-scale graph agent-based model.

In this thesis, a network model has proposed that it has both the perfect Power-Law degree distribution and the high clustering. Furthermore, in this model, the clustering's degree, measured by the clustering coefficient, is shown to be tunable and thus controllable by adjusting the parameters of the model. Figure 3.18 shows that the growing tumor model to more than 10000 cells is clustered into 29 distributed clusters based on the Power-Law probability distribution.

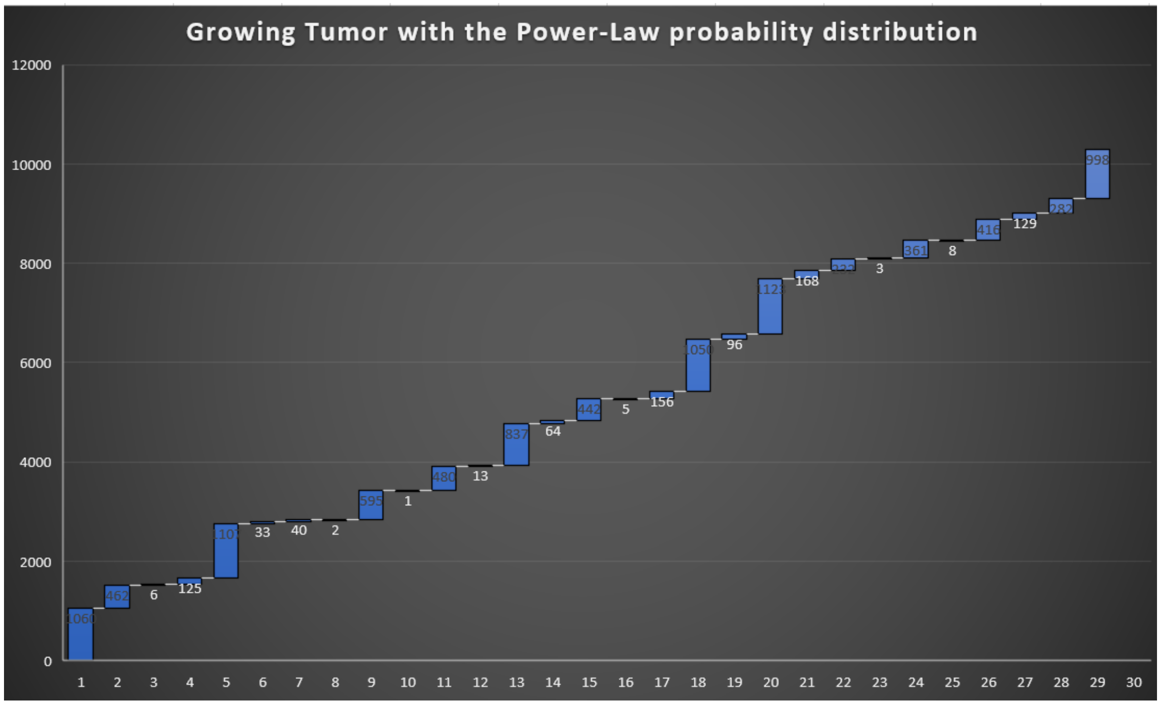


Figure 3.18: Growing Tumor to more than 10000 cells with the Power-Law probability distribution.

3.4.2 Egocentric Network Analysis

It is necessary to estimate many analyses, such as agent profiles, agent populations' patterns of interaction, and structured data collection to simulate a sizeable multi-scale tumor network model. The simulation must translate the most relevant expression of a tumor growth model's significant sub-networks, which are functionally dependent on each other. From a new perspective, an ego-network module extracting from Network X returns induced subgraph of neighbors centered at node *n* within a given radius by single-source Dijkstra's algorithm. This module has four input parameters: generated agent-based graphs, *n* node, the radius, and the distance parameters. The radius parameter works as the cut-off fact. This fact includes all neighbors from *n* that their distance is less than the radius criterion.

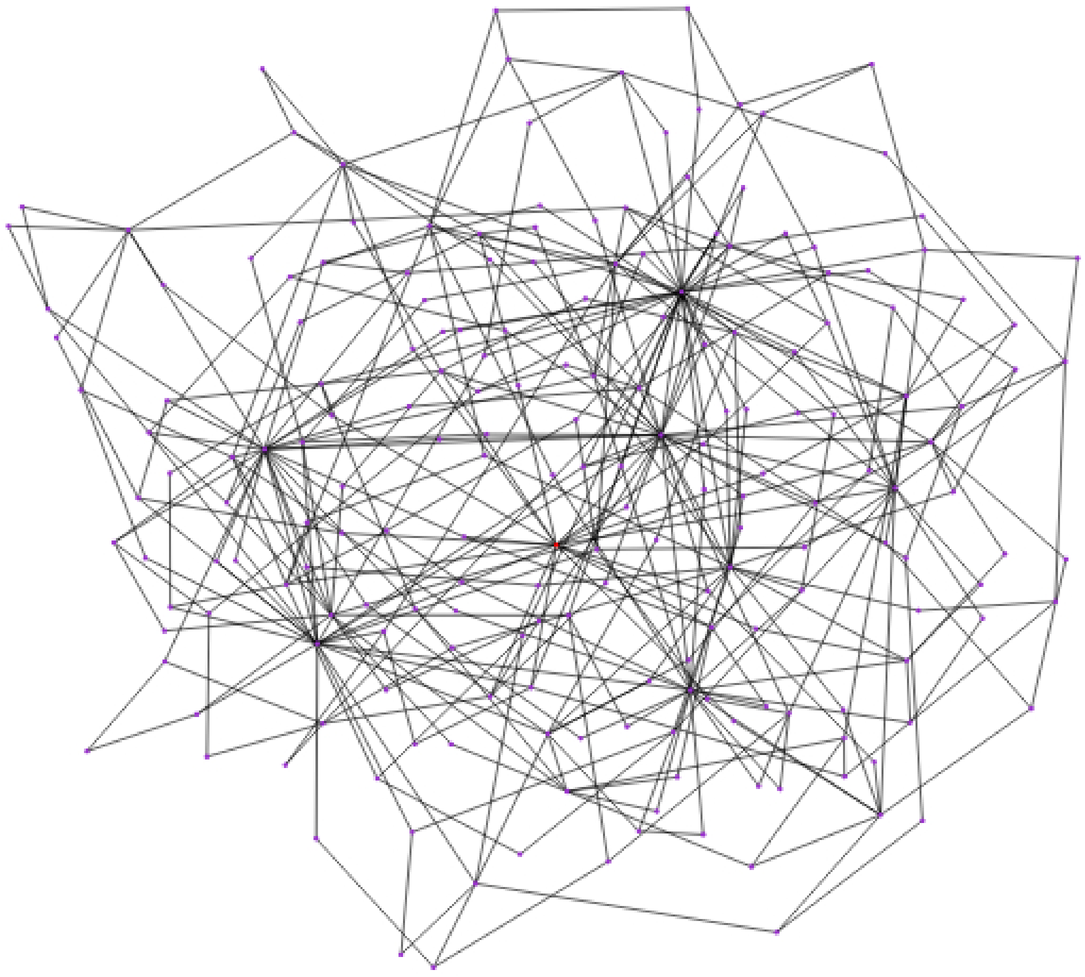


Figure 3.19: Egocentric graph of a tumor model (simulation of 600 cells).

Figure 3.19 shows a tumor graph agent-based model for ego-network simulation of 600 cells subdivided into seven subgraphs. Figure 4.4 shows the egocentric visual format of the tumor growth model, which returns seven subgraphs of neighbors in the network of 2000 cells.

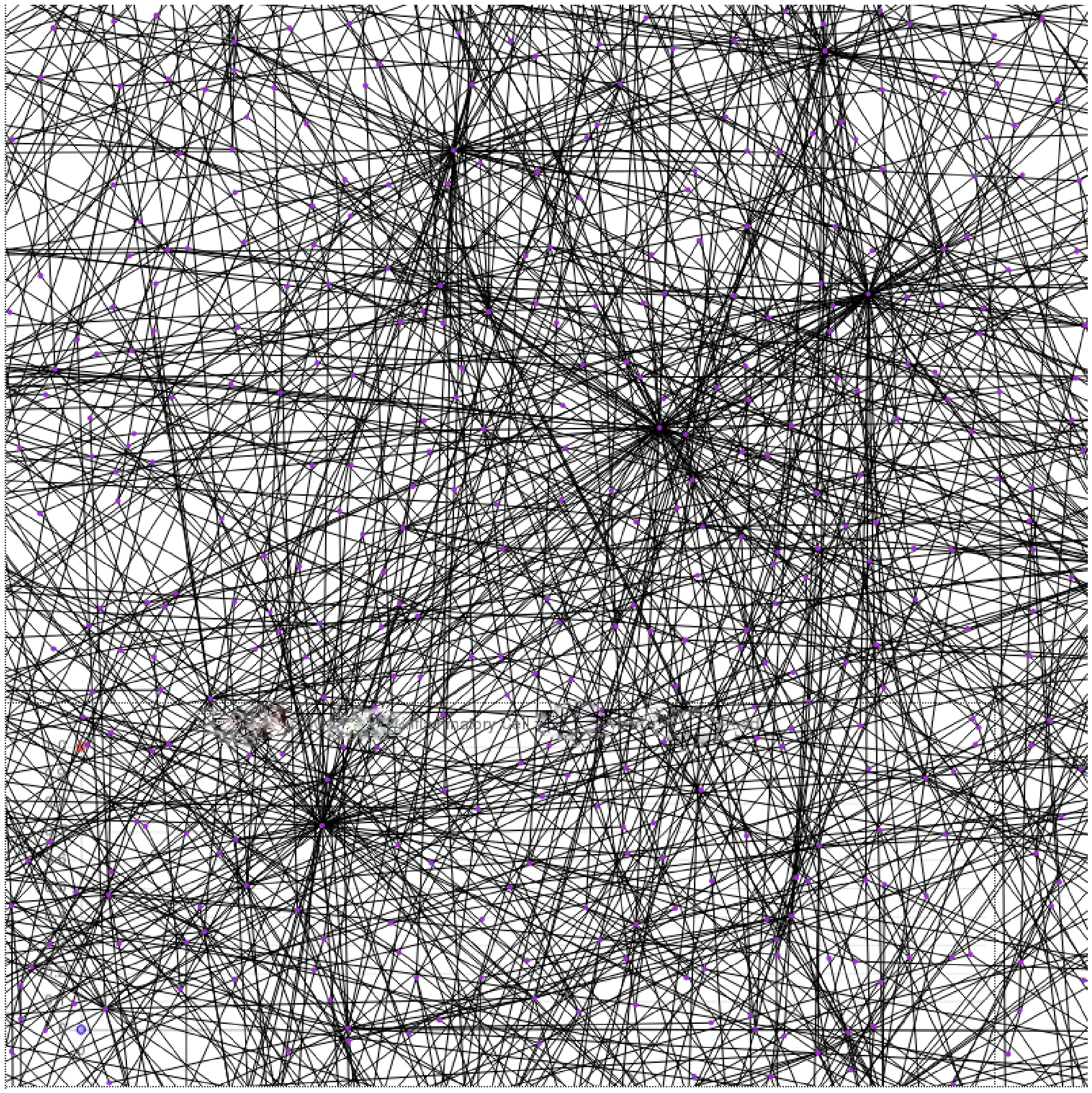


Figure 3.20: Egocentric form of tumor growth model which returns seven identified subgraphs in the simulated network of 2000 cells.

As can be seen in figure 3.20, each subgraph has its different ego-node, which has its agent's role and works as a hub-node among the other nodes in the subgraph essentially.

Since the egocentric algorithm's implementation is nested in the agent-based growing model of the tumor, each ego-node shows a different color based on the agents' different defined states. Six red ego-agents are the metastasis, and one purple ego-agent is inflamed but not yet become metastatic. Finding these seven ego-agents was made possible after growing the cells by the rule of the Power-Law distribution algorithm.

Because each ego-node works as a hub-node among the other nodes in its Subgraph network, each ego-agent in the model becomes the same. The optimized egocentric each subgraph has its different ego-node, which has its agent's role also. Each ego-node works as a hub-node among the other nodes among the subgraph networks.

3.4.3 Discovery of Subgraph Centrality

The subgraph centrality measure works as a benchmark for assessing the tumor-derived cells' essential genome profile in the simulated tumor model. One of the most critical tasks in the analysis of cell to cell interaction is to predict a group or cluster of transiently(cancer cells) interacting cells as the way that each genome profile at the molecular level is related to the transiently interacting proteins and identify as hub genes. Together, these groups of cells can accomplish a biological function, and they can be mapped to specific subgraphs in the network. Characterizing nodes in a network by subgraph centrality is according to the number of closed walks starting and ending at the node. Each closed walk is associated with a connected subgraph, and the measure counts the times that a node takes part indifferently connected subgraphs of the network. The node behaves like a hub. Assessing the essential genome profile of the tumor-derived cells could be an optimal partitioning function for the classification of tumor cells of the sizeable multi-scale model clustering and analysis.

Accordingly, the exclusive characteristic distribution of tumor-derived cells is illustrated by the subgraph centrality measuring in Figure 3.21. These measures are labeled as genome profiles of tumor-derived cells. Figure 3.21, from another point of view, it shows the genomic evolution of the tumor-derived cells, which causes biological function, such as changing the state of the neighbor cells in the subgraphs.

In each run, a dictionary of nodes with subgraph centrality measures as the values are mined. The subgraph centrality value of a tumor-derived cell is the sum of weighted closed walks of all lengths starting and ending at the cell.

At the molecular level, tumor-derived cells act as a hub gene and key pathways among the others, so their genome profile could be the critical factor affecting the graph clustering and data classification of heterogeneous information in multi-scale models. In this experiment, 30 tumor-derived cells are identified among 1200 cells, but not all of them are a metastasis. Egocentric networking helps identifying how many of them are metastasis and the obtained value of subgraph centrality gives an estimate of the scope of influence of them on the others. There are identified six metastasis cells in the egocentric graph of the simulated tumor model.

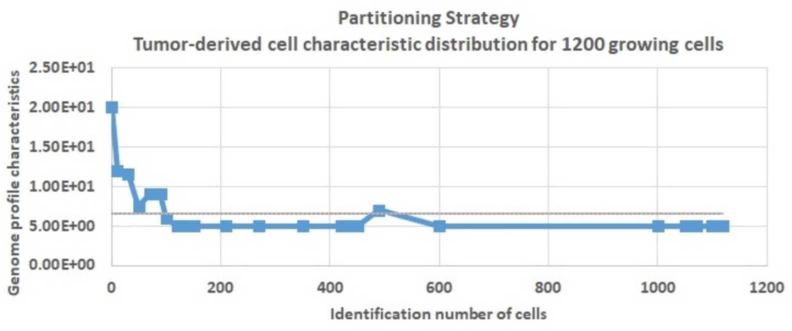


Figure 3.21: Tumor-derived cell identification, distribution for 1200 growing cells.

After the graph optimization step by translating the generated graph into an egocentric network based on the obtained Identical profiles, there is a need for partitioning these subgraphs in different clusters.

The identical cell profiles, taken as the sub-graph centrality values, become a key value to individual clusters. In a partitioned egocentric network, each cluster includes all the neighbors of a determined distance less than the intended radius of the essential cells (ego-nodes), and the connected neighbors in the same class of the cell profile are collected in the same cell cluster.

Figure 3.22 shows six prepared clusters from Q1 to Q6 based on the obtained subgraphs extracted as dictionaries of nodes based on sub-graph centrality as the values. They are considered the tumor-derived cell clusters, and six metastasis hub genes of 1200 growing cancer cells are considered the hub-nodes that create the clusters. Each cluster includes all the neighbors of a determined distance less than the hub node's intended radius, and the connected neighbors in the same class of the genomic profile collect in the same cluster.

Tumor cells in cluster number five (Q5) have the most effective behavior and essential genome profile, which probably can accomplish an aggressive function among the others or become the genetic reason for the metastasis.

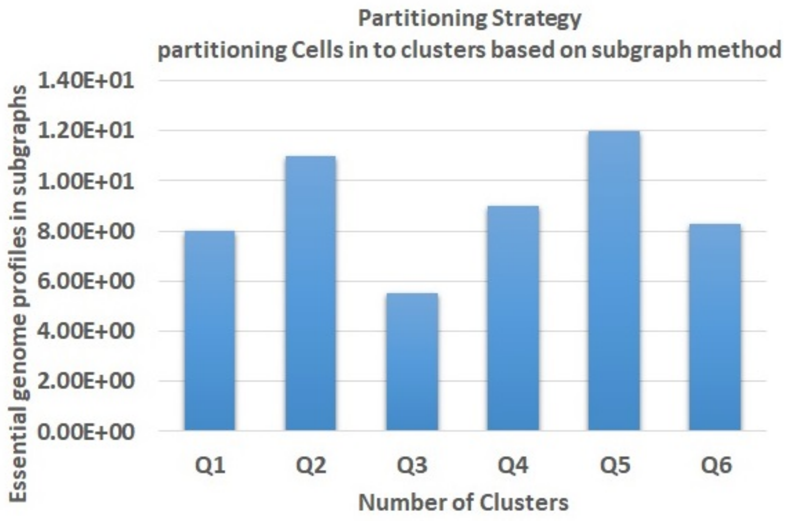


Figure 3.22: Partitioning cells into clusters based on subgraph centrality.

The output dictionaries from the simulation are saved in Graph-ML and XML format and ready at the containers to move the workloads in parallel to the pool of nodes on the Cloud-HPC infrastructures.

3.5 Conclusion

Observing the above laboratory results based on a large amount of simulated data and the significant variation of input parameters, it is obvious developing a discrete model from a growing graph agent-based simulation helps simulate at different scales. As the model grows in multi-scale, multi patterns, and multi-stages, there is an urgent need for high-performance simulation.

Since each agent is transitioning to several possible states and is dependent and in communication with the other agents in the graph, distributing the model could be a substantial challenge and an achievement for dynamic modeling and simulation. The past experiences showed that computation-intensive and open scale agent-based models should not be scheduled on one computing node. Then it is more profitable to allocate a group of sub-models that communicate with each other intensively on more than one computing node.

In this chapter, after developing the model in a graph's format, we fed functional ML clustering techniques into the model to generate subgraphs for distributed executions. The statistical method and applied Mathematics used in this chapter help cluster the model into submodels in subgraphs, extract, and transfer simulated data in different distribution files.

The performance monitoring of the simulation has been relegated to the future as an open line. Also, handling the sub-models' communication and dependencies is considered an open line of this research in chapter five.

The accepted publications with the peer review for this chapter are P3, P4, and P5 mentioned in section 4.3 of the thesis. Also, there are two submitted manuscripts in the revision process, which are D1 and D2. The complete code for the experiments of this chapter is available in appendix B.

CHAPTER 4
CONCLUSION AND OPENING LINES

4.1 Conclusion

In the scientific domain, many types of research today focus on the distribution and parallelization of data-intensive applications or models to execute distributed memory and processor architectures. The goal of porting to such architectures improves performance, and so the distributed memory and processor model is pertinent and facilitates scalability and enables high-performance gains.

The evaluation technology of computing systems consists of many parts and processes. There is a need for performance monitoring and evaluation of the simulation system, which will guide us to the different computation and integration of infrastructures. For many years, the past experiences showed that computation-intensive and open scale simulation models should not be scheduled on one computing node. It's more profitable to allocate a group of sub-models that communicate with each other intensively on more than one computing node.

The evaluation must test the running speed, CPU and memory usage, and network I/O traffic of the physical/virtual simulation system. This test keeps the quality of performance. This thesis is focused on the distribution techniques of data-intensive models to execute distributed memory and processor architectures. The evolving agent-based network model is one of the most complex models for distribution because the agents behave in a stochastic, nonlinear manner over time, so there is a need to discover a class of the agents based on their relation, dependencies, and communication at their community.

The ability to leverage hardware acceleration is a challenge for resource discovery. There is a need for an automated decision-making system to predict network growth patterns, classify sub-models, agent/node states, and dependencies to assign distributed computing resources considering this challenge. In this regard, a workflow modeling and simulation system is proposed to construct a complex, scalable agent-based model on a tumor growth scenario by applying three applicable algorithms to train the tumor network growing in different patterns and discover and classify the sub-models.

4.2 Opening lines

Distributed systems present new challenges, and the most important one is the efficient parallelization of the training process and the creation of a coherent model. Challenges and opportunities of distributed machine learning over conventional (centralized) mostly discuss the techniques and performance. The HPC community has identified machine learning as an emerging high-value workload and has started to apply HPC methodology.

Many types of research and experiences these days focused on these methodologies such as training extensive parametric network on the High-Performance Computing, Optimized the training of a neural network on Intel's, Designing a chip for HPC applications, Using deep learning to be optimized and scaled efficiently on large parallel HPC systems or even Scheduling deep neural network applications on cloud computing infrastructure by modeling the workload demand with techniques like lightweight profiling, which are borrowed from HPC. Like for other large-scale computational challenges, there are two fundamentally different and complementary ways of accelerating workloads: adding more resources to a single machine (vertical scaling or scaling up) and adding more nodes to the system (horizontal scaling or scaling out). Over the years, GPUs have shifted to more flexible architectures.

Nonetheless, the diverging branches are still inefficient. The proliferation of GPGPUs (General-Purpose GPUs, i.e., GPUs that can execute arbitrary code) must lead the vendors to design custom products that can be added to conventional machines as accelerators. For example, the NVIDIA Tesla GPU series is meant for highly parallel computing and designed to deploy supercomputers and clusters. While there are many different strategies to increase the processing power of a single machine for large-scale machine learning, there are reasons to prefer a scale-out design or combine the two approaches, as often seen in HPC.

The first reason is the generally lower equipment cost. The second reason is the resilience against failures because, when a single processor fails within an HPC application, the system can continue operating by initiating a partial recovery. The third reason is the increase in aggregate I/O bandwidth compared to a single machine. Training ML models is a highly data-intensive task, and the ingestion of data can become a severe performance bottleneck. Since every node has a dedicated I/O subsystem, scaling out is a useful technique for reducing I/O's impact on the workload performance by effectively parallelizing the reads and writes over multiple machines. A significant challenge of scaling out is that not all ML algorithms lend themselves to a distributed computing model, which can only be used for algorithms that can achieve a high degree of parallelism [162, 163].

The lines between traditional supercomputers, grids, and the cloud are increasingly getting blurred when it comes to the best execution environment for demanding workloads like machine learning. On the other hand, GPUs and accelerators are now more common in major cloud data centers. As a result, parallelization of the machine learning workload has become paramount to achieving acceptable performance on a large scale.

There are two fundamentally different ways of partitioning across all machines: parallelizing the data or the model when it comes to distribution and parallelization of the machine learning workload. These two can also be applied simultaneously. The idea of this thesis is based on parallelizing the model. Partitioning the model makes everything ready at the containers to move the workloads on the pool of nodes. However, in the following, there is a need for an extensive relational database to register each sub-model and indexes them based on their current states and dependencies.

Considering the Lookup performance table, indexing strategies, and a decision-making locator and distributor engine for allocating the most appropriate and available resources are the thesis's opening lines. The proposed architecture of this idea is shown in Figure 4.1.

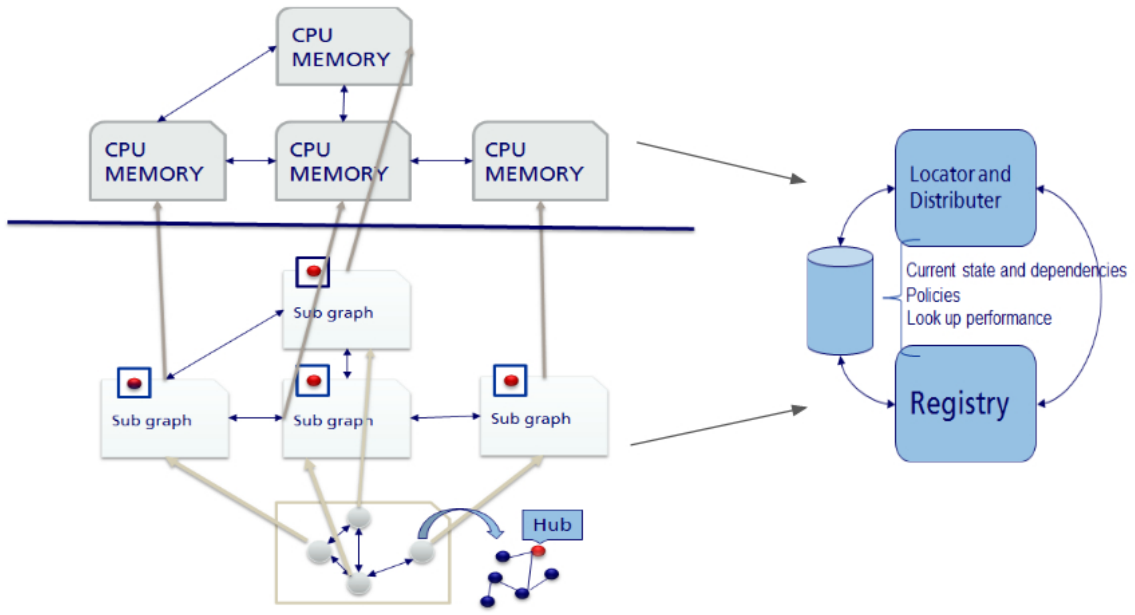


Figure 4.1: A locator and distributor engine for allocating linked Submodels which are dependent.

Also, from the other point of view, developing and implementing new AI methodologies, techniques, and policies to allocate proper HPC or cloud resources to the large-scale distributed agent-based models are demanded.

4.3 Publications

Accepted publication with the peer review process:

P1 High Performance Computing for Tumor Propagation Agent-based Model (Best Paper Award), XXIII Congreso Argentino de Ciencias de la Computación (La Plata), 2017, Ghazal Tashakor, Emilio Luque, Remo Suppi

P2 HPC for ABM using Netlogo, Jornadas Sarteco (Sociedad Científica Informática de España), 2017, Ghazal Tashakor, Emilio Luque, Remo Suppi

P3 Agent-Based Model for Tumor-Analysis Using Python+MESA, The European Modeling and Simulation Symposium (EMSS), 2018, Ghazal Tashakor, Remo Suppi

P4 Simulation and computational analysis of multiscale graph agent-based tumor model, IEEE International Conference on High-Performance Computing & Simulation (HPCS), 2019, Ghazal Tashakor, Remo Suppi

P5 Simulation and analysis of a multi-scale tumor model using agent clustered network, International Conference on Modeling and Applied Simulation 2019 (MAS19), Ghazal Tashakor, Remo Suppi

Accepted collaborative publication with the peer review process:

C1 Crowd Evacuation Modeling and Simulation Using Care HPS, International Conference on Computational Science (ICCS), 2017, Mohammed J. Alghazzawi1, Ghazal Tashakor, Francisco Borges, and Remo Suppi

C2 Performance Analysis of ABM Distributed Simulation for Real Crowd Evacuation Scenarios, XXIII Congreso Argentino de Ciencias de la Computación (La Plata), 2017, Mohammed J. Alghazzawi, Ghazal Tashakor, Fancisco Borges, Remo Suppi

Submitted manuscripts in revision process:

D1 Feeding graph machine learning into the agent based model for network analysis, IEEE International Conference on High-Performance Computing & Simulation (HPCS), 2020, Ghazal Tashakor, Alvaro Wong, Remo Suppi

D2 Ghazal Tashakor, Remo Suppi,“Incorporating Data Science into the Agent-Based Models for Constructing a Scalable Tumor model”, Journal of Artificial Intelligence and Soft Computing Research (JAISCR), 2020

BIBLIOGRAPHY

[1] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, Gabriel Balan. Mason: A multi-agent simulation environment. Simulation. 2005, Vol. 81 (7), pp 517-527.

[2] North, M.J., N.T. Collier, and J.R. Vos. Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit. ACM Transactions on Modeling and Computer Simulation. ACM, New York, USA.2006., Vol. 16 (1), pp. 1-25.

[3] Simon Coakley, Marian Gheorghe, Mike Holcombe, Shawn Chin, David Worth, Chris Greenough. Exploitation of High Performance Computing in the FLAME Agent-Based Simulation Framework. Proceedings of the 14th International Conference on High Performance Computing and Communications.2012, IEEE. Vol. 1. pp. 538-545.

[4] Dauer, J., & Dauer, J. A framework for understanding the characteristics of complexity in biology. International Journal of STEM Education, 2016, 3(1), 13.

[5] Minar, Nelson, et al. ”The swarm simulation system: A toolkit for building multi-agent simulations.” (1996): 96-06.

[6] Bellifemine F, Poggi A, Rimassa G. Developing multi-agent systems with JADE. In-International Workshop on Agent Theories, Architectures, and Languages 2000 Jul 7 (pp. 89-103). Springer, Berlin, Heidelberg.

[7] Fachada, N., Lopes, V. V., Martins, R. C., & Rosa, A. C. Parallelization Strategies for Spatial Agent-Based Models. International Journal of Parallel Programming, Springer.2016, pp. 1–33.

[8] Allan, Robert John. Survey of agent based modelling and simulation tools. London: Science & Technology Facilities Council, 2010.

[9] Grimm V, Revilla E, Berger U, Jeltsch F, Mooij WM, Railsback SF, Thulke HH, Weiner J, Wiegand T, DeAngelis DL. Pattern-oriented modeling of agent-based complex systems: lessons from ecology. science. 2005 Nov 11;310(5750):987-91.

[10] An G. Introduction of an agent-based multi-scale modular architecture for dynamic knowledge representation of acute inflammation. Theoretical Biology and Medical Modelling. 2008 Dec 1;5(1):11.

[11] Tero Aittokallio, Benno Schwikowski, Graph-based methods for analysing networks in cell biology, Briefings in Bioinformatics, Volume 7, Issue 3, September 2006, Pages 243–255

[12] Vazquez A, Dobrin R, Sergi D, et al.The topological relationship between the large-scale attributes and local interaction patterns of complex networks, Proc Natl Acad Sci USA, 2004, vol. 101 (pg. 17940-5)

[13] Pradines JR, Farutin V, Rowley S, et al. Analyzing protein lists with large networks: edge-count probabilities in random graphs with given expected degrees, J Comput Biol, 2005 ,vol. 12 (pg. 113-28)

[14] Estrada E. Virtual identification of essential proteins within the protein interaction network of yeast, Proteomics , 2006, vol. 6 (pg. 35-40)

[15] Croes D, CoucheF, WodakS J, et al. Metabolic PathFinding: inferring relevant pathways in biochemical networks, Nucleic Acids Res, 2005, vol. 33 (pg. W326-30)

[16] Shlomi T, Segal D, Ruppin E, et al. QPath: a method for querying pathways in a protein–protein interaction network, BMC Bioinformatics, 2006, vol. 7 pg. 199

[17] Soleimani S, Shamsi M, Ghazani MA, Modarres HP, Valente KP, Saghafian M, Ashani MM, Akbari M, Sanati-Nezhad A. Translational models of tumor angiogenesis:

a nexus of in silico and in vitro models. Biotechnology advances. 2018 Jul 1;36(4):880-93.

[18] Wilensky, Uri. ”Center for connected learning and computer-based modeling.” NetLogo. Northwestern University, 1999.

[19] Edelman LB, Eddy JA, Price ND. In silico models of cancer. Wiley Interdisciplinary Reviews: Systems Biology and Medicine. 2010 Jul;2(4):438-59.

[20] Li S, Li Q. Cancer stem cells and tumor metastasis. International journal of oncology. 2014 Jun 1;44(6):1806-12.

[21] Weis SM, Cheresh DA. Tumor angiogenesis: molecular pathways and therapeutic targets. Nature medicine. 2011 Nov;17(11):1359-70.

[22] Carmeliet P, Jain RK. Angiogenesis in cancer and other diseases. nature. 2000 Sep;407(6801):249-57.

[23] Granger DN, Senchenkova E. Inflammation and the Microcirculation. InColloquium series on integrated systems physiology: from molecule to function 2010 Jul 7 (Vol. 2, No. 1, pp. 1-87). Morgan & Claypool Life Sciences.

[24] Thorne BC, Bailey AM, Peirce SM. Combining experiments with multi-cell agent-based modeling to study biological tissue patterning. Briefings in bioinformatics. 2007 Jul 1;8(4):245-57.

[25] Tashakor G, Luque E, Suppi R. High Performance Computing for tumor Propagation Agent-based Model. In: XXIII Congreso Argentino de Ciencias de la Computación (La Plata, 2017)

[26] Estrada E, Rodríguez-Velázquez JA. Subgraph centrality and clustering in complex hyper-networks. Physica A: Statistical Mechanics and its Applications. 2006 May 15;364:581-94.

[27] Hartwell LH, Hopfiled JJ, Leibler S, et al. From molecular to modular cell biology, Nature , 1999, vol. 402 6761 Suppl(pg. C47-52)

[28] Lee TI, Rinaldi NJ, Robert F, et al. Transcriptional regulatory networks in Saccharomyces cerevisiae, Science, 2002, vol. 298 (pg. 799804)

[29] Milo R, Shen-Orr S, Itzkovitz S, et al. Network motifs: simple building blocks of complex networks, Science, 2002, vol. 298 (pg. 824-7)

[30] Ideker T, Ozier O, Schwikowski B, et al. Discovering regulatory and signalling circuits in molecular interaction networks, Bioinformatics, 2002, vol. 18 Suppl 1(pg. S233-40)

[31] Tashakor G, SuppiR. : Agent-based model for tumor-analysis using Python+Mesa. In: European Modeling and Simulation Symposium (EMSS),2018; pp. 248-253,

[32] Cao L, Gorodetsky V, Mitkas PA. Agent mining: The synergy of agents and data mining. IEEE Intelligent Systems. 2009 May;24(3):64-72.

[33] Railsback SF, Grimm V. Agent-based and individual-based modeling: a practical introduction. Princeton university press; 2019 Mar 26.

[34] M. Girvan, ME. Newman, ”Community structure in social and biological networks,” in Proceedings of the national academy of sciences, vol.99(12), 2002; pp. 7821-7826.

[35] ME. Newman, ”Detecting community structure in networks,” The European Physical Journal B., vol. 38(2), 2004, pp. 321-330.

[36] McGillivray P, Clarke D, Meyerson W, Zhang J, Lee D, Gu M, Kumar S, Zhou H, Gerstein M. Network analysis as a grand unifier in biomedical data science. Annual Review of Biomedical Data Science. 2018 Jul 20;1:153-80.

[37] Barabási AL, Bonabeau E. Scale-free networks. Scientific american. 2003 May 1;288(5):60-9.

[38] Barabási AL, Albert R. Emergence of scaling in random networks. science. 1999 Oct 15;286(5439):509-12.

[39] Klamt S, Haus UU, Theis F. Hypergraphs and cellular networks. PLoS computational biology. 2009 May 29;5(5):e1000385.

[40] Pavlopoulos GA, Secrier M, Moschopoulos CN, Soldatos TG, Kossida S, Aerts J, Schneider R, Bagos PG. Using graph theory to analyze biological networks. BioData mining. 2011 Dec;4(1):10.

[41] Rejniak KA, Anderson AR. Hybrid models of tumor growth. Wiley Interdisciplinary Reviews: Systems Biology and Medicine. 2011 Jan;3(1):115-25.

[42] MMasoudi-Nejad A, Bidkhori G, Ashtiani SH, Najafi A, Bozorgmehr JH, Wang E. Cancer systems biology and modeling: microscopic scale and multiscale approaches. InSeminars in cancer biology 2015 Feb 1 (Vol. 30, pp. 60-69). Academic Press.

[43] Shannon P, Markiel A, Ozier O, et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks, Genome Res, 2003, vol. 13 (pg. 2498-504)

[44] Reiss DJ, Avila-Campillo , Thorsson V, et al. Tools enabling the elucidation of molecular pathways active in human disease: application to Hepatitis C virus infection, BMC Bioinformatics , 2005, vol. 6 pg. 154

[45] Albrecht M, Huthmacher C, Tosatto SC, et al. Decomposing protein networks into domain-domain interactions , Bioinformatics, 2005, vol. 21 Suppl 2(pg. ii220-1)

[46] Breitkreutz BJ, Stark C, Tyers M. Osprey: a network visualization system, Genome Biol , 2003, vol. 4 pg. R22

[47] Breitling R, Amtmann A, Herzyk P. Graph-based iterative Group Analysis enhances microarray interpretation, BMC Bioinformatics, 2004, vol. 5 pg. 100

[48] Gopalacharyulu PV, Lindfors E, Bounsaythip C, et al. Data integration and visualization system for enabling conceptual biology, Bioinformatics, 2005, vol. 21 Suppl 1(pg. i177-85)

[49] Hu Z, Mellor J, Wu J, et al. VisANT: data-integrating visual framework for biological networks and modules, Nucleic Acids Res , 2005, vol. 33 (pg. W352-7)

[50] Myers CL, Robson D, Wible A, et al. Discovery of biological networks from diverse functional genomic data, Genome Biol, 2005, vol. 6 pg. R114

[51] Hwang D, Rust AG, Ramsey S, et al. A data integration methodology for systems biology , Proc Natl Acad Sci USA, 2005, vol. 102 (pg. 17296-301)

[52] Hwang D, Rust AG, Ramsey S, Smith JJ, Leslie DM, Weston AD, De Atauri P, Aitchison JD, Hood L, Siegel AF, Bolouri H. A data integration methodology for systems biology. Proceedings of the National Academy of Sciences. 2005 Nov 29;102(48):17296-301.

[53] Breitkreutz BJ, Stark C, Tyers M. Osprey: a network visualization system. Genome biology. 2002 Dec;3(12):1-6.

[54] Baitaluk M, Qian X, Godbole S, et al.PathSys: integrating molecular interaction graphs for systems biology, BMC Bioinformatics, 2006, vol. 7 pg. 55

[55] Li W, Kurata H. A grid layout algorithm for automatic drawing of biochemical networks. Bioinformatics. 2005 May 1;21(9):2036-42.

[56] Dogrusoz U, Erson EZ, Giral E, Demir E, Babur O, Cetintas A, Colak R. PATIKA web: a web interface for analyzing biological pathways through advanced querying and visualization. Bioinformatics. 2006 Feb 1;22(3):374-5.

[57] Scholtens D, Vidal M, Gentleman R. Local modeling of global interactome networks. Bioinformatics. 2005 Jan 1;21(17):3548-57.

[58] D'haeseleer P, Liang S, Somogyi R. Genetic network inference: from co-expression clustering to reverse engineering. Bioinformatics. 2000 Aug 1;16(8):707-26.

[59] Hartemink AJ. Reverse engineering gene regulatory networks. Nature biotechnology. 2005 May;23(5):554-5.

[60] Ott S, Hansen A, Kim SY, Miyano S. Superiority of network motifs over optimal networks and an application to the revelation of gene network evolution. Bioinformatics. 2005 Jan 15;21(2):227-38.

[61] Wagner A. Reconstructing pathways in large genetic networks from genetic perturbations. Journal of Computational Biology. 2004 Jan 1;11(1):53-60.

[62] di Bernardo D, Thompson MJ, Gardner TS, Chobot SE, Eastwood EL, Wojtovich AP, Elliott SJ, Schaus SE, Collins JJ. Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. Nature biotechnology. 2005 Mar;23(3):377-83.

[63] Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED. Advances to Bayesian network inference for generating causal networks from observational biological data. Bioinformatics. 2004 Dec 12;20(18):3594-603.

[64] Rand, William. ”Machine learning meets agent-based modeling: when not to go to a bar.” In Conference on Social Agents: Results and Prospects. 2006.

[65] Wolpert DH, Wheeler KR, Tumer K. General principles of learning-based multi-agent systems. InProceedings of the third annual conference on Autonomous Agents 1999 Apr 1 (pp. 77-83).

[66] Mitchell TM. Artificial neural networks. Machine learning. 1997;45:81-127.

[67] Dehmer, Matthias, and Subhash C. Basak. Statistical and machine learning approaches for network analysis. Hoboken: Wiley, 2012.

[68] Spirin, Victor, and Leonid A. Mirny. ”Protein complexes and functional modules in molecular networks.” Proceedings of the national Academy of sciences 100, no. 21 (2003): 12123-12128.

[69] D’haeseleer P. How does gene expression clustering work?. Nature biotechnology. 2005 Dec;23(12):1499-501.

[70] Dunn, Ruth, Frank Dudbridge, and Christopher M. Sanderson. ”The use of edge-betweenness clustering to investigate biological function in protein interaction networks.” BMC bioinformatics 6, no. 1 (2005): 39.

[71] Rives, Alexander W., and Timothy Galitski. ”Modular organization of cellular networks.” Proceedings of the national Academy of sciences 100, no. 3 (2003): 1128-1133.

[72] Tsuda, Koji, and William Stafford Noble. ”Learning kernels from biological networks by maximizing entropy.” Bioinformatics 20, no. suppl_1 (2004): i326-i333.

[73] Kernighan BW, Lin S. An efficient heuristic procedure for partitioning graphs. The Bell system technical journal. 1970 Feb;49(2):291-307.

[74] U. Luxburg, A tutorial on spectral clustering in Statistics and Computing,2007; 17(4), 395–416

[75] M.E.J. Newman, Modularity and community structure in networks. PNAS, 2006; 103(23), 8577– 8582

[76] Fortunato S. Community detection in graphs. Physics reports. 2010 Feb 1;486(3-5):75-174.

[77] Palla G, Derényi I, Farkas I, Vicsek T. Uncovering the overlapping community structure of complex networks in nature and society. nature. 2005 Jun;435(7043):814-8.

[78] Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences. 2008 Jan 29;105(4):1118-23.

[79] Wu FY. The potts model. Reviews of modern physics. 1982 Jan 1;54(1):235.

[80] Wu FY. Potts model and graph theory. Journal of statistical physics. 1988 Jul 1;52(1-2):99-112. .

[81] Newman ME, Leicht EA. Mixture models and exploratory analysis in networks. Proceedings of the National Academy of Sciences. 2007 Jun 5;104(23):9564-9.

[82] Coakley S, Gheorghe M, Holcombe M, Chin S, Worth D, Greenough C. Exploitation of high performance computing in the FLAME agent-based simulation framework. In2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems 2012 Jun 25 (pp. 538-545). IEEE.

[83] Wilensky U, Rand W. Making models match: Replicating an agent-based model. Journal of Artificial Societies and Social Simulation. 2007 Oct 31;10(4):2.

[84] Walker E, Guiang C. Challenges in executing large parameter sweep studies across widely distributed computing environments. InProceedings of the 5th IEEE workshop on Challenges of large applications in distributed environments 2007 Jun 25 (pp. 11-18).

[85] Walker E, Gardner JP, Litvin V, Turner EL. Creating personal adaptive clusters for managing scientific jobs in a distributed computing environment. In2006 IEEE Challenges of Large Applications in Distributed Environments 2006 Jun 19 (pp. 95-103). IEEE.

[86] Cordasco G, De Chiara R, Mancuso A, Mazzeo D, Scarano V, Spagnuolo C. Bringing together efficiency and effectiveness in distributed simulations: the experience with D-MASON. Simulation. 2013 Oct;89(10):1236-53.

[87] Was, Jaroslaw, and Pawel Topa. ”Special issue on complex collective systems.” (2019): 68-69.

[88] Collier N, Ozik J, Macal CM. Large-scale agent-based modeling with repast HPC: A case study in parallelizing an agent-based model. InEuropean Conference on Parallel Processing 2015 Aug 24 (pp. 454-465). Springer, Cham.

[89] Kiran M, Bicak M, Maleki-Dizaji S, Holcombe M. Flame: A platform for high performance computing of complex systems, applied for three case studies. Acta Physica Polonica. Series B, Proceedings Supplement. 2011 Jan 1;4(2).

[90] Suryanarayanan, Vinoth, Georgios Theodoropoulos, and Michael Lees. ”Pdes-mas: Distributed simulation of multi-agent systems.” Procedia Computer Science 18 (2013): 671-681.

[91] Wittek P, Rubio-Campillo X. Scalable agent-based modelling with cloud HPC resources for social simulations. In4th IEEE International Conference on Cloud Computing Technology and Science Proceedings 2012 Dec 3 (pp. 355-362). IEEE.

[92] Bujas, Jakub, Dawid Dworak, Wojciech Turek, and Aleksander Byrski. ”High-performance computing framework with desynchronized information propagation for large-scale simulations.” Journal of Computational Science 32 (2019): 70-86.

[93] Clarke L, Glendinning I, Hempel R. The MPI message passing interface standard. InProgramming environments for massively parallel distributed systems 1994 (pp. 213-218). Birkhäuser, Basel.

[94] Standish RK, Leow R. EcoLab: Agent based modeling for C++ programmers. arXiv preprint cs/0401026. 2004 Jan 27.

[95] Coakley S, Gheorghe M, Holcombe M, Chin S, Worth D, Greenough C. Exploitation of high performance computing in the FLAME agent-based simulation framework. In2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems 2012 Jun 25 (pp. 538-545). IEEE.

[96] Ginot V, Le Page C, Souissi S. A multi-agents architecture to enhance end-user individual-based modelling. Ecological modelling. 2002 Nov 15;157(1):23-41.

[97] Rubio-Campillo, Xavier. ”Pandora: a versatile agent-based modelling platform for social simulation.” Proceedings of SIMUL (2014): 29-34.

[98] North MJ, Collier NT, Vos JR. Experiences creating three implementations of the repast agent modeling toolkit. ACM Transactions on Modeling and Computer Simulation (TOMACS). 2006 Jan 1;16(1):1-25.

[99] Luke S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G. Mason: A multiagent simulation environment. Simulation. 2005 Jul;81(7):517-27.

[100] Abar, Sameera, Georgios K. Theodoropoulos, Pierre Lemarinier, and Gregory MP O’Hare. ”Agent Based Modelling and Simulation tools: A review of the state-of-art software.” Computer Science Review 24 (2017): 13-33.

[101] Reuillon R, Leclaire M, Rey-Coyrehourcq S. OpenMOLE, a workflow engine specifically tailored for the distributed exploration of simulation models. Future Generation Computer Systems. 2013 Oct 1;29(8):1981-90.

[102] Hold-Geoffroy Y, Gagnon O, Parizeau M. Once you SCOOP, no need to fork. In-Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment 2014 Jul 13 (pp. 1-8).

[103] Ayllón D, Railsback SF, Vincenzi S, Groeneveld J, Almodóvar A, Grimm V. InSTREAM-Gen: Modelling eco-evolutionary dynamics of trout populations under anthropogenic environmental change. Ecological Modelling. 2016 Apr 24;326:36-53.

[104] Railsback, S., Ayllón, D., Berger, U., Grimm, V., Lytinen, S., Sheppard, C., & Thiele, J.Improving execution speed of models implemented in Netlogo. JASSS, 2017, 20(1).

[105] Foster I. Globus toolkit version 4: Software for service-oriented systems. Journal of computer science and technology. 2006 Jul;21(4):513-20.

[106] Asadzadeh P, Buyya R, Kei CL, Nayar D, Venugopal S. Global grids and software toolkits: A study of four grid middleware technologies. High performance computing: paradigm and infrastructure, laurence yang and minyi guo (eds). 2005:431-58.

[107] Borges F, Gutierrez-Milla A, Suppi R, Luque E, de Brito Arduino M. An agent-based model for assessment of aedes aegypti pupal productivity. In2015 Winter Simulation Conference (WSC) 2015 Dec 6 (pp. 159-170). IEEE.

[108] Auld J, Hope M, Ley H, Sokolov V, Xu B, Zhang K. POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations. Transportation Research Part C: Emerging Technologies. 2016 Mar 1;64:101-16.

[109] Klijn W, Diaz-Pier S, Morrison A, Peyser A. Staged deployment of interactive multi-application HPC workflows. arXiv preprint arXiv:1907.12275. 2019 Jul 29.

[110] Faraway JJ. Does data splitting improve prediction?. Statistics and computing. 2016 Jan 1;26(1-2):49-60.

[111] Meade, A., Deeptimahanti, D. K., Buckley, J., & Collins, J. J. An empirical study of data decomposition for software parallelization. Journal of Systems and Software, 125, 2017, 401–416.

[112] Miller, Patrick J. Parallel, distributed scripting with Python. No. UCRL-JC-148522. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2002.

[113] Masad D, Kazil J. MESA : an agent-based modeling framework. In:Proceedings of the 14th Python in Science Conference, SCIPY 2015; pp. 53-60,

[114] Hagberg A, Swart P, S Chult D. Exploring network structure, dynamics, and function using NetworkX. Los Alamos National Lab.(LANL), Los Alamos, NM (United States); 2008 Jan 1.

[115] Hu P, Niu Z, He T, Chan KC. Learning Deep Representations in Large Integrated Network for Graph Clustering. In2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE) 2018 Sep 26 (pp. 101-105). IEEE.

[116] P. Calabrese, D. Shibata, ”A simple algebraic cancer equation: calculating how cancers may arise with normal mutation rates,” BMC Cancer,vol. 10(1), 2010

[117] E. Vidal, F.Thollard, C. De La Higuera, F.Casacuberta, RC. Carrasco,”Probabilistic finite-state machines-part I,” IEEE transactions on pattern analysis and machine intelligence, vol. 27(7), 2005, pp. 1013-1025.

[118] E. Vidal, F.Thollard, C. De La Higuera, F.Casacuberta, RC. Carrasco, ”Probabilistic finite-state machines-part II,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27(7), 2005, pp. 1026-1039

[119] RS. Kerbel, ”Tumor Angiogenesis,” New England Journal of Medicine,vol. 358(19), 2008, pp. 2039-2049.

[120] Quail DF, Joyce JA. Microenvironmental regulation of tumor progression and metastasis. Nature medicine. 2013 Nov;19(11):1423-37.

[121] C. Jolly, P. Van Loo, ”Timing somatic events in the evolution of cancer,” Genome biology, vol. 19(1), 2018

[122] Lc. Freeman, ”A set of measures of centrality based on betweenness,” Sociometry, vol. 1, 1977, pp. 35-41

[123] A. Goodspeed, LM. Heiser, JW. Gray, JC.Costello, ”Tumor-derived cell lines as molecular models of cancer pharmacogenomics,” Molecular Cancer Research, vol. 14(1), 2016, pp. 3-13.

[124] BELLO, Opeyemi, et al. Application of artificial intelligence methods in drilling system design and operations: a review of the state of the art. Journal of Artificial Intelligence and Soft Computing Research, 2015, vol. 5, no 2, p. 121-139.

[125] Jianhong, Artificial intelligence and data mining: algorithms and applications, 2003.

[126] Mellit A. Artificial Intelligence technique for modelling and forecasting of solar radiation data: a review. International Journal of Artificial intelligence and soft computing. 2008 Jan 1;1(1):52-76.

[127] Liu W, Pellegrini M, Wang X. Detecting communities based on network topology. Scientific reports. 2014 Jul 18;4:5739.

[128] Sugimori M, Hayakawa Y, Boman BM, Fields JZ,Awaji M, Kozano H, Tamura R, Yamamoto S, Ogata T, Yamada M, Endo S. Discovery of power-law growth in the self-renewal of heterogeneous glioma stem cell populations. PloS one. 2015 Aug 18;10(8):e0135760.

[129] Yang R, Bai Y, Qin Z, Yu T. EgoNet: identification of human disease ego-network modules. BMC genomics. 2014 Dec;15(1):314.

[130] Shen R, Goonesekere NC, Guda C. Mining functional subgraphs from cancer protein-protein interaction networks. BMC systems biology. 2012 Dec;6(3):S2.

[131] Goodspeed A, Heiser LM, Gray JW, Costello JC. Tumor-derived cell lines as molecular models of cancer pharmacogenomics. Molecular Cancer Research. 2016 Jan 1;14(1):3-13.

[132] Park J, Barabási AL. Distribution of node characteristics in complex networks. Proceedings of the National Academy of Sciences. 2007 Nov 13;104(46):17916-20.

[133] Estrada, Ernesto, and Juan A. Rodríguez-Velázquez. ”Subgraph centrality and clustering in complex hyper-networks.” Physica A: Statistical Mechanics and its Applications 364 (2006): 581-594.

[134] Kollias, Giorgos, et al. ”Fast parallel algorithms for graph similarity and matching.” Journal of Parallel and Distributed Computing 74.5 (2014): 2400-2410.

[135] Prettejohn BJ, Berryman MJ, McDonnell MD. Methods for generating complex networks with selected structural properties for simulations: a review and tutorial for neuroscientists. Frontiers in computational neuroscience. 2011 Mar 10;5:11.

[136] Pradines JR, Farutin V, Rowley S, et al. Analyzing protein lists with large networks: edge-count probabilities in random graphs with given expected degrees, J Comput Biol, 2005 ,vol. 12 (pg. 113-28)

[137] Perry, Brea L., Bernice A. Pescosolido, and Stephen P. Borgatti. Egocentric network analysis: Foundations, methods, and models. Vol. 44. Cambridge University Press, 2018.

[138] Wölfer R, Faber NS, Hewstone M. Social network analysis in the science of groups: Cross-sectional and longitudinal applications for studying intra-and intergroup behavior. Group Dynamics: Theory, Research, and Practice. 2015 Mar;19(1):45.

[139] Tashakor G, Suppi R. Simulation and computational analysis of multiscale graph agent-based tumor model. IEEE International Conference on High Performance Computing & Simulation (HPCS). 2019 Sep 4,252-258

[140] Tashakor G, Suppi R. Simulation and analysis of a multi-scale tumor model using agent clustered network International Conference on Modeling and Applied Simulation ,2019, 64-71

[141] Rai A, Pradhan P, Nagraj J, Lohitesh K, Chowdhury R, Jalan S. Understanding cancer complexome using networks, spectral graph theory and multilayer framework. Scientific reports. 2017 Feb 3;7:41676.

[142] Kamburugamuve S, Wickramasinghe P, Ekanayake S, Fox GC. Anatomy of machine learning algorithm implementations in MPI, Spark, and Flink. The International Journal of High Performance Computing Applications. 2018 Jan;32(1):61-73.

[143] T. White, Hadoop: The Definitive Guide, 1st ed. O'Reilly Media, Inc., 2009.

[144] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10.

[145] "Apache Flink: Scalable Batch and Stream Data Processing." [Online]. Available: https://flink.apache.org/

[146] P. Carbone, G. F'ora, S. Ewen, S. Haridi, and K. Tzoumas, "Lightweight asynchronous snapshots for distributed dataflows," arXiv preprint arXiv:1506.08603, 2015.

[147] B. Saha, H. Shah, S. Seth, G. Vijayaraghavan, A. Murthy, and C. Curino, “Apache tez: A unifying framework for modeling and building data processing applications,” in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015, pp. 1357–1369.

[148] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. Fern’andez- Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt et al., “The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing,” Proceedings of the VLDB Endowment, vol. 8, no. 12, pp. 1792–1803, 2015

[149] N. Doekemeijer and A. L. Varbanescu, “A survey of parallel graph processing frameworks,” Delft University of Technology, 2014.

[150] X. Lu, N. S. Islam, M. Wasi-Ur-Rahman, J. Jose, H. Subramoni, H. Wang, and D. K. Panda, “High-performance design of hadoop rpc with rdma over infiniband,” in 2013 42nd International Conference on Parallel Processing. IEEE, 2013, pp. 641–650.

[151] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser,and G. Czajkowski, “Pregel: a system for large-scale graph processing,”in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010, pp. 135–146.

[152] S. Seo, E. J. Yoon, J. Kim, S. Jin, J.-S. Kim, and S. Maeng, “Hama: An efficient matrix computation with the mapreduce framework,” in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010, pp. 721–726.

[153] [17] Apache Giraph. Accessed: December 14 2016. [Online]. Available: http://giraph.apache.org/

[154] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, “Graphx: Graph processing in a distributed dataflow framework,” in 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), 2014, pp. 599–613.

[155] N. Doekemeijer and A. L. Varbanescu, “A survey of parallel graph processing frameworks,” Delft University of Technology, 2014.

[156] G. Fox, J. Qiu, S. Kamburugamuve, S. Jha, and A. Luckow, “Hpcabds high performance computing enhanced apache big data stack,” in Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on, May 2015, pp. 1057–1066.

[157] Intel Data Analytics Acceleration Library (Intel DAAL). Accessed: January 02 2016. intel-daal

[158] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” arXiv preprint arXiv:1603.04467, 2016.

[159] H2O. Accessed: January 02 2016. [Online]. Available: http://www.h2o.ai/h2o/

[160] S. Owen, R. Anil, T. Dunning, and E. Friedman, “Mahout in action,” 2012.

[161] Someren EV, Wessels LF, Backer E, Reinders MJ. Genetic network modeling. Pharmacogenomics. 2002 Jul 1;3(4):507-25.

[162] Hellkvist M, Özçelikkale A, Ahlén A. Generalization Error for Linear Regression under Distributed Learning. arXiv preprint arXiv:2004.14637. 2020 Apr 30.

[163] Verbraeken J, Wolting M, Katzy J, Kloppenburg J, Verbelen T, Rellermeyer JS. A Survey on Distributed Machine Learning. ACM Computing Surveys (CSUR). 2020 Mar 13;53(2):1-33.

Appendices

APPENDIX A
HPCNETLOGO

Hpcnetlogo is a frontend for the concurrent execution of Netlogo experiments defined in Behavior Space (BS). The goal of this environment is that non-technological users can use HPC to run their experiments in the BS without worrying about the configuration of SGE or Netlogo-Java.This environment sends the experiments to be executed to a cluster using SGE (but the scripts can be adapted for other queues systems such as SLURM).

The Github Link: `https://github.com/hpcnetlogo/hpcnetlogo/tree/master/jobs_project/job_manager`

This frontend uses:

a) Netlogo Program (https://ccl.northwestern.edu/netlogo/5.2.1/) executed as headless in a cluster with SGE.

b) Split_nlogo_experiment (`https://github.com/ahrenberg/split_nlogo_experiment`) to create the xml files from Behavior Space.

c) Directory Lister (http://www.directorylister.com) to show the output files (result of execution).

HPCNetlogo code:

```
import os
import paramiko
from django.template.loader import get_template
from django.template import Context

def submit_job(job, hostname, username, password, model_name, cluster_number):
    """
    Executed when clicking the play button, the parameters are what passed
```

```
from the interface to this function
"""
# Directory of the split command. Change if necessary.
actual_dir = '/var/www/html/jobs_project/job_manager/'
# Netlogo directory-version
netlv = 'netlogo-5.2.1'
# Cluster Queues names
clusg = 'cluster.q@clus*.hpc.local'
clus1 = 'cluster.q@clus'
clus2 = '.hpc.local'

try:
   try:
      expand = job.e_e
      #making a directory for splitted experiments, it is current directory/experiments
      xml_dir = os.path.join(os.path.dirname(job.file_first.path), 'experiments-%d-%d'
          % (job.id, job.latest_run))
      os.mkdir(xml_dir)

      #splitting the model using command line and piutting the result in xml_dir =
          current /experiments
      if expand:
         os.system(actual_dir + 'split_nlogo_experiment.py --repetitions_per_run 1 %s
             %s --output_dir %s' % (job.file_first.path, job.experiment_name, xml_dir))
      else:
         os.system(actual_dir+'split_nlogo_experiment.py %s %s --output_dir %s' %
             (job.file_first.path, job.experiment_name, xml_dir))
   except Exception as e_s:
      print('*** Caught Split exception: ' + str(e_s.__class__) + ': ' + str(e_s))

   #setting server (cluster master node) key directories
   netlogo_dir = '/home/%s/netlogo-sge' % username
   java_prefs_dir = '/home/%s/.java' % username

   # setting job-# directory and run, e.g. job-2/1
   run_dir = os.path.join(netlogo_dir, 'job-%s' % job.id, '%d' % job.latest_run)
   output_dir = os.path.join(run_dir, 'output')
   error_dir = os.path.join(run_dir, 'error')
   model_filepath = os.path.join(run_dir, os.path.split(job.file_first.path)[1])
   experiment_filepath = ''
```

```
# login to master node
s = paramiko.SSHClient()
s.set_missing_host_key_policy(paramiko.AutoAddPolicy())
s.connect(hostname, username=username, password=password)

s.exec_command('mkdir -p %s' % run_dir)
# cd to run directory
s.exec_command('cd %s' % run_dir)

#copy experiment files to server
for experiment_filename in os.listdir(xml_dir):
   file_path = os.path.join(xml_dir, experiment_filename)
   destination_path = os.path.join(run_dir, experiment_filename)
   copy_file(s, file_path, destination_path)
experiments_count = len(os.listdir(xml_dir))

# create output dir
s.exec_command('mkdir -p %s' % output_dir)

# create error dir
s.exec_command('mkdir -p %s' % error_dir)
simulator_name = 'run-simulator-%d-%d.sh' % (job.id, job.latest_run)

# copy job submit script
context_dict = {
   'experiment_name': job.experiment_name,
   'experiment_count': experiments_count,
   'work_dir': run_dir,
   'simulator_dir': run_dir,
   'netlogo_dir': os.path.join(netlogo_dir, netlv),
   'simulator_src_dir': run_dir,
   'output_dir': output_dir,
   'model_name': model_name,
   'error_dir': error_dir,
   'simulator_name': simulator_name,
}

if experiments_count < 10:
   job_submit_script = get_script('job-submit.sh', context_dict)
elif experiments_count < 100:
   job_submit_script = get_script('job-submit0010.sh', context_dict)
```

```
elif experiments_count < 1000:
   job_submit_script = get_script('job-submit0100.sh', context_dict)
else:
   job_submit_script = get_script('job-submit1000.sh', context_dict)
j_s = run_dir+'/job-submit.sh'
copy_script(s, job_submit_script, j_s, xml_dir+'/job-submit.sh')
s.exec_command('chmod +x %s' % j_s)

# copy sge script
if cluster_number == '*':
   cluster_string = clusg
else:
   cluster_string = clus1 + cluster_number + clus2
context_dict = {
   'cluster_number': cluster_string,
   'work_dir': run_dir,
   'simulator_dir': run_dir,
   'netlogo_dir': os.path.join(netlogo_dir, netlv),
   'simulator_src_dir': run_dir,
   'output_dir': output_dir,
   'model_name': model_name,
   'error_dir': error_dir,
   'simulator_name': simulator_name,
}
sge_script = get_script('sge-script2.sh', context_dict)
s_s = run_dir+'/sge-script2.sh'
copy_script(s, sge_script, s_s, xml_dir+'/sge-script2.sh')
s.exec_command('chmod +x %s' % s_s)

# copy run simulator script
context_dict = {
   'work_dir': run_dir,
   'model_file': model_filepath,
   'experiment_file': experiment_filepath,
   'netlogo_dir': os.path.join(netlogo_dir, netlv),
   'output_dir': output_dir,
   'java_prefs_dir': java_prefs_dir,
}
run_simulator_script = get_script('run-simulator.sh', context_dict)
r_s = netlogo_dir + '/' + netlv
r_version = 'run-simulator-%d-%d.sh' % (job.id, job.latest_run)
```

```
        copy_script(s, run_simulator_script, r_s + '/' + r_version,
            xml_dir+'/run-simulator-sh')
        s.exec_command('chmod +x %s' % r_s + '/' + r_version)

        # copy .nlogo file
        copy_file(s, job.file_first.path, model_filepath)

        # execute job-submit
        s.exec_command('/bin/bash %s/job-submit.sh >> %s/output.txt' % (run_dir, run_dir))
        s.exec_command('touch %s/ok' % run_dir)
        job.latest_run += 1
        job.save()
        s.close()
        return 'Successfully ran the job'
    except Exception as e:
        print('*** Caught Run Job exception: ' + str(e.__class__) + ': ' + str(e))

def get_script(script_name, context_dict={}):
    """ Get an Script & Context """
    template = get_template(script_name)
    context = Context(context_dict)
    return template.render(context)

def copy_file(ssh, local_filepath, remote_filepath):
    """ Copy a file """
    ftp = ssh.open_sftp()
    ftp.put(local_filepath, remote_filepath)

def copy_script(ssh, script, filename, local):
    """ Copy a Script """
    f = open(local, 'w')
    for line in script.split('\n'):
        f.write(line+'\n')
    f.close()
    copy_file(ssh, local, filename)

def stop_job(job, hostname, username, password):
    """ Stop a Job """
    try:
        # login to master node
        s = paramiko.SSHClient()
```

```
      s.set_missing_host_key_policy(paramiko.AutoAddPolicy())
      s.connect(hostname, username=username, password=password)

  # send killall command - Warning: V.1.1 this will erase all the user's jobs.
  #Improvement: find the id and delete selectively.
      s.exec_command('qdel -u %s' % username)

      s.close()
      return 'Successfully stop job'
   except Exception as e:
print('*** Caught Stop Job exception: ' + str(e.__class__) + ': ' + str(e))
```

APPENDIX B

MAIN METHODS FOR AGENT BASED TUMOR MODEL IN PYTHON USING MESA AND NETWORKX

ABM Model:

```
class State(Enum):
   METASTAISE = 1 # Tumor Cells
   Transitory = 2 # daughter cell
   Stem = 3 # Normal Cells
   Dead = 0# Apoptotic Cells

def number_state(model, state):
   return sum([1 for a in model.grid.get_all_cell_contents() if a.state is state])

def number_METASTAISE(model):
   return number_state(model, State.METASTAISE)

def number_Stem(model):
   return number_state(model, State.Stem)

def number_M(model):
   return number_state(model, State.Transitory)

def number_dead(model):
   return number_state(model, State.Dead)

class TumorModel(Model):
   """A Tumor model with some number of agents"""

   def __init__(self, num_nodes, avg_node_degree, initial_outbreak_size,
       Angiogenesis_chance, Mitosis_frequency,
             recovery_chance, Angioprevention_chance):

      self.num_nodes = num_nodes
      prob = avg_node_degree / self.num_nodes
      # self.G = nx.erdos_renyi_graph(n=self.num_nodes, p=prob)
      self.G = nx.fast_gnp_random_graph(n=self.num_nodes, p=prob, directed=True)
      self.M = nx.to_numpy_matrix(self.G)
      self.M = nx.write_adjlist(self.G, "test.adjlist")
      self.N = nx.write_graphml(self.G, "test.graphml")
      self.C = nx.betweenness_centrality(self.G, k=None, normalized=True, weight=None,
          endpoints=False, seed=None)
      self.D = nx.density(self.G)
      self.grid = NetworkGrid(self.G)
      self.schedule = RandomActivation(self)
      self.initial_outbreak_size = initial_outbreak_size if initial_outbreak_size <=
          num_nodes else num_nodes
      self.Angiogenesis_chance = Angiogenesis_chance
      self.Mitosis_frequency = Mitosis_frequency
      self.recovery_chance = recovery_chance
      self.Angioprevention_chance = Angioprevention_chance

      self.datacollector = DataCollector({"METASTAISE": number_METASTAISE,
                                 "Transitory": number_M,
                                 "Dead": number_dead,
                                 "Stem": number_Stem})
      t_start = time.time()
      # Create agents
      for i, node in enumerate(self.G.nodes()):

         a = TumorAgent(i, self, State.METASTAISE, self.Angiogenesis_chance,
             self.Mitosis_frequency,
                    self.recovery_chance, self.Angioprevention_chance)
         self.schedule.add(a)
         # Add the agent to the node
         self.grid.place_agent(a, node)

      t_end = time.time()
      print("[ Lapse: {} seconds".format(t_end - t_start))
      # Transient some nodes
      infected_nodes = random.sample(self.G.nodes(), self.initial_outbreak_size)
      for a in self.grid.get_cell_list_contents(infected_nodes):
         a.state = State.Transitory
      self.running = True
      self.datacollector.collect(self)
      #np.savetxt('TV.txt', self.C, fmt='%10.5f', delimiter=',', newline='\n', header='',
          footer='', comments='# ', encoding=None)
          def Stem_Transitory_ratio(self):
              try:
                  return number_state(self, State.Stem) / number_state(self, State.METASTAISE)
              except ZeroDivisionError:
                  return math.inf

          def Density_Calculation(self):
                  try:
                      return self.D
                  except ZeroDivisionError:
                      return math.inf

          def step(self):
              self.schedule.step()
              # collect data
              self.datacollector.collect(self)

          def run_model(self, n):
              print("model run")
              for i in range(n):
                  self.step()

      class TumorAgent(Agent):
          def __init__(self, unique_id, model, initial_state, Angiogenesis_chance,
               Mitosis_frequency,
                     recovery_chance, Angioprevention_chance):
              super().__init__(unique_id, model)

              self.state = initial_state

              self.Angiogenesis_chance = Angiogenesis_chance
              self.Mitosis_frequency = Mitosis_frequency
              self.recovery_chance = recovery_chance
              self.Angioprevention_chance = Angioprevention_chance

          def try_to_infect_neighbors(self):
              neighbors_nodes = self.model.grid.get_neighbors(self.pos, include_center=False)
              transitory_neighbors = [agent for agent in
                  self.model.grid.get_cell_list_contents(neighbors_nodes) if
                                 agent.state is State.METASTAISE]
              for a in transitory_neighbors:
                  if random.random() < self.Angiogenesis_chance:
                      a.state = State.Stem
                  else:
                      a.state = State.Transitory

          def try_gain_Quiescent(self):
              if random.random() < self.Angioprevention_chance:
                  self.state = State.Dead
              else:
                  self.state = State.Transitory
                  self.try_to_infect_neighbors()



          def try_kill_cancer(self):
              # Try to kill
              if random.random() > self.recovery_chance:
                  # Failed
                  self.state = State.METASTAISE
                  self.try_gain_Quiescent()
              else:
                  # Success
                  self.state = State.Stem

          def try_check_situation(self):
              if random.random() > self.Mitosis_frequency:
                  # Checking...
                  if self.state is State.Transitory:
                      self.try_kill_cancer()

          def step(self):
              if self.state is State.Transitory:
                  self.try_to_infect_neighbors()
              self.try_check_situation()
```

Main methods for Data and Graph Visualization:

```
from mesa.visualization.ModularVisualization import ModularServer
from mesa.visualization.UserParam import UserSettableParameter
from mesa.visualization.modules import ChartModule
from mesa.visualization.modules import NetworkModule
from mesa.visualization.modules import TextElement
from .model import TumorModel, State, number_METASTAISE

def network_portrayal(G):
   # The model ensures there is always 1 agent per node

   def node_color(agent):
      if agent.state is State.Transitory:
         return '#008000'
      elif agent.state is State.METASTAISE:
         return '#FF0000'
      else:
         return '#808080'

   def edge_color(agent1, agent2):
      if agent1.state is State.Stem or agent2.state is State.Stem:
         return '#000000'
      return '#e8e8e8'

   def edge_width(agent1, agent2):
      if agent1.state is State.Stem or agent2.state is State.Stem:
         return 3
      return 2

   portrayal = dict()
   portrayal['nodes'] = [{'id': n_id,
                 'agent_id': n['agent'][0].unique_id,
                 'size': 2,
                 'color': node_color(n['agent'][0]),
                 }
                for n_id, n in G.nodes(data=True)]

   portrayal['edges'] = [{'id': i,
                 'source': source,
                 'target': target,
                      'color': edge_color(G.node[source]['agent'][0],
                          G.node[target]['agent'][0]),
                      'width': edge_width(G.node[source]['agent'][0],
                          G.node[target]['agent'][0]),
                      }
                     for i, (source, target, _) in enumerate(G.edges(data=True))]

   return portrayal

network = NetworkModule(network_portrayal, 800, 800, library='d3')
chart = ChartModule([{'Label': 'METASTAISE', 'Color': '#FF0000'},
                {'Label': 'Transitory Cell', 'Color': '#008000'},
                {'Label': 'Dead', 'Color': '#808080'}])

class RatioElement(TextElement):
   def render(self, model):
      ratio = model.Stem_Transitory_ratio()
      ratio_text = '&infin;' if ratio is math.inf else '{0:.2f}'.format(ratio)
      return 'Stem Cell/ Transitory Cells Ratio: ' + ratio_text

class METASTAISERemainingElement(TextElement):
   def render(self, model):
      infected = number_METASTAISE(model)
      return 'Metastatic remain: ' + str(infected)

class Density(TextElement):
      def render(self, model):
         Density = model.Density_Calculation()
         return 'Density: ' + str(Density)

text = RatioElement(), METASTAISERemainingElement(), Density()
n_slider = UserSettableParameter('slider', "Number", 100, 2, 200, 1)

model_params = {
   'num_nodes': UserSettableParameter('slider', 'Number of agents', 10, 10, 100000, 1,
                             description='Choose how many agents to include in the
                                 model'),
   'avg_node_degree': UserSettableParameter('slider', 'Avg Node Degree', 3, 3, 8, 1,
                                 description='Avg Node Degree'),
   'initial_outbreak_size': UserSettableParameter('slider', 'Initial Tumor Size', 1, 1,
       3200, 1,
                                    description='Initial Tumor Size'),
   'Angiogenesis_chance': UserSettableParameter('slider', 'Angiogenesis Chance', 0.4,
       0.0, 1.0, 0.1,
                                   description='Probability that vascular neighbor will
                                       be infected'),
   'Mitosis_frequency': UserSettableParameter('slider', 'Tumor propagation Frequency',
       0.4, 0.0, 1.0, 0.1,
                                    description='Frequency the nodes check whether they
                                        are infected '
                                            ),
   'recovery_chance': UserSettableParameter('slider', 'Recovery Chance', 0.3, 0.0, 1.0,
       0.1,
                                description='Probability that the tumor will be
                                    removed'),
   'Angioprevention_chance': UserSettableParameter('slider', ' Quiescent Chance', 0.5,
       0.0, 1.0, 0.1,
                                     description='Probability that a recovered agent
                                         will become '
                                              'dead in the future'),

}

server = ModularServer(TumorModel, [network, chart, *text],'Tumor Model', model_params)

server.port = 8521
```

Main methods for Power-Law Distribution and Clustering algorithm

```
self.GR = nx.read_graphml("GrowPatterm.graphml")
        self.num_nodes = nx.number_of_nodes(self.GR)
        self.prob = avg_node_degree / self.num_nodes
        self.G = nx.gnr_graph(self.redirected_nodes, self.prob, create_using=self.GR,
             seed=None)
        self.G = nx.powerlaw_cluster_graph(n=self.num_nodes, m=2, p=self.prob, seed=None)
        #self.G = nx.random_powerlaw_tree(n=self.num_nodes, gamma=3, seed=None, tries=10)
        self.GP = nx.write_graphml(self.G, "Growcluster.graphml")
        self.D = nx.density(self.G)
        self.O = nx.write_graphml(self.G,"Output.graphml", encoding='utf-8',
             prettyprint=True)
        self.M = nx.to_numpy_matrix(self.G)
        self.C = dict(nx.betweenness_centrality(self.G, k=None, normalized=True,
             weight=None, endpoints=False, seed=None))

        np.savetxt('Matrix.out', self.M, fmt='%s')
        w = csv.writer(open("Dictionary6.1.csv", "w"))
        for key, val in dict.items(self.C):
            w.writerow([key, val])

 Part2: Discovery of Sub-Graph Centrality through Egocentric Network Analysis
   self.G = nx.read_graphml("Growcluster.graphml")
        self.E = nx.ego_graph(self.G, n="3", radius=1, center=True, undirected=False,
             distance=None)
        self.SubG = dict(nx.subgraph_centrality(self.E))

         w = csv.writer(open("Subgraphdic.csv", "w"))
          for key, val in dict.items(self.SubG):
            w.writerow([key, val])
```