Accelerating Spatially Explicit Simulations of Spread of Lyme Disease*

Dhananjai M. Rao and Philip A. Wilsey
Dept. of ECECS, PO Box 210030, Cincinnati, OH 45221–0030

Abstract

*The factors influencing spread of Lyme disease are often studied using computer-based simulations and spatially explicit models. However, simulating large and complex models is a time consuming task, even when parallel simulation techniques are employed. In an endeavor to accelerate such simulations, an alternative approach involving dynamic (*i.e., *during simulation) changes to spatial resolution of the model via a novel methodology called Dynamic Component Substitution (DCS) is proposed. Changes to the resolution are performed such that the total number of interactions between the entities in the model is optimized, thereby improving overall performance but introducing minor ($< \pm 1\%$) deviations in the results. This paper explores the effectiveness and issues involved in applying DCS to accelerate sequential and parallel simulations of spatially explicit Lyme disease models. The paper also presents a brief description of the simulation environment along with empirical results. Our experiments indicate that performance improvements can be obtained using the proposed approach.*

1. Introduction

Lyme disease is a bacterial (*Borrelia burgdorferi*), vector-borne disease that is transmitted through a complex chain of interactions between the vectors carrying the bacteria, other organisms in the Eco system, and humans [5, 11]. It is prevalent in many parts of the world, including: northeastern US, central Europe, and central Asia. In the year 2002, more than 23,000 cases were reported in United States alone [8]. Consequently, analysis of various aspects of Lyme disease particularly its spread and transmission is an important and active area of research and study [8, 5, 11].

The study and analysis of the *spread* of Lyme disease, the domain of interest in this study, is a task complicated by the numerous interactions and symbiotic relationships between the vector (deer ticks), the primary hosts (white footed mice), and their Eco system. For example, it has been shown that in oak forests, there is a direct correlation between the number of acorns in the Eco system to the intensity of Lyme disease [11]. The correlation arises because increases in acorns (a primary food for mice) encourages an increase in mice population which results in increased proliferation of the disease causing bacteria [11]. Analyzing and comprehending these secondary factors makes the study of spread of Lyme disease a complex task.

Several experimental and analytical approaches have been employed for the study and analysis of Lyme disease. Amongst the various methods, computer-based simulations have gained significant importance primarily because they provide a powerful and intuitive mechanism to explore scenarios that cannot be analyzed using other approaches. Simulations of Lyme disease are typically performed using spatially explicit models [7, 6]. Spatially explicit models provide a convenient conceptual representation for modeling Eco systems as they have an explicit notion of space or environment [7, 6, 16, 17]. The motivation to treat the environment as a separate dimension is that it undergoes its own sequence of changes. In addition, it eases modeling the influence of the environment on all interactions between the entities that reside in them [3, 16, 17].

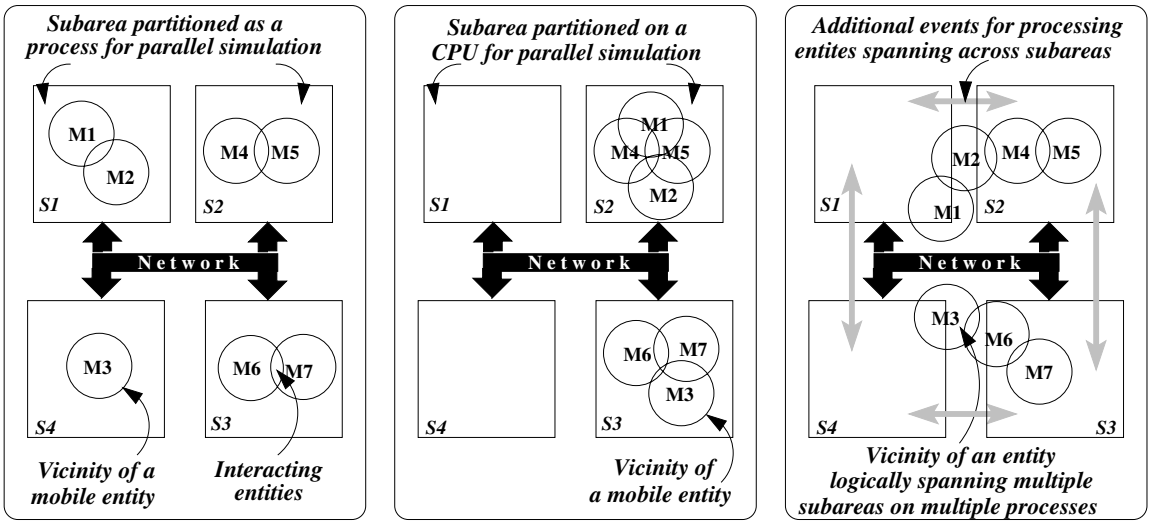
Unfortunately, simulating large, complex models is a time consuming task [7, 13]. In addition, capacity (or resource) limitations of the workstations often prevent simulation of large models [13]. Different approaches have been proposed to address these bottlenecks [7, 6, 13]. Amongst the various approaches, the application of parallel simulation methodologies have shown to provide considerable capacity and performance improvements [7, 6].

In order to achieve efficient parallel simulations, spatially explicit models are subdivided into smaller, non-overlapping regions [7, 6]. Subdivision of space (say into k subspaces) reduces the average time complexity for simulating interactions between each pair of n entities in the simulation from $\mathcal{O}(n^2)$ to $\mathcal{O}(k * [(n/k)^2])$. Furthermore, subdivision is necessary for optimal parallel simulations [7, 6]. Figure 1(a) presents an example in which the space containing 7 entities ($M1 \cdots M7$) has been divided into 4 subareas ($S1 \cdots S4$). The division of space into smaller units is usually performed *statically*, that is either when the model is developed or just before simulation commences. However, the dynamics of the model make static partitioning in-

*Support for this work was provided in part by the Ohio Board of Regents.

Proceedings of the 38th Annual Simulation Symposium (ANSS'05)
1080-241X/05 $20.00 © 2005 **IEEE**

IEEE COMPUTER SOCIETY



(a) Optimal scenario (b) Suboptimal scenario 1 (c) Suboptimal scenario 2

Figure 1. Example of optimal & suboptimal scenarios in static subdivision

effective as the average number of entities in a given subspace may not be optimal throughout the course of simulation. Figure 1(b) illustrates a scenario where the mobile entities crowd into few or just one subspace and the performance deteriorates. Figure 1(c) illustrates another suboptimal or possibly worst case scenario, where the entities are at the edges of subspaces. In this scenario the vicinity of the entities span across multiple subspaces. Consequently, additional events need to be exchanged by the logical processes modeling the subspaces in order to correctly capture all interactions between various entities. Note that in this scenario, these extra events would not be necessary if the space was modeled as a whole, but a single subspace would be suboptimal in other cases. These issues present a challenging dilemma and are a significant bottleneck preventing efficient simulation of spatially explicit models [6, 13].

In an endeavor to address the complications with static division of the simulated space, this study proposes to dynamically change the subdivisions during the simulation runtime. The spatial resolution of the model is continuously adapted to the changing conditions in the simulated system. More specifically, the resolution of the model is changed depending on the relative positions of the mobile, interacting entities (mice) in the model to improve performance with acceptable tradeoffs in fidelity. The dynamic changes to the resolution of the model are achieved by applying a novel methodology called Dynamic Component Substitution (DCS) [13].

This paper presents the issues involved in applying DCS to accelerate the simulation of the spread of Lyme disease using multi-resolution, spatially explicit models. A brief biological background on the spread of Lyme disease pertinent to this study is presented in Section 2. Section 3 includes a brief review of some of the related research activities along with a brief background on spatially explicit models. A summary on Dynamic Component Substitution (DCS) is presented in Section 4. An overview of WESE, the modeling and simulation environment used in this study is presented in Section 5 along with a brief description of WESE's infrastructure for DCS. The models developed as a part of this study along with the strategy for using DCS is presented in Section 6. Section 7 presents the statistics and inferences collated from the various sequential as well as parallel simulations conducted to evaluate the effectiveness of the proposed approach. Finally, Section 8 concludes the paper and presents pointers to future work.

2. Biological Background

The Lyme disease causing spirochete (*Borrelia burgdorferi*) is transmitted to humans by tick (*Ixodes scapularis*) bites. Although the ticks are the major vectors for carrying the spirochete they are relatively immobile and the spread of the disease is primarily driven through a complex chain of interactions with their mobile hosts such as deer and mice. The dominant hosts for ticks in many habitats are white footed mice (*Peromyscus leucopus*) [11]. Accordingly, in consensus with earlier biological and simulation studies on spread of Lyme disease, the model used in this research also employs mice as the main, mobile host [5, 11]. Furthermore, in concordance with earlier research, interactions with other hosts such as humans is ignored because such interactions do not have a significant bearing on the spread of

IEEE COMPUTER SOCIETY

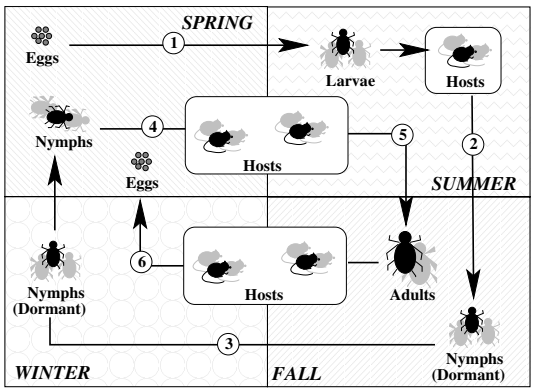


Figure 2. Chain of interactions causing spread of Lyme disease

the disease [5, 11]. Accordingly, a brief description of the interactions between ticks and mice is presented in the following paragraph; more detailed discussions are available elsewhere [5, 11, 13].

Figure 2 presents an overview of the typical sequence of interactions between ticks and mice that has been used to develop the simulation model. The interactions occur over a two year period starting with summer. In summer, the tick eggs hatch and become larvae. If a larvae comes in close contact with a mouse it bites the mouse to feed on its blood. Larvae typically feed for a few days and may be carried to a different area by the mouse. If the mouse was infected earlier with the disease causing spirochete, the bacteria is transmitted from the mouse to the larva. Larvae that successfully feed on a host drop off and molt into a nymph. Nymphs that survive the winter actively seek a blood meal in spring. If an infected nymph bites a mouse, the bacteria is transmitted from the nymph to the mouse. Nymphs that find sufficient food molt into adult ticks. Female ticks lay eggs and the cycle continues.

3. Related Research

A large number of processes in nature are strongly influenced by the environment or "space" in which they occur. Space is usually used to denote processes or factors of interest that encapsulate and affect the system under study. The actual process or factors referred to as space depends on the domain and context of a study. For example, in demographic analysis the plant-animal-human ecology is typically considered as space [1]. On the other hand, Deelman *et al* present an Eco system model to study the spread of Lyme disease in which they consider ticks in the environment to be the background or space [6]. In order to analyze such systems using computer-based simulations, models that have an explicit notion of space in their conceptual representations have been developed [1, 6]. Such models are classified as *spatially explicit models*. Spatially explicit models are widely used in a number of domains such as: ecology, economics, and physics [3, 6, 16, 17].

One of the complex tasks in simulation of spatially explicit models is capturing the interaction between the entities involved in the simulation. In order to ease model development, analysis, and optimize simulation performance, the space is divided into subspaces. The choice of the subspace dimensions plays a crucial role in the overall accuracy and efficiency of the simulation [2, 6, 7, 9, 17]. However, in related studies [2, 6, 7, 9, 17], the space is broken into smaller blocks during model development or before simulation commences, *i.e., statically*. In this study, we proposes to *dynamically* (at simulation runtime) abstract and refine subspaces to further improve performance. Dynamic changes to the spatial resolution conspicuously distinguishes this research from the earlier efforts.

Since the proposed methodology utilizes models at different levels of resolution, this study also falls under the broad umbrella of multi-resolution modeling and simulation. A number of studies have been reported on selectively abstracting parts of a model to enable efficient tradeoffs between several model and simulation related parameters (*e.g.,* model resolution, fidelity, and performance). Most of these earlier studies deal with static abstractions or refinement where the abstraction or refinement is performed during model development or prior to simulation commencement. On the other hand, dynamically changing the resolution of the model enables more optimal tradeoffs between several interrelated parameters such as: model observability, fidelity, resource requirements, and performance [13].

The previous multi-resolution simulation studies closest to this research activity was performed by Natrajan *et al* [10]. They propose the use of Multiple Resolution Entities (MRE) to process events at various levels of abstraction or resolution of a model [10]. MREs are entities that are capable of interacting with the remainder of the model at different levels of resolution by maintaining internal consistency across multiple, concurrent levels of resolution. In contrast, this study utilizes a novel methodology called Dynamic Component Substitution (DCS) for changing the resolution of the model during simulation. Unlike MREs that may maintain state at all possible resolutions at all times, DCS operates using a single state at a given resolution for each component (or simulation object) at a given instant of time. In DCS, states at different levels of resolution are created on-demand whenever resolution changes [13].

4. Dynamic Component Substitution (DCS)

Dynamic Component Substitution (DCS) is used to change the resolution or level of abstraction of a hierarchical, component-based model [13]. DCS is performed by

IEEE COMPUTER SOCIETY

substituting a set of components called a *module* with an *equivalent* component or vice versa. The equivalent component of a module must satisfy the following criteria: (i) it must have an interface that is identical to that of the module; and (ii) its externally visible functionality must be the same (or within some acceptable delta) as that of the original module. Substituting a module with its equivalent component or vice versa is synonymous to abstracting or refining a model respectively. Note that the DCS transformations induce changes to the model which in-turn impacts the overall simulation.

The strategies that may used by a model developer for triggering DCS are broadly categorized into *proactive* and *reactive* strategies. Approaches in which DCS transformations are scheduled to occur in the future (with respect to simulation time) are classified as proactive strategies. Proactive strategies are used when the scenarios for triggering DCS are known apriori either during model development or can be identified during simulation. In reactive strategies, DCS transformations are scheduled at the current simulation time or in the past. Proactive strategies are easier to implement but identifying scenarios for proactive triggering can be complicated. In contrast, reactive DCS is much easier to trigger, but it is more complex to support and it may incur additional overheads during simulation [13]. Note that a combination of proactive and reactive strategies may be employed in a single model. In this study, only the proactive DCS transformation strategy has been utilized.

The primary objective of applying DCS is to enable a more optimal tradeoff between several interrelated modeling and simulation related parameters. DCS transformations may be performed statically or dynamically. Static DCS transformations occur just before simulation commences while dynamic DCS transformations occur during the course of simulation. The latter subset of transformations provide a mechanism to continuously adapt the model to the dynamic scenarios in the simulation. Typically, dynamic DCS is used to abstracts parts of a model that are inconsequential to a given study in order to improve simulation performance [13]. A more detailed description of the strategy for applying DCS is discussed in Section 6

5. WESE

This section presents a brief overview of a Web-based Environment for Systems Engineering (WESE) and its infrastructure for DCS [14, 15]. WESE provides a component based modeling language, a framework for developing a web-based repository of components, and the infrastructure for distributed simulation. WESE is an asynchronous, distributed environment in which the WESE Server plays a central role in orchestrating the various activities. All interactions with the WESE Server are performed through suitable modeling and simulation interfaces. Model development in WESE involves two phases. First, a set of components involved in the model are developed using WESE's API and deployed as WESE factory. A WESE factory is a repository of components with added capability for sequential and parallel simulation. In the second phase, the actual models are developed by suitably interconnecting the components using a hierarchical modeling language called the System Specification Language (SSL).

The specification of a model or a SSL design file consists of a set of interconnected *modules*. Each module consists of three main sections, namely: (i) the *component definition section* that contains the details of the components to be used to specify a module (such as the Universal Resource Locator (URL) of a factory and name of the source object along with initial parameters); (ii) the *component instantiation section* that defines the various components constituting the module; and (iii) the *netlist section* that defines the interconnectivity between the various instantiated components. SSL permits an equivalent component to be associated with each module. DCS is performed by replacing the module with its equivalent component or vice versa.

The WESE server performs the task of collaborating with the distributed factories and coordinating the simulations. The server has several coordinated, loosely coupled subsystems that perform the various tasks in setting up a distributed simulation. Once the simulation commences, the simulation subsystems of each WESE factory involved in the simulation handle further processing. The *simulation subsystem* of a WESE factory has been developed using the WARPED simulation kernel. WARPED is an API and implementation for a general purpose discrete event simulation [12]. WESE utilizes the Time Warp [12] based simulation kernel of WARPED. A WARPED simulation is organized as a set of interacting logical processes (LP). Accordingly, in WESE each component is mapped to a WARPED LP. Furthermore, the API provided by WARPED is used for exchanging events, optimistic synchronization, and garbage collection.

In conjunction with SSL, the simulation subsystem of WESE provides the infrastructure for triggering and automatically carrying out DCS transformations to a model. The WESE server contains a *DCS Analyzer* module that is used to extract model information for DCS (such as related sets of components at various levels of abstraction) through static analysis of the SSL description. In WESE, an event-driven mechanism has been employed to sequence the various phases involved in DCS. This approach makes the DCS implementation immune to the idiosyncrasy of the synchronization mechanism. A component can trigger DCS by merely scheduling an appropriate kernel event. WESE also provides an API for mapping states of components during DCS. A more detailed description of DCS and WESE is available elsewhere [12, 14, 15].

IEEE COMPUTER SOCIETY

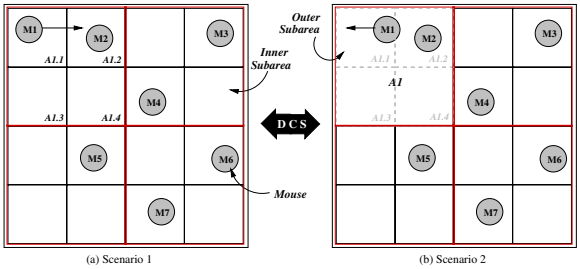


Figure 3. Overview of strategy and DCS transformations

6. Model & Strategy for DCS

As described in Section 2, the primary challenge in this model is to track any possible interactions between mice and ticks that may occur in a large area of a given Eco system. As explained earlier, large areas are subdivided into smaller subsections to minimize the overheads. In order to dynamically change the composition of the subareas, the space is divided into a hierarchy of subareas and DCS is used to aggregate and disaggregate them. Figure 3 presents an example of this approach. As shown in Figure 3, the spatially explicit Lyme disease model is partitioned into a hierarchical set of subspaces. The innermost subareas, namely $A1.1$, $A1.2$, $A1.3$, and $A1.4$ constitute the next hierarchical level namely $A1$. DCS is then used to dynamically abstract and refine the subareas depending on the position and vicinity of the mice in the model. In Figure 3(a), the vicinity of all the mice are completely contained in a given subarea. Consequently, retaining the space at its highest level of disaggregation provides maximum performance.

However, as the mouse, $m1$ moves from subarea $A1.1$ toward $A1.2$, its vicinity spans 2 (or more) subareas and all the mice in the subareas have to be considered for interaction processing. Accordingly, as shown in Figure 3(b), the subareas ($A1.1$ through $A1.4$) are abstracted into a larger area ($A1$). Furthermore, the abstract area (namely $A1$) does not account for movement of ticks, unlike the smaller subareas (namely $A1.1 \cdots A1.4$) to minimize overheads. This discrepancy in behavior does introduce some deviation (about $\pm 1\%$ in our studies) which is assumed to be acceptable for this study. deviation in the results with and without DCS. However, in this study it has been assumed that this deviation is acceptable. Conversely, when the mice are in non-overlapping areas, the larger subarea $A1$ is refined into smaller subareas to improve performance. During DCS transformations, WESE's state mapping API is used to pass information about the position and life cycle status of ticks in the area (in the form of kernel events).

The spatially explicit model of spread of Lyme disease has been developed based on descriptions and parameters obtained from previous studies performed by Deelman *et al* [5, 6] and Ostfeld *et al* [11]. The model reflects the the interaction cycle described in Section 2. As explained in Section 5, the first (`Mouse`) and `EcoArea` components were developed to capture the various interactions shown in Figure 2. As described by Deelman *et al* [5], the mice in the simulation move around with a range of 400m^2 per day in a random fashion scouting for food. If a sexually active male and female mice come in vicinity of each other, the female mice gets pregnant. A pregnant mouse scouts around for a nesting site and once a nest has been built, the mouse moves back and forth to the nest looking for food. Mice give birth to a litter (of 1-5 mice) after about 17-23 days. The babies mature in about 10-14 days and move away from the nesting site. The mother moves away from the nest once all the babies have left the nest. A mouse has a random life time of 30 to 365 days at the end of which it dies. Birth and death of mice is logical and a pool of inactive `Mouse` components is maintained in a `MouseStore` component for this purpose. The inactive `Mouse` components do not actively participate in the simulation.

The `EcoArea` components are used to model the environment under study. The `EcoArea` components also encapsulate a number of ticks that are at various stages in their life-cycle. Since the number of ticks is large (about 1600/400m^2), the ticks are not modeled as separate entities and the `EcoArea` components are responsible for modeling the life cycle and movements of ticks. A similar approach has been previously employed by Deelman *et al* as well [4, 6] to improve performance. The adult ticks in the area move about a foot every 2 to 3 days [11]. However, it must be noted that the abstract subareas do not perform tick movements in order to minimize overheads. The `EcoArea` components report statistics on the ticks at the end of simulation. The statistics include the number of ticks and percentage of infected ticks in each sub area. These statistics are used to verify that consistent results are obtained with and without DCS. In concordance with WESE's API the models were bundled into a `EcoFactory` for deployment and further processing.

The interactions between mice and ticks are performed through the `EcoArea`. Every mouse reports its position to an appropriate `EcoArea` component. Each `EcoArea` maintains the list of mice in its vicinity and provides feedback on proximity of mice to each other. In addition, they check and trigger interactions with ticks in the vicinity of a mouse. When DCS has been enabled, each `EcoArea` utilizes the position of the mice to decide when to aggregate and disaggregate themselves. A proactive DCS triggering strategy has been employed as the scenarios in which transformations need to occur can be detected apriori. The proactive DCS transformations are triggered via API

IEEE COMPUTER SOCIETY

calls provided by WESE. Since DCS is performed using WESE's API, the changes to the subspace is transparent to the `Mouse` components in the simulation. The `EcoAreas` also generate and update their state with position of mice whenever DCS is triggered using the state updation API of WESE. The models for simulation have been developed by suitably interconnecting the aforementioned `EcoArea` and `Mouse` components.

7. Experiments

The experiments were conduced by developing different configurations of Lyme Disease models. Some of the salient characteristics of the models used in the experiments are shown in Table 1. The `#Subareas` column indicates the number of subareas into which the total area was subdivided. Note that the space was divided into a square matrix and the number in parentheses (in the `#Subareas`) indicates the number of rows and columns into which the areas was subdivided. The size of each subarea was fixed to be 200 square meters . For example, the total area in `Eco1` model was 0.6 square kilometers which was subdivided into 4x4 grid consisting of 16 subareas. In all the models, 4 adjacent subareas were abstracted into a single, more abstract subarea. The top level of the hierarchy consisted of a single large area. As indicated in Table 1, the different models contained different areas and densities of mice. `Eco3` model had a higher density of mice than `Eco2`. However, all of the models contained 3 hierarchies of subspaces. Each subarea contained 1600 tick eggs out of which 15% were seeded to be infected with the bacteria. All the simulations start at late spring and the Eco system was simulated for 2 years in terms of simulation time.

The simulations were performed using a network of workstations. Each workstation had two Athlon MP (2.0 GHz comparable) processors with 1 Gigabyte of RAM running Linux. The workstations were networked using gigabit Ethernet. The models were simulated using the sequential simulation mode of WESE as well as the parallel simulation mode. For parallel simulation, the `EcoFactory` was deployed on multiple machines and components from various factories were used in the simulation. The default, random partitioning provided by WESE was used for parallel simulations. Verification of the results obtained from the simulations was performed by comparing the number of infected ticks at the end of simulation and ensuring the results were sufficiently similar. Note that deviations arise because the abstract `EcoAreas` do not perform tick movements to minimize overheads. In our experiments the deviations were less than $\pm 1\%$.

The statistics collated from the sequential and parallel simulations performed using the different models are tabulated in Table 2 and Table 3 respectively. The data shown in both the tables are average values computed from 3 simulation runs. The statistics in `No DCS` column (in Table 2 and Table 3) did not involve any DCS and are base reference numbers for performance comparisons. The reference performance statistics were collated from optimal *static* configurations of the model to ensure fair comparison and to highlight the effectiveness of DCS. These optimal, static (without DCS) configurations were determined empirically — simulations with different subarea sizes were conducted and the configuration that ran the fastest has been used as the reference configuration. Note that the simulation time varies from one static configuration to another because of overheads and tradeoffs between the following interrelated factors: (i) tick life cycle processing; (ii) state saving overheads; and (iii) mouse-tick interaction processing [5].

A trend of the DCS transformations that occur in the `Eco2` Lyme disease model is shown in Figure 4. The trend is typical and reflective of the characteristics of other models as well. As described earlier, the subareas are abstracted whenever the vicinity of a mouse spills across subarea boundaries. On the other hand, refinement is triggered when all mice in a given vicinity are in independent subareas. Regions in which the number of active components is higher indicates phases of higher performance. Initially, the model starts off at an abstract level and consequently there is a ramp up to the optimal operating region. Furthermore, it must be noted that the DCS activity never stabilizes to a quiescent point because of the constant movement of mice in the simulation. As a result, there are additional DCS related events (recollect that WESE uses an event-based mechanism to sequence various DCS related operations) processed in the case of simulations with DCS.

As indicated by the performance gains (`Speedup` column) in Table 2 and Table 3, DCS approach consistently performs better than the non-DCS case in sequential and parallel simulations respectively. As explained earlier, the performance gains are measured against the most optimal static configuration. As shown by the statistics, the proposed strategy provides good performance improvements even in the `Eco3` model, even though the model undergoes a large number of DCS transformations due to the high mice density. This observation indicates that the overhead for

Model ID	Simulation Time (in seconds)		Speedup
	No DCS	DCS	
Eco1	3672.5	3123.5	14.89%
Eco2	17756.1	12668.2	28.65%
Eco3	23670.3	16872.5	28.71%

Table 2. Statistics from sequential simulations

IEEE COMPUTER SOCIETY

Model ID	Eco Area		Number of Mice			Total
	Area (KM^2)	#Subareas (Row x Col)	Male Mice	Female Mice	Inactive Mice	No. Of Comps.
Eco1	0.6	16 (4x4)	10	10	20	58
Eco2	2.56	64 (8x8)	25	25	50	127
Eco3	2.56	64 (8x8)	50	50	100	266

Table 1. Model characteristics

Model ID	# Of CPUs	Simulation Time (in seconds)		Network Msgs. (in millions)		Rollbacks (in millions)		Speedup
		No DCS	DCS	No DCS	DCS	No DCS	DCS	
Eco1	2	5872.5	5177.6	25.44	19.21	4.5	3.97	11.84
	4	6872.3	6342.7	29.43	22.71	6.1	4.93	7.71%
Eco2	2	40942.2	38047.3	91.45	72.9	38.66	17.97	7%
	4	42944.5	39153.4	99.4	78.1	44.71	23.42	8.82%
Eco3	2	19827.4	16733.4	99.1	78.4	13.4	8.24	15.6%
	4	18227.4	15987.8	102.1	82.3	11.2	6.37	12.28%

Table 3. Statistics from parallel simulations

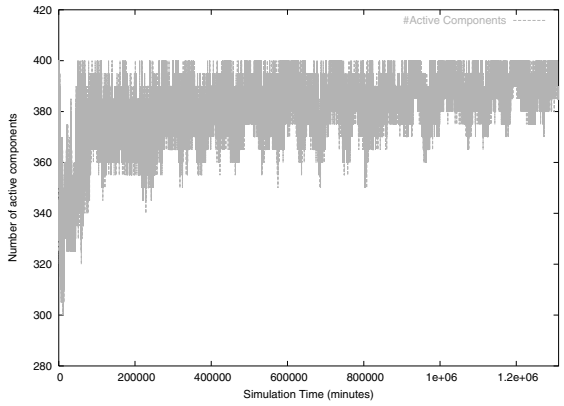


Figure 4. DCS trends in Eco simulation

DCS transformations is compensated by the performance gains accrued by dynamically minimizing the model overheads. The DCS-based approach provided greater performance improvements in the sequential case primarily because of the reduction in the computational time complexity as described earlier.

On the other hand, the performance gains in parallel simulations were modest because of the following three factors: (i) when multiple CPUs were used, the computations were distributed across the CPUs thereby mitigating the gains of reducing the time complexity; (ii) a large number of events were dispatched over the network (indicated by the `Network Msgs.` column) increasing communication overheads; and (iii) a large number of rollbacks that occur due to the nature of the model. Note that the simulations were throttled to control the number of rollbacks that occur. In fact the `Eco2` and `Eco3` models do not simulate on more than two CPUs without throttling. However, the inherent characteristics of the model which require events to be scheduled well into the future (with respect to current simulation time) prevented aggressive throttling. As shown in Table 3, the smaller models, namely `Eco1` and `Eco2` did not benefit from parallel simulation because the overheads of parallel simulation outweighed the benefits. Conversely, the larger model `Eco3` had sufficient parallelism and work load to benefit from parallel simulations. In this context, it must be noted that the objective of this study is not to compare the effectiveness of sequential and parallel simulations but compare DCS versus non-DCS approaches. Furthermore, our experiments suggest that as the size of the model increases, the benefits accrued by employing parallel simulations and DCS increases. As indicated by the experimental results, the proposed DCS-based approach accelerates the simulations by 7% to 28%.

8. Conclusion

This paper presented an alternative methodology for accelerating simulations of spatially explicit Lyme disease models by dynamically changing the resolution of the model at simulation runtime. The proposed approach improves performance by minimizing computational overheads and trading off some fidelity (less than $\pm 1\%$). Changes to the resolution of the models were performed

IEEE COMPUTER SOCIETY

by applying Dynamic Component Substitution (DCS). The DCS transformations proactively changed the spatial resolution to minimize the overheads of simulating interactions between the entities in the model without significant loss in fidelity, thereby improving performance. The paper described the hierarchical, multi-resolution models developed as a part of this study. An overview of the modeling and simulation environment used to develop and simulate the models was also presented. The paper presented the statistics obtained from sequential and parallel simulations to highlight the effectiveness of the proposed approach.

The results collated from the experiments were presented as empirical evidence to highlight the gains accrued by applying the proposed approach. The gains accrued in sequential simulations were more prominent than the gains from the parallel simulations. Examination of the results indicate that the overheads of parallel simulations mitigate the benefits. The primary source of overheads were large volumes of events that need to be exchanged over the network between the workstations participating in the parallel simulations. A dynamic process migration strategy may be used to address this bottleneck and is a good candidate for future investigations.

The results obtained from this study indicate that the proposed strategy holds promise for accelerating the simulations of other spatially explicit models. More specifically, dynamic changes a model's resolution can be used to accelerate simulations of spatially explicit models which the time complexity of computing the interactions between n given entities is $\mathcal{O}(n^2)$ or greater. For example the proposed strategy maybe suitably adapted and applied to simulation of spatially explicit models of wireless ad-hoc networks. Furthermore, the general purpose nature of this approach opens up a number exciting opportunities to improve the overall efficiency of simulations in several application domains.

References

[1] N. P. R. Anten. Evolutionarily stable leaf area production in plant populations. *Journal of Theoretical Biology*, 127(1):15–32, Jan. 2002.

[2] R. L. Bagrodia and W.-T. Liao. Parallel simulation of the sharks world problem. In *Proceedings of the 22nd Winter Simulation Conference*, pages 191–198, Dec. 1990.

[3] K. P. Bell and E. G. Irwin. Spatially explicit micro-level modelling of land use change at the rural-urban interface. *Journal of Agricultural Economics*, 27(3):217–232, Nov. 2002.

[4] T. Caraco, G. Gardner, W. Maniatty, E. Deelman, and B. K. Szymanski. Lyme disease: self-regulation and pathogen invasion. *Journal of Theoretical Biology*, 193(4):561–575, Aug. 1998.

[5] E. Deelman and B. K. Szymanski. Breadth-first rollback in spatially explicit simulations. In *Proceedings of the ACM/IEEE/SCS 11th Workshop on Parallel and Distributed Simulation (PADS'97)*, June 1997.

[6] E. Deelman and B. K. Szymanski. Simulating spatially explicit problems on high performance architectures. *Journal of Parallel and Distributed Computing*, 62(3):446–467, Mar. 2002.

[7] E. Deelman, B. K. Szymanski, and T. Caraco. Simulating lyme disease using parallel discrete event simulation. In *Proceedings of the 1996 Winter Simulation Conference*, pages 44–50, Dec. 1997.

[8] C. for Disease Control and P. (CDC). Lyme disease. Online website. Available online at: http://www.cdc.gov/ncidod/dvbid/lyme/.

[9] P. Hontalas, B. Beckman, M. DiLoreto, L. Blume, P. Reiher, J. Ruffles, K. Sturdevant, L. Warren, J. J. Wedel, F. Wieland, S. Bellenot, and D. Jefferson. Performance of colliding pucks simulation on the time warp operating system. In *Distributed Simulation Conference*. Society for Computer Simulation, 1989.

[10] A. Natraj and P. Reynolds. Multi-resolution modeling entities. *TOMACS*, 1999.

[11] R. S. Ostfeld, E. M. Schauber, C. D. Canham, F. Keesing, C. G. Jones, and J. O. Wolff. Effects of acorn production and mouse abundance on abundance and borrelia burgdorferi infection prevalence of nymphal ixodes scapularis ticks. *Vector-Borne and Zoonotic Diseases*, 1(1):55–63, Mar. 2001.

[12] R. Radhakrishnan, D. E. Martin, M. Chetlur, D. M. Rao, and P. A. Wilsey. An Object-Oriented Time Warp Simulation Kernel. In D. Caromel, R. R. Oldehoeft, and M. Tholburn, editors, *Proceedings of the International Symposium on Computing in Object-Oriented Parallel Environments (ISCOPE'98)*, volume LNCS 1505, pages 13–23. Springer-Verlag, Dec. 1998.

[13] D. M. Rao. *Study of Dynamic Component Substitution*. PhD thesis, University of Cincinnati, Cincinnati, OH 45221-0030. USA, Aug. 2003.

[14] D. M. Rao and P. A. Wilsey. Dynamic component substitution in web-based simulation. In *In Proceedings of the 2000 Winter Simulation Conference (WSC'2000)*, Dec. 2000.

[15] D. M. Rao, P. A. Wilsey, and H. W. Carter. Optimizing costs of web-based modeling and simulation. In *Proceedings of the First International Workshop on Internet Computing and E-Commerce (ICEC'01)*, Apr. 2001.

[16] Y.-P. Wang and S. T. Bentley. Development of a spatially explicit inventory of methane emissions from australia and its verification using atmospheric concentration data. *Journal of Atmospheric Environment*, 36(31):965–4975, Oct. 2002.

[17] J. Wu and J. L. David. A spatially explicit hierarchical approach to modeling complex ecological systems: theory and applications. *Journal of Ecological Modeling*, 153(1):7–26, Oct. 2002.

IEEE COMPUTER SOCIETY