# MODELING AND SIMULATION OF A GENERAL MOTORS CONVEYOR SYSTEM USING A CUSTOM DECISION OPTIMIZER

by

MIRZA MOHAMMAD LUTFE ELAHI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2008

To my parents Md. Fozle Elahi and Latifa Begum,

my younger brothers Babu, Monzur and Tanvir

and

my beloved country

Bangladesh.

ACKNOWLEDGEMENTS

First, I would like to thank Almighty Allah for giving me the strengths and courage to overcome all the difficulties that I have faced throughout my life so far.

I would like to express my deepest gratitude and admiration to my supervisor Dr. Gergely Záruba for his continuous help and excellent guidance throughout my research work.

I want to extend my sincere appreciation and special thanks to Dr. Jay Rosenberger for helpful discussions and suggestions. I would like to thank Mr. David Levine for his time in serving as my thesis committee member. Special thanks to Karthik Rajpurohit for being an extremely helpful project partner.

I would also like to extend my appreciation to University of Texas at Arlington for providing me the financial support, academic resources, and facilities during my graduate study. I would like to thank Dr. Bahram Khalili for his valuable advices and proper guidance as a graduate advisor.

. I would like to thank all my relatives, friends for their help, suggestions and good wishes throughout my studies.

Above all I would like to thank my parents, my brothers and Neha for their intense support, blessings, love and inspiration.

November 25, 2008

# ABSTRACT

MODELING AND SIMULATION OF A GENERAL MOTORS CONVEYOR SYSTEM USING A CUSTOM DECISION OPTIMIZER



Mirza Mohammad Lutfe Elahi, M.S.

The University of Texas at Arlington, 2008

Supervising Professor: Gergely V. Záruba


The temporal behavior of conveyor systems can be modeled using discrete event (DE) simulations. DE modeling provides a quick and cost effective method for analyzing complex problems as different scenarios can be tested rapidly without affecting the day to day activities of production systems. A simulation model coupled with decision optimization on routing choices enables the evaluation of different decision strategies.

In the General Motors paint shop at Arlington, Texas, the complex conveyor system moving cars to the paint booths has been observed to mix up same color batches of cars coming from the body shop. At the paint booth, every time two consecutive cars have a different color the paint head needs to be cleaned and primed with the new color; the cost of such paint head changes accumulates to a significant expense annually. By observing the decision making process on the conveyor system

it was apparent that better routing decisions could be made, thus reducing the resource wastage.

In this work a DE based simulation model is developed for the General Motors paint shop conveyor system. The simulation model interacts with a decision optimizer at four critical points in the system trying to regroup batches of different color cars. Simulation results of the current decision making policies are compared to that of the proposed optimized policies, showing that significantly better performance can be achieved in terms of number of paint head changes.

TABLE OF CONTENT

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

Most manufacturing systems are characterized by a list of operations in a particular sequence, that a part or job has to undergo before it exists the system. Developing a simulation model that covers the explicit details of manufacturing systems, to represent reality is a complex task. The simulation model offers great opportunities for studying system response when structural relationships within the model are altered. System responses can be further studied for change in decision policies in a timely and cost effective manner.

Discrete event (DE) simulation offers many conveniences that makes it attractive tool for analysis. It can “compress time” so that years of activities can be simulated in minutes and in some cases in seconds. Simulation of a complex system simplifies the study of alternatives and helps in developing optimal or near-optimal policies for system management. Simulation of complex manufacturing systems provides an analytical tool to identify bottlenecks and to control system parameters.

## 1.1 Problem Statement

General Motors (GM) Corporation is the largest automotive manufacturer in the world. In the Arlington, Texas paint shop of GM, about 1,000 vehicles a day get painted in two shifts. Cars in same-color batches enter the paint shop on a conveyor from the body shop. Between two consecutive color-batches there may already be some mix up of colors as cars come in from body shop. Shuffling cars inside the paint shop creates more mixing between consecutive batches. At the paint booth, which is completely automated, those color-batch shuffles can result in a significant number of paint changes in the paint heads. As a paint robot at each paint booth, receives a new car it obtains the color code to which the car needs to be painted.  If the next car is of a different color then the paint robot has to flush its paint head with solvent and prime it with the new color. Such “paint head replacements” (we are going to refer to the change of paint in the paint head as paint head change) cost USD 15 in average and thus the cumulative cost of the replacements over a year is well above USD 1,200,000. This incurs a huge impact on GM’s production cost and moreover it contributes to environmental pollution. Thus plant leaders were looking at finding a solution to reduce the number of paint head changes. Experimenting with the facility’s conveyor layout is not an option that can be pursued, as it incurs huge setup cost and can cause significant down times. However, a simulation study of the current facility could be an effective method to study the response to changes in the system or how decisions on routing cars are made in the system. Such a modeling and simulation approach can save a significant amount of cost, time, and engineering effort.

In order to understand how the conveyor system could shuffle up batches of cars we need to take a look at how the paint shop works. There are two work stations in the paint shop where human interactions are needed. During union mandated shift breaks those work stations cease working, i.e., the conveyors stop delivering vehicles to next conveyors and previous conveyors have to be stopped not to deliver more cars into these work stations. The usual way to deal with such “burstiness” in supply is solved by introducing buffers. Thus, during shift breaks and at the end of each shift, buffers conveyors are used to accumulate and store cars. Every shift in paint shop starts with those buffers filled to their capacity. These buffer conveyors subsystems usually consist of several parallel conveyors. At the beginning and end of these buffer conveyors there are critical points where decisions have to be made on how to split or merge to/from the parallel conveyors. One subsystem we have identified as a bottleneck in the paint shop contains five parallel buffer conveyors with same size. Originally splitting and merging were done in round robin fashion in this subsystem, i.e., although this subsystem did not introduce any more shuffling, it was not used to "unshuffle" cars back to their correct order. Another subsystem contained two buffer conveyors with different sizes – one is referred to by GM as *short loop* while the other has a descriptive name: *long loop*. The *long loop* is almost ten times longer than *short loop*. In the original set-up, the plant management gave preference to the *short loop* for splitting at the ingress of this block if there is space at the start of *short loop*. At the egress the subsystem another decision has to be made as to which of the vehicles should be pulled first from these two conveyors. Again, currently the *short loop* is preferred if there is a vehicle in it readily available. As a result, this splitting and merging adds more shuffling between different color-batches of cars.

To reorder or to shuffle as little as possible those mixes poses a difficult scheduling problem with high complexity due to physical space constraints. Moreover new scheduling algorithms cannot be tested on the system as their outcome may be questionable and cause a significant production loss at the paint shop.

### 1.2 Contribution

Discrete event simulations of temporal systems can provide a good way to evaluate various decision strategies and to optimize their parameters. Different algorithms can be studied for their effectiveness in optimizing the paint head changes without affecting the day to day activities at the plant. In this work we develop a discrete event simulation model of the GM Arlington, Texas paint shop conveyor system which interacts with the decision optimizer. Although our simulator was developed specifically for this paint shop, the objects can be reused to create simulations of other conveyor systems. We have interfaced our simulator with decision optimization libraries and used optimization strategies developed by colleagues. We will show simulation results for two scenarios – one without the decision optimizer which reflects the current paint shop's conveyor system and another one using the decision optimizer. By analyzing the output traces we can compare the performance (in terms of paint head changes) between the current system and the proposed optimized decisions.

### 1.3 Organization

The rest of this Thesis is organized as follows. Chapter 2 describes the background information needed on simulation studies, discrete event simulations, modeling conveyor systems and optimization techniques. Important concepts relevant to those areas will be discussed. Chapter 3 describes the current model of GM’s plant and provides the simulation model we developed. It also outlines the decision optimizer that we used in our simulations. In Chapter 4 we detail our experimental set up for our simulation. Discussions on results can be found in Chapter 5. Chapter 6 concludes the work with the summary of the project and points to some possible future work.

CHAPTER 2

BACKGROUND

## 2.1 Simulation

Simulation is a powerful and cost effective tool for modeling real world systems. Software modeling of complex real world system has become one of the most used decision making tools in many fields, especially in engineering, manufacturing, telecommunication, military and transportation. Simulation provides a cost effective way of collecting information for decision making. The size and complexity of real systems rarely allow physical study on the system itself to provide information. Hence discrete event simulation has become a method of choice to analyze complex systems. The success in applying discrete event simulation depends on the depth and breadth of the underlying model as an approximation of the system [1]. A survey of simulation modeling using discrete event simulation in manufacturing plants can be found in [2].

## 2.2 Simulation of Manufacturing Plants in Automotive Industry

Simulation of industrial systems has been prevalent since the early 1960s. Simulation has been used to study wide range of problems in automotive manufacturing. A simulation study of an entire GM assembly plant was performed to increase its production throughput in [3]. Simulation and optimization studies helped to improve car body production at PSA Peugeot Citroën [4]. A simulation-based decision support was developed in Training, Operations, and Planning for Visteon's Sterling

Plant [5]. The simulation models available in literature for automotive industry are developed mostly by the experts within the plant. The simulation model developed in this study utilizes the data collected from The Arlington, Texas GM plant and covers the details of the paint shop of the plant in detail.

### 2.3 Modeling Conveyors and Queuing Systems

Queuing systems have been extensively simulated and studied in past. Mostly two kinds of queuing systems are used which are either open loop or closed loop systems. In an open loop system, work arrives from outside the system at the rate independent of system and beyond the control of system managers. Closed loop systems from open loop systems can be distinguished by control over work arrival time. A lot of the manufacturing, transportation and telecommunication systems are characterized by closed loop systems. In addition to choosing routing, sequencing and scheduling policies, the ability to control jobs arrival based on state of the system offers system managers a valuable tool for decision making. This motivates the study of routing and scheduling policies for simulation models designed to optimize the response of the system.

Conveyors, since their invention, have been the backbone of industries for transporting materials within the manufacturing plant. In simulation modeling there are generally two types of conveyors: accumulating and non-accumulating conveyors. Most of conveyors that appear in simulation models are simplistic. Although simulation of the conveyors is simplistic in nature, it is necessary to obtain appropriate information, such as, the type of the conveyor, conveyor speed, capacity and other factors to model it as realistically as possible. For a production line where the movement of jobs is completely controlled by the conveyors, the aforementioned conveyor parameters play an important role in developing a synchronized system.

## 2.4 Decision Optimization

Mathematical optimization techniques have been widely used in variety of fields, especially in job scheduling, inventory planning, transportation, logistics planning, telecommunication optimization and resource planning. Such optimization techniques have been used in past and are still in use in the automotive industry for the same purposes.

### 2.4.1 Linear Programming

Linear programming gives the direction of a method to achieve the optimal result in a given mathematical model (objective function), given a set of constraints represented as a set of linear equations. This method is an optimization technique of a linear objective function, subject to linear equality or inequality constraints. Integer linear programming is a particular case of linear programming where some or all of the unknown variables can only take integer values. While there are polynomial solutions to linear programming problems, solving integer linear programming problems has been shown to be NP-hard. There are three common variations within integer linear programming:

- Pure Integer Programming (IP) -- all variables are integer.
- Mixed Integer Linear Programming (MILP) -- some variables are integers.
- Zero-one IP -- the integer variables are binary which can take either one (yes) or zero (no).

Integer linear programming can be applied extensively to various fields in financial planning, resource planning and also to solve some complex engineering problems. Linear programming is used in manufacturing, transportation, energy, and telecommunications industries. It has proven to be useful in modeling diverse types of problems. For example the simulation study in [3] uses, linear programming to optimize job scheduling and thereby increase its production throughput.

### 2.4.2 Dynamic Programming

Dynamic programming which is similar to divide and conquer method, solves a problem by integrating the solutions of sub problems. When sub problems overlap, dynamic programming is a good choice to solve that problem. It reduces the computation of the same problem again and again by solving it only once. It saves the solution in a table and uses the saved solution when it encounters the same sub problem next time.

The development of a dynamic-programming solution can be broken into a sequence of four steps [6].

1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Compute the value of an optimal solution in a bottom-up fashion
4. Construct an optimal solution from computed information

Dynamic programming has been used widely in search and optimization techniques. It has been used in inventory management, job scheduling, resource planning and work force scheduling.

CHAPTER 3

MODEL AND IMPLEMENTATION

## 3.1 Current Model

The paint shop at GM can be divided into different logical blocks. Figure 3.1 depicts the current GM paint shop plant which serves as a basis of our model.

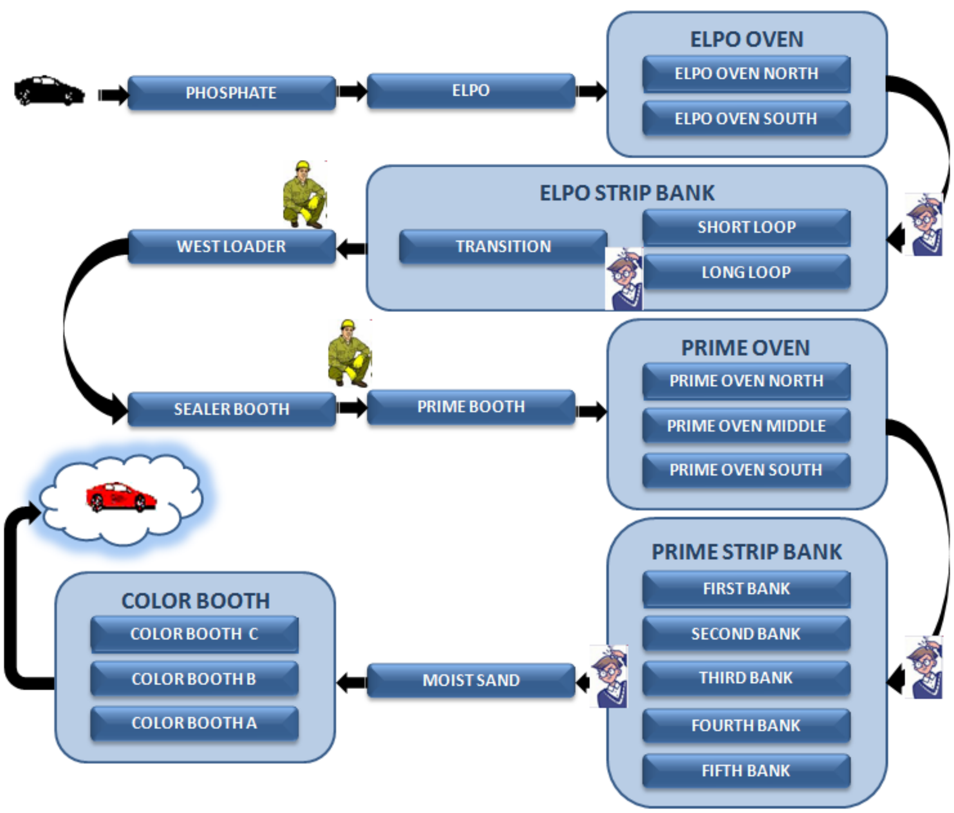



Figure 3.1 Logical block diagram of paint shop

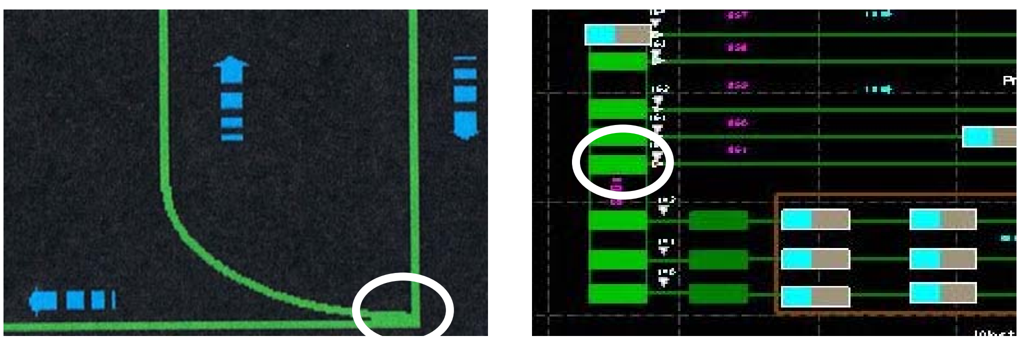
Each block represents a painting (sub) process. Each process consists of conveyors with different capacities and speeds (both of which are fixed). Some of the processes have more than one conveyor in order to synchronize the entire system. For our study, we have considered 11 processes in the paint shop. Each car starts with the *phosphate* process and goes through all of the other processes once to complete the paint job.

The entire shop works in two shifts per day. The first shift starts at 6:00 AM in the morning. There are three shift breaks during this shift. First one occurs at 8:30 AM for 23 minutes. Next one is at 11:00 AM for 30 minutes and the last one is at 1:30 PM for another 23 minutes. The shift ends at 2:30 PM.

Each car enters paint shop from body shop and starts with the *phosphate* process where body of the vehicle is cleaned and coated with phosphate solution. In elpo (electrophoretic priming operation) the body of the vehicle is covered with a special substance to protect it from corrosion. Then the body needs to be heated and baked to finish the elpo process; the cars enter the *elpo oven* which is a slower process than the previous two. For this reason it has two parallel conveyors to synchronize the speed of the conveyors with the previous process. After finishing the *elpo oven* cars enter the *elpo strip bank*; this is basically a buffer. It has two conveyors, one is *short loop* and another is *long loop*. The capacity of *long loop* is almost 10 times the capacity of *short loop*. The next process is *west loader* and this is the first place in the paint shop where human interaction is needed. In this process cars are moved from one support frame (skid) to another manually. The two conveyors from the *elpo strip bank* merge just before the *west loader*. The next process in the sequence is the *sealer* *booth* after which cars are delivered to the *prime booth*. The purpose of the prime *booth* is to spray primer on the body of the car which improves the adherence of the paint to body of the car. Then the body of the car needs to be baked again as it enters the prime oven. It takes long for a car to pass through the oven area thus the conveyors needs to be slow inside the oven; instead of having one long oven, the prime oven is implemented using a relatively short oven with three parallel lines. Each car enters one of the conveyors in sequence from *prime booth*. The cars from prime oven are delivered in the same order as they enter. Then cars enter the *prime strip bank* which has five parallel buffer conveyors (the prime strip bank is one of the buffers used to store cars while human labor is suspended). Cars pulled from *prime strip bank* are merged to form a single queue and are sent to *moist sand* where any scratch is identified and fixed. Once the cars pass through this process they are scheduled to be painted in one of the three *color booth*s. Cars in the repair section of the *paint booth* are some time scheduled to enter back to either *color booth* or *moist sand*. This part of the paint shop is not considered for our study.

There are two places in the paint shop where human interaction is needed. One is just after *elpo strip bank*, i.e., the *west loader* and another is just before the *prime booth*. During shift breaks, *elpo strip bank* stops delivering any car to west loader. The *sealer booth* also stops delivering cars to the *prime booth*; it does not have to accommodate cars from previous process because *elpo strip bank* does not deliver any car to *west loader*. The conveyors in *elpo strip bank* and *prime strip bank* are used as buffer conveyors. At the end of each shift, there should be no cars in any of the processing conveyors except in *elpo strip bank* and *prime strip bank* (which are used as buffers).

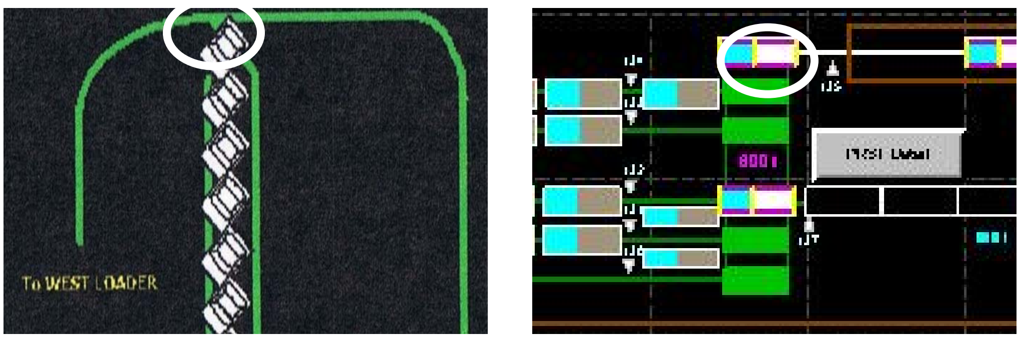
There are thus four major decision points in the paint shop (as depicted with a scratched head in Figure 3.1). Two points are for splitting from a single conveyor to multiple conveyors and two for merging from multiple conveyors to a single conveyor. Splitting points are at the entry points of *elpo strip bank* and *prime strip bank*. There are two conveyors in *elpo strip bank*, short loop and long loop; originally, short loop is given preference over long loop. The *prime strip bank* is another buffer. There are five conveyors in parallel which have equal capacities and conveyor speeds. Currently, cars are pushed to the *prime strip bank* in a round robin fashion. Figure 3.2 shows the splitting and the merging points (white circles) in *elpo strip bank* and *prime strip bank*.



(a) (b)

Figure 3.2 (a) *elpo strip bank split* (b) *prime strip bank split*

Merging points are at the end of *elpo strip bank* and *prime strip bank* where one conveyor has to be selected for the next outgoing car. Currently, short loop is given preference over long loop for merging to a single queue at end of *elpo strip bank*. Merging at the end of *prime strip bank*, works in the same way as splitting. For example, if the last car was pulled from first conveyor then the next car should be pulled from second conveyor. If no car is available to pull from second then it takes from third and so on. These four points have been considered for optimization and regrouping the batch sizes. Figure 3.3 shows the merging points (white circles) in *elpo strip bank* and *prime strip bank*.




(a) (b)

Figure 3.3 (a) *elpo strip bank merge* (b) *prime strip bank merge*

There are more splitting and merging points at the beginning and ending of *elpo oven*, *prime oven* and *color booth*. However all of the splitting and merging algorithms currently used do not affect the response variable, i.e., cars per head changes. The order of the batches is well maintained in these work stations. . The order of the batches is well maintained at these points.

### 3.2 Implemented Model

In order to create a DE simulation model we have decided to develop a custom C++ based model, in order to have the most flexibility when interfacing with optimization packages. It was also important to use an object oriented language especially since conveyor systems can be built (inherited) from smaller conveyor units.

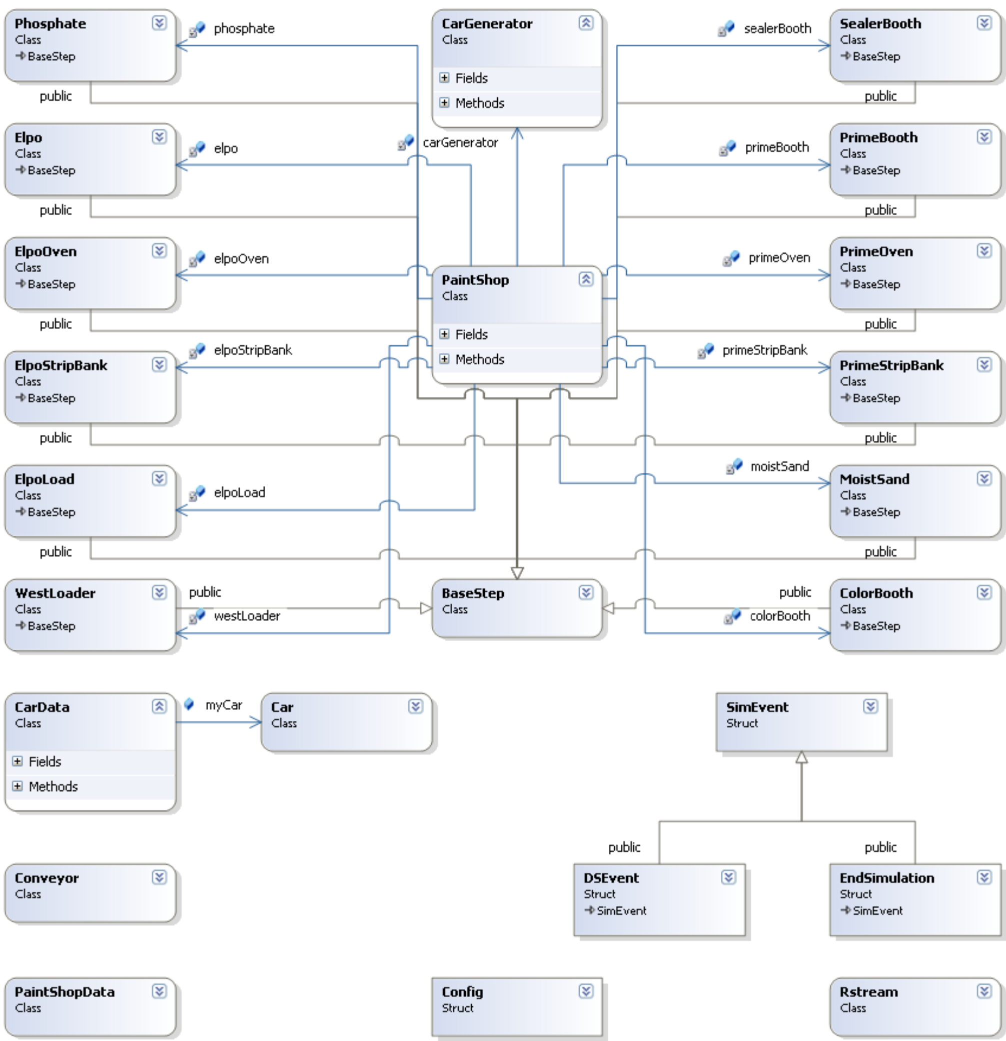



Figure 3.4 Class Diagram of simulation model

Figure 3.4 depicts the class diagram that represents the entire simulation model as implemented. Thus, the DE simulation model is developed in C++ and a PERL script is used to run the different iterations of the simulation for different set of inputs and to reach a high confidence level on a low mean error.

In Figure 3.4 *PaintShop* is the main class that represents the entire paint shop. All processes are members of the class *PaintShop*. Processes are inherited from *BaseStep* which contains the common functionalities of each process. Each process is associated with either a single or multiple numbers of *Conveyor* objects according to the configuration.

Class *CarGenerator* creates car object and schedules them for delivery in same color batches to the *PaintShop*. This is also the object that is used to model mixing of car colors among batches trying to model the output of the body shop of GM.

Structures *SimEvent*, *DSEvent* and *EndSimulation* represent the discrete event simulation engine [7]. They manage the event queue and thus the simulation clock.

Class *Conveyor* has all the functionalities of a conveyor. It maintains a queue for car objects and simulates car movement by scheduling events into the event queue.

Class *Config* [7], which is a global object, reads all the configuration data from a file; this data is managed by class *PaintShopData* which is a singleton object. When data is needed for any part of the program like conveyor capacities for any processes then *PaintShopData* provides the required information. Class *Rstream* [7] which is also a global object is used to generate random numbers that are needed to mix the batches.

### 3.3 Decision Optimizer

In our simulation the decision points (see Figure 3.1) in *elpo strip bank* and *prime strip bank* are used for optimization. When each car reaches any of the two splitting points we call the decision optimizer with the set of oncoming cars and cars placed into the conveyors and wait for a solution (i.e., conveyor number to send the car to) and the car is sent to that conveyor.

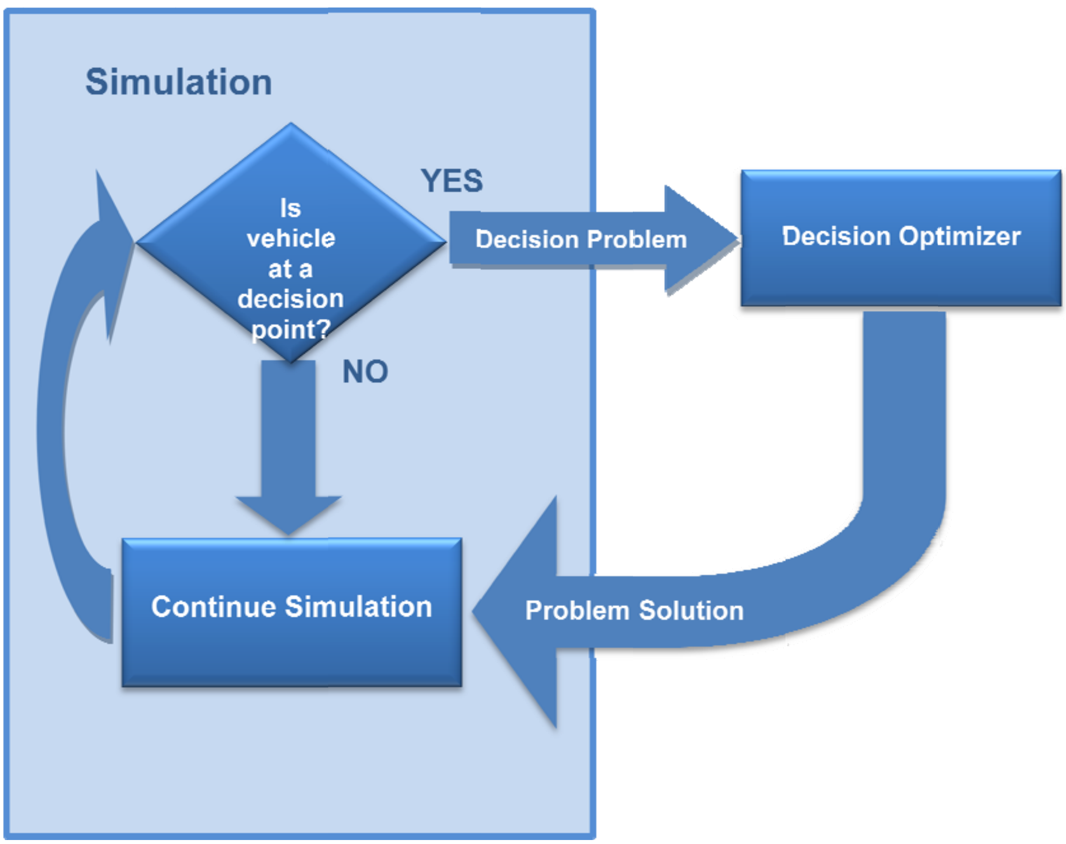



Figure 3.5 Simulation model with Decision Optimizer

For merging the simulation checks the conveyors at the end of each bank periodically (with an expected period of car arrival). For *elpo strip bank* if both of the loops have cars ready to deliver then the decision optimizer is called which will make a decision on from which conveyor to take the next car. If only one of the conveyors have a car ready to be delivered then the decision optimizer is not called and the trivial solution is going to be used. Similar process is followed for the *prime strip bank*. Figure 3.5 depicts this interaction between the discrete event simulation and the decision optimizer.

### 3.3.1 Splitting Single Queue to Multiple Queues

In optimizing the decisions made for splitting a single queue to a multiple queues the color codes of the previous cars in each of the destination queues must also be taken into account. An integer linear programming model optimizes the grouping of cars of similar colors. The optimization model is based on minimum cost multi commodity flow model. It is a (0,1) boolean programming model that takes into account the color code of the last car that was pushed to different conveyors and also the color code of the car at the decision point. The cost of pushing a car to a particular conveyor is ‘0’, if the color codes are the same as the previous car in the conveyor and ‘1’ otherwise. The model developed is solved using CBC [8] open source solver. In addition, the optimizer receives the following parameters that could influence the decision:

1. Total Number of cars in a single queue to be split.

2. Number of empty skids available in different conveyors.

For the optimization to give a feasible result, the capacity constraints, i.e., the sum of individual commodities flowing across an arc should be equal to one and flow balance constraints, i.e., number of arcs coming into a node should be equal to number arcs flowing out of the node must be satisfied. In addition to these two constraints the total number of cars in the queue about to be split must always be either less than or equal to numbers of empty skids available at all the conveyors.

The number of cars considered (look ahead) in the single queue before being processed may or may not have an impact on the final batch sizes as the result solely depends on the colors codes of cars. The optimization model is called for each car during the simulation before the decision is made to which conveyor the car needs to be forwarded. There should be at least one empty skid available in two or more conveyors for the optimization to have any effect on the batch sizes.

### 3.3.2 Merging Multiple Queues to a Single Queues

The currently employed algorithm at GM for merging cars from different conveyors to a single queue is just a round robin system with the color code of last car that was merged not playing a role. This also results in huge mixing of cars of different colors thus reducing the batch sizes. A dynamic programming model was developed to optimize the merging process and so trying to regroup cars into same color batches. The dynamic programming model was created to calculate the total number of color changes for each conveyor involved in the merging process. The model also took in to account the color code of the last car that that was merged in to the single queue. The outcome of the dynamic programming optimization depends also on the number of cars chosen in each conveyor (look ahead) for the process of merging.

The look ahead in each conveyor for the purpose of optimization plays an important role in the problem size of the dynamic programming problem. The size of the problem increases exponentially as the number of cars selected in each conveyor for optimization increases. During our experiments we have not seen an advantage to increase the look ahead beyond five cars. Thus for our study we limited this number to five in each conveyor for the five-to-one as well; for the merging between the short loop and the long loop we have used an asymmetric look ahead with five and ten respectively. The optimization model is called for each car during the simulation before the decision is made to merge to a single conveyor. There should be at least one car in two different conveyors for the optimization to have any effect on the batch sizes.

CHAPTER 4

EXPERIMENTAL SETUP

In this chapter we describe the discrete event simulation based experimental setup we used for our evaluation study.

### 4.1 Input Parameters

Simulator uses a configuration file for setting input and initialization parameters. Thus, we can control the simulation by changing the configuration file. The configuration file contains the following parameters:

- Total simulation time (Total simulation time is the simulation clock time for which the simulation has to run to simulate the entire shift)

- Car arrival time from car generator (body shop model) to paint shop. (Car arrival time is the inter arrival time of the individual skids containing cars coming from the body shop to the paint shop.)

- Percentage of mixing in the car arrival process between two consecutive batches. (Mixing percentage represents the number of cars that can be randomly mixed at the beginning and end of batches. Generally, cars from the body shop are not batched strictly with the same color and thus there are some shuffled periods of colors between two consecutive batches. For example an input value of 20% mix means that 10% cars at the end of one batch mix with another 10% of the cars of the next batch.)

- Length of the conveyors of each process (number of cars) of each conveyor. (Each conveyor can have a different capacity - how many cars or skids it can hold maximally. Some painting sub-processes have multiple (parallel) conveyors with the same capacity while some of these compound processes have multiple conveyors with different capacities.)

- Time it takes to move a skid from one “slot” to the next inside individual conveyors. (Cycle time is defined as the time it takes on a conveyor to move the skid ahead exactly one slot (i.e., from the time it takes the conveyor to move the beginning of one skid to the beginning of the next skid.)

- Number of look ahead cars in both splitting and merging. (The optimizer always takes the decision based on the look ahead cars for both splitting and merging.)

- Color codes of different batches

- Size of the batches. (We experimented for different batch sizes ranging from 10 to 100.)

- Car arrival checking time at the end of processes which have multiple conveyors (for merging). (At the end of some processes which have multiple conveyors like *elpo strip bank* and *prime strip bank*, the simulator has to check whether there is a car ready to pull for delivering it to next process. Thus, car arrival checking times represents how frequently the simulator has to check those conveyors.)

Table 1 shows the name of the processes and the cycle time required by those processes which are provided by GM. Table 2 lists values we used for other parameters.

Table 4.1 Processes and their cycle time

| **Process** | **Cycle time (min)** |
|---|---|
| *Phosphate* | 1 |
| *Elpo* | 1 |
| *Elpo strip bank* | 2.128 |
| *Elpo load* | 0.933 |
| *West loader* | 0.933 |
| *Sealer booth* | 0.933 |
| *Prime booth* | 0.933 |
| *Prime oven* | 2.804 |
| *Prime strip bank* | 0.857 |
| *Moist sand* | 0.857 |
| *Color booth* | 2.034 |

Table 4.2 Different parameters and their values

| **Parameter** | **Value** |
|---|---|
| Simulation time (min.) | 1050 (min) |
| Batch sizes | 10 – 100 |
| Mix percentages (%) | 0 - 30 |
| Car arrival checker at ELPO STRIP BANK | 1.00001 |
| Car arrival checker at PRIME STRIP BANK | 0.9 |
| Car arrival checker from body shop (min.) | 1.00001 |

### 4.2 Initial Conditions of Conveyors

The conveyors in the simulation model start off being empty. However, GM starts each shift with two buffer blocks *elpo strip bank* and *prime strip bank* filled to capacity and *phosphate* is started 36 minutes before the shift begins, as this process takes some time before it starts depositing cars into the buffer conveyors. In order to cope with this two issues our simulation is primed (with a stopped paint booth) until all buffer conveyors filled up. In addition, we start the car generator 36 minutes before the shift starts so that *phosphate* remains full before the actual shift simulation starts. So the steady state condition for our simulation modeling is that *phosphate*, *elpo strip bank* and *prime strip bank* should be full at the beginning of the shift. Hence, we filled the buffer blocks first and then started the car generator 36 minutes before the shift starts. The warm up time for the system to reach the primed state is 476 minutes.

## CHAPTER 5

## EXPERIMENTS AND RESULTS

We ran experiments using the simulation model with different parameters for four different scenarios mainly distinguished by the mixing percentage. We considered four different kinds of mixing ratios: 0%, 10%, 20% and 30%.

### 5.1 Cars per Paint Head Change

In our experiment we traced color changes at the three paint booths after the simulation ran for an entire shift. We recorded the total number of color changes and total number of cars that finished the paint job from paint shop. Thus we could calculate the *cars per paint head change* value for particular mixing ratio and for specific batch sizes.

### 5.2 Savings Calculation

In order to translate the recorded values into numbers easily communicated to GM management, we calculated the financial benefit between the currently used model and the decision processes proposed. We have done that using the following simple calculation:

$c$ = cost for changing a paint head

$d$ = total working days in a year

$n$ = total number of finished paint jobs per shift

$s$ = number of shifts per day

$C_{ij}$ = total savings for batch $i$ and mixing ratio $j$

$d_{ij}$ = cars per head change for batch $i$ and mixing ratio $j$ using current algorithms

$o_{ij}$ = cars per head change for batch $i$ and mix ratio $j$ using optimized algorithms

Therefore, $C_{ij} = c \times d \times n \times s \times \left( \frac{1}{d_{ij}} - \frac{1}{o_{ij}} \right)$ (5.1)

### 5.3 Confidence Level

Using the mixing ratios for batch sizes involve the employment of a random number generator when generating batches. This turns our simulation into a Monte Carlo simulation, where depending on the random number seed each simulation run has a different outcome. Thus, an average over same setup but different seed simulations is obtained which is used to estimate the mean of the underlying random process as defined by the simulation. To obtain a 95% confidence that the relative error (compared to the average) of the average to the mean is less than 5%, the appropriate number of simulations were conducted.

Thus, if

*N* is the sample sizes,

*m* is the average over all samples,

*s* is the standard deviation of total sample,

*Z*($\alpha$) is the normal percentile for $\alpha$ confidence level, then the absolute error is:

$$e = Z(\alpha)\frac{s}{\sqrt{N}} \quad , \tag{5.2}$$

and thus relative error is:

$$r_e = \frac{e}{m} \tag{5.3}$$

If the target relative error $r_e^t$ is $5\%$ but after an initial *N=30* samples $r_e > r_e^t$ then we estimate the number of samples needed that will satisfy our target as:

$$N_e = \left\lceil \left( \frac{Z(\alpha) * s}{r_e^t * m} \right)^2 \right\rceil \tag{5.4}$$

**5.4 Experimental Results and Discussions**

Here we are going to list our experimental results and provide discussions of them. We have run four different sets of experiments with the main difference being in the mixing ratios. For each set of experiments we used the batch size as the factor (and thus in the figures the batch size is depicted on the horizontal axis). For each set of experiments we have two figures, one showing the number of cars per paint head change for both the original GM as well as the optimized approach; the other shows the savings that can be achieved if GM was to switch over to the new optimization method (in US Dollars).

### 5.4.1 Case 1 - 0% Mixing Ratio

The first set of experiments looked at the optimal case, when clean batches come in from the body shop. Thus, since there are three paint booths, we can define an upper bound on the performance of any routing decision maker algorithm, which would be a linear (in the figure) with a value of one third of the batch size. In Figure 5.1 we can see that the optimizing decision maker operates not far away (and sometimes right on) from that theoretical maximum, while the current decision making can be as much as three times worse. This can result in savings in the USD 300,000 range as depicted in Figure 5.2.

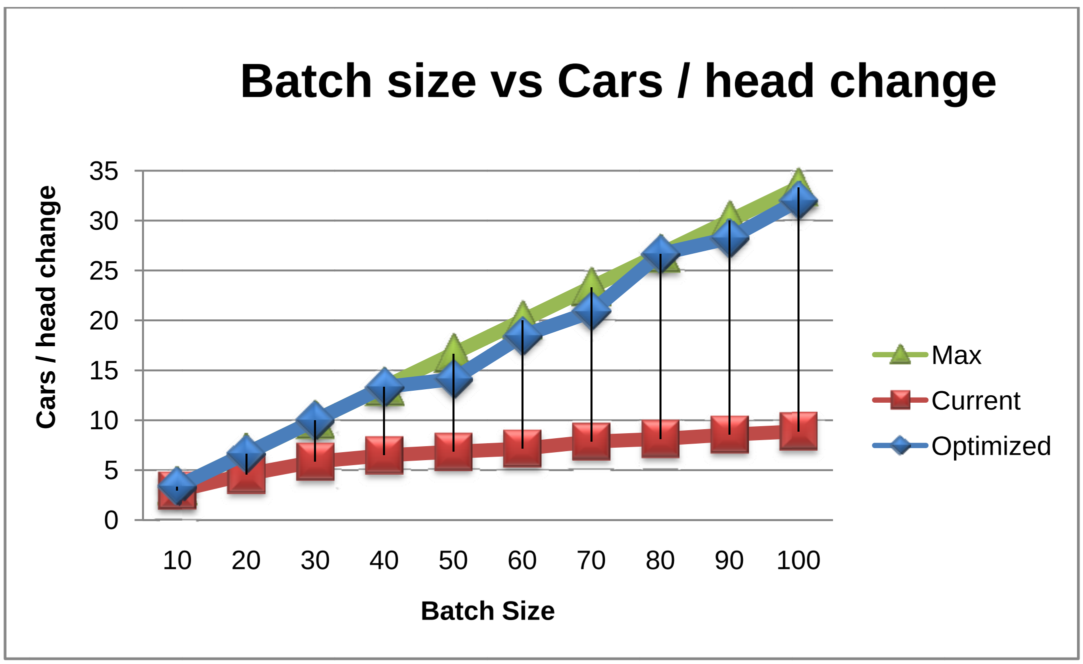



Figure 5.1 Batch size vs cars/head change for 0% mixing ratio

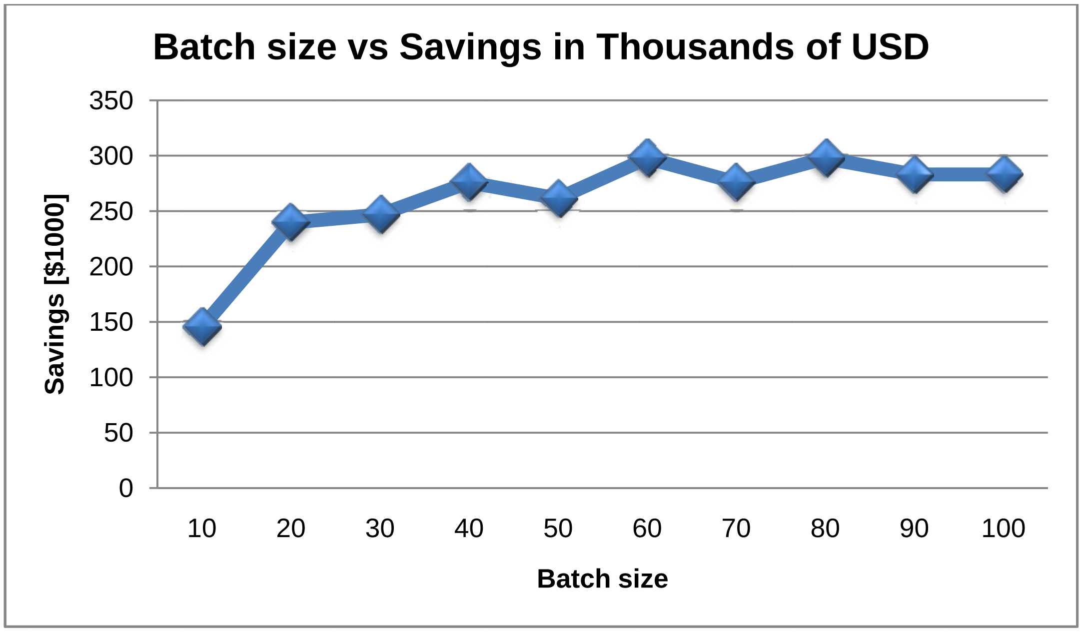



Figure 5.2 Batch size vs savings in thousands of USD for 0% mixing ratio

### 5.4.2 Case 2 - 10% Mixing Ratio

Introducing 10% mixing ratio between batches in the second set of experiments shows that cars *per head change* decreases compared to clear cut batches (0% mixing ratio). In Figure 5.3 we can see that even if the batches are getting mixed (10%), the performance of the optimal algorithms is close to the performance of the previous set of experiments. This indicates that the decision optimizer also performs well with mixing. The results in savings for this mixing ratio are depicted in Figure 5.4 which ranges around USD 350,000.

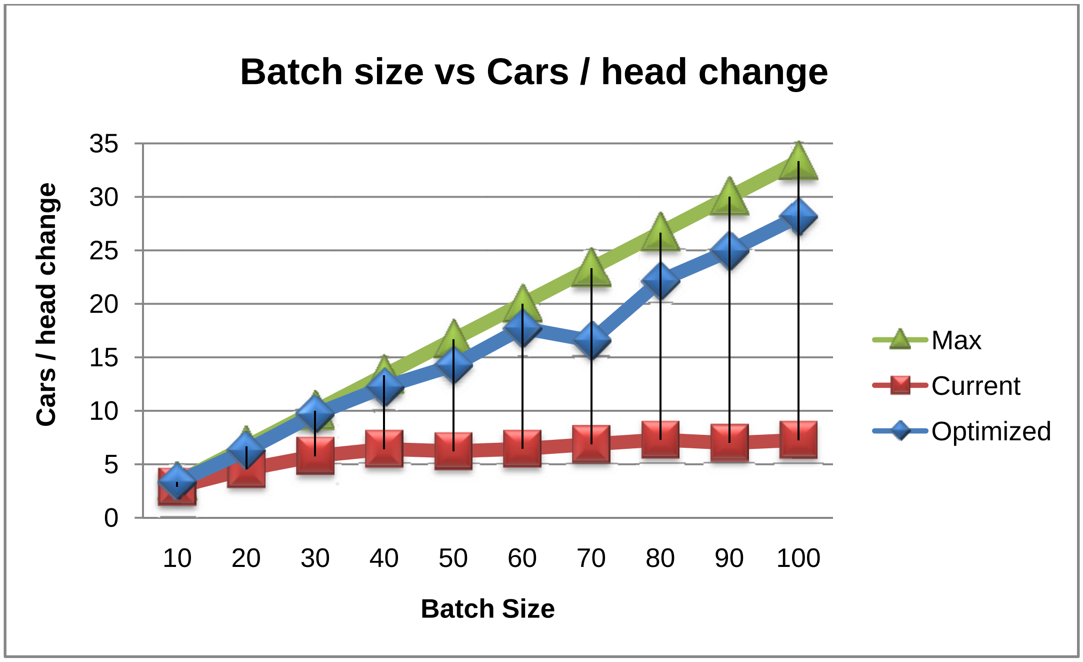



Figure 5.3  Batch size vs cars/head change for 10% mixing ratio

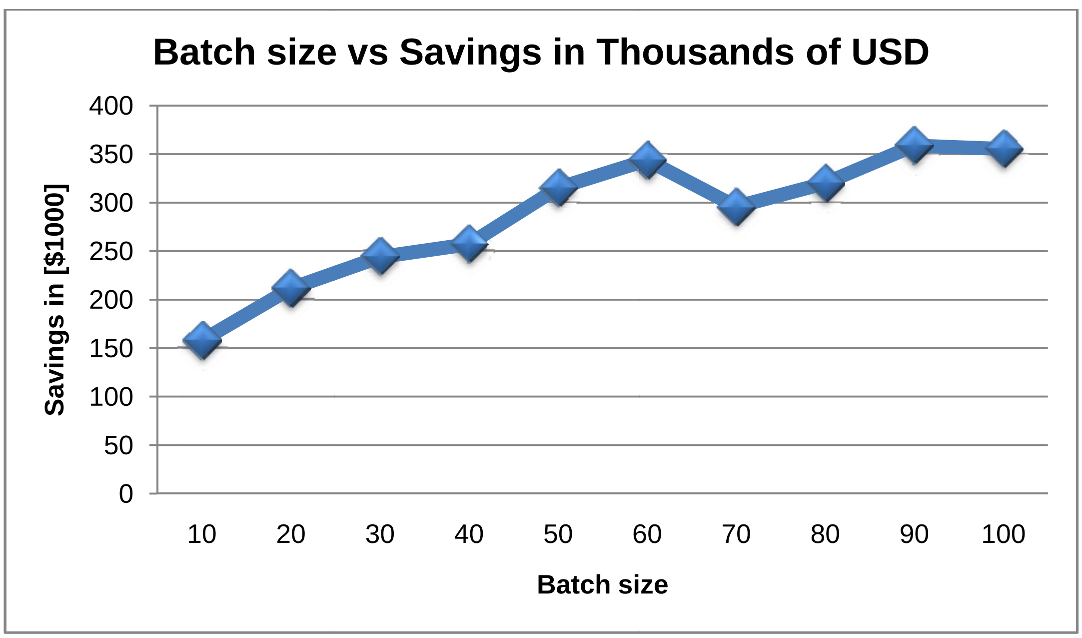



Figure 5.4 Batch size vs savings in thousands of USD for 10% mixing ratio

### 5.4.3 Case 3 - 20% Mixing Ratio

In the third set of experiments we increased the mixing ratio to 20%. Figure 5.5 represents the effect of this mixing ratio in *cars per head change* which shows that the decision optimizer still performed much better than the current one. The performances of the decision optimizer comparing to current one could save around USD 450,000 which can be observed from Figure 5.6. This is even better than the savings in the previous set of experiments.

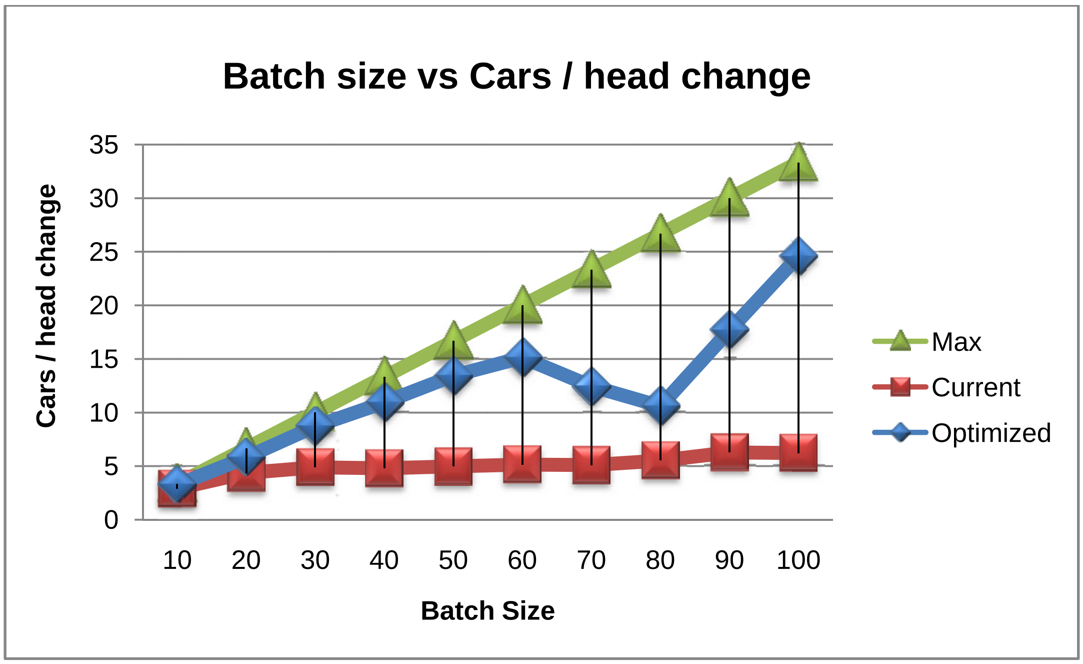



Figure 5.5 Batch size vs cars/head change for 20% mixing ratio

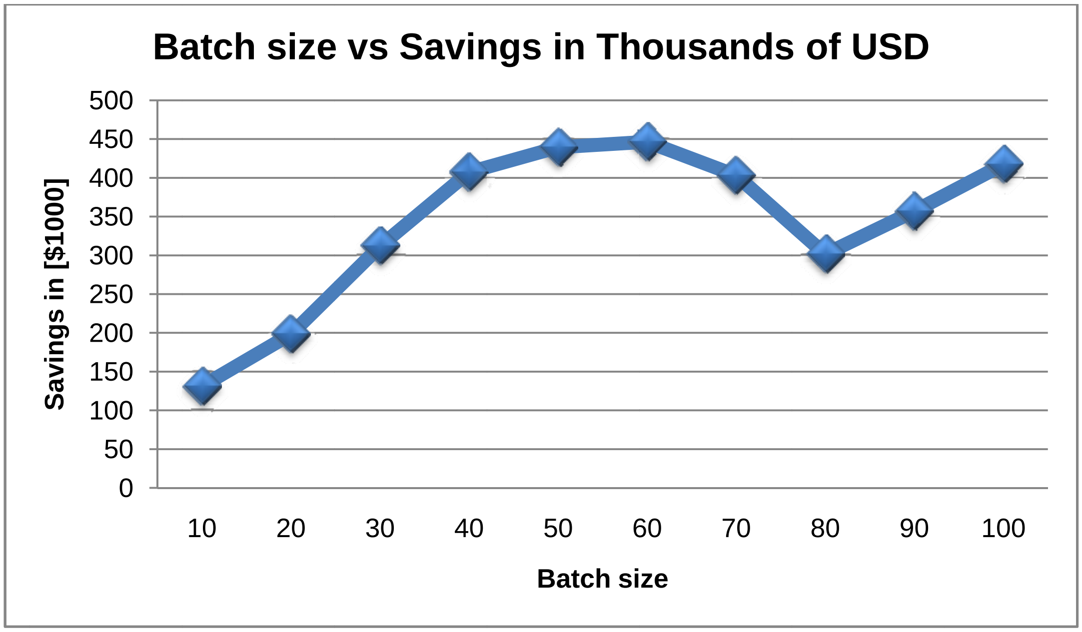



Figure 5.6 Batch size vs savings in thousands of USD for 20% mixing ratio

### 5.4.4 Case 4 - 30% Mixing Ratio

The fourth set of experiments, with highest mixing ratio of 30%, reflects the actual scenario of GM's paint shop. From Figure 5.7, we can observe that the decision optimizer performs very well regarding *cars per head change* for most of the batches in comparison with the current one. The savings curve in Figure 5.8 shows that savings around USD 550,000 (which is also highest among all the mixing ratios) can be achieved if the decision optimizer is applied.

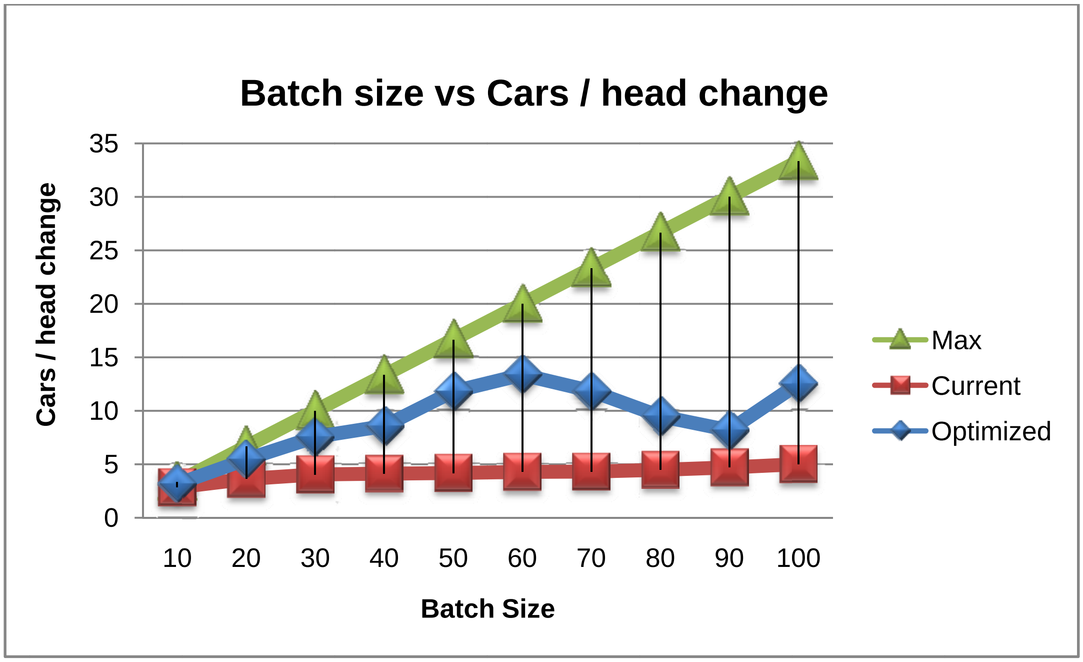



Figure 5.7 Batch size vs cars/head change for 30% mixing ratio

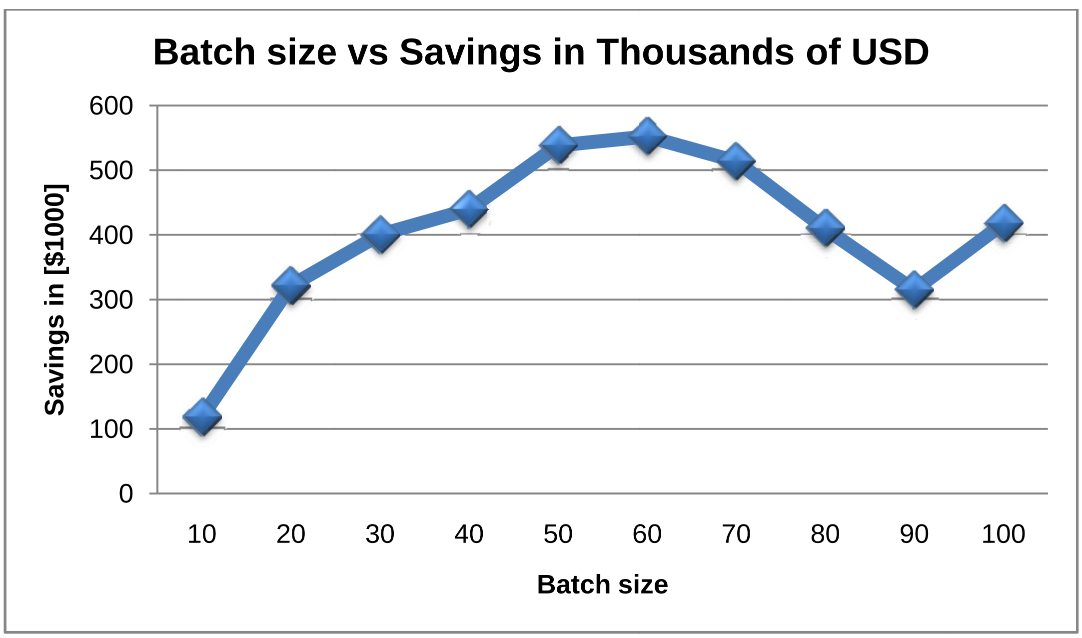



Figure 5.8 Batch size vs savings in thousands of USD for 30% mixing ratio

From Figures 5.3, 5.5, 5.7 we can observe that the *cars per head* change decreases when the batch size grows to 70-80 for mixing ratios 10%, 20% and 30%. To identify the unusual behavior we performed more experiments for three mixing ratios and traced same-color cars group at the three paint booths for different batch sizes and calculated the standard deviations. Figure 5.9, 5.10 and 5.11 depict the results of these experiments.

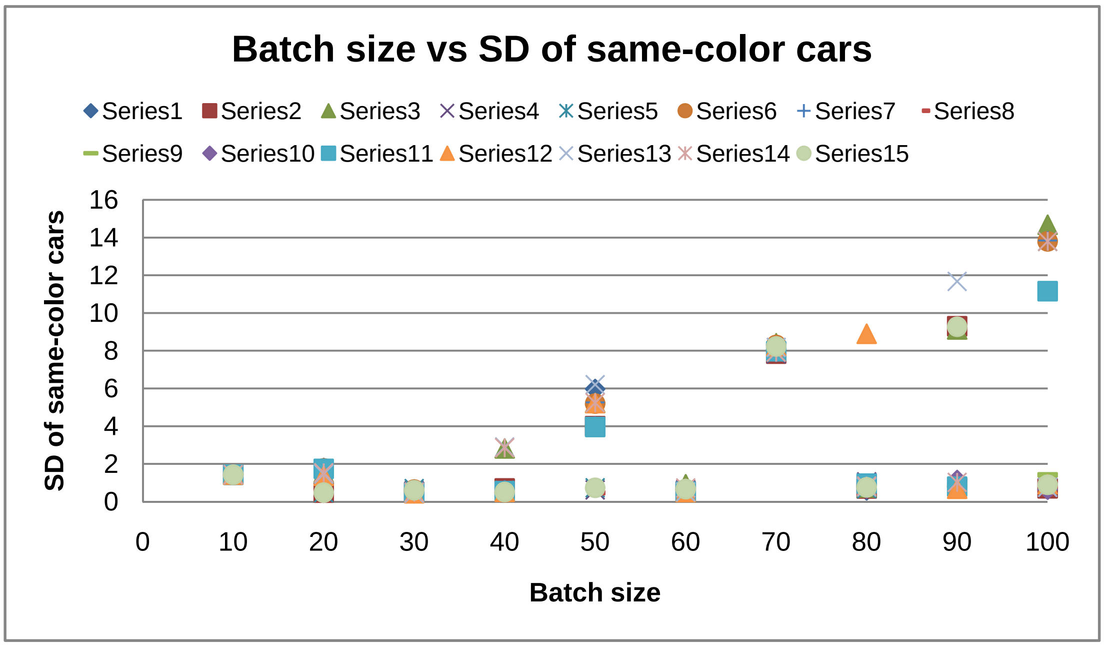



Figure 5.9 Batch size vs standard deviations of same-color cars for mixing ratio 10%

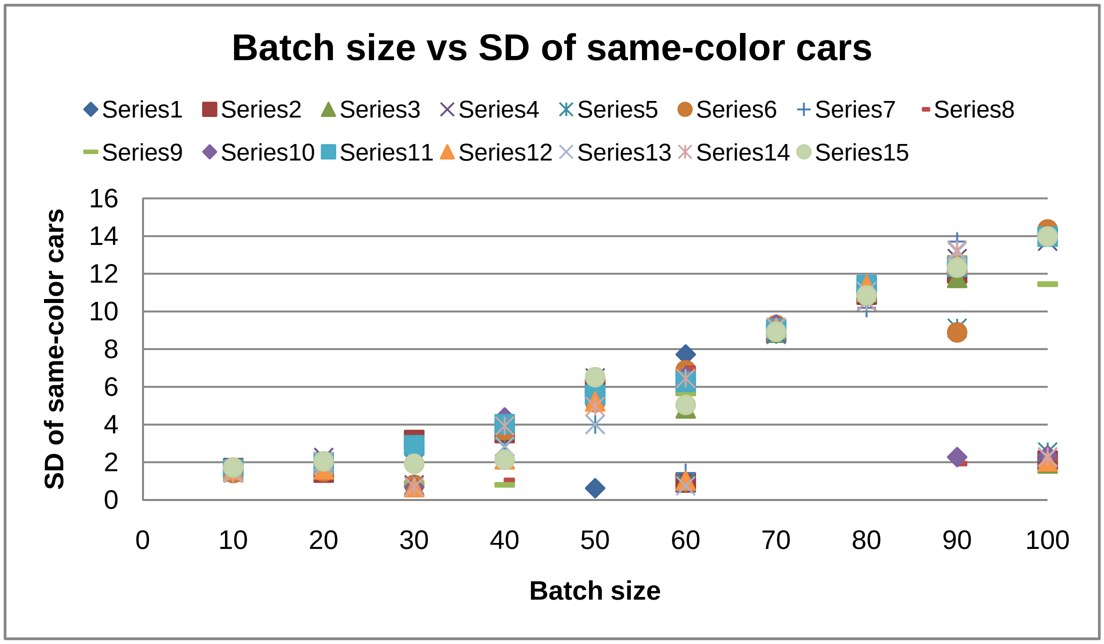



Figure 5.10 Batch size vs standard deviations of same-color cars for mixing ratio 20%

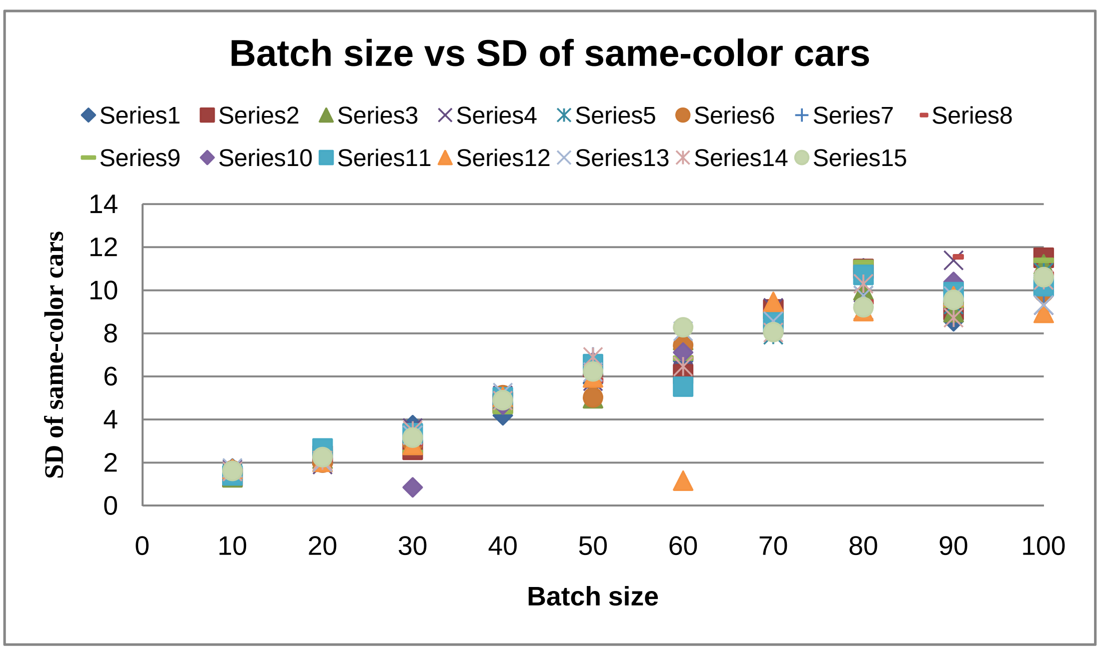



Figure 5.11 Batch size vs standard deviations of same-color cars for mixing ratio 30%

From the figures we can observe that the standard deviation of same-color cars at three paint booths are large for higher batch sizes, especially around 70 and 80. These results provide a hint that the low average cars per head change may be caused by the large differences between same-color cars group around batch sizes 70 and 80.

# CHAPTER 6

# CONCLUSIONS AND FUTURE WORK

In this thesis we presented a simulation study to gain insights into the paint shop conveyor system of the General Motors plant in Arlington, Texas. Simulation modeling of such complex systems can help in evaluating new routing decision making algorithms without interrupting day to day operations of the plant. In the GM paint shop cars to be painted are entering on a single conveyor from the body shop. When ordering the incoming cars at the body shop, attention is paid to group same color code cars thus creating batches of cars which need to be painted using the same color. Sometimes (due to the real-time behavior of the conveyors) the transition between batches is not clear cut. The paint shop conveyor system has been observed to make the situation worse, i.e., mixing the batches even more. This represents a problem, as by the time the cars reach the actual paint booths, they are out of sequence (when looking at their colors). If same color cars are painted then the color in the paint heads does not need to be changed which results in savings on solvent and paint. Thus, increasing the same color batch sizes at the paint booths is very beneficial. Better algorithms could be used at the paint shop conveyors to reorder cars to clean batches. In order to effectively and efficiently evaluate such algorithms a simulation model is needed.

We have connected our simulation to an optimization framework that receives data from the simulator to make routing decisions and relays these decisions back to the simulator. We experimented with different parameters to analyze the plant's current performance (in the terms of how many cars can in average be processed at the plant without changing the paint in the paint head) and performance when using decision optimizer. We have analyzed the conveyor system using our model for various original batch sizes coming from the body shop and various degrees of how clear cut the transitions are (mixing ratio). We have done so for both the current decision making process as well as the optimizing approach. We have shown that GM could save as much as USD 550,000 if calculating with an average cost of USD 15 for a paint head change (a value confirmed by GM's plant).

Change in conveyor speed in long loop certainly has an effect on cars per head change at the paint booth. Increasing the speed of this conveyor will increase the number of available cars at the end of long loop to make merging decision. Further analysis may be required to study the effect of conveyor speed on cars per head change and the optimal batch size.

REFERRENCES


[1] George S. Fishman, Discrete-Event Simulation: Modeling, Programming, and Analysis, Edited by Peter Glynn, Stephen M. Robinson, Springer-Verlag New York Inc., 2001.

[2] Semini, M., Hakon Fauske, Jan Ola Strandhagen, “Application of Discrete-Event Simulation to support Manufacturing Logistics Decision-Making: A Survey,” *Simulation Conference, 2006. WSC 06. Proceedings of the Winter*, pp. 1946-1953, Monterey, California, 3-6 Dec. 2006.

[3] Jeffrey M. Alden, Lawrence D. Burns, Theodore Costy, Richard D. Hutton, Craig A. Jackson, David S. Kim, Kevin A. Kohls, Jonathan H. Owen, Mark A. Turnquist, David J. Vander Veen, “General Motors Increases Its Production Throughput,” *Interfaces,* Vol. 36, No. 1, pp. 6-25, January-February 2006.

[4] Alain Patchong, Thierry Lemoine, Gilles Kern, “Improving Car Body Production at PSA Peugeot Citroën,” *Interfaces,* Vol. 33, No. 1, pp. 36-49, January-February 2003.

[5] George Pfeil, Ron Holcomb, Charles T. Muir, Shahram Taj, “Visteon's Sterling Plant Uses Simulation-Based Decision Support in Training, Operations, and Planning,” *Interfaces,* Vol. 30, No. 1, pp. 115-133, January-February 2000.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithm, Second Edition, MIT Press and McGraw-Hill, 2001.

[7] A simple, elegant, yet easy to use C++ based discrete event simulation engine (event queue), http://crystal.uta.edu/~zaruba/usefulsutff.html

[8] COIN-OR, Computational Infrastructure for Operations Research, http://www.coin-or.org

# BIOGRAPHICAL INFORMATION

Mirza Mohammad Lutfe Elahi, the eldest son of Md. Fozle Elahi and Latifa Begum, was born in February 7, 1981, in Dhaka, Bangladesh. He received his Bachelor of Science Degree from the University of Dhaka, Bangladesh in July, 2005 majoring in Computer Science and Engineering. He started his graduate studies in the University of Texas at Arlington in August, 2006 and received Master of Science degree in Computer Engineering in December, 2008. During his graduate studies, Mirza conducted his research on modeling and simulation using discrete-event simulation technique under the supervision of Dr. Gergely Záruba.