Simulating Self-Assembly of Nanoparticles in Tumor Environments

by Ofir Nachum

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology

~~May 2014~~ [June 2014]

Author .... Signature redacted
Department of Electrical Engineering and Computer Science
May 2, 2014

Certified by .... Signature redacted
Sabine Hauert
Postdoctoral Fellow at MIT Koch Institute
Thesis Supervisor

Certified by .... Signature redacted
Mieszko Lis
Graduate Student at MIT CSAIL
Thesis Supervisor

Certified by .... Signature redacted
Sangeeta Bhatia
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by .... Signature redacted
Srini Devadas
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ..... Signature redacted ....
Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Simulating Self-Assembly of Nanoparticles in Tumor Environments

by

Ofir Nachum

Submitted to the Department of Electrical Engineering and Computer Science
May 2, 2014
In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Self-assembly is important in nanomedicine and increasingly plays a role in drug-delivery or imaging applications in tumors. Predicting behavior and dynamics of nanoparticle systems is very difficult, especially when assembling and disassembling particles are involved. To address this challenge, the Bhatia lab has developed NanoDoc (`http://nanodoc.org`), an online game that allows users around the world to design and simulate nanoparticle treatments. During this project, we were able to implement mechanisms to effectively describe and simulate self-assembly in NanoDoc. As a bench mark for our simulator, we show that we are able to reproduce laboratory experiments in the literature. The simulator was then made available to the crowd and a challenge was proposed that requires users to perform self-assembly in a scenario aimed at improving the accumulation of imaging agents in tumors.

Thesis Supervisor: Sabine Hauert
Title: Postdoctoral Fellow at MIT Koch Institute

Thesis Supervisor: Mieszko Lis
Title: Graduate Student at MIT CSAIL

Thesis Supervisor: Sangeeta Bhatia
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Srini Devadas
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

Many thanks goes of course to my thesis advisors.

Sabine, for her incredible vision for NanoDoc and what it should and could be. I have Sabine to thank for allowing me to play such a large role in NanoDoc's early development and guiding me through this MEng Thesis project. Sabine also deserves thanks for her careful editing and review of my thesis proposal, several poster and oral presentations of this work, as well as this final MEng Thesis.

Mieszko, for his helpful comments regarding SSC's performance, which were instrumental in several of the performance improvements we present in this thesis. Thanks to Mieszko also for reviewing this thesis report and offering valuable suggestions.

Sangeeta and Srini, for their wisdom and guidance through this project.

Thanks as well goes to Justin Lo, who was also instrumental in NanoDoc's early development and continued to help Sabine and I with suggestions and several UI designs for NanoDoc's new self-assembly UI.

Thanks to the MIT Course 6 Department, for offering such wide and varied opportunities (classes, research, and projects) for me and many other students.

Thanks to my family for their support and encouragment during my time at MIT, as well as making sure I had abundant opportunities to learn on my own schedule before MIT.

Thanks to my friends for keeping me entertained during my time at MIT.

Contents

List of Figures

Chapter 1

Introduction

Increasingly sophisticated nanoparticles are being engineered to transport treatments and imaging agents to tumors. These particles are able to passively accumulate in tumor tissue through the enhanced permeability and retention effect. Their diffusion and binding characteristics then allow them to distribute in the tumor tissue in a controllable manner [6]. Recent work in nanomedicine further uses self-assembly to dynamically change nanoparticle properties. Self-assembly of nanoparticles refers to their ability to bind and interact with each other to form a supramolecular structure exhibiting emergent properties not present in the original collection of nanoparticles. Self-assembly has been studied in the past in robotics, chemistry, and nanomedicine [23]. Treatments utilizing self-assembly have been shown to have superior sensitivity for molecular targets [12], thus enhancing a treatment's role as a drug-delivery vehicle or tumor imaging agent.

Designing nanoparticles is typically done empirically through trial and error over the course of lengthy experiments. Recently, the Bhatia lab has been able to simulate the transport of nanoparticles in tumor tissue. Using this simulator they have created an online game called NanoDoc that outsources the exploratory phase of nanoparticle design to the crowd. The project builds on previous success in crowdsourcing scientific problems such as protein-folding [2]. We believe that the desire of the general population to learn about nanomedicine and help in the fight against cancer will encourage large-scale crowd participation [7].

NanoDoc utilizes the stochastic simulation compiler (SSC) developed at MIT [11] to simulate nanoparticle treatments. The system is able to simulate nanoparticles that diffuse, bind to nanoparticles or cells in their environment, and release cargos. The next step is to allow nanoparticles to self-assemble into functional structures. Unfortunately, simulating self-assembly is not simple. Prior to this work, simulating self-assembly in tumor enironments has been time-intensive, limited in abilities, and overall impractical.

We analyze and breakdown simulation steps in SSC to better understand it. We then introduce a novel method for simulating self-assembly. We use this efficient method to implement self-assembly in NanoDoc both by implementing new simulation methods in the backend and updating the user interface appropriately. Our new system is able to recreate real laboratory experiments (namely, [18]). In addition, our system can handle simulating treatments which include particles that self-assemble into compounds with up to ≈ 18 nanoparticles in a reasonable amount of time (< 2 minutes). We complete the project by releasing a self-assembly-related challenge to the crowd, and hope that it can lead to new and improved tumor treatment mechanisms.

Chapter 2

State of the Art

2.1 Self-Assembly

The ability for nanoparticles to self-assemble has shown promise for treatment and imaging applications in cancer. A self-assembling collection of nanoparticles is a group of nanoparticles whose individual properties and attributes cause them to bind to each other. The binding of nanoparticles yields larger and more complex nanoparticles which exhibit emergent properties which are not present in the original collection of nanoparticles. This self-assembly can be a natural course of action or triggered (or destabilized) by an outside factor such as heat, light, a magnetic field, or the presence of some other particle. Recent work has shown how self-assembly can enable effective and specific nanoparticle treatments [20].

In essence, a nanoparticle treatment is a drug delivery system which is able to kill cancerous cells while minimizing harmful effects on healthy cells [20]. Examples of self-assembly can be found in the literature. Micelles for example can effectively encapsulate drugs for killing tumor cells as shown by Rösler et al. [20]. The effectiveness of the treatment is dependent on releasing the drugs with great specificity in the cancerous region. In this respect, the self-assembly properties of copolymers can be utilized. Rösler et al. show that the time of release can be adjusted by small modifications to the copolymer structure. Additionally, a change in surrounding temperature can destabilize the structure and lead to a release of the drug. This way, the self-assembling copolymers provide the ability to design effective treatments.

A similar method of drug delivery has been verified by [26]. This method uses amphiphilic γ-PGA, which again self-assemble to encapsulate antigenic proteins and deliver them to antigen-presenting cells. This then elicits an immune response by cytotoxic T lymphocytes, which kill the antigen-presenting cells. Experiments showed that the nanoparticles were able to deliver the drug to antigen-presenting cells with high efficiency, successfully inhibiting the growth of tumors in mice.

An application of self-assembly for cancer therapy, though not drug delivery-related, is shown by Nam et al. [15]. They use "smart" gold nanoparticles that are triggered by a pH change to self-assemble and form aggregate structures in tumor tissue. These aggregate structures exhibit emergent light absorption properties. Notably, the structures allow photothermal cancer therapy to be more selective and efficient at killing tumor cells.

Self-assembling nanoparticles can also be valuable for their disassembling properties. Wong et al. [25] present a proof of concept multistage nanoparticle treatment: a 100nm nanoparticle leaks into tumor tissue via the enhanced permeation and retention effect. The nanoparticle is subsequently exposed to proteases that are expressed in the tumor environment, causing the particle to disassemble into 10nm nanoparticles. These smaller particles are better able to quickly diffuse deep into tumor tissue.

Another example of self-assembly (albeit not directly tumor-related) is the discovery of magnetic nanoparticles that self-assemble in the presence of viral particles. These nanoparticles self-assemble to create large structures with enhanced magnetic properties in the presence of adenovirus-5 and herpes simplex virus-1 [17]. This property is used in the detection of viral infection as well as imaging of the viral distribution.

Highlighting the potential for self-assembling nanoparticles to target tumors with high specificity, an article by von Maltzahn et al. [12] exhibits a collection of nanoparticles which mimick **AND** and **OR** boolean operations. The self-assembly of these particles is directed via these operations, which are governed by enzymatic activity associated with aspects of tumorigenesis. This gives added support to the idea of creating nanoparticle treatments which are increasingly tailored to a specific tumor scenario. In a similar fashion, Gröschel et al. [5] exhibit methods of specially designing nanoparticles to yield specific large, spatial configurations, some of which can reach sizes of several micrometers (on the order of a thousand times bigger than a single component nanoparticle).

A paper that we will come back to several times is a paper of Perrault et al. which designs a tumor imaging mechanism via in-vivo self-assembly [18]. Co-assembling nanoparticles work together to yield a fast and efficient fluorescent tumor-imaging mechanism. The key to their method is to inject a large diameter biotin-GNP (gold nanoparticle coated with biotin) that accumulates in the tumor tissue over 24 hours. Subsequently, small diameter fluorescent streptavidin are injected. These particles distribute fast due to their small diameter. Biotin and streptavidin have a high binding affinity, causing the fluorescent particles to stick to the biotin-GNP, yielding high fluorescence in the tumor tissue. Thus, the large diameter biotin-GNP act as a tumor-specific anchor to the small fluorescent particles. Perrault et al. analyze the effects of this in-vivo method of fluorescence imaging versus a pre-assembled method in which GNP-A750, a pre-assembled fluorescent particle, is injected. They find that the method of in-vivo co-assembling particles yields rapid accumulation and more effective tumor imaging.

There are many additional examples in the literature that show that self-assembly can play a quintessential role in drug delivery via nanoparticle treatment. It is thus imperative that self-assembly simulation mechanisms be devised to sufficiently model this phenomenon, so that bioengineers are better able to predict the behavior of their nanoparticle treatments.

2.2 Simulation

Computer simulations and mathematical models are useful in predicting the behavior of nanoparticle treatments. Some of the most detailed simulations make calculations and model interactions on individual atoms. While these simulations can be very accurate, they are infeasible for practical applications. Recent work has shown that a few thousand processors can, in the span of a day, simulate systems containing 1 million particles (corresponding to 10 million atoms) for 100ns [10]. This is impractical for the needs of many bioengineers researching tumor environments, where simulations must be able to predict behaviors of particles over 24 or 48 hours in a timely manner.

In order to allow for simulations to give feedback to bioengineers on their designs in a timely and accurate manner, many molecule-to-molecule interactions (fluid dynamics, diffusion, cell binding and internalization, etc.) must be abstracted away via mathematical relations. Much work has gone into creating mathematical models of tumor environments. For example, the kinetics of diffusion, tumor uptake, and nanoparticle-receptor and tissue interactions have been studied and mechanisms for stochastically simulating these devised [4, 3]. The ability for these models and simulations to help bioengineers is shown in several examples in the literature.

Thurber et al. studied the distribution of antibodies in tumor environments [22]. They analyze the behavior of two types of antibodies and then compare the observed behavior to that predicted by theoretical models. The theoretical models used properties of the antibodies such as diffusion coefficient, void fraction, and bivalent binding to predict tumor distribution. They find the models to be validated by experimental results, and thus state that these models can help bioengineers optimize their treatments.

Hauert et al. provide a notable example of using simulation and modelling to guide nanoparticle design [6]. They work with both a deterministic and stochastic computer model to determine strategies for increased tissue penetration and accumulation of nanoparticles. The computer model is able to simulate nanoparticle potency, diffusion, as well as binding and cell internalization kinetics. The models were able to guide the bioengineers to a strategy of 'delayed binding' for nanoparticles that allows for better tissue penetration and accumulation.

In a similar fashion, Wittrup et al. use mathematical models to optimize for tumor uptake [24]. They take mathematical models for tumor uptake and tumor clearance of nanoparticles, and analyze the relationships of these rates to particle size. The optimum particle size for tumor uptake is found to be about the size of Immunoglobulin G molecules.

Models and simulations can give bioengineers a better understanding of the kinetics of the biochemical system. Martinez-Veracoechea et al. show this benefit by investigating observed 'super selectivity' of nanoparticles with multiple ligands binding to a surface with a certain concentration of receptors [13]. A model is constructed which helps to show that the phenomenon is due to a nonlinear increase of ligand-receptor binding arrangements. Moreover, methods for designing nanoparticles for optimal binding are described. In addition to mathematical models, the group also validates predictions with a stochastic simulation.

These examples show that simulation of diffusion, uptake, clearance, and cell binding and internalization in tumor environments is well-studied in the literature. However, simulation of self-assembling particles in tumor environments is not as developed. Examples in the literature of simulations of self-assembling particles often simulate at the molecular stage and can do so for simulations of at most several hundred nanoseconds [16, 8, 10]. While these simulations are useful in providing accurate representations of the atom-to-atom and molecule-to-molecule kinetics of self-assembly of micelles, monolayers, diblock copolymers, and other structures, they are impractical for bioengineers studying tumor environments, which require a time- and space-scale several magnitutes larger.

Chapter 3

Methods

3.1 Self-Assembly Simulation Requirements

We concretely define what capabilities any self-assembly simulation must have:

- **Assembly**: Particles or compounds A, B with matching receptors can bind to form a new compound AB. This assembled compound can go on to bind with other particles or compounds, as long as it has open and matching receptors.

- **Disassembly**: Conversely, an assembled compound AB can disassemble to yield A and B.

- **Emergence**: An assembled compound may have different and emergent properties from any of its individual components. For example, a compound AB will have a lower diffusion rate than either A or B, individually.

- **Retainment**: An assembled compound may retain some of the properties of its individual components. For example, if A can individually bind to and internalize in a cell, then so can compound AB.

3.2 SSC as a Simulation Tool

To simulate self-assembly we want to make as few assumptions as possible. For this reason, stochastic simulations, which apply biochemical rules (like diffusion or binding) at probabilistically chosen time points that reflect the rule's specified propensity, are best. The stochastic simulation compiler (SSC) is an open-source tool for creating exact stochastic simulations of biochemical reaction networks. It was created and developed by Mieszko Lis with the Devadas lab [11]. We use SSC in our project as the backbone to our self-assembly simulation mechanisms.

3.2.1 How SSC Works

The input to SSC is a high-level description of the reaction system. The reaction grid is discretized into cubic subvolumes, *regions*. In the case of most of our simulations, the reaction grid is a two-dimensional array of cells. This array of cells is discretized into cubic regions, each holding one cell, which is either vessel, healthy tissue, or cancerous tissue. SSC has the capacity to represent adjacent regions of the same type as a single *coalesced* region.

Reactions and diffusion in SSC can be specifed as actions which alter compounds of a region. These reactions and diffusions can be restricted to any subregion. In the case of most of our simulations, nanoparticles are initially present only in vessels and subsequently diffuse into tissue. Rules for diffusion into and through tissue are input to SSC. Additionally, rules for binding to or internalizing in cells are implemented. Each rule has an associated propensity. A propensity p associated with rule R means that R is applied rougly once every $1/p$ seconds.

In order to create a simulator, SSC takes as input a `rxn` file which specifies the regions, initial particles, and rule set. SSC first expands the set of present compounds for each distinct region. It does so by applying the reaction rules to any possible combination of substrates. This creates an expanded set of reachable compounds and the reactions which yield these compounds. This expanded set is then used to generate assembly code for a binary executable which simulates the biochemical reaction system.

The generated binary can then take in a `cfg` file (a file including some simulation parameters), simulate the system, and output the simulation to a separate file. The binary simulates by summing all the propensities to yield P. At each step, an exponential random variable r with mean $1/P$ is generated. The simulator moves forward r seconds, picks a rule to apply at random (so that the probability of choosing rule R is proportional to the propensity of R), and then applies that rule.

Appendix A includes an example `rxn` file and a corresponding example `cfg` file for a simple 3×3 grid with two distinct particles that can bind to each other.

3.2.2 Performance of SSC

Before we devise effcient methods for simulating reaction networks, it is important to better understand SSC and to single out specific simulation parameters that make SSC performance faster or slower. Essentially, we must identify and isolate bottlenecks in our methods of compiling and executing tumor scenarios.

To this end, we devised a basic simulation - a 10×10 grid of tumor cells (all with receptor A) and one vessel cell located in the upper left corner of the grid. The simulation injects 10,000 nanoparticles over 48 hours. The nanoparticles follow the simple rules: binding to and internalizing in cells. Binding happens at propensity 1×10^{-7} and internalizing at propensity 1×10^{-5}. The nanoparticles diffuse at propensity 7.572×10^{-3}.

With this basic simulation as a starting point, we varied the following parameters:

- Grid size.
- Number of nanoparticles.
- Number of reactions (x reactions means that instead of only A receptors, we included $A_1, \ldots, A_x$ receptors on different cells).
- Diffusion rate.

For each variation of these parameters one at a time, we create two simulations - one with coalescing (multiple neighboring and equivalent regions are merged into a single region), and one without. The simulation length is set to 48 hours (172800 seconds). For each variation of the basic challenge, we compiled and executed the binary 10 times, averaged the times, and analyzed the results. All time measurements (throughout this project) were performed on the `rous.mit.edu` Linux cluster using x86 architecture, 96 processing cores, and a 304GB RAM.

We find that compile time is mostly determined by the "complexity of the challenge regions." Namely, the number of distinct regions (grid size) and the number of distinct reactions determines compile time. Moreover, coalescing helps reduce the compile time. This makes sense, because SSC compilation must expand the reaction rule set for each defined region.

On the other hand, execute time is determined by the "number of things going on." Specifically, the number of nanoparticles, the number of reactions, and diffusion rate have a strong effect on execute time. Coalescing does not reduce execute time. This makes sense, because these parameters contribute to the total propensity, and the higher the total propensity, the more often the executable must stop to perform a reaction.

Results of these experiments, including graphs of compile and execute times, are shown in Appendix B.

Knowing the key parameters affecting performance helps us devise efficient methods for simulating challenges using self-assembly. Specifically, we come to realize that performance of self-assembly using SSC is mostly a problem of compilation. That is, the main issue of including self-assembly in simulations is that it drastically increases the number of compounds SSC must expand during compilation. Reducing the number of compounds is a key step in gaining efficient methods for defining self-assembly.

3.3 Defining Self-Assembly

We devise two methods for defining self-assembly. They are best introduced in the context of a scenario, so consider the following: A particle A can self-assemble with itself linearly as well as with a particle B. The compounds formed in this scenario are thus strings of A's with several B's attached, as shown in Figure 3-1.

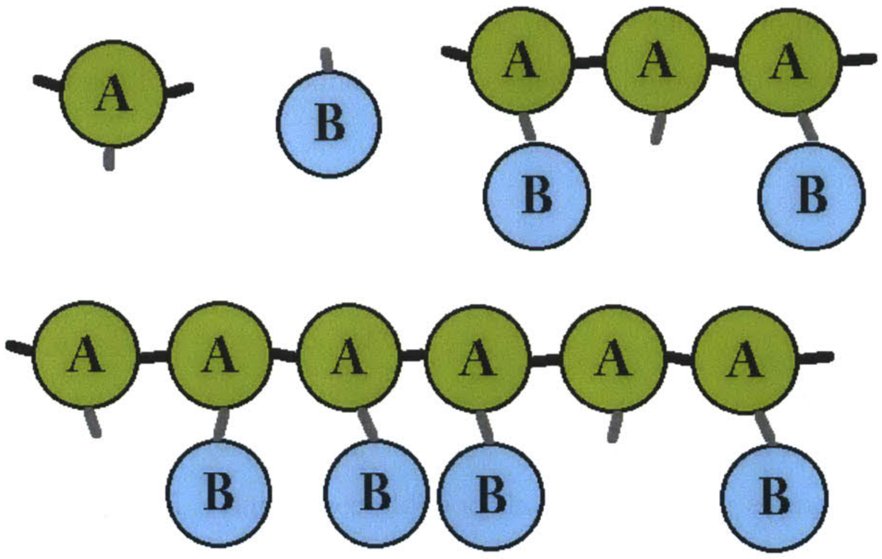


Figure 3-1: Some of the possible compounds in our hypothetical scenario involving particles A and B.

The first method for defining self-assembly in SSC is the *natural method.* It follows the rule-based spirit of SSC to define self-assembly in the most natural way. The scenario described would be thus defined by two rules:

$$A + A \rightarrow A\#A$$

$$A + B \rightarrow A\#B.$$

This method is succinct and simple. Assembly and disassembly are easy to define using the # operator in SSC denoting binding. Emergence has to be defined specifically for every compound, but retainment is given for free (SSC applies any rule for A to any compound containing A). The downside of this method is that it can (and often) leads to exponentially many different compounds, each of which must be computed by SSC. Specifically in our scenario, SSC will compute all $\Theta(2^n)$ possible strings of A's and B's during compilation, where n is the limit on the length of the strings.

The second method for defining self-assembly in SSC is the *precomputation method.* In this method, we treat a compound as a 'bag' of its components and their open receptors, regardless of the specific structure. This method requires us to precompute the possible compounds our scenario could yield and efficiently present them to SSC. The scenario described would be defined by a number of rules of the form

$$\begin{aligned} &COMP(A = a_1, B = b_1) + COMP(A = a_2, B = b_2) \\ &\qquad\qquad \rightarrow COMP(A = a_1 + a_2, B = b_1 + b_2). \end{aligned} \tag{3.1}$$

This method is unfortunately not as succinct and simple. Defining assembly and disassembly requires precomputing all possible compounds. Emergent and retained properties must also be defined for each precomputed compound. However, the advantage of this method is that it reduces the number of compounds that SSC needs to consider. For example in our scenario, the number of distinct compounds is $\Theta(n^2)$, where n is the limit on the length of the string of particles.

We can analytically compute the number of distinct compounds in the general case as well. Suppose we have a reaction with k distinct particles and we are interested in how many compounds these particles can form with up to n particles. For the sake of simplicity, assume that each particle can bind with each other particle and that there is no limit on how many particles a single particle can bind. In addition, assume that there exists at least one n-particle compound.

In the natural method, each compound essentially forms a graph structure, where vertices represent particles and edges represent bindings between particles. A compound corresponds to a specific graph with a specific vertex-particle configuration. Thus, for each obtainable graph with l vertices, there are k^l distinct compounds. Therefore, if $t(l)$ is the number of possible graphs that this particle set can yield on

l vertices (particles), the number of distinct compounds with up to n particles is

$$N_{nat} = \sum_{l=1}^{n} t(l)k^l \geq k^n.$$

In the precomputation method, a compound is simply represented by the number of each particle component it has. Thus, the number of compounds with l particles is $\binom{l+k-1}{k-1} \leq (l+k)^{k-1}$. Therefore, the total number of distinct compounds with up to n particles is

$$N_{pre} = \sum_{l=1}^{n} \binom{l+k-1}{k-1} \leq (n+k)^k.$$

Now we have a lower bound on N_{nat} and an upper bound on N_{pre}. We can treat k as a constant, since applications of self-assembly in the literature rarely use more than $5-6$ distinct particles. Thus, we realize that N_{nat} is bounded below by an exponential and N_{pre} is bounded above by a polynomial in terms of the limit n. This shows analytically that the number of compounds in the precomputation method is polynomial, rather than the exponential number in the natural method.

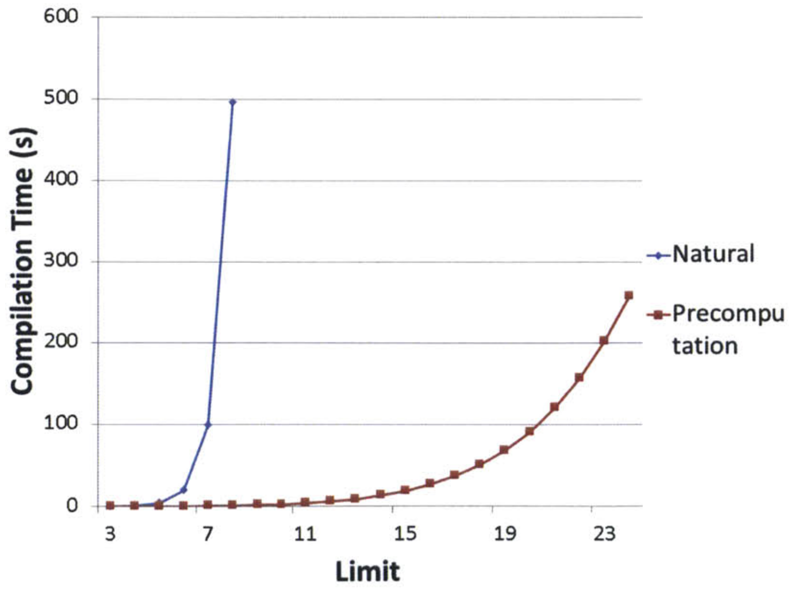


Figure 3-2: Self-assembly compile times using the natural method (blue diamonds) and the precomputation method (red squares). The precomputation method is far superior and can handle much higher expansion limits. Here, an expansion limit of 20 means that compilation resulted in compounds with up to 20 nanoparticles.

Figure 3-2 shows the performance results of the natural and precomputation methods on our example scenario. We see that the natural method becomes infeasible to simulate beyond compounds of 7 particles while the precomputation method is able to simulate compounds consisting of 20 or more particles in a reasonable amount of time.

One might posit that while the precomputation method does exploit symmetry to reduce the number of distinct compounds and results in a much smaller compilation time, it may also lead to a new bottleneck: the precomputation step. While this is a reasonable statement, it turns out to be false. Automatic precomputation can be implemented efficiently using a series of 'combining' rounds in which each pair of existing compounds is combined to form a new compound. Thus, in total, there is only a polynomial number of pairs of compounds necessary to consider. Once we implemented automatic precomputation (described in Section 3.5.2), we were able to analyze the performance of precomputation and comfirmed that it is negligible (< 0.01 seconds). Namely, Figure 4-2 exhibits this. Therefore, it is clear that the precomputation method of expressing self-assembly is superior.

3.4 Recreating Laboratory Experiments

The precomputation method is able to fulfill our requirements for a self-assembly simulation mechanism: assembly, disassembly, emergence, and retainment. Moreover, we can reproduce laboratory experiments using this method. Specifically, we recreate Perrault et al.'s experiments using in-vivo co-assembling particles to yield rapid and efficient tumor imaging.

Recall that Perrault et al. initally inject large-diameter biotin-GNP and subsequently inject small-diameter fluorescent streptavidin. The streptavidin bind to the biotin-GNP 'anchors,' thus localizing the fluorescent particles in the tumor tissue [18]. This in-vivo assembly tumor-imaging method is compared against an injection of pre-assembled fluorescent GNP-A750.

To recreate these experiments in our simulations, we model the tumor as a sphere with diameter $2r_{tumor} = 0.75\text{cm}$, given by [18]. The volume of a mouse was estimated as $V_{mouse} = 20\text{cm}^3$ from a weight of 20g given in [14]. Cells are modeled as cubes with side lengths $s = 10\mu\text{m}$. The volume of a cell is then $V_{cell} = s^3$.

Perrault et al. mention the injection amounts of biotin-GNP and GNP-A750 as 1×10^{12} particles, of which 0.72% accumulate in the tumor. To estimate the amount of streptavidin reaching the tumor tissue, we calculated the total streptavidin injected (2.5μM solution of 150μL) and multiplied this by the ratio V_{tumor}/V_{mouse}, assuming uniform distribution.

Simulating these experiments also requires knowing the binding rates. The biotin-streptavidin dissociation rate is given by $k_d = 2.4 \times 10^{-6}\text{s}^{-1}$ [19]. The association rate is then calculated as $k_a = \frac{k_d}{K_d} \cdot \frac{1}{10^3 \cdot V_{cell} A}$ where $K_d = 4 \times 10^{-14}$M is the dissociation constant [9] and $A = 6.022 \times 10^{23}$ is Avogadro's number. We are also given that biotin-GNP can bind at most 500 streptavidin particles [18].

Radii of particles are taken from [18]: $r_{GNP} = 37.5\text{nm}$ and $r_{strep} = 3.5\text{nm}$. The compound formed by biotin-GNP bound to fluorescent streptavidin and the compound GNP-A750 is calculated to have radius $r_{GNP} + 2r_{strep}$. The diffusion rates are then calculated by the Stokes-Einstein equation [21].

The depletion rate of the small-diameter fluorescent streptavidin is calculated as $\frac{1}{Ld}$ where d is the diffusion rate and L is the mean length of a discrete random walk in a space of tumor volume V_{tumor} and tumor surface area (envelope) A_{tumor}. $L = \frac{4V_{tumor}}{A}$ is given by Blanco et al. [1].

Therefore, our rule-set for the in-vivo assembly experiments is given by the appropriate diffusion and depletion rules as well the assembly and disassembly rules. These rules are determined via the precomputation method as

$$GNP(strep = k) + strep \text{ at } k_a \rightarrow GNP(strep = k + 1)$$

and

$$GNP(strep = k + 1) \text{ at } (k + 1) \cdot k_d \rightarrow GNP(strep = k) + strep,$$

where k ranges from 0 to 499. Note that for dissociation we multiply the base rate by $k + 1$. This is because there are $k + 1$ possible individual fluorescent streptavidin complexes that can dissociate from the biotin-GNP. Each of these dissociates at propensity k_d and thus any one of them dissociates at propensity $(k + 1) \cdot k_d$.

The rule-set for the pre-assembled experiments is given simply by the diffusion of GNP-A750.

The numerical simulation results and a comparison of these results with the results of Perrault et al. is presented in Section 4.1.

3.5 Generalization

In order to make this efficient and realistic method of simulating self-assembly accessible to bioengineers, we implemented it in NanoDoc. NanoDoc is a tool that provides a nanoparticle treatment design interface for an SSC-based backend. NanoDoc previously was only able to simulate diffusion, cell-binding, and nanoparticle bind and release. By implementing self-assembly capabilities, we hope that bioengineers and the general public find it easier to develop promising tumor treatments.

3.5.1 NanoDoc

NanoDoc is the union of a user-friendly browser interface and a server with a database and simulator. It aims to provide bioengineers with the capability to design tumor scenarios, which they can then release to the crowd, so that ordinary users can design and simulate effective nanoparticle treatments. The website presents a series of bioengineer-created *challenges*. A challenge is composed of a *scenario*, an arrangement of cancerous and healthy cells in the tissue of a hypothetical patient, as well as a *mission* and a *fitness* which define the objectives and measures of success (e.g. killing all the tumor cells). The user must design a *treatment*, a specific assortment of nanoparticles (*agents*) to be released into the patient's bloodstream, that adequately fulfills the mission and fitness of the challenge.

Figure 3-3 is a simplified diagram of the whole system.

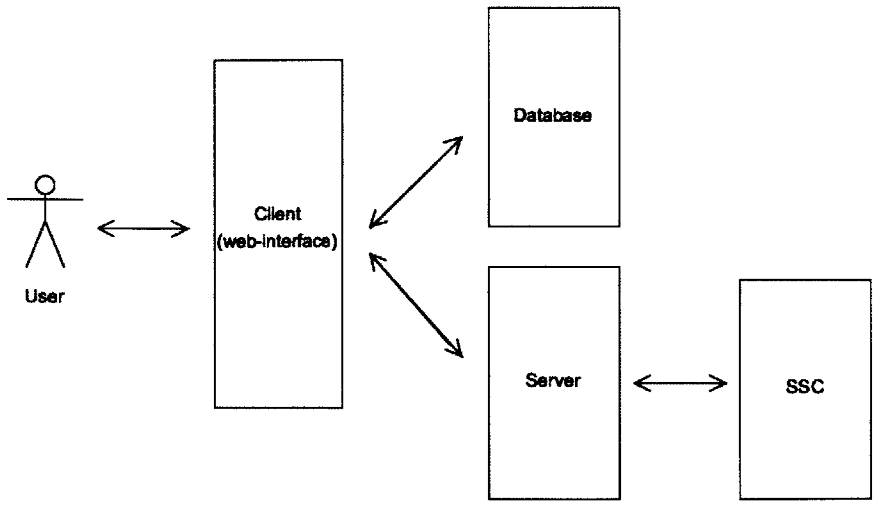


Figure 3-3: The system design of the project. The user (bioengineer or player) only sees the web-interface via their client's browser. The client communicates with the NanoDoc database and the server which handles simulations using SSC.

Server-Side Components

NanoDoc's server acts to connect the user's browser to the database and SSC, the simulator used for NanoDoc's user-defined treatments.

The web-interface relies on a centralized server to connect the user to the database. The database stores information on the user, as well as the nanoparticles, treatments, and challenges the user has already created. These data are sent to the browser to be graphically represented.

The server is also the component which runs the simulations of treatments on tumors. It does so by taking treatment and tumor scenario specifications from the user's browser and then translating those into a format to be input to SSC. The input to SSC is the collection of cancerous and healthy cells (specified by the challenge) as well as the nanoparticles (specified by the treatment) and how they interact with patient cells or each other via diffusion and binding. SSC returns the results of the simulation, which are subsequently sent to the user's browser to be displayed graphically. Figure 3-4 exhibits the steps the server takes to convert a user's treatment data to an SSC simulation.

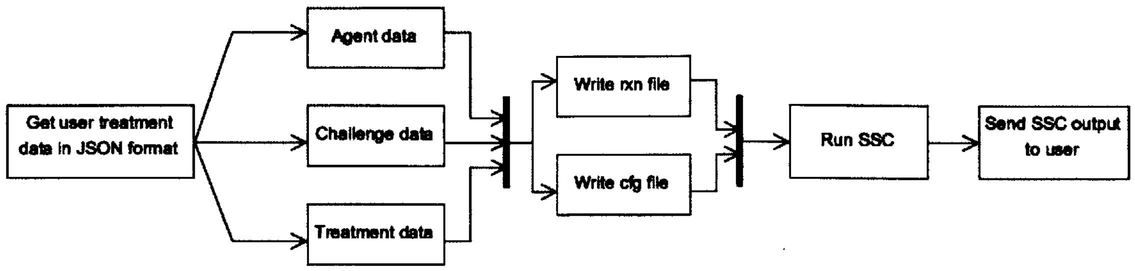


Figure 3-4: The server takes in user data from clients, and uses it to write appropriate `rxn` and `cfg` files which are then input to SSC.

User Experience

NanoDoc has an extensive user-interface built to educate and motivate the user to build effective treatments. The most recent version of NanoDoc is currently available at `http://nanodoc.org`.

The web-interface initially introduces the user to the capabilities of the system through a series of *levels*. Each level is composed of a challenge. The introductory levels are designed to train the user on how to build effective treatments that respond appropriately to a specific challenge. For example, the first levels show the user how the size of a nanoparticle affects its rate of diffusion. Later levels show how the molecular coat (sensor) surrounding a nanoparticle allows it to bind with different strengths (affinities) to tumor cells. Subsequently, the concept of an effector, a molecule that can be released by an agent upon specific triggering, is taught.

Once the user has passed these introductory levels, she will be able to tackle real-life challenges designed by bioengineers. Bioengineers are designated special users who may create a challenge which mimics a tumor scenario. Bioengineers can submit their challenge to be released to the crowd.

Figure 3-5 shows a use-case diagram for the bioengineer. Note that a bioengineer has all the capabilities of a normal user plus the ability to design and edit challenges.

The bioengineer may create a challenge, edit this challenge, and eventually submit it to the crowd. Once it is submitted to the crowd, players will be able to create and edit potential treatments for the challenge. A bioengineer designing a treatment may

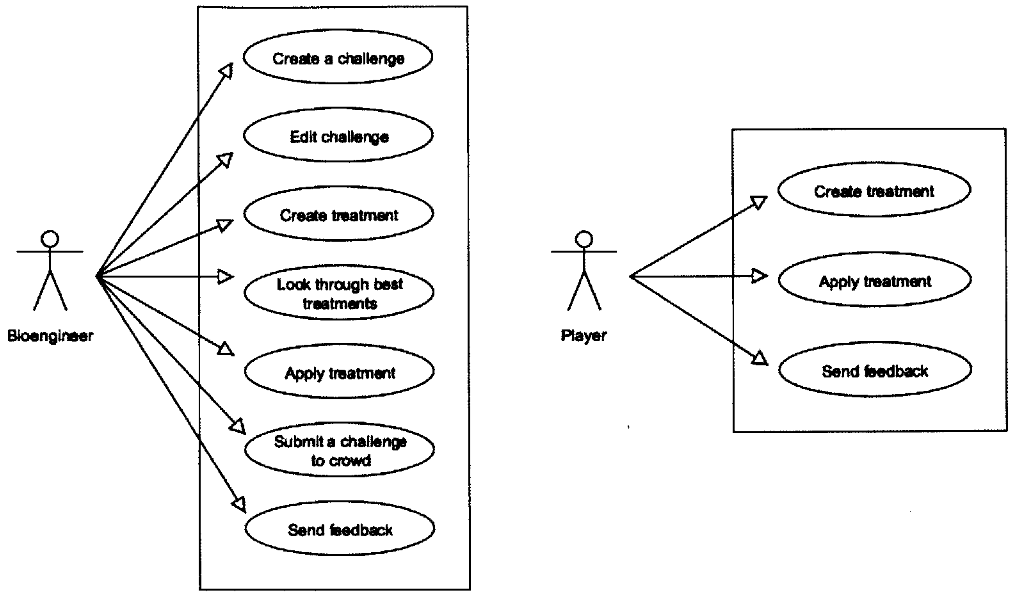


Figure 3-5: Use-case diagrams for bioengineers and players using the interface. Each bubble shows a possible use of the interface.

simulate the treatment on the tumor challenge via applying the treatment. The original bioengineer can track the crowd's progress on the challenge by looking through the best treatments (pooled from the crowd). The bioengineer also has the ability to send feedback to us, the developers.

Appendix C includes Figures C-1 to C-4 which are flow charts that exhibit the intended work-flow.

3.5.2 Implementing Self-Assembly in Simulations

Figure 3-6 exhibits the steps we added to the server to convert a user's treatment data to an SSC simulation. Note the addition of a coalescing stage, a precomputation stage, and a `traj` parsing stage (simulating a coalesced scenario results in SSC outputting a `traj` file with the simulation results).

The precompuation step is expanded further in Figure 3-7. The precomputation is performed by repeatedly combining compounds to add to a working set of compounds. The initial set of compounds is taken as the collection of agents in the treatment as well as the cell receptors present in the scenario. A compound is represented as a

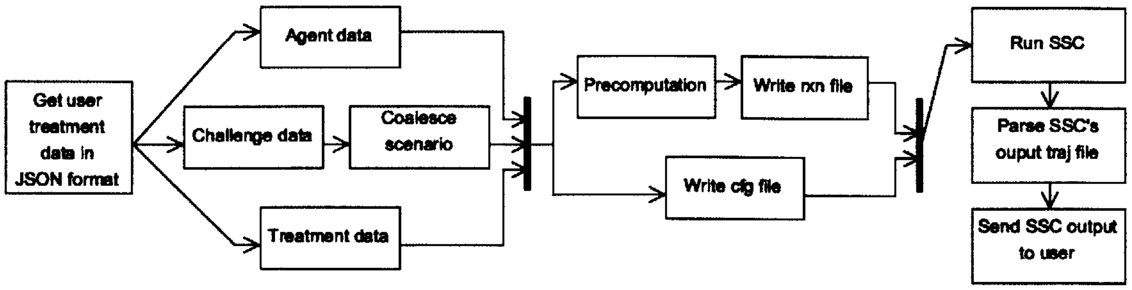


Figure 3-6: The server takes in user data from clients, and uses it to write appropriate `rxn` and `cfg` files which are then input to SSC.

list of components (agents, effectors, or cell receptors) and a list of open receptors (unbound ligands). Note that there is no sense of structure in this representation. Also, there are possibly many distinct formations of the same compound.

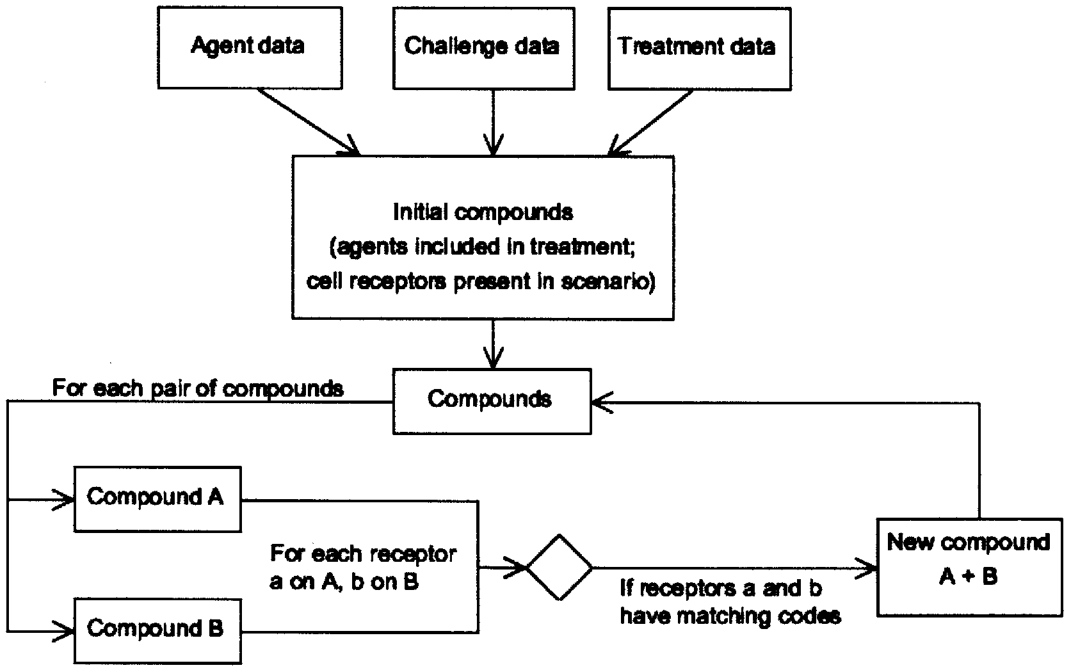

Agent data
Challenge data
Treatment data
Initial compounds (agents included in treatment; cell receptors present in scenario)
For each pair of compounds
Compounds
Compound A
Compound B
For each receptor a on A, b on B
If receptors a and b have matching codes
New compound A + B

Figure 3-7: Precomputation of compounds.

In each combining round, every pair of compounds A, B is chosen. After checking that A and B have not been combined already, we check if A and B have matching open receptors (that is, A and B can bind to each other). For each receptor a on

A matching a receptor b on B, we combine A and B via binding $a \leftrightarrow b$ to yield compound C. If this compound is new, we add it to the working set. We also store the reaction $A + B \leftrightarrow C$.

We set the association propensity $(A + B \rightarrow C)$ to

$$f(a, b) \cdot (\text{\# of receptors of type } a \text{ on } A)(\text{\# of receptors of type } b \text{ on } B).$$

The function $f(a, b)$ is based on the user-defined strengths of a and b. This accounts for the fact that the binding of A and B via a and b can be made via any pair of a on A and b on B.

We set the dissociation propensity $(A + B \leftarrow C)$ to

$$g(a, b) \cdot (\text{\# of ways } C \text{ decomposes to } A + B).$$

For example, if B is an agent X bound to 10 agents Y and A is a single agent Y, then C is an agent X with 11 agents Y. The number of ways that C decomposes to $A + B$ is 11. The function $g(a, b)$ is a function of the receptors a and b. In NanoDoc this function is constant $g(a, b) = 0.0001$.[1] This calculation accounts for the multiple ways in which a compound can break down into its components.

The precomputation is completed when either no new compounds are formed in the combining step or the total number of compounds exceeds a predefined limit.

As an example, consider a precomputation initialized with two particles X and Y, each with two ligands that match (so X, Y self-assemble to form line-shaped compounds of the form $X_k Y_{k\pm1}$). Then the initial set of compounds is $C_0 = \{X_1 Y_0, X_0 Y_1\}$. in the first combining round, we find one reaction yielding a new compound:

$$X_1 Y_0 + X_0 Y_1 \leftrightarrow X_1 Y_1.$$

The set of compounds becomes $C_1 = \{X_1 Y_0, X_0 Y_1, X_1 Y_1\}$ (note that the structurally-

[1]In a more accurate system, this dissociation propensity would rather be a function of an experimentally established K_D dissociation constant. However, in NanoDoc there is as of yet no way for user-defined dissociation constants, and so this approximation was necessary [7].

distinct compounds XY, YX are both equivalent to X_1Y_1).

In the second combining round, we find three reactions yielding new compounds:

$$X_1Y_0 + X_1Y_1 \leftrightarrow X_2Y_1,$$

$$X_0Y_1 + X_1Y_1 \leftrightarrow X_1Y_2,$$

and

$$X_1Y_1 + X_1Y_1 \leftrightarrow X_2Y_2.$$

The set of compounds becomes $C_2 = \{X_1Y_0, X_0Y_1, X_1Y_1, X_2Y_1, X_1Y_2, X_2Y_2\}$. These rounds continue, and will be terminated when the set of compounds exceeds the predefined limit.

Once the precomputation step is done, the server generates the **rxn** and **cfg** files. Specifically, for each reaction stored from the precomputation step, we write the corresponding assembly and disassembly reactions. Subsequently, we go through each compound. For each compound A we check it for any retained properties (specifically, internalization or bind and release) and write the appropriate reactions to file.

For each compound A we also need to compute its diffusion propensity. Knowing the diffusion requires knowing the size or radius of the compound. Since we do not have any structural information regarding the formation of the compound, we estimate the radius assuming a spherical compound. Specifically, we compute the volume of the compound by summing the volumes of its components and then solve for the radius. Using this estimated radius, the diffusion propensity is calculated via the Stokes-Einstein equation [21].

We also include in the new server options for particle depletion. This is meant to simulate the fact that particles in the tumor tissue eventually diffuse out of the tumor and are eliminated by the body. The rate of depletion of a particle was calculated as $\frac{1}{Ld}$ where d is the diffusion rate and L is the mean length of a discrete random walk in a space of tumor volume V and tumor surface area (envelope) A. $L = \frac{4V}{A}$ is given in [1].

Note that our server has the capability to express assembly, disassembly, emergent properties (diffusion), and retained properties (internalization, bind and release), as desired.

3.5.3 Implementing Self-Assembly in the User Interface

NanoDoc's user interface was updated to allow players to define self-assembly in their treatments. We updated the user controls to allow specifying a number of binders on agents' sensors. Also, to help players better design treatments using nanoparticle self-assembly we implemented an informational window which appears when a player hovers over a sensor or effector. This informational window shows information regarding the molecule's parameters as well as a list of other agents or cell receptors for which the molecule has matching receptors. Figure 3-8 shows an example of this informational window. We intend this to be a valuable feature for players to efficiently design self-assembling nanoparticle treatments.

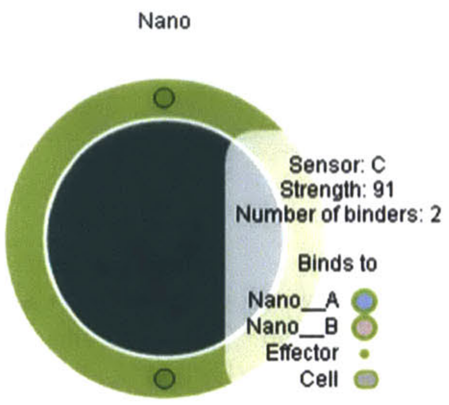
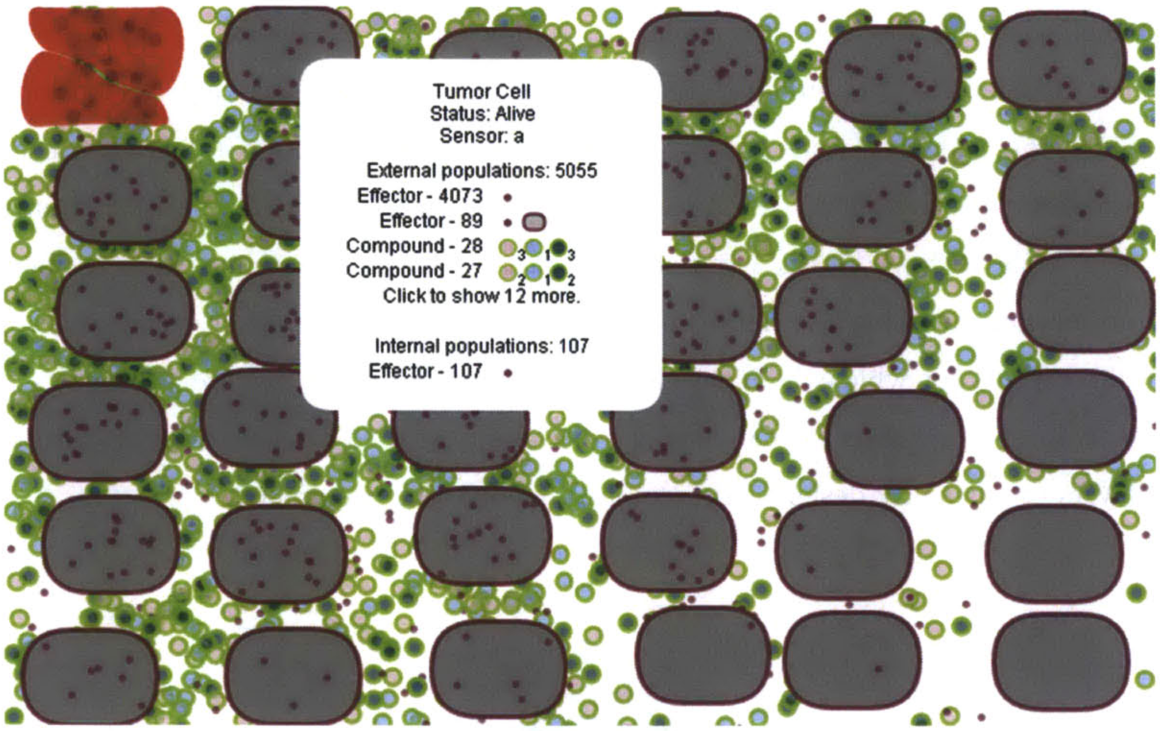


Figure 3-8: The informational window visible to the player when hovering over an agent's sensor. The window contains a list of molecules and cells to which the sensor can bind.

NanoDoc players often use trial and error when designing their treatments. For this reason, it is important that the graphical representation they see of their treatment give useful feedback. Thus, we implemented a similar informational window for when the player hovers over a cellular region in the simulation. The window lists and shows a graphical representation of the molecules and compounds external and internal to the cell. Graphical representations of compounds (multiple particles bound via self-assembly) show their components (agents or molecules) as well as the number of each component. Figure 3-9 shows an example of this informational window.


Tumor Cell
Status: Alive
Sensor: a

External populations: 5055
Effector - 4073 ●
Effector - 89 ●
Compound - 28 ●3 ●1 ●3
Compound - 27 ●2 ●1 ●2
Click to show 12 more.

Internal populations: 107
Effector - 107 ●

Figure 3-9: The informational window visible to the player when hovering over an a cellular region during simulation. The window contains a list of molecules and compounds internal and external to the cell.

Chapter 4

Results

4.1 Recreating Laboratory Experiments

Perrault et al. describe a method for imaging tumors in mice via accumulation of self-assembling fluorescent particles [18]. The key to their method is to inject a large diameter biotin-GNP (gold nanoparticle coated with biotin), that accumulates in the tumor tissue over 24 hours. Subsequently, small diameter fluorescent streptavidin are injected. These particles distribute fast due to their small diameter. Biotin and streptavidin have a high binding affinity, causing the fluorescent particles to stick to the biotin-GNP, yielding high fluorescence in the tumor tissue. Perrault et al. analyze the effects of this in-vivo method of fluorescence imaging versus a pre-assembled method in which GNP-A750, a pre-assembled fluorescent particle, is injected.

We recreated these experiments using our devised precomputation method for simulating self-assembly. Figure 4-1 shows the results of our simulations as well as the results presented by Perrault et al.

Figures 4-1A and 4-1C are results taken from Perrault et al. [18]. Figure 4-1A shows the net (tumor minus body) fluorescence over a 24 hour period. This 24 hour period corresponds to the time after fluorescent streptavidin was injected in the in-vivo experiment, or after GNP-A750 was injected in the pre-assembled experiment. Perrault et al. note that while the two fluorescence signals reach a roughly equal level after 24 hours, the in-vivo fluorescence signals are much quicker and more efficient at

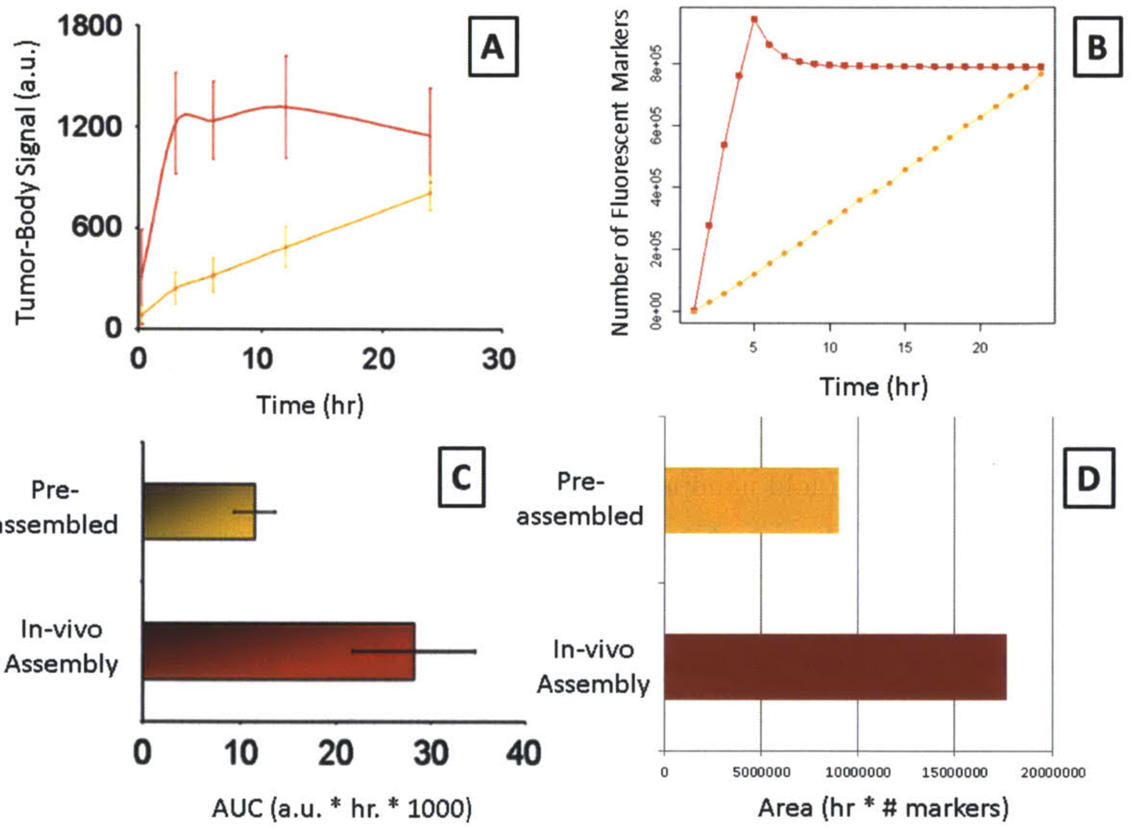


Figure 4-1: Results from Perrault's experiments are shown in A and C [18]. The analogous results from our NanoDoc simulations are shown in B and D. In all graphs, red represents the in-vivo assembly experiments and yellow represents the pre-assembled experiments.

reaching this fluorescence level. After approximately 4 hours, the in-vivo experiment reaches a fluorescence level that the pre-assembled experiment takes 24 hours to reach.

Figure 4-1B shows the corresponding results from our NanoDoc simulations. It shows the number of fluorescent markers (bound to biotin-GNP or unbound and in the extravascular tissue) over a 24 hour period. We note the similarities between our simulation and Perrault's experiments: Namely, that both the in-vivo and the pre-assembled experiments reach an equal level of fluorescence after 24 hours. The pre-assembled fluorescence reaches this fluorescence in a linear fashion, as seen in Perrault's results. The simulations of the in-vivo experiment however reach a spike in fluorescence at 4 hours and subsequently decrease slightly, as also seen in Perrault's results. We can attribute this decrease to depletion of small-diameter fluorescent streptavidin from the tumor tissue.

Figures 4-1C and 4-1D show the 'area under the curve' for Figures 4-1A and 4-1B, respectively. Perrault's results in Figure 4-1C shows a rough 2-3x difference between the in-vivo and pre-assembled experiments. Our simulation's results in Figure 4-1D show a similar 2x difference. Therefore, our simulation of Perrault et al.'s experiments yield results with similar characteristics.

We note that our simulations are much faster than experiments in the laboratory, while achieving similar characteristics. Laboratory experiments can take days or weeks to complete, while these simulations took approximately 10 minutes to run. Therefore, a bioengineer can quickly analyze the results of simulations of experiments, before taking the time to perform the laboratory experiment. Of course, while simulation results will not always be accurate and should not be used as a definitive experiment result, it is still possible to use simulations as a proof-of-concept or a trial-and-error period before verifying results with laboratory experiments.

4.2 Performance Improvements

We analyzed the performance of our new server, first gauging it on simulations that the original server could also handle, and then gauging it on simulations utilizing self-assembly.

4.2.1 Comparisons With Original Server

We compared the performance of the new NanoDoc server with the old version. We devised two scenarios to measure performance. Scenario A is a 10×10 grid of cells of three distinct receptor types. We simulated the injection of 30,000 cell-binding nanoparticles over 48 hours on this scenario. Scenario B is a 10×10 grid of cells of only one receptor type. We simulated the injection of 10,000 nanoparticles of type X and 10,000 nanoparticles of type Y over 48 hours. Type X nanoparticles were designed to bind to type Y nanoparticles, which then induced a release of an effector. These scenarios were designed to encompass all the simulating capabilities of the old server.

Figures 4-2 and 4-3 show the generation times and compilation times of the old server and the new server on the two scenarios. Each simulation was run 10 times on each server. The average of the generation and compilation times is shown in the figures.

Generation time is the time it takes to generate the `rxn` and `cfg` files. In the new server, this includes the precomputation step. Note that generation is very fast for both servers (< 0.01 seconds), showing that precomputation is not a computation-intensive step.

Compilation time measures the time SSC takes to compile the `rxn` file. The new server performs better on both scenarios. We can attribute some of the speed-up in Scenario B to precomputation, since it includes some particle-to-particle interaction. However, Scenario A contains no particle-to-particle interaction, so most of its speed up is most likely due to coalescing.

Rewriting several components of the server inevitably led to several changes to reaction propensities (e.g. slightly different diffusion calculations and association rates). For this reason, we do not include a comparison of execution times. However, as we show in the subsequent section, execution times are usually not time-intensive on the new server.

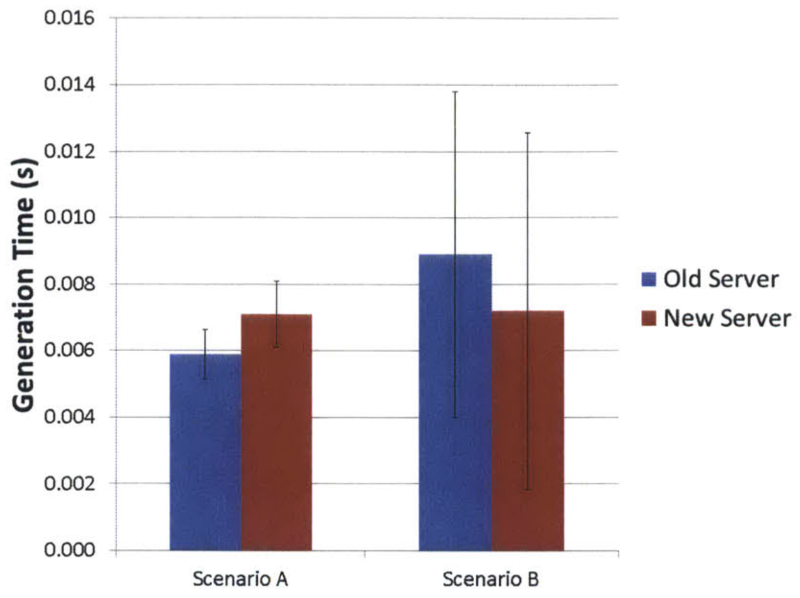


Figure 4-2: Generation times (time to generation `rxn` and `cfg` files) for the old and new servers. Each scenario on each server was generated 10 times. The average of those times is shown in here. Both servers have comparable and very small (< 0.01 seconds) generation times.

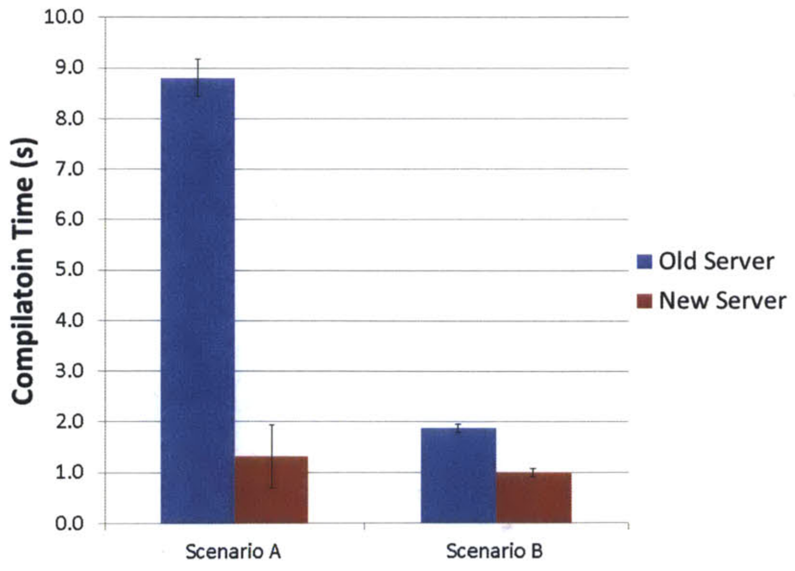


Figure 4-3: Compilation times (time for SSC to compile `rxn` file) for the old and new servers. Each scenario on each server was compiled 10 times. The average of those times is shown in here. The new server performs better in both scenarios.

4.2.2 Simulations With Self-Assembly

We gauged the new server's capabilities to simulate self-assembling particles. We tested simulations on a scenario of 10×10 cell grid. First, we tested simulations of 2,000 line-assembling nanoparticles (two types of nanoparticles with matching receptors, each receptor has two ligands). Then we tested simulations of 2,000 mesh-assembling nanoparticles (two types of nanoparticles with matching receptors, each receptor has 5 ligands).

These simulations were run on the new server with various expansion limits. Each simulation on each expansion limit was run 10 times and the resulting generation, compilation, and execution times were averaged. The results of these tests are shown in Figures 4-4 and 4-5. In both simulations generation time is negligible. This shows that even when self-assembly is a large component of the simulation, precomputation is not a time-intensive step. Also note that execution time is never a considerable part of the overall time. It is usually only a fraction of compilation time.

Compilation time only becomes impractical for NanoDoc players once the expansion limit exceeds 30. Thus, we can expect to simulate up to expansion limit 30, which corresponds to lines with up to 18 nanoparticles in line-assembly and meshes with up to 16 nanoparticles in mesh-assembly. Note that this means that our line-assembling simulation has similar compilation times to those shown in 3-2, and thus the implementation of precomputation in our server matches the performance we expect.

4.3 Self-Assembly Crowd Challenges

To introduce NanoDoc's new self-assembly capabilities to players, we introduced a challenge shown in Figure 4-6.

The challenge takes inspiration from the tumor imaging experiments of Perrault et al. Players are presented with a 10×10 grid of cells with a single vessel in the upper left corner. The cells are distinguished between tumor cells located near the vessel and healthy tissue cells located far from the vessel. Unlike all previous challenges,

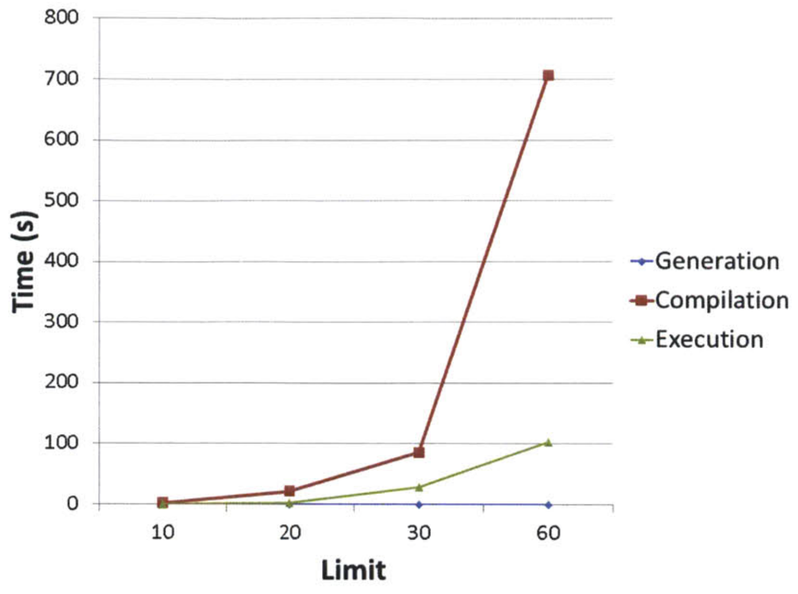


Figure 4-4: Generation (blue diamond), compilation (red square), and execution (green triangle) times for line-assembling nanoparticles. Here, the expansion limit refers to the total number of compounds produced. Thus an expansion limit of 30 corresponds to compounds with up to 18 nanoparticles in a line.

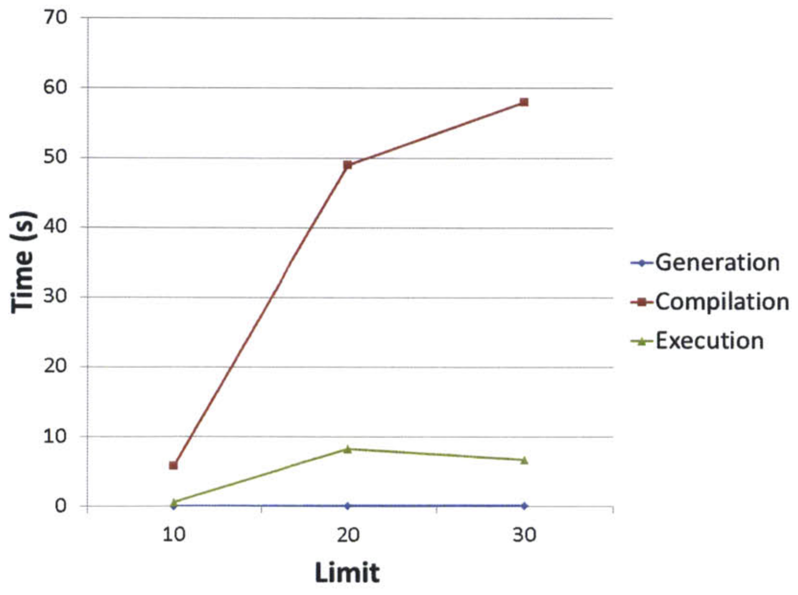


Figure 4-5: Generation (blue diamond), compilation (red square), and execution (green triangle) times for mesh-assembling nanoparticles. Here, the expansion limit refers to the total number of compounds produced. Thus an expansion limit of 30 corresponds to compounds with up to 16 nanoparticles in a mesh.

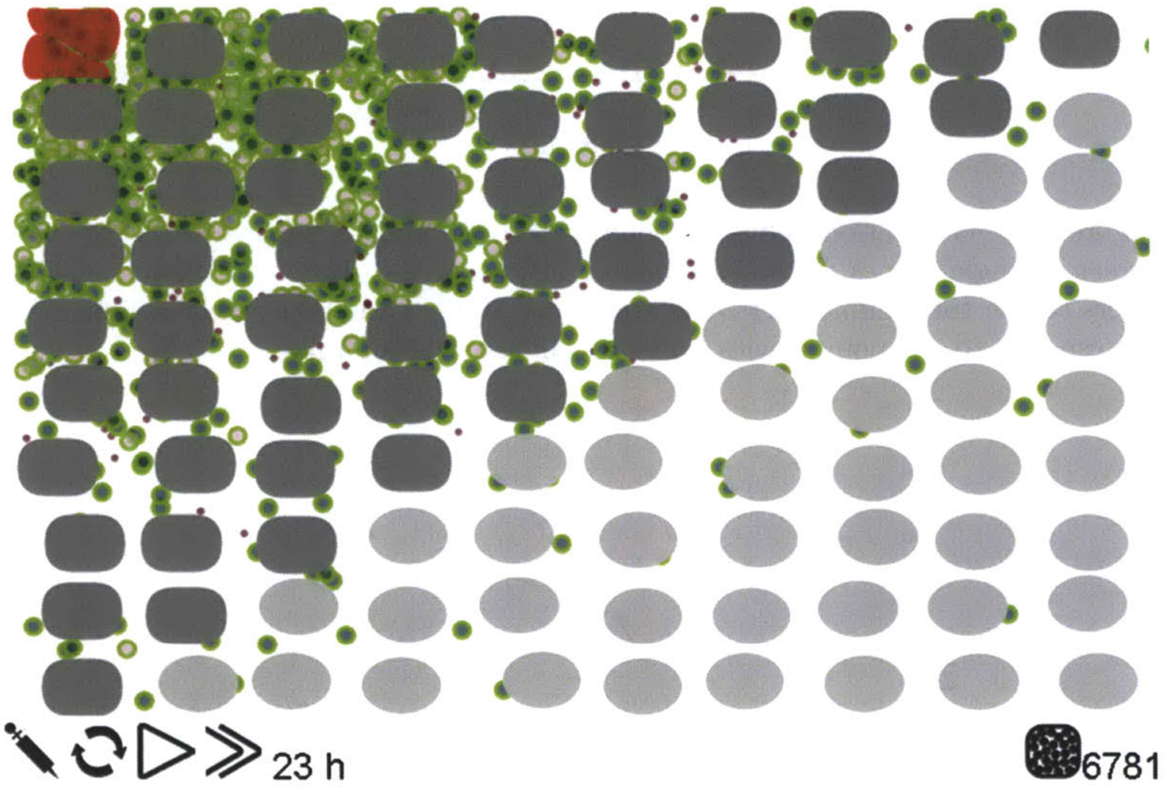


Figure 4-6: The results of a simulated treatment on the new challenge. The number in the lower right corner shows the net accumulation.

which require the player to kill as many tumor cells and save as many healthy cells as possible, we instead tasked the player with achieving a high net accumulation of particles in the tumor tissue. That is, we count the total number of particles in the tumor tissue and subtract the total number of particles in the healthy tissue. This net accumulation is shown in the lower right corner of the simulation. We maintain a leaderboard of best accumulations on the website.

Chapter 5

Discussion and Conclusion

Nanoparticle treatments are becoming increasingly common in drug-delivery or imaging applications for tumors. Nanoparticles' ability to interact with their environment via binding and diffusion as well as interact with each other via self-assembly is the key to their effectiveness. However, predicting behavior and dynamics of nanoparticle systems is difficult. For this reason, simulation mechanisms like NanoDoc, an online game to crowdsource nanomedicine, have been created. For NanoDoc to be used effectively, it is imperative that simulations be fast, comprehensive, and accurate.

Through this project, we significantly expanded NanoDoc's abilities to quickly and accurately simulate self-assembly of nanoparticle treatments in tumor environments using a precomputation method. This method precomputes possible compounds by repeatedly combining nanoparticles with each other. We found that this method resulted in much better performance, both analytically - the number of compounds is reduced from an exponential number to a polynomial number - and empirically - compilation times of example scenarios show corresponding improvements.

We then aimed to show that self-assembly simulated via precomputation can capture experimental results. To this end, we recreated laboratory experiments by Perrault et al. highlighting in-vivo assembly of nanoparticles for tumor imaging applications. The results of the simulations are similar to results of Perrault's laboratory experiments, giving credibility to our simulation methods.

We implemented the precomputation method in NanoDoc's server so that NanoDoc is now able to simulate line-assembling and mesh-assembling particles with compounds of up to 16 and 18 nanoparticles, respectively, in a practical amount of time (less than 2 minutes to generate, compile, and execute each simulation). We made corresponding updates to NanoDoc's user interface, including mechanisms for defining multiple binding sites and visual representations of self-assembly. These updates aim to help players design nanoparticles and to give them feedback on their simulations. Then, to entice players to use self-assembly in their treatments we created a challenge which encourages them to create self-assembling particles that can efficiently accumulate in tumor tissue.

We hope that NanoDoc's new capabilities will act to empower bioengineers to improve and more efficiently design nanoparticle treatments. In the future, we intend to further expand NanoDoc's self-assembly simulating capabilities. For example, self-assembling particles in NanoDoc only exhibit emergent properties of diffusion and retained properties of cell-binding. There are many more possible properties to be simulated (e.g. charge, light absorption) that can play a role. Additionally, triggers for self-assembly should also be implemented in simulations (e.g. heat- or light-triggered self-assembly). We also believe self-assembly simulations can be made faster. It may be possible to pre-compile scenarios so that simulation time does not rely on compile time.

Appendix A

SSC Code Examples

```
region vessel_0_0 move 0 0 0 box width 1 height 1 depth 1
region cell_0_1 move 0 1 0 box width 1 height 1 depth 1
region cell_0_2 move 0 2 0 box width 1 height 1 depth 1
region cell_1_0 move 1 0 0 box width 1 height 1 depth 1
region cell_1_1 move 1 1 0 box width 1 height 1 depth 1
region cell_1_2 move 1 2 0 box width 1 height 1 depth 1
region cell_2_0 move 2 0 0 box width 1 height 1 depth 1
region cell_2_1 move 2 1 0 box width 1 height 1 depth 1
region cell_2_2 move 2 2 0 box width 1 height 1 depth 1
subvolume edge 1

new A at NumA in vessel_0_0
new B at NumB in vessel_0_0

diffusion A at DiffA
diffusion B at DiffB
diffusion A(ligand#1) B(ligand#1) at DiffAB

rxn x:A(ligand#) y:B(ligand#) at AtoBrate -> x.ligand # y.ligand

limit at 3
record all
```

Figure A-1: An example **rxn** file for SSC. The file defines a 3×3 grid of cells, with vessel in the top left corner. Nanoparticles A and B are defined, which can react to form compound AB.

```
NumA = 1000
NumB = 1000

DiffA = 0.02
DiffB = 0.03
DiffAB = 0.04

AtoBrate = 0.1
```

Figure A-2: An example `cfg` file for SSC. The file provides parameter values for the the variables introduced in the `rxn` file.

Appendix B

SSC Performance Graphs

The following figures present the performance results of SSC on the basic simulation described in Section 3.2.2. The figures correspond to performance measures averaged over 10 simulations. Figures contain error bars measuring the standard deviation, although often standard deviation is too small to be noticable.

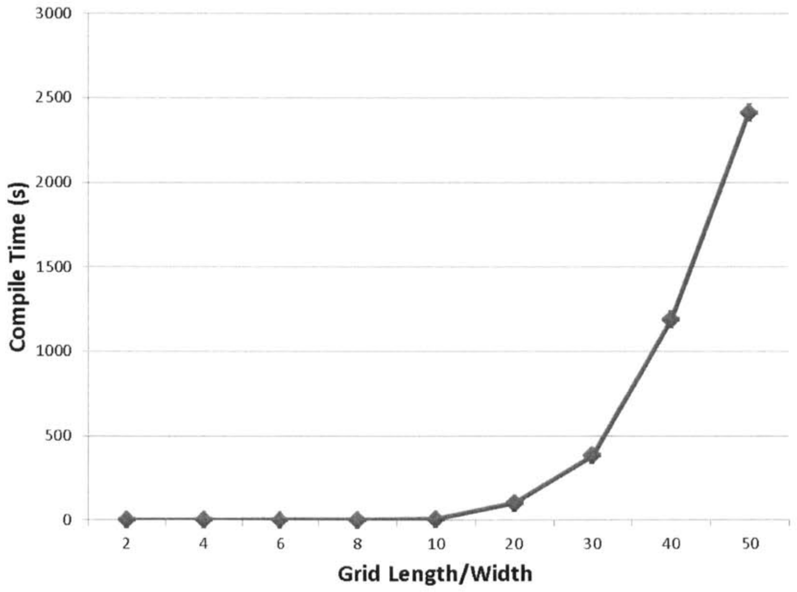


Figure B-1: Compile time of basic scenario against grid size. A 20 on the x-axis corresponds to a 20×20 grid. The relation between compile time and grid length/width is roughly quadratic (linear with the number of regions).

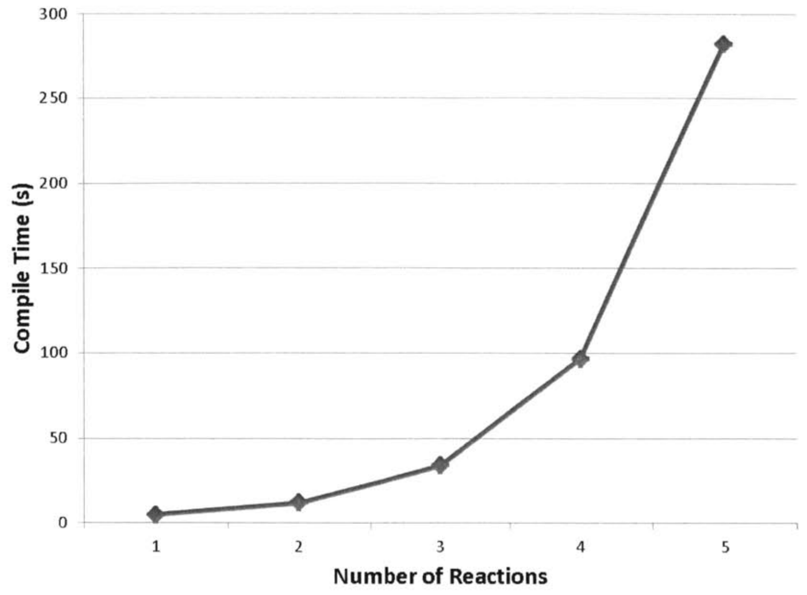


Figure B-2: Compile time of basic scenario against number of reactions (x reactions means that instead of only A receptors, we included $A_1, \ldots, A_x$ receptors on different cells, and allowed each nanoparticle to bind to any receptor). We see a roughly exponential relationship. This is due to the fact that when a nanoparticle can bind to x different receptors, SSC computes every possible binding of those receptors (each receptor is either bound or unbound, so 2^x distinct possibilities).

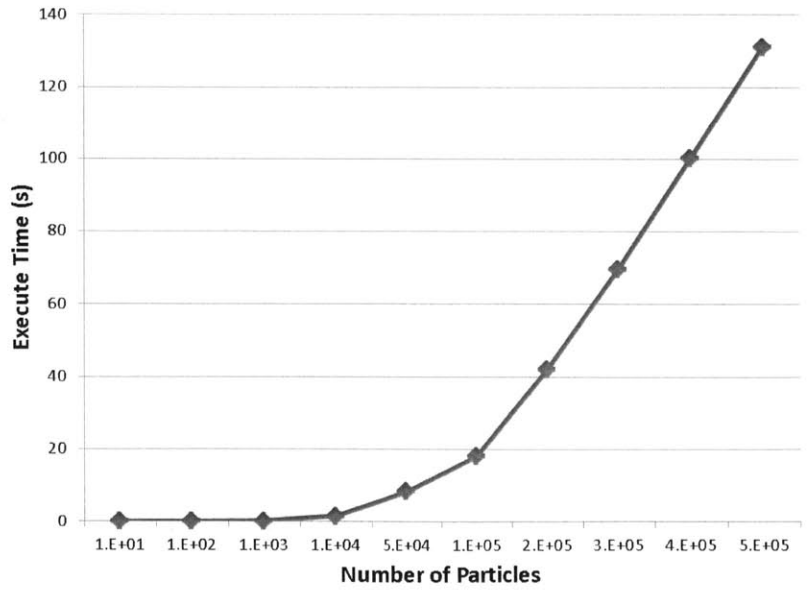


Figure B-3: Execute time of basic scenario against number of particles. We see a roughly linear relationship. This is expected since the more particles, the more rules the simulator must apply during execution.

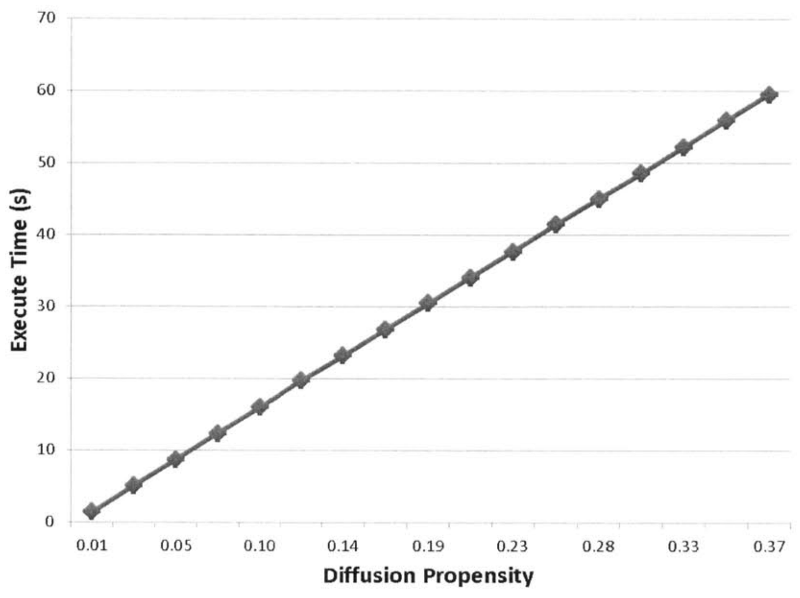


Figure B-4: Execute time of basic scenario against diffusion rate. We see a roughly linear relationship. This is expected, as the diffusion rate increases the total propensity, and thus the number of rules the simulator must execute, in a linear fashion.

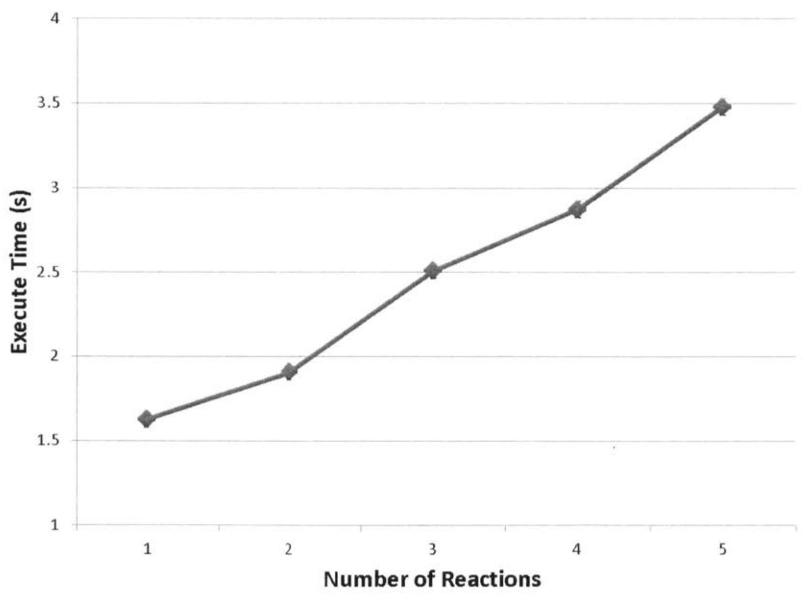


Figure B-5: Execute time of basic scenario against number of reactions. We see a roughly linear relationship. Again, as the number of reactions increases the total propensity, and thus the number of rules the simulator must execute, in a linear fashion.

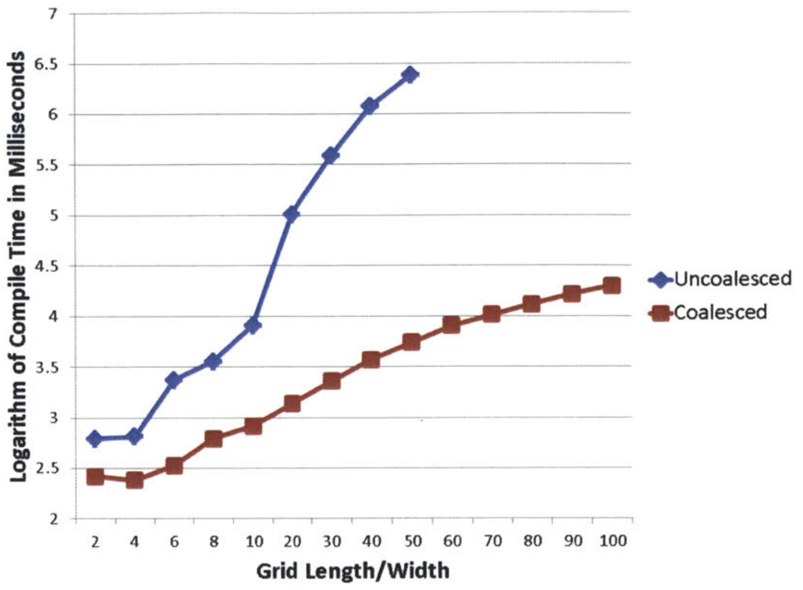


Figure B-6: Compile time of basic scenario against grid size with coalescing and without. Coalescing gives a significant performance improvement.

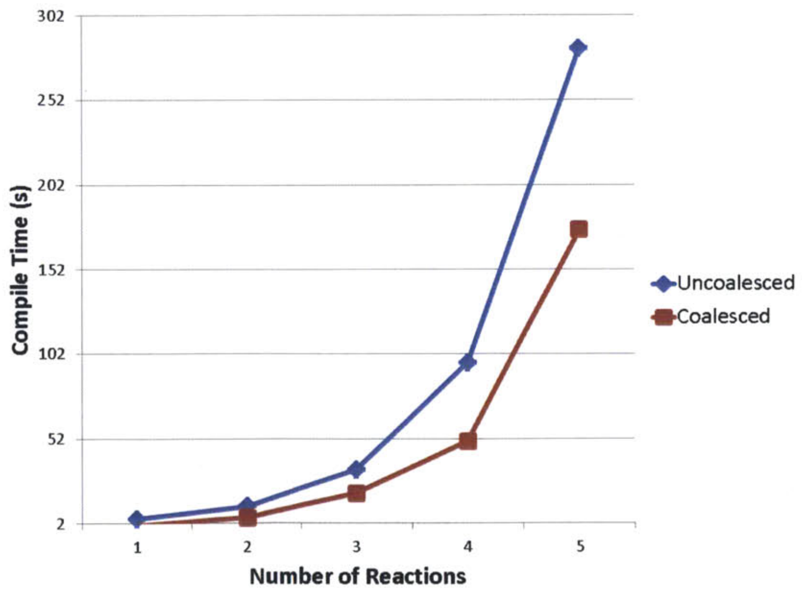


Figure B-7: Compile time of basic scenario against number of reactions with coalescing and without. Coalescing gives a modest performance improvement.

Appendix C

Flow Diagrams

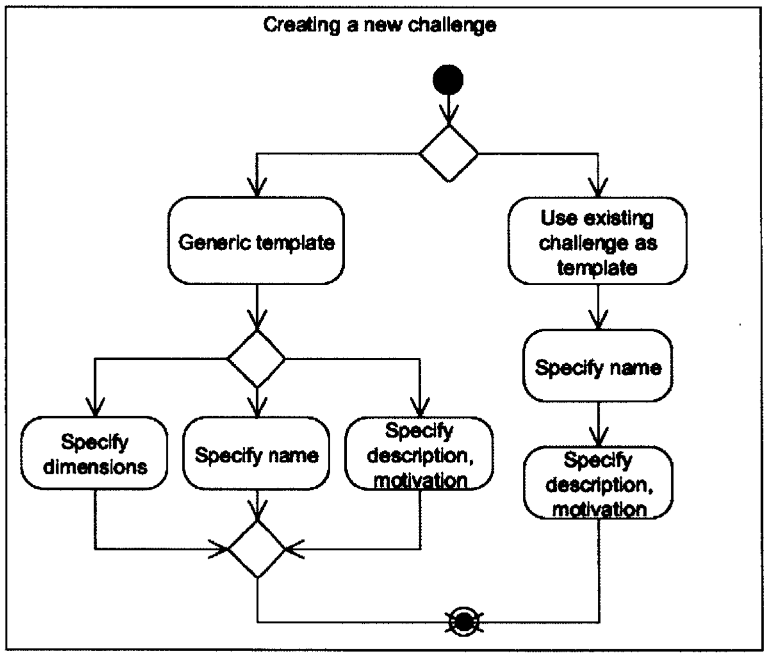

Creating a new challenge
Generic template
Use existing challenge as template
Specify name
Specify dimensions
Specify name
Specify description, motivation
Specify description, motivation

Figure C-1: The work-flow for creating a new challenge.

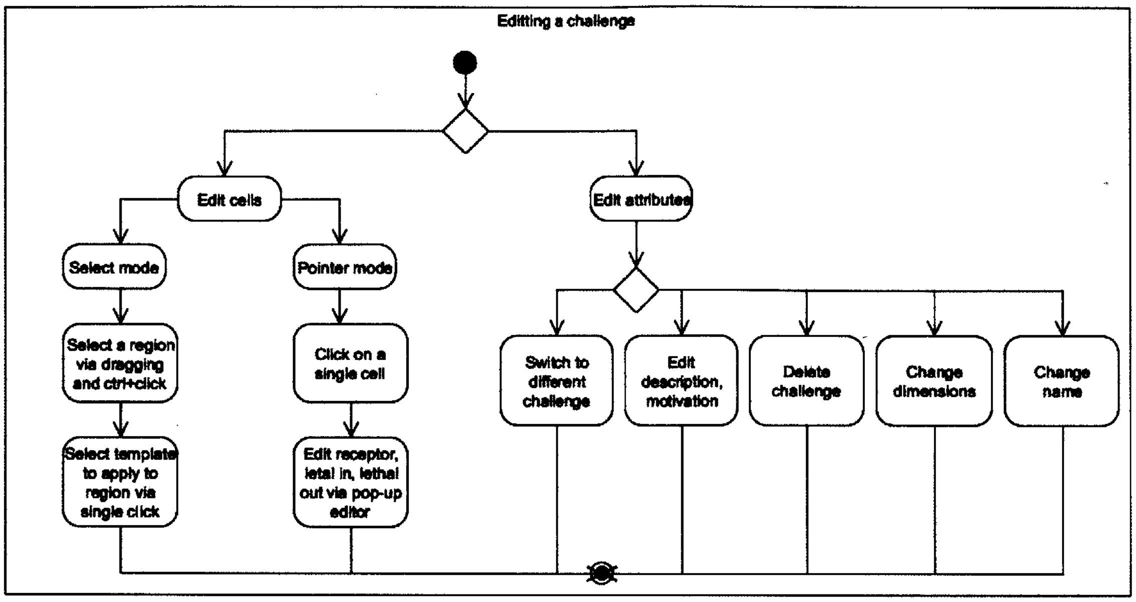

Editting a challenge
Edit cells
Edit attributes
Select mode
Pointer mode
Select a region via dragging and ctrl+click
Click on a single cell
Select template to apply to region via single click
Edit receptor, letal in, lethal out via pop-up editor
Switch to different challenge
Edit description, motivation
Delete challenge
Change dimensions
Change name

Figure C-2: The work-flow for creating editing challenge.

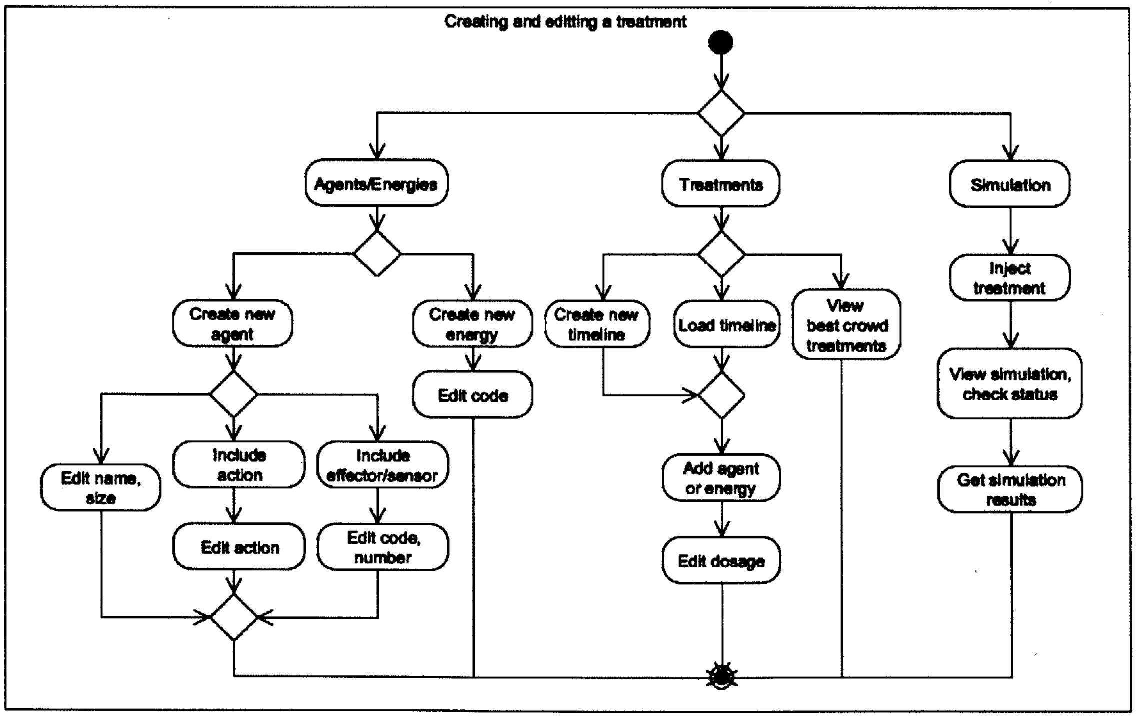


Figure C-3: The work-flow for creating and editing a treatment. Note that the individual nanoparticles are created and then added to a treatment, which is later applied to the tumor challenge.

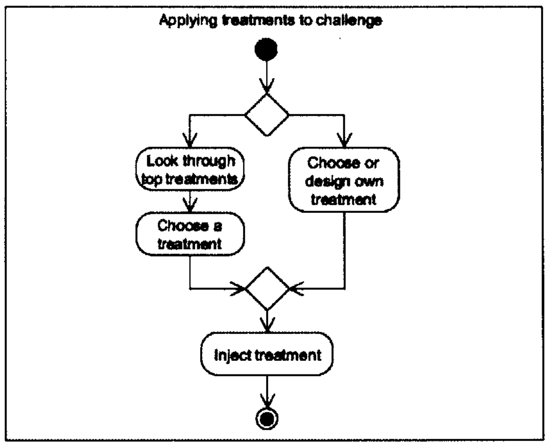

Applying treatments to challenge
Look through top treatments
Choose or design own treatment
Choose a treatment
Inject treatment

Figure C-4: The work-flow for a bioengineer looking through the best treatments produced by the crowd.

Bibliography

[1] S. Blanco and R. Fournier. 'An invariance property of diffusive random walks.' *Europhysics Letters.* Vol. 61. January 15, 2003. Pages 168-173.

[2] Christopher B. Eiben, Justin B. Siegel, Jacob B. Bale, Seth Cooper, Firas Khatib, Betty W. Shen, Foldit Players, Barry L. Stoddard, Zoran Popovic, and David Baker. 'Increased Diels-Alderase activity through backbone remodeling guided by Foldit players.' *Nature Biotechnology.* Vol. 30. October 25, 2011. Pages 190 – 192.

[3] Alexander T. Florence. ' "Targeting" nanoparticles: the constraints of physical laws and physical barriers.' *Journal of Controlled Release.* Vol. 164, Issue 2. December 10, 2012. Pages 115-124.

[4] Alexander T. Florence. 'Reductionism and complexity in nanoparticle-vectored drug targeting.' *Journal of Controlled Release.* Vol. 161, Issue 2. July 20, 2012. Pages 399-402.

[5] André H. Gröschel, Andreas Walther, Tina I. Löbling, Felix H. Schacher, Holger Schmalz, and Axel H. E. Müller. 'Guided hierarchical co-assembly of soft patchy nanoparticles.' *Nature.* November 3, 2013.

[6] Sabine Hauert, Spring Berman, Radhika Nagpal, Sangeeta N. Bhatia. 'A computational framework for identifying design guidelines to increase the penetration of targeted nanoparticles into tumors.. *Nano Today.* Vol. 8, Issue 6. December 27, 2013. Pages 566-576.

[7] Sabine Hauert, Justin H. Lo, Ofir Nachum, Sangeeta Bhatia. 'NanoDoctor: Crowdsourcing the design of swarming nanorobots.'

[8] Berk Hess, Carsten Kutzner, David van der Spoel, Erik Lindahl. 'GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation.' *Journal of Chemical Theory and Computation.* Vol. 4, No. 3. March 12, 2008. Pages 435447.

[9] A. Holmberg, A. Blomstergren, O. Nord, M. Lukacs, J. Lundeberg, and M. Uhlén. 'The biotin streptavidin interaction can be reversibly broken using water at elevated temperatures.' *Electrophoresis.* Vol. 26. No. 3. February 2005. Pages 501-510.

[10] Michael L. Klein, Wataru Shinoda. 'Large-Scale Molecular Dynamics Simulations of Self-Assembling Systems.' *Science.* Vol. 321, Issue 5890. August 8, 2008. Pages 798800.

[11] Mieszko Lis, Maxim N. Artymov, Srinivas Devadas, Arup K. Chakraborty. 'Efficient stochastic simulation of reaction.diffusion processes via direct compilation.' *BIOINFORMATICS.* Vol. 25, No. 17. 2009. Pages 2289 – 2291.

[12] Geoffrey von Maltzahn, Todd J. Harris, Ji-Ho Park, Dal-Hee Min, Alexander J. Schmidt, Michael J. Sailor, and Sangeeta N. Bhatia. 'Nanoparticle Self-Assembly Gated by Logical Proteolytic Triggers.' *Journal of the American Chemical Society.* Vol. 129, No. 19. April 21, 2007. Pages 6064 – 6065.

[13] Francisco J. Martinez-Veracoechea and Daan Frenkel. 'Designing super selectivity in multivalent nano-particle binding.' *Proceedings of the National Academy of Sciences of the United States.* Vol. 108, Issue 27. July 5, 2011. Pages 1096310968.

[14] "MGI-Mouse Facts." *MGI.* The Jackson Laboratory, March 25, 2014. April 5, 2014. `http://www.easybib.com/reference/guide/mla/website`.

[15] Jutaek Nam, Nayoun Won, Ho Jin, Hyokyun Chung, and Sungjee Kim. 'pH-Induced Aggregation of Gold Nanoparticles for Photothermal Cancer Therapy.' *Journal of the American Chemical Society.* Vol. 131, No. 38. September 8, 2009. Pages 13639 – 13645.

[16] Steve O. Nielsen, Carlos F. Lopez, Goundla Srinivas, Michael L. Klein. 'Coarse grain models and the computer simulation of soft materials.' *Journal of Physics: Condensed Matter.* Vol. 16, No. 15. April 2, 2004. Pages 482512.

[17] J. Manuel Perez, F. Joseph Simeone, Yoshinaga Saeki, Lee Josephson, and Ralph Weissleder. 'Viral-Induced Self-Assembly of Magnetic Nanoparticles Allows the Detection of Viral Particles in Biological Media.' *Journal of the American Chemical Society.* Vol. 125, No. 34. August 5, 2003. Pages 10192 – 10193.

[18] Steven D. Perrault and Warren C. W. Chan. 'In vivo assembly of nanoparticle components to improve targeted cancer imaging.' *Proceedings of the National Academy of Sciences of the United States of America.* Vol. 107, No. 25. June 22, 2010. Pages 1194-1199.

[19] Uri Piran and William J. Riordan. 'Dissociation rate constant of the biotin-streptavidin complex.' *Journal of Immunological Methods.* Vol. 133. Issue 1. October 4, 1990. Pages 141-143.

[20] Annette Rösler, Guido W.M Vandermeulen, Harm-Anton Klok. 'Advanced drug delivery devices via self-assembly of amphiphilic block copolymers.' *Advanced Drug Delivery Reviews.* Vol. 53, Issue 1. December 3, 2001. Pages 95 – 108.

[21] William Sutherland. 'A dynamical theory of diffusion for non-electrolytes and the molecular mass of albumin.' *Philosophical Magazine Series 6.* Vol. 9, No. 54. 1905. Pages 781-785.

[22] Greg M. Thurber, K. Dane Wittrup. 'Quantitative Spatiotemporal Analysis of Antibody Fragment Diffusion and Endocytic Consumption in Tumor Spheroids.' *Cancer Research.* Vol. 68. No. 9. 2008. Pages 3334-3341.

[23] George M. Whitesides, Mila Boncheva. 'Beyond molecules: Self-assembly of mesoscopic and macroscopic components.' *Proceedings of the National Academy of Sciences of the United States.* Vol. 99. No. 8. April 16, 2002. Pages 4769-4774.

[24] K. Dane Wittrup, Greg M. Thurber, Michael M. Schmidt, John J. Rhoden. 'Practical Theoretic Guidance for the Design of Tumor-Targeting Agents.' *Methods in Enzymology.* Vol. 503. 2012. Pages 255-268.

[25] Cliff Wong, Triantafyllos Stylianopoulos, Jian Cui, John Martin, Vikash P. Chauhan, Wen Jiang, Zoran Popović, Rakesh K. Jain, Moungi G. Bawendi, and Dai Fukumura. 'Multistage nanoparticle delivery system for deep penetration into tumor tissue.' *Proceedings of the National Academy of Sciences of the United States.* Vol. 108, No. 6. December 7, 2010. Pages 2426 – 2431.

[26] Tomoaki Yoshikawa, Naoki Okadaa, Atsushi Odaa, Keisuke Matsuoa, Kazuhiko Matsuoa, Hiroyuki Kayamuroa, Yumiko Ishiia, Tomoyo Yoshinagaa, Takami Akagib, Mitsuru Akashib, Shinsaku Nakagawa. 'Nanoparticles built by self-assembly of amphiphilic γ-PGA can deliver antigens to antigen-presenting cells with high efficiency: A new tumor-vaccine carrier for eliciting effector T cells.' *Vaccine.* Vol. 26, Issue 10. March 4, 2008. Pages 1303 – 1313.