Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

A CONCEPTUAL ARCHITECTURE FOR STATIC FEATURES IN PHYSICAL SECURITY SIMULATION

Volkan Ustun
Haluk Yapicioglu
Skylab Gupta
Abishek Ramesh
Jeffrey S. Smith

Department of Industrial and Systems Engineering
207 Dunstan Hall
Auburn University
Auburn,AL,36849,U.S.A.

ABSTRACT

The aim of this paper is twofold: First, to propose a data model that enables the user to model a physical facility at different levels of detail and explicitly incorporate interactions among the components of the facility. Second to suggest a methodology for line-of-sight, which is the primary factor in recognition of threats in physical security settings.

1 INTRODUCTION

This paper describes the conceptualization of the static aspects of a facility such as geometry, structure etc. and defines the relations between these aspects and the active entities in the simulation of physical security systems. Jordan *et al.* (1998) discuss the general problem of simulation of physical security systems and state that the use of discrete event system simulation in the design and evaluation of physical protection systems is limited and current approaches are either very low-cost analytical tools or highly costly stochastic models involving human participants. Discrete event system simulation might be an alternative for those who want to know the responses of a given physical protection system under various threat scenarios at a moderate cost.

Smith *et al.* (1999) describe a general system structure for physical security discrete-event simulation. In this structure, there are two different types of simulation entities: intruders and guards. Intruder entities move through the facility in order to reach or acquire a specified target and guard entities are trying to detect the intruder entities in order to prevent them from achieving their goals. Discussions in this paper assume a similar setting.

One inherent feature in physical security system simulation is its explicit attention to spatial features and spatial behavior. Hence, it is important to formally define a conceptual infrastructure to represent the spatial features of the facilities that would then allow modeling these aspects in an object-oriented design. Precise reflection of spatial features is of vital importance for physical security system simulations since most of the cognition and decision making activities of the entities will be based on these spatial features.

Threat detection is primarily performed by vision and recognition in our physical security system context. Therefore, vision is the primary method that is used to perceive the environment and it is the main moderator of behavior of the entities in our physical security system simulation. Static data model and vision are closely related and they provide the necessary infrastructure that will let to add advanced behavioral features to the entities that participate in the simulation.

The remainder of the paper is organized as follows. In Section 2, the details of the conceptual data model are given. Then, the relation between this data model and the actual simulation is discussed. Methodology used for vision and some preliminary findings on the performance are provided in Section 4. Finally, we conclude in Section 5.

2 CONCEPTUAL DATA MODEL

Static aspects of the facilities (i.e. structure and the location of buildings, other objects present within the area, properties of the materials) will be used by several components of the simulation software and hence a common data structure should be used in order to increase the overall effectiveness of the system. This common data structure might also be thought of as a standard that dictates design rules for different components of the simulation software. The data model described in this section defines the conceptual relationships between different types of data and provides a conceptual tool to extract information from facility drawings. Moreover, it provides a representation hierarchy, which allows a way to describe the facility at different levels of abstraction.

The data model is inspired from a natural hierarchical leveling such as, at the top level there is the whole facility, and as we go deeper in the hierarchy there are buildings, floors, rooms and objects in the rooms. However, this natural representation is not sufficient to cover the relationships between different data types that are consumed by the software. Furthermore, it also creates some ambiguity while running the simulation at various levels of detail. The data model that we propose enhances this natural decomposition. It defines three data types to represent the static features of a facility and describes the relationship between these data types. To simplify the physical modeling, all data types use eight point convention to define the corner points of any object of any of the three data type. Three different data types are:

- A **solid object** is any 3-dimensional solid shape.
- A **zone** is a 3-dimensional volume that is described by its bounding objects. Its distinction from a solid object is that zones allow entity movement in them. Hence, a zone is basically a hollow shape either empty or filled with something that allows entity movement (e.g. water).
- A **portal** is a 3-dimensional shape that is included in a bounding object that connects two or more zones either visually or letting movement between zones (e.g. windows, doors etc.).

Our data model is represented in the form of a tree-like structure; the root node (starting node) of the tree corresponds to the *highest level* of abstraction (lowest level of detail); as we go further down the tree, the level of abstraction decreases (higher level of detail). There are several levels in the data model and these levels determine the desired precision (fidelity) in the simulation. For example, we may want to deal with a building as a solid object. A higher level of detail may be defining the object as a hollow 3-dimensional object, which is defined by a zone in this data model. The next level of detail may be partitioning the building into rooms and so on. Each structure in the facility may have different representations at different levels. For example, whole facility may be represented as a solid cube object in level 0 and it may be represented as a hollow cube (zone) in level 1 representation.

There are three types of relations that define the connections between different data types in the data hierarchy either vertically or horizontally. These relations are:

- **Bounding relation:** This relation is used to define a zone using some other objects (referred as bounding objects) as the boundaries for the zone. This relation is a horizontal relation in the data hierarchy.
- **Inclusion relation:** Inclusion defines a different type of relation for different data types.
  - **Zone-object:** Objects can be related to zones by inclusion relation. This is a horizontal relationship. Defines the objects that are included in a volume. (e.g. desk in a room)
  - **Zone-zone:** Zones can include other zones in a horizontal relationship. An example for this having rooms in a floor.
  - **Object-portal:** This relationship can only be defined for bounding objects. It connects a portal to an object. (E.g. door in a wall). This is again a horizontal relationship.
- **Parent-child relation:** This type of relation connects different data types at different hierarchical levels. Hence, this is a vertical relation. There are three different types of parent-child relationships.
  - **Zone-zone:** A zone can be represented as a combination of different zones in a lower hierarchical level. This relation defines an exact partitioning and hence the combination of zones used in the lower hierarchical level should exactly form the zone in the higher hierarchical level.
  - **Object-object:** An object can be defined as a combination of different objects in a lower hierarchical level. (e. g. at the higher level, a table can be defined as a solid cube which can be represented by 8-points. This solid cube is then represented as a combination of 5 different objects in the lower level, one top and four legs. The important point here is combination of these 5 objects doesn't necessarily give the cube that is used to represent the table at the higher level.)
  - **Object-zone:** An object can be represented as a zone in a lower hierarchical level. Example for this type of relationship is having a building defined as an object in the higher hierarchical level and as a zone bounded by walls, a floor and a ceiling in the lower hierarchical level.

In this context, different levels in the tree structure represent a vertical relationship, which is a partitioning of an object into smaller pieces in order to obtain a better fidelity. Inclusion and bounding relationships are horizontal relationships and they represent the relationships between objects at a single level.

A simple example facility is used to illustrate the conceptual data model described. This sample facility (Figure 1) comprises of two buildings and one building is divided into two rooms. First building has a door (D1) and a window (W1). Second building has a main entrance door (D2) and two windows (W2,W3). There is also another door that is located in the wall that separates the two rooms in this building.

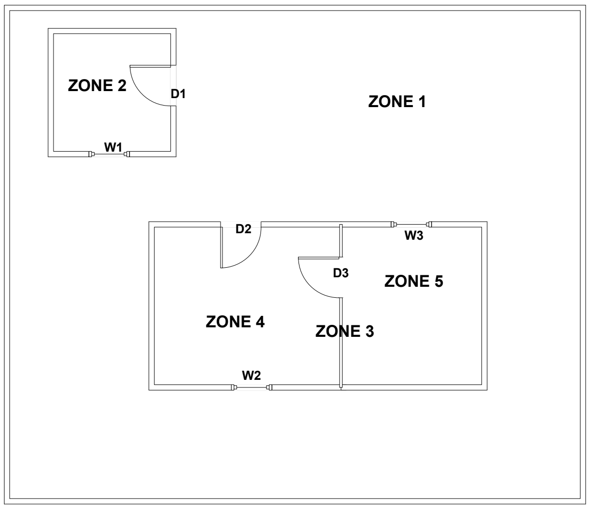

Figure 1: Sample Facility

This sample facility is conceptually described using the data model defined in this section (Figure 2). There are three hierarchical levels in this conceptual description. First level shows the whole facility as a single solid object, which is the root of the conceptual description tree. Second level introduces three zones. First zone (Zone 1) covers the whole facility excluding the two buildings. Zone 2 and Zone 3 presents the two buildings in the facility and these zones are included in Zone 1. Each zone is bounded by solid objects. These bounding objects are simply walls for the buildings. Bounding objects for Zone 1 represent the boundaries for the facility. One of the bounding the objects of Zone 2 includes a door (D1) and another one includes a window (W1) as portals. Bounding objects for Zone 3 include a door (D2) and two windows (W2,W3) as portals. For simplicity, only one bounding object for each zone is presented in Figure 2.

At level 3, Zone 3 is partitioned into two zones (Zone 4 and Zone 5). Zone 4 and Zone 5 share the all bounding objects with Zone 3 except one. Both Zone 4 and Zone 5 are also bounded by the separator wall, which includes a door (D3) as a portal.

This conceptual description facilitates running our simulations at different fidelity levels. For example, if the simulation is running at level 1, only Zone 1, Zone 2, and Zone 3 will be present in the simulation along with their bounding and included objects. If we need a higher precision, simulation can be run at level 3. In this level, Zone 1 and Zone 2 will be present along with their associated objects. However, Zone 4 and Zone 5 will replace Zone 3.

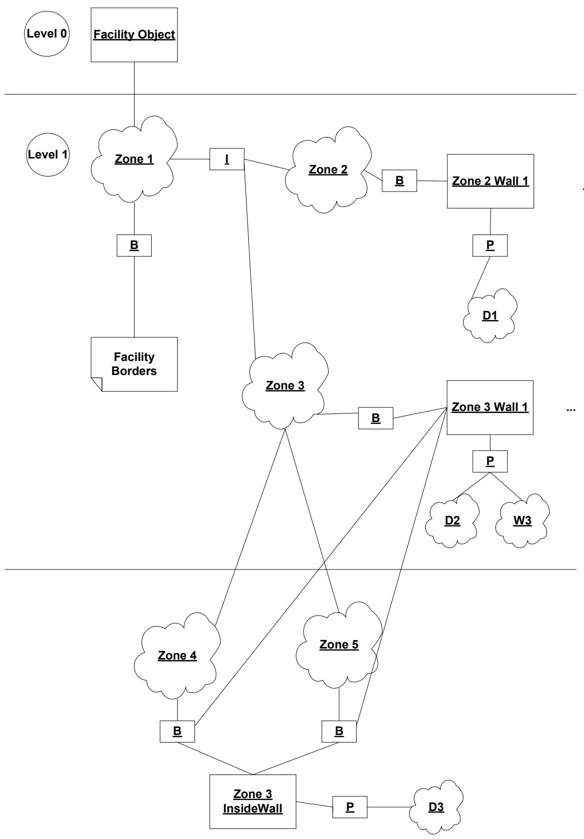

Figure 2: Sample Facility Conceptual Data Model

3 USE OF DATA MODEL IN THE SIMULATION

Spatial features of the environment will affect the entities in two general ways. First, they will limit the movements of the entities and secondly, they will obstruct the vision and hence impact “who sees whom” information. These interactions are captured using two different graphs. The former graph represents the possible movements available to an entity while the latter represents the possibility of visibility allowed by the environment.

Both of these graphs can be defined by means of “portals”. For example, consider the facility in Figure 1. For this example facility, we have three doors that can allow movement and visibility between zones and three windows that can allow visibility between zones depending on their states. There can be several states for a portal, some of them can be accepted as “open” states (that allow movement and/or visibility through) and some of them can be accepted as a “close” states (movement and/or visibility is not allowed).

Zone movement graph defines the possible movements between zones. Entities in the simulation can move from one zone to another zone if there is an edge between these two zones and the conditions for this edge are satisfied. Entities can move freely within the zones (not colliding with the objects that are in the zone), however, movements between zones will be controlled using the zone movement graph. Zone movement graph for example facility is shown in Figure 3. This figure shows the access paths to the different zones of the facility. From Zone 1, Zones 2, 3, and 4 are accessible, and Zone 5 is accessible through Zone 4 only. It should also be noted that Zone 3 decomposes to Zone 4 and Zone 5 in a higher levels of detail.

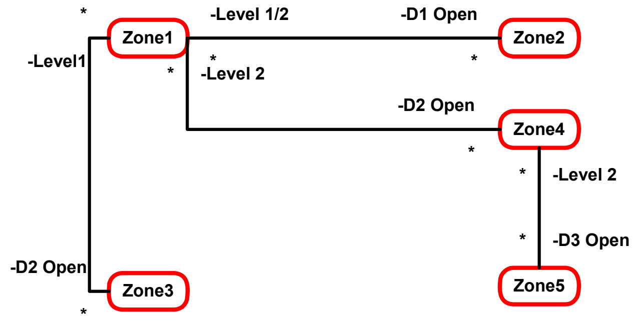

Figure 3: Zone Movement Graph

Teller introduced the concept of potentially visible sets (Teller 1992) and discussed the importance of their application in virtual environments with a high degree of occlusion. Portal Visibility Graphs use a similar approach and show which portals are visible from a certain portal. This information is important for the line of sight calculations that are introduced in Section 4. Portal Visibility Graph decreases the number of objects that are incorporated in visibility calculations and hence the computation required to perform line of sight calculations. Portal Visibility Graphs are constructed in the pre-processing stage by casting lines from several points in a portal to other portals. If any of these lines can reach the target portal then an edge is added to the Portal Visibility Graph between the vertices that are representing these two portals. An example Portal Visibility Graph for the sample facility is depicted in Figure 4.

Figure 4 can be read as follows: from the location of door 1, D1, there is a possibility of visibility through door D2, and windows W1 and W3. Now assume that there is an entity in the simulation which is in Zone 1 and it can only see the portal W2. Through portal W2, it may see portals W3, D2 and D3 but portals D1 and W1 are invisible to this specific entity. The first immediate conclusion is nothing inside Zone 2 is visible to this entity. The second conclusion is Zone 2 can be treated as a single object instead of using the bounding objects of Zone 2 in line of sight calculations for this entity.

4 VISIBILITY DETECTION

Line of Sight (LOS) visibility computations form an important aspect of simulation models for physical security

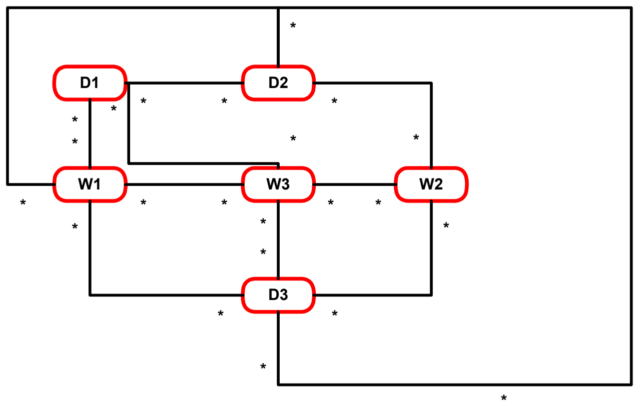

Figure 4: Portal Visibility Graph

systems. The basic question is to determine all entities that a particular entity can see; indeed, in a simulation model, such visibility questions arise at every simulation step, and these need to be answered a large number of times. The answer to visibility questions determines an entity's future behavior, and so it is important that LOS computations be as accurate and efficient as possible. As noted in Darken (2004), LOS detection is basically a problem of visibility of an entity's surface. Furthermore, in a 3-dimensional simulation model, visibility calculations need to take into account not only the attributes of the entity (such as how far can it see), but other facility characteristics such as any *barriers* (walls for instance) that obstruct the view.

Line-of–sight module described by *Smith et al.* (1999) uses planar barriers and the entities in this module are represented by a single point. Visibility calculations in this module are performed by basic point-to-point ray casting and checking whether or not this ray intersects with any planes that are representing the barriers. As noted earlier, the models of the environment (such as buildings and rooms) described in this paper consist of 3-dimensional shapes (represented by eight points) and entities are defined by multiple points. Line-of-sight calculations discussed in this paper use these 3-dimensional definitions.

We define the field of vision of an entity by three parameters – the looking direction, LOS range, and the cone half-angle. The LOS range determines how far the entity can see, and the looking direction is defined by a vector (in 3-dimensional space) in which the entity is looking; lastly, the actual region of view is represented as a 3-dimensional cone (with the vertex at the viewpoint of the entity) that is centered around the looking direction vector, with a half cone-angle specified by the third parameter above. Visibility detection then reduces to finding all entities that are within the field of view (the 3-dimensional cone) of an entity.

We follow the Multiple Ray Casting Approach similar to the one discussed in Darken (2004), to determine if an entity *i* can see entity *j*. Here, we compute the visibility of multiple candidate points on entity *j*'s surface by casting rays from the viewpoint of entity *i*; if all the candidate points are not visible, then we conclude that entity *j* is not visible, and if *at least* one candidate point is visible, we conclude that entity *j* is visible. As per the precision level desired, a predetermined number of points representing the target entity *j* are chosen (these can be the positions of the entity's arms, feet, head, and so on). Thus, the basic assumption here is that if a sufficiently large number of points representing entity *j* are chosen, they will correctly represent the 3-D object(s) representing the entity in the model.

We now briefly describe the LOS visibility algorithm. The algorithm starts by forming a list of all possibly visible target entities; this list contains all entities that are within the cone of view of the viewing entity (the 'source' entity). For each entity in this list, we then check if any candidate point representing the entity is visible; a candidate point might not be visible if there is a barrier obstructing its view. This basically calls for checking if the ray cast from the source entity's viewpoint to a candidate point of a target entity intersects any barriers in the environment. The list of possibly obstructing barriers is then constructed, mainly to speed up the overall algorithm. This is done by first translating/rotating the 8 corners of a barrier and then applying front and back clipping procedures to determine if the barrier can obstruct the view; these clipping procedures also determine the exact portions of a barrier that may possibly obstruct the view. After the list of possibly obstructing barriers is formed, the algorithm then loops through each barrier in the list and checks if the barrier obstructs any of the candidate points of the target entity; if all candidate points are obstructed by some barrier, then the target entity is not visible, otherwise the algorithm concludes that (a portion of) the target entity is visible. The computations to determine if a barrier actually obstructs vision are rather involved, and we briefly describe them next.

The main idea here is to define a plane perpendicular to the 'looking direction' vector, and then project all barriers from the set of obstructing barriers to this plane; this plane is called the *Viewing Plane*. The candidate points of a target entity are also projected onto the viewing plane, and a check is made to see if the projected points are within the boundaries of any of the projected barriers. If all of the projected candidate points are within the boundaries of a (projected) barrier, then the barrier obstructs the view, otherwise not. We basically perform *Perspective Projection*, since it more realistically models practical situations. It should be noted that since we are using perspective projection, the size of objects projected on the viewing plane depends on the focal length, and this focal length is also a parameter defined for each viewing entity. It remains to determine if a candidate point lies within any projected barrier; and this is accomplished by first determining the convex hull of the projected corners of the barrier (using Graham's Scan algorithm), and then using the Crossing Number algorithm to determine if a particular candidate point is within the convex hull polygon corresponding to a barrier.

Thus, LOS computations form a major part of simulation models for physical security systems. The proposed visibility detection methodology uses techniques from computational geometry and carefully optimizes and blends these together, resulting in an efficient and effective LOS visibility algorithm.

4.1 Computational Efforts of LOS Visibility Algorithm

Since LOS calculations are one of the most intensely used component of the simulation package, the execution time of the LOS visibility algorithm is of vital importance. In order to obtain a basic idea on the computational time requirement of the algorithm, we tested the algorithm using randomly generated test cases. Results are summarized in the following Table 1.

Table 1: Computational Time

n	*Avg (in seconds)*	*CV*
10	0.0006	7.7645
20	0.0007	8.4203
30	0.0006	8.5912
50	0.0008	6.7139
75	0.0009	6.0339
100	0.0024	5.3497
150	0.0032	4.4853
200	0.0033	3.6408
250	0.0033	3.5069
300	0.0036	3.2191
400	0.0045	2.8980
500	0.0058	2.7095
600	0.0064	2.5243
800	0.0076	2.0492
1000	0.0089	2.0931

In Table 1, *n* denotes the number of barriers in each test case. Note that the test problems are generated such that all the barriers created randomly within the cone of view. The LOS visibility algorithm is coded with Java and executed using a PC with a Pentium-IV 3 GHz processor and 512 MB of RAM running a Windows XP operating system.

As can be clearly seen from Figure 5, the computational time requirement changes linearly with respect to the number of barriers in the cone of view. The trend line fitted to the plot has an R^2 value of 96%. Even with a number of barriers of 1000, the execution of the LOS visibility algorithm does not take more than 0.01 seconds. Based on this, we can claim that the LOS visibility algorithm is fast enough to be accommodated within the simulation package.

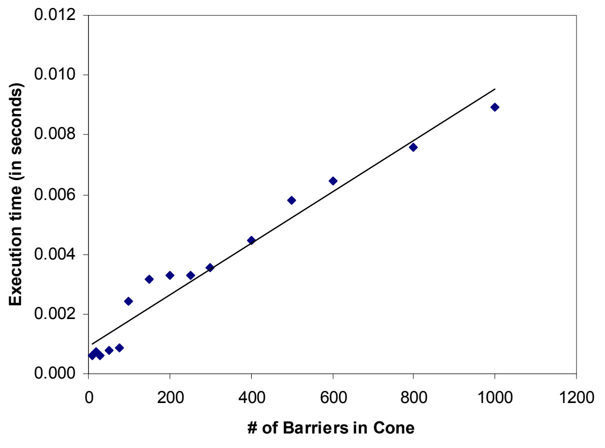

Figure 5: Average LOS Visibility Algorithm Execution Time (in seconds)

5 CONCLUSION

Effective modeling of complex human behavior and the corresponding decision-making processes is critical for simulation-based analysis of physical security systems. However, most of the research reported in the literature encapsulates very simple, largely pre-determined entity behavior models. The central role of visual cognition on entity behavior and the difficulty of explicitly incorporating this visual cognition in simulation models is a major cause of the limited use of complex entity behavior within these simulation models. Our perception is that the difficulty of representing the spatial features of the environment is a significant component in the difficulty of modeling visual cognition. The conceptual architecture described in this paper is an important milestone in this regard since it allows us to formally define the environment and to effectively incorporate the spatial features of the environment for visual cognition. Performance tests conducted for the line-of-sight algorithm generated good results and we believe that it is possible to extensively use this algorithm for visibility calculations in physical security system simulations.

REFERENCES

Darken, C. 2004. Visibility and concealment algorithms for 3D simulations. *Proceedings of Behavior Representation in Modeling and Simulation (BRIMS).*

Jordan S.E., M.K. Snell, M.M. Madsen, J.S. Smith, and B.A. Peters. 1998. Discrete-event simulation for the design and evaluation of physical protectionsystems. *Proceedings of the 1998 Winter Simulation Conference,* ed. D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan, 899-905. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Smith J.S., B.A. Peters, S.E. Jordan, and M.K. Snell. 1999 Distributed real-time simulation for intruder detection system analysis. *Proceedings of the 1999 Winter Simulation Conference,* ed. P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, 1168-1173. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Teller S.J. 1992. Visibility computations in densely occluded polyhedral environments. Ph.D. Thesis. University of California at Berkeley.

AUTHOR BIOGRAPHIES

VOLKAN USTUN is a Ph.D. student in the Industrial & Systems Engineering department at Auburn University. He received his B.S. and M.S. degrees in Industrial Engineering from Middle East Technical University (METU), Turkey in 1997 and 2000, respectively. Prior to joining Ph.D. program at Auburn University, he has worked as a software engineer at The Scientific and Technical Research Council of Turkey (TUBITAK). His email address is <ustunvo@auburn.edu>.

HALUK YAPICIOGLU is a Ph.D. student in the Industrial & Systems Engineering department at Auburn University. He received his B.S. degree in Industrial Engineering from Anadolu University, Turkey and M.S. degree in Industrial Engineering from Middle East Technical University (METU), Turkey in 1997 and 2001, respectively. He worked as a research assistant in the Department of Industrial Engineering at Anadolu University during 1997-2002. His email address is <yapicha@auburn.edu>.

SKYLAB R. GUPTA is a Ph.D. student in the Industrial & Systems Engineering Department at Auburn University, AL. He received his B.S. in Mechanical Engineering from the M.S. University of Baroda, India and M.S. in Industrial Engineering from Auburn University. His research interests include logistics, scheduling, optimization and simulation model development. His email address is <guptasr@auburn.edu>.

ABISHEK RAMESH is a M.S. student in the Industrial & Systems Engineering department at Auburn University. He received his B.S. in Mechanical Engineering from Bharathiyar University, India in 2004. He has worked part time in a mechanical industry during 2002-2004. His e-mail address is <ramesab@auburn.edu>.

JEFFREY S. SMITH is a professor in the Industrial & Systems Engineering department at Auburn University. Prior to joining Auburn, Dr. Smith was an associate professor in the Industrial Engineering Department at Texas A&M University. He received the B.S. in Industrial Engineering from Auburn University in 1986 and the M.S. and

Ph.D. degrees in Industrial Engineering from Penn State University in 1990 and 1992, respectively. In addition to his academic positions, Dr. Smith has held industrial positions at Electronic Data Systems and Philip Morris U.S.A. Dr. Smith is an active member of IIE and INFORMS. His email and web addresses are <`jsmith@auburn.edu`> and <`http://sim.auburn.edu/~jsmith`>.