A UNIFIED SIMULATION-BASED OPTIMIZATION WITH INTERPRETATIONAL MACHINE LEARNING

Shugang Kang
Scientific Forming Technology Corporation, Columbus, Ohio, USA

ABSTRACT

With the advancement of information technology, more and more data are being collected to monitor the operation of manufacturing systems. This has provided a material foundation for applying real-time simulation-based optimization to improve the efficiency of complex manufacturing systems. In order to facilitate effective implementation of simulation-based optimization, in this paper, a unified framework for modeling, simulating and optimizing complex dynamic manufacturing systems is proposed. The modeling method, the simulation, optimization and machine learning mechanisms are investigated and presented.

Keywords: dynamic system, real-time data, decision making, simulation, optimization, machine learning

บทคัดย่อ

ด้วยความก้าวหน้าทางเทคโนโลยีสารสนเทศ ทำให้มีการเก็บข้อมูลจำนวนมากเพื่อตรวจสอบการดำเนินงานของระบบการผลิต ซึ่งเป็นพื้นฐานในการประยุกต์ใช้แบบจำลองเพื่อหาค่าเหมาะสมที่สุดแบบทันที เพื่อปรับปรุงประสิทธิภาพของระบบการผลิตที่ซับซ้อน บทความนี้นำเสนอกรอบการสร้างแบบจำลองต่าง ๆ และการหาค่าเหมาะสมที่สุดสำหรับระบบการผลิตที่ซับซ้อนและมีการเปลี่ยนแปลงอยู่ตลอดเวลา

INTRODUCTION

Let us start with a real world story. In a large auto manufacturing plant, the production facility is sophisticated. The shop floor is segmented into different production zones and areas. In each area, various types of equipment are running, and workers are busy along the production lines. The material handling systems are moving parts and assemblies

*Dr. Kang, PhD, MSc, BSc, has engineering degrees from universities in Beijing, Tsinghua and Hong Kong. His present position is Principal Research Scientist. Email: kangshugang@gmail.com

across the main production lines and sub production lines. The whole system is a complex dynamic system. As the production facility is capital intensive, in order to recover the high cost and make a profit, the manufacturing system must work efficiently to maximize its throughput.

To maximize the throughput, the plant needs to keep the equipment running, instead of being stopped or idle. In other words, the plant must minimize the "down time" (down time means that some workstations on the production lines stop working for certain period of time). Down time entails production lost. Therefore downtimes are strictly managed. Each incident of the downtime is recorded, and root cause analyses are performed.

In order to keep track of the down time, the plant has invested a lot of money and effort for monitoring the equipment on the shop floor. Advanced sensors and a software system are used to collect data from the equipment. Huge amounts of data are being collected and stored into a large database. From the collected data, the running status of each piece of equipment on the shop floor can be visualized and displayed on computer screens in real time. Reports are generated to analyze the down times and evaluate the performance of the manufacturing system.

For each workstation, its down time could be "direct down time" or "indirect down time". Direct down time could be "equipment error" or "production error". Equipment error means that the equipment breaks down due to mechanical or electrical failures. Production error could be defective parts or human operator errors etc. Indirect down time means the workstation is either blocked or starved due to the downtime of other workstations in the upper-stream or downstream.

The plant operation staffs can understand and perform root cause analysis on the direct down time without problems. However, it is difficult for them to trace the sources of indirect down times. They cannot clearly understand how the down time in one area affects the other areas. The existing data collecting software system could not provide effective help on this problem. According to the plant manager, "no textbook formula is applicable to our problem".

For the plant manager, the major concern is to make sure that "right parts arrive at the right workstations at the right time". Although it is easy to say, it is not easy to implement. If a down time happens at any parts of the system, it may propagate to the other parts of the system. Depending on the current system status, it may or may not cause problems. It is not feasible to handle exceptions with fixed rules. However, as said before, there is no effective tool to help the plant operation staffs quickly figure out how the down time will affect other areas. In cases where multiple down times happen in

different areas, it is harder to analyze the consequences. Unfortunately, if no appropriate actions are taken in a timely manner, a small down time may propagate to other areas and escalate into big problems. When this happens, it is usually too late to effectively take remedial actions.

In order to get better understanding on the system behavior, the plant has conducted simulation studies with discrete event simulation software systems. The simulation studies reveal some facts that are contrary to common sense. For example, in certain situations, slowing down certain production areas might help in clearing some buffers and improve the overall throughput of the system. This indicates that the system's overall performance cannot be achieved by simply letting each production area do their best. Different parts of the production system need to work in a cooperative manner, especially when exceptions occur. However, how to effectively achieve this is the real challenge.

Due to the complexity of the system, even though the plant has a strong interest in finite capacity scheduling systems to facilitate the manufacturing operations, it is difficult to find a suitable software package on the market to fit their need. The difficulty comes from the complex production processes and constraints. For example, the buffer size constraints are usually not sufficiently considered by the scheduling software systems. As a result, even though huge amounts of data are available, the plant does not have an effective tool to make use of the data for guiding the operation decisions.

For a complex manufacturing system, computer simulation probably is the only viable solution for adequately modeling it. Actually, the plant manager is looking forward to a "real time simulation" system, which can do simulations based on the real-time data, look ahead to what is going to happen, perform various what-if analyses, and figure out the best course of action to guide the manufacturing system operations.

From this real-world story, we can see an acute demand for modeling and simulating complex dynamic systems to guide the operation decisions. The simulation system needs to:

1. make use of the collected data; thus the simulation results will be relevant to the real-time operation need.
2. perform different what-if analyses; thus figure out the best course of actions by comparing different alternatives.
3. perform the simulation and optimization efficiently; thus we can quickly get optimized results to guide the real-time shop floor control

Motivated by the above requirements, a unified framework for simulating, optimizing, and controlling complex dynamic systems is developed in this paper.

The rest of this paper is organized as follows. In the next section (No.2), related studies

in the literature are briefly reviewed. Section 3 describes the modeling method. Section 4 discusses the integrated simulation and optimization engine, and how to integrate this with the existing information system. Section 5 discusses the knowledge-based machine learning mechanism. Section 6 discusses how to integrate the simulation and optimization with the existing information system. The last section concludes this paper and points out future research directions.

Section 2: RELATED RESEARCH LITERATURE

To run manufacturing systems efficiently, extensive research effort has been put into the area of production scheduling (Pinedo, 2012). In general, production scheduling deals with the problem of allocating limited production resources to the production tasks over time, with the objective of accomplishing the production tasks in a timely and cost-effective manner (Smith, 2005).

Scheduling problems has been traditionally studied as a class of combinatorial optimization problems (Jain & Meeran, 1999). These problems are notoriously NP-hard (the hardest of problems; NP = Non-deterministicPolynomial-time). Due to this nature of these problems, it is very difficult to solve the scheduling problems optimally when the problem size is large. Various approximation and heuristic methods have been developed to tackle the combinatorial optimization problems, such as shifting bottleneck procedure (Adams, Balas, & Zawack, 1988), tabu search (Widmer, 1991), simulated annealing (Vakharia & Chang, 1990), and genetic algorithms (Biegel & Davern, 1990).

Although significant research effort has been focused on tackling the NP-hard property of scheduling problems which aims at finding efficient methods to solving scheduling problem optimally or near optimally optimally, static scheduling solutions are seldom carried out in the real world of manufacturing practice. Because of the dynamic nature of the manufacturing systems, an optimized scheduling can quickly become infeasible due to the constant changes. Therefore, simple dispatching rules (DR), such as SPT (shortest process time), EDD (earliest due date), FIFO (first in first out), etc. are widely adopted by the industry in the dynamic manufacturing environments, because of their adaptive nature, low computation effort, and easiness to implement (Liu, 1998; Min & Yih, 2003; Mouelhi-Chibani & Pierreval, 2010; Chang, Dong, & Yang, 2013).

As pointed out by many researchers, no dispatching rule dominates other rules in all circumstances (Wu & Wysk, 1989; Min & Yih, 2003; Metan, Sabuncuoglu, & Pierreval, 2010). That is, one dispatching rule may perform well in some situations, while performing poorly in other situations. Based on this observation, dynamically selecting different dispatching rules could be a potential way to improve the performance of the manufac-

turing system; this has been proven to be effective by different researchers (Wu & Wysk, 1989; Liu, 1998; Metan et al., 2010).

To evaluate the performance of different dispatching rules, simulation study is usually employed. Actually, due to the complexity and dynamic nature of the real world manufacturing systems, simulation is believed to the only feasible approach to study the behavior of complex manufacturing systems without over-simplifying them (Wu & Wysk, 1989; Aqlan, Lam, & Ramakrishnan, 2014).

Traditionally, simulation studies are used for evaluating the performance of a production facility when the facility is newly built or modified. After the production facility is put into operation, the simulation models are usually thrown away. Wysk and colleague proposed to reuse the simulation models for system control purposes (Son & Wysk, 2001). This has evolved into an important research direction that is attracting increasing interest in the production scheduling and control research area (Wu & Wysk, 1989; Peng & Chen, 1998; Borenstein, 2000; Son & Wysk, 2001; Arakawa, Fuyuki, & Inoue, 2003; Guo, Chen, Wang, & Zhou, 2003; Min & Yih, 2003; Dangelmaier, Mahajan, Seeger, Klopper, & Aufenanger, 2006; Xia, Tian, Sun, & Dong, 2008; Mahdavi, DEL: Shirazi, & Solimanpur, 2010; Metan et al., 2010; Mouelhi-Chibani & Pierreval, 2010; Hu & Zhang, 2012; Shirazi, Mahdavi, & Solimanpur, 2012; Puergstaller & Missbauer, 2012; Chang et al., 2013; Riegler, Spangl, Weigl, Wimmer, & Muller, 2013; Remenyi & Staudacher, 2014).

Recall the discussion in the beginning that the plant manager is looking forward to a "real-time simulation" system to guide operation decisions. We can see that the industry and academic community are actually converging on the same spot: "real-time simulation based optimization". This is an important shift for the production scheduling research, which has been largely disconnected from industrial practice in the past (Herrmann & Words, n.d.; Jacobs & Weston, 2007).

Basically, simulation based optimization for real-time control mainly deals with the dynamic selection of dispatching rules by evaluating the performance of different rule combinations based on the real-time system status. The major challenges for real-time simulation and optimization includes three aspects: modeling effort and flexibility, solution quality, and computation efficiency.

Firstly, traditional simulation packages usually cannot provide enough flexibility for simulation-based optimization, and the modeling effort is significant. To alleviate this problem, Borenstein (2000) proposed an object-oriented modeling framework to improve the flexibility of the simulation model, and Son and Wysk (2001) proposed automatic generation of simulation models to reduce the modeling effort. How to flexibly and ef-

fortlessly model the complex manufacturing systems by taking into account the various alternatives for simulation-based optimization remains an open research issue.

Secondly, although various studies have testified to the effectiveness of simulation-based optimization for improving the manufacturing system efficiency, simulation-based optimization was often implemented in ad hoc ways by grafting the existing simulation packages with optimization routines. How to achieve high quality solutions is still not a simple and straightforward task. Therefore, it would be helpful to rethink the problem. Instead of handling simulation and optimization as two separated problems, we may need to consider simulation and optimization as a whole problem, and design a unified solution to resolve it. With such an improved methodology, we may develop effective and reusable software tools to make high quality simulation-based optimization easily achievable.

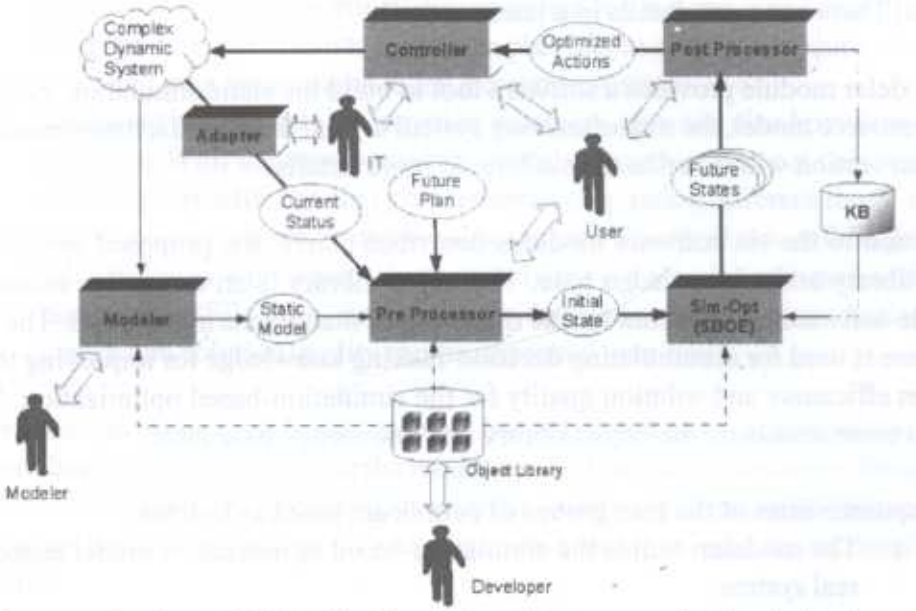
Thirdly, for real-time control purpose, the simulation-based optimization must produce optimized rule selections quickly. For a complex manufacturing system, the simulation often requires heavy computation. Moreover, for optimization purposes, we need to run multiple simulations to figure out the best rule combinations. To reduce the computation effort, some researchers proposed to use offline training mechanisms, such as artificial neural network (ANN), to improve the online computation efficiency (Xia et al., 2008; Metan et al., 2010; Mouelhi-Chibani & Pierreval, 2010; Shiue, Guh, & Lee, 2012). More research effort in this direction is needed to make simulation-based optimization practicable.

Based on the above observations, in this study, a unified framework is proposed to seamlessly integrate simulation, optimization, and machine learning into a total solution to systematically address the above challenges, and which provides a practical and easily implementable software tool to help the industry improve operation efficiency through real-time simulation-based optimization.

Section 3: AN OVERVIEW OF THE MODELING SYSTEM

Figure 1 depicts the overall infrastructure of the proposed system, which includes the six software modules: the simulation-based optimization engine (SBOE), the pre-processor, the post-processor, the modeler, the controller, and the adapter. Four groups of people are involved in implementing it in a real manufacturing system: the modelers, the users, the IT staff, and the developers. In this section, we will briefly introduce the functionalities of each module, how these modules work together, what are the responsibilities of different groups of people, and how the people will interact with the system. The following sections will describe each module in more details.

Figure 1: System overview



Source: Author (NOTE: All Figures in this paper are by the Author)

The Simulation-Based Optimization Engine (SBOE) is the most important module of the system. The input of the SBOE is the initial state of the system, the output of the SBOE is the future states of the system. Based on principles of discrete event simulation, the SBOE simulates the system state changes from the initial state to the future states.

As the major goal of simulation-based optimization is to dynamically select the best rule combination, for a given initial state, the SBOE needs to evaluate different possible rule combinations, each of which will lead to a different future state. Hence, there will be multiple possible future states, from which we need to pick the best one and use it to guide the real-time control. (There will be more details of SBOE in a later section).

The pre-processor prepares the initial state, which is the input for the SBOE, based on the static model, the dynamic system status, and the future plan of the system. The post processor interprets and presents the simulation results (the future states) to the manufacturing operation staffs Thus they can get insights into what will happen for the system, and quickly figure out the best course of actions to handle all kinds of system disruptions.

The controller module receives the optimized results from the post-processor and translates them into executable instructions for controlling the real system. The adapter module interprets the raw data collected from the system, and translates them into meaningful information for representing the current system status, and feeds the real-time system status information into the pre-processor. Both the controller module and the adapter

module need to be customized, based on the existing IT system in the real manufacturing system. (There are more details in a later section).

The modeler module provides a software tool to build the static simulation model, such as the resource model, the manufacturing system layout, the manufacturing process, etc. The next section will describe the modeler in more details.

In addition to the six software modules described above, the proposed system has an object library and a knowledge base. The object library is an extensible collection of reusable software components for the modeling of manufacturing systems. The knowledge base is used for accumulating decision-making knowledge for improving the computation efficiency and solution quality for the simulation-based optimization. We will get into more details on the object library and knowledge base later.

The responsibilities of the four groups of people are listed as follows:

1. The modeler: builds the simulation-based optimization model based on the real system
2. The user: makes use of the simulation-based optimization results for controlling the real system
3. The IT staff: interface the existing IT system with the simulation-based optimization system.
4. The developer: builds and maintain the reusable object library based on the industry's need.

The procedure for implementing such a system into a real manufacturing system is described as follows:

1. The software system is provided with six software modules, a standard object library, and an empty knowledge base.
2. The trained modelers (either employee or consultant of the manufacturing company) build a simulation-based optimization model based on the real system.
3. The IT folks hook up the existing IT system, including the data collection system and execution system, with the simulation-based optimization system.
4. Start up the system: the real-time data are flowing into the system, which continuously simulates the possible future states of the system, automatically performs what-if analysis, and figures out the best courses of actions.
5. The users get benefits from the system. With the real-time simulation-based optimization system, the users will not only know what is going on now, but also what is going to happen in the near future, and what are the actions to control the system for the best possible performance.

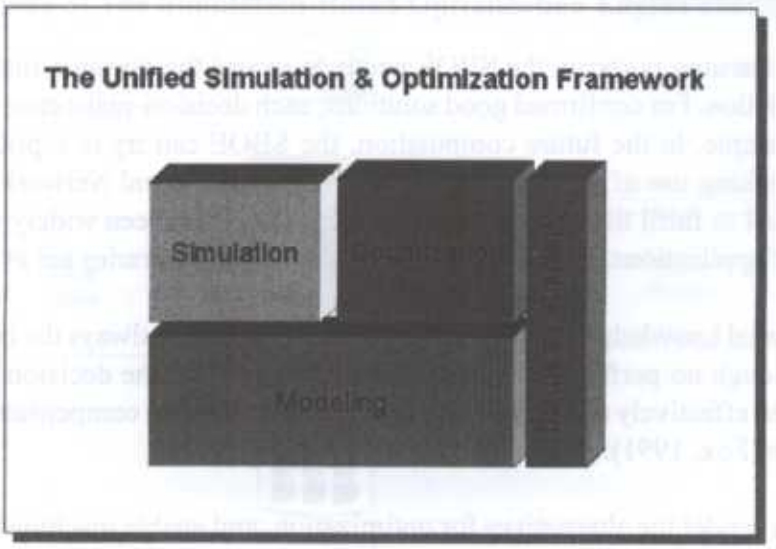
6. The system not only provides useful information and insight to the users, but also helps the users to effectively carry out the actions. The controller software module can translate the action plan into to concrete execution instructions and pass instructions to the execution system.
7. The users are not only information receivers, they are also knowledge contributors. With the post-processor module, the users can not only view the simulation results, but also give feedback (e.g. rating different future system states) based on their professional judgment. The feedback will be gathered into the knowledge base, which will enable the simulation-based optimization engine (SBOE) to make smarter decisions, thus the computation efficiency and solution quality can be improved over time.

The above process is not a single shot activity. Instead, it will be an incremental process. The system begins small. Over time, the model can be fine-tuned, the object library can be extended and customized, the interface with the existing system can be polished, and the knowledge base can grow larger. The system will become more powerful, intelligent, and valuable.

Section 4: MODELING FOR INTEGRATED SIMULATION AND OPTIMIZATION

As discussed in the previous sections, in this study, simulation and optimization are considered as a whole problem, instead of two separated problems. This is illustrated in figure 2, which depicts the relationship among modeling, simulation, optimization, and

Figure 2: The Unified Simulation and Optimization Framework



machine learning. From this picture, we can see that modeling is not only the foundation for simulation and optimization, but also the enabler for machine learning.

For simulation purpose, the major focus of modeling is to specify the resource model, the process model, and facility layout, etc. The modeler software module provides a design environment for the modeler people to conveniently set up the simulation model through easy to use graphical user interfaces (GUI). Basically, they can pick objects from the object library, set up connections between objects, and input the properties for the objects and connections.

For optimization purpose, the simulation and optimization model must specify the possible alternatives. Thus, the simulation-based optimization engine (SBOE) can try different combinations and find the best combination. Instead of trying to make any judgment at the design time, the modelers specify all possible combinations in the model. The SBOE will figure out which combination is good.

Due to the combinatorial explosive nature of the problem, the number of possible combinations could be huge, for example, in thousands, millions, billions, or even more. It is not possible to simulate and evaluate each possible combination. How to quickly find high quality solutions from the huge solution space, is a real challenging problem that has intrigued the scheduling research community into spending a huge amount of effort to tackle it, as discussed earlier.

To get a high quality solution with limited computation effort, the SBOE needs to intelligently choose the most promising combinations to simulate and evaluate, instead of blindly trying out some random combinations. To make the SBOE intelligent, machine learning will be necessary.

For machine learning purpose, the SBOE needs to record the decision-making history for each simulation. For confirmed good solutions, each decision-make case will become a training example. In the future computation, the SBOE can try to replicate the best practice by making use of previous experience. Artificial Neural Network (ANN) is a very useful tool to fulfil the machine learning task. ANN has been widely adopted in a wide range of applications, and many mature ANN program libraries are available.

The accumulated knowledge cannot guarantee each decision is always the best. Here the belief is: although no perfect knowledge can be acquired for the decision making, the knowledge can effectively reduce search effort, while search can compensate for the lack of knowledge (Fox, 1991).

To effectively model the alternatives for optimization, and enable machine learning, the

following decision-making primitive is proposed at the object level:

make_decision *(decision_name)* **among** *(list of alternatives)* based on *(relevant information)*.

This decision-making primitive enables the modeler to specify what kind of decision needs to be made, what are the alternatives, and what information is relevant to this decision making.

This decision primitive enables the SBOE to record sufficient data for machine learning. At the same time, it enables the SBOE to improve the decision-making quality by applying the previously learned decision-making knowledge.

No programming skills are required for the modeler people, and they do not have to be experts in optimization and machine learning. The modeling method proposed in this study provides sufficient flexibility, and meanwhile requires minimal skills and effort.

SIMULATION AND OPTIMIZATION

For simulation-based optimization, there are three important design considerations:

5. Multiple simulations will be performed for different decision-making combinations.
6. It is not practical to simulate all decision combinations, because the number of possible combinations can be very large.
7. Machine learning is essential for simulation-based optimization.

Figure 3 depicts the scope of the SBOE, which is by nature a generic processor. SBOE

Figure 3: The Simulation-Based Optimization Engine (SBOE)

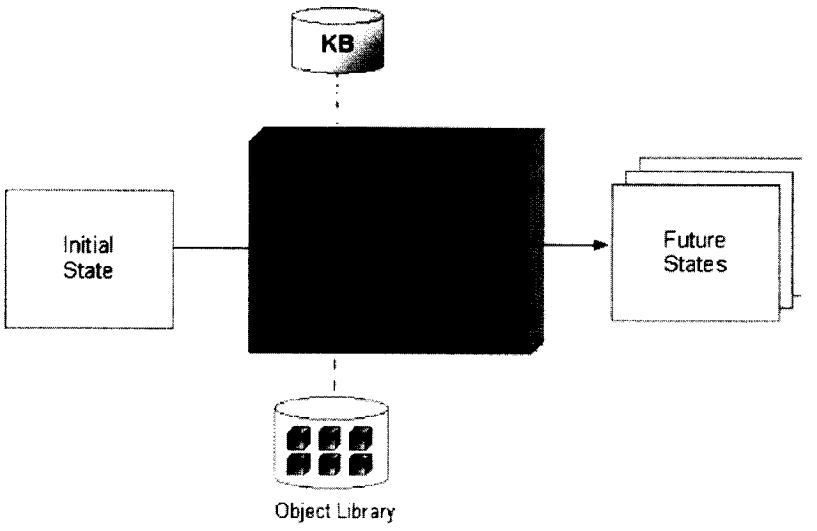
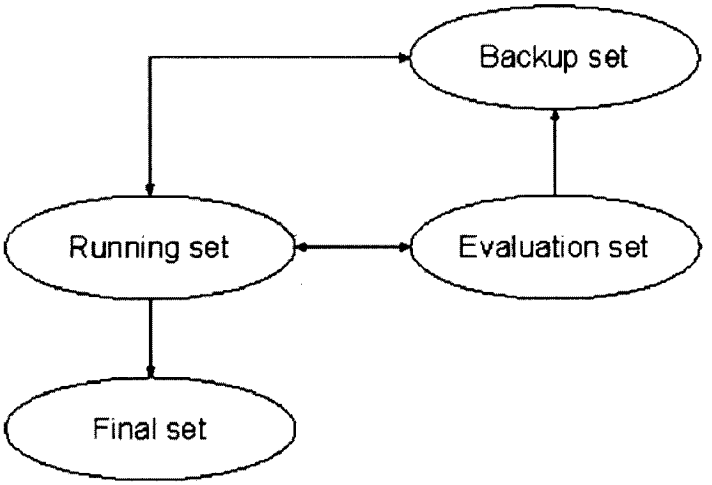


Figure 4: The sets of system states



can simulate the behavior of a complex system by progressively changing the system state from the initial states to the future states. The domain specific logic is programmed in the object library, and the decision-making knowledge is captured into the knowledge base. The captured knowledge can be used to guide the decision-making.

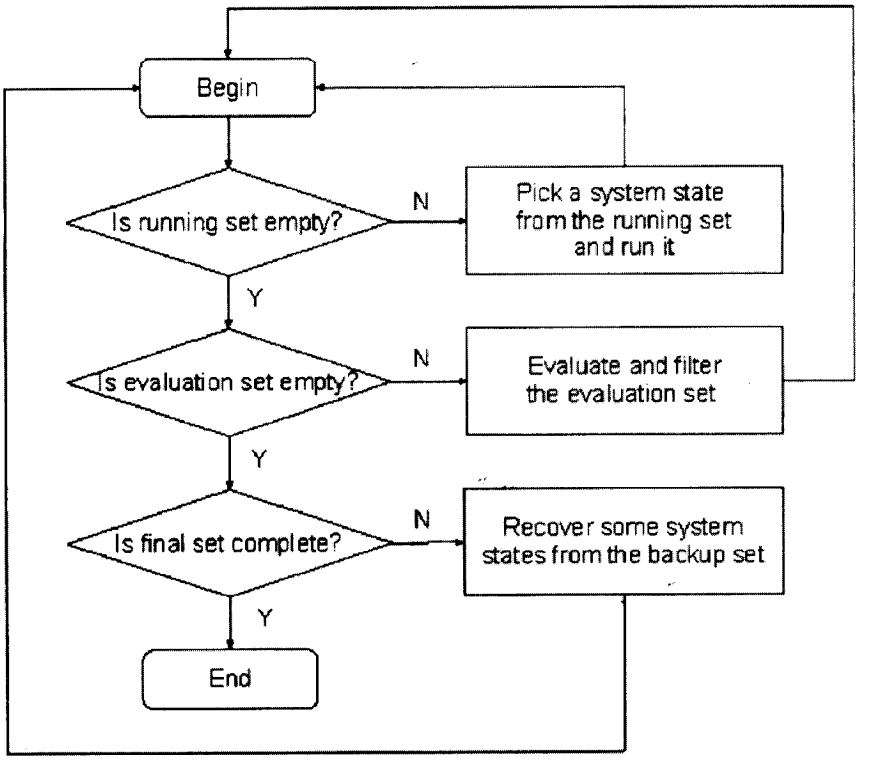
As SBOE simulates multiple decision-making scenarios, the single initial states will be developed into multiple different future states. Therefore, SBOE maintains a population of system states, and manages them with the following four non-overlap sets:

1. The running set: contains the system states for the simulations that are ready to run.
2. The final set: contains the system states of the finished simulations.
3. The evaluation set: contains the system states for the simulations that are waiting for performance evaluation and filtering.
4. The backup set: contains the inferior simulations are going to be discarded. In some cases, some of them may be recycled to the running set.

At the very beginning of the simulation-based optimization, there is only one single system state (the initial state prepared by the pre-processor) in the running set. The control logic of the SBOE is depicted in figure 5. It contains the following steps:

1. Check whether the running set is empty or not
   a) If not, pick a simulation (system state) from the running set and run the simulation; depending on the simulation result, after running the simulation, the system states may be moved to the finished set, the evaluation set, or backup set, or may be destroyed. (Details will be discussed later).
   b) If yes, next step (step 2)
2. Check whether the evaluation set is empty or not
   a) If not, evaluate and compare the performance of the simulations (system

Figure 5: The control logic of SBOE



states) in the evaluation set. For the simulations (system states) with good performance, move them to the running set. For the simulations (system states) with inferior performance, move them to the backup set. If the backup set exceeds the maximum size, discard some of them.

b) If yes, next step (step 3)

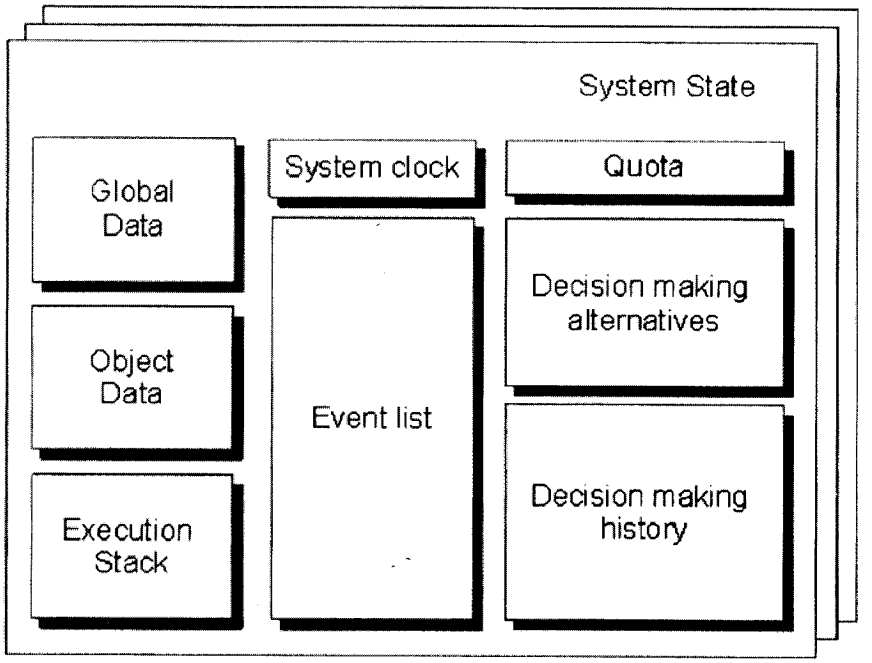
3. Check whether the final set contains enough number of simulations.
   a) If yes, done
   b) Recover some simulations (system states) from the backup set, and go to step 1.

Figure 6 depicts the data structure for storing the system state, which includes three parts:

1. The generic part: the global data, the object data, and the execution stack
2. The simulation part: system clock and event list
3. The decision-making part: quota, the decision-making alternatives, the decision-making history

The first part is to support generic computation, the second part is to support discrete event simulation, and the third part is to support decision-making, machine learning and optimization.

Figure 6: The Data Structure for System State

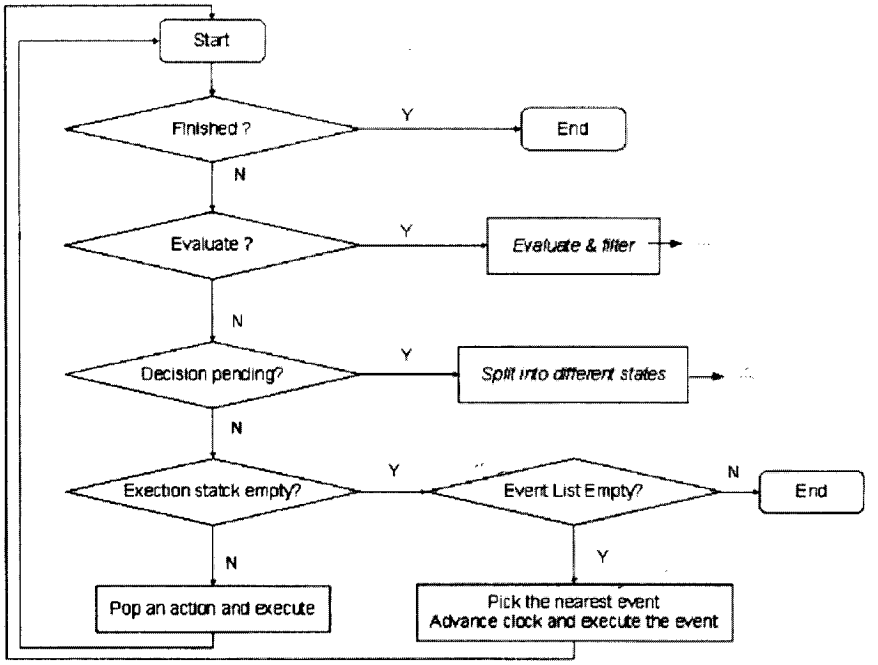


For each simulation instance, its system state contains all the information for running it. The system state can be quickly and easily copied into multiple ones. Thus, at a decision-making point, the SBOE can continue to simulate multiple decision-making scenarios, without having to re-run the simulations from the initial state for each decision-making scenario. This can significantly reduce computation effort.

The working mechanism of the SBOE is illustrated as Figure 7. The main logic includes the following steps:

1. Check whether the system state is marked as finished
   a) If yes, move this system state into the final set, and end this simulation
   b) If no, next step (step 2)
2. Check whether the system state is marked as to be evaluated
   a) If yes, move the system state to the evaluation set, and end the simulation
   b) If no, next step (step 3)
3. Check whether there is a decision needing to be made, by checking whether there are decision making alternatives stored in the decision-making alternatives area.
   a) If yes, then the SBOE will split the current system status into multiple copies. For each copy of system state, push a different alternative action into the execution stack. Move the copied system states in to the running set.
   b) If no, next step (step 2).

Figure 7: the simulation logic of the SBOE



4. Check whether the execution stack is empty.
   a) If yes, go to step 5
   b) If no, go to step 6.
5. Check whether the event list is empty
   a) If yes, end simulation and move the system state into the final list.
   b) If no, pick up the nearest future event from the list, advance the simulation clock to the time of the picked event, and then execute the event. This will call the program code of an object in the object library. After running the object program code, go back to step 1.
6. Pop-up an action from the execution stack and execute the action. Similar to execute an event, this will call the program code of an object in the object library. After running the object program code, go back to step 1.

To hedge the computation effort, a quota allocation mechanism is designed. Each simulation (system state) is assigned a quota, which means the maximum number of this simulation (system states) which can be split from this simulation. At a decision making point, one simulation can be split into multiple simulations, the quota is also split into multiple simulations, but the total number of quota remains unchanged. With the quota mechanism, we can limit the total number of simulations performed, and thus ensure that the computation effort is controlled within an acceptable level.

After performance evaluation, the quota will be reallocated. High quality simulations

will get more quota, inferior simulations will get less quota. Poor quality simulation will be moved to the backup set.

With the above described computation method, the SBOE can perform simulation. Starting from one single initial state, the SBOE can develop multiple final future states. The simulation results will be further processed by the post-processor, which converts them to easily comprehendible format, such as reports, charts, animations, etc., to enable the user's ability to easily get useful information from the simulation and optimization results.

THE OBJECT LIBRARY

We have seen that the SBOE is very generic. The rich functionality and flexibility of the simulation comes from the object library. For each object, it can access the system state through a set of API (Application Programming Interface) functions. At the same time, each object must expose its functionality to the simulation engine by implementing the predefined interfaces. Thus the SBOE can invoke the objects, and the objects can access the system states through the SBOE.

One important thing the object developer needs to keep in mind is each object must keep all its information in the object data area of the system states. The object cannot keep information through the storage mechanism provided by the OO language, such as C++ member variables. This is because the system states can be copied into multiple copies. The storage mechanism provided by the OO language cannot support the system state copy mechanism of SBOE.

With such a design, the object developers have unlimited flexibilities to implement any possible functionality to simulate complex dynamic systems. At the same time, the developer does not need to worry about the optimization and machine learning, as it is implemented at the SBOE level, instead of the object level.

Section 5: THE KNOWLEDGE BASED LEARNING MACHINE

As discussed in the previous sections, each simulation will keep a full record of the decision making history. Each decision-making record includes the following information:

1. The object ID
2. The decision making name
3. The alternative action list
4. The selected alternative action

5. The relevant information for this decision making

The knowledge base is organized as a map; the key of each map item is: "object id + decision name"; the value of the map item is a trained ANN instance. The purpose of the ANN is to learn the association between the related information and the decision making result (the selection of the alternative action).

At the beginning, the KB is empty. By performing simulation-based optimization computation, the KB can be expanded by the decision-making history of the high quality solutions. Through the post-processor, the users can rate the simulation results, and thus the user's professional judgment will affect the training examples, and thus the user's preference can be captured into the knowledge base. Such implicit knowledge is unusually difficult to be explicated formulated. This is one important advantage for the machine learning.

After the KB is trained, the SBOE can make use of the KB to guide its decision making. At each decision making point, the SBOE first locates the ANN by the object ID and decision making name, then it consults the trained ANN on which alternatives are good choices based on the related information. The relative score provided by the ANN for each alternative will be used for filtering the alternatives, and the allocation of quota. Thus, the SBOE can make sure that the computation effort always focuses the most promising possibilities.

INTEGRATION WITH THE EXISTING IT SYSTEM

Although often overlooked by academic researchers, the integration with the existing IT system is an important issue for the real-world implementation of a practical system. In this aspect, the proposed system provides three modules to help with this problem: the pre-processor, the controller and the adapter. When designing the object library, integration with the existing IT system is an important design consideration. The object library provides functionality and flexibility, and at the same time, eliminates the barriers for effective integration of the proposed system with the existing IT system.

CONCLUSION

In this paper, a unified simulation-based optimization framework is proposed, and a reference design of the system is described in detail. The proposed system is featured by the following characteristics:

1. Systematically consider simulation, optimization, and machine-learning into a

unified framework.

2. Object-orient modeling is employed to provide extensibility and flexibility
3. A decision-make primitive is proposed for modeling and machine learning
4. ANN-based machine learning is utilized to improve solution quality and reduce computation effort.
5. Implementation procedures, the responsibility of different groups of related people are explored.
6. Integration with the existing IT system is considered.

At this point of time, this study remains at the prototype stage, more effort is needed to:

1. develop a practicable software solution
2. evaluate and demonstrate its feasibility and performance with some example problems
3. work with the industry to test it out in real-world settings
4. reduce computation time by employing parallel computation for better support of real-time application

The author wishes that this research work could be helpful to academic researchers and industrial practitioners, and that it could promote further research and practice in the simulation-based optimization research area.

REFERENCES

Adams, J., Balas, E., & Zawack, D. (1988). The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science,* 34(3), 391-401.

Aqlan, F., Lam, S. S., & Ramakrishnan, S. (2014). An integrated simulation-optimization study for consolidating production lines in a configure-to-order production environment. *International Journal of Production Economics,* 148, 51-61.

Arakawa, M., Fuyuki, M., & Inoue, I. (2003). An optimization-oriented method for simulation-based job shop scheduling incorporating capacity adjustment function. *International Journal of Production Economics,* 85(3), 359-369. DEL: doi:10.1016/S0925-5273(03)00122-1

Biegel, J., & Davern, J. (1990). Genetic Algorithms and Job Shop Scheduling. *Computers & Industrial Engineering,* 19(1-4), 81-91. DEL: doi:10.1016/0360-8352(90)90082W

Borenstein, D. (2000). Implementation of an object-oriented tool for the simulation of manufacturing systems and its application to study the effects of flexibility. *International Journal of Production Research,* 38(9), 2125-2142. DEL: doi:10.1080/002075400188537

Chang, X., Dong, M., & Yang, D. (2013). Multi-objective real-time dispatching for

integrated delivery in a Fab using GA based simulation optimization. *Journal of Manufacturing Systems,* 32(4), 741-751. DEL: doi:10.1016/j.jmsy.2013.07.001

Dangelmaier, W., Mahajan, K. R., Seeger, T., Klopper, B., & Aufenanger, M. (2006). Simulation assisted optimozation and real-time control aspects of flexible production systems subject to disturbances. In *Proceedings of the 38 conference on Winter simulation* (pp.1785-1795). Winter Simulation Conference. Retrieved from http://dl.acm.org/citation.cfm?id=1218436

Fox, M. (1991). Introduction to artificial intelligence and expert systems. Piscataway, N.J.: Institute of Electrical and Electronics Engineers.

Guo, Y., Chen, L. P., Wang, S. T., & Zhou, J. (2003). A new simulation optimisation system for the parameters of a machine cell simulation model. *International Journal of Advanced Manufacturing Technology,* 21(8), 620-626. DEL: doi:10.1007/s00170002-1380-5

Herrmann, J. W., & Words, K. (n.d.). *A History of Decision-Making Tools for Production Scheduling.*

Hu, H., & Zhang, H. (2012). A simulation-based two-stage scheduling methodology for controlling semiconductor wafer fabs. *Expert Systems with Applications,* 39(14), 11677-11684. DEL: doi:10.1016/j.eswa.2012.04.042

Jacobs, F., & Weston, F. C. (2007). Enterprise resource planning (ERP)-A brief history. *Journal of Operations Management,* 25(2), 357-363. DEL: doi:10.1016/j.jom.2006.11.005

Jain, A. S., & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research,* 113(2), 390-434. DEL: doi:10.1016/ S0377-2217(98)00113-1

Liu, K. C. (1998). Dispatching rules for stochastic finite capacity scheduling. *Computers & Industrial Engineering,* 35(1-2), 113-116. DEL: doi:10.1016/S0360-8352(98)000333

Mahdavi, I., Shirazi, B., & Solimanpur, M. (2010). Development of a simulation-based decision support system for controlling stochastic flexible job shop manufacturing systems. Simulation Modelling Practice and Theory, 18(6), 768-786. DEL: doi:10.1016/j.simpat.2010.01.015

Metan, G., Sabuncuoglu, I., & Pierreval, H. (2010). Real time selection of scheduling rules and knowledge extraction via dynamically controlled data mining. *International Journal of Production Research,* 48(23), 6909-6938. DEL: doi:10.1080/00207540903307581

Min, H.-S., & Yih, Y. (2003). Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system. *International Journal of Production Research,* 41(16), 3921-3941. DEL: doi:10.1080/0020754031000118099

Mouelhi-Chibani, W., & Pierreval, H. (2010). Training a neural network to select dispatching rules in real time. *Computers & Industrial Engineering,* 58(2), 249-

256. doi:10.1016/j.cie.2009.03.008

Peng, C., & Chen, F. F. (1998). Real-time control and scheduling of flexible manufacturing systems: An ordinal optimisation based approach. *International Journal of Advanced Manufacturing Technology,* 14(10), 775-786. DEL: doi:10.1007/BF01438229

Pinedo, M. L. (2012). Scheduling: *Theory, Algorithms, and Systems* (4th ed. 2012 edition.). New York: Springer.

Puergstaller, P., & Missbauer, H. (2012). Rule-based vs. optimisation-based order release in workload control: A simulation study of a MTO manufacturer. *International Journal of Production Economics,* 140(2), 670-680. DEL: doi:10.1016/j.ijpe.2011.09.012

Remenyi, C., & Staudacher, S. (2014). Systematic simulation based approach for the identification and implementation of a scheduling rule in the aircraft engine maintenance. *International Journal of Production Economics,* 147, 94-107. DEL: doi:10.1016/j.ijpe.2012.10.022

Riegler, M., Spangl, B., Weigl, M., Wimmer, R., & M?ller, U. (2013). Simulation of a real-time process adaptation in the manufacture of high-density fibreboards using multivariate regression analysis and feedforward control. *Wood Science and Technology,* 47(6), 1243-1259. DEL: doi:10.1007/s00226-013-0571-6

Shirazi, B., Mahdavi, I., & Solimanpur, M. (2012). Intelligent decision support system for the adaptive control of a flexible manufacturing system with machine and tool flexibility. *International Journal of Production Research,* 50(12), 3288-3314. DEL: doi:10.1080/00207543.2011.574504

Shiue, Y.-R., Guh, R.-S., & Lee, K.-C. (2012). Development of machine learning-based real time scheduling systems: using ensemble based on wrapper feature selection approach. *International Journal of Production Research,* 50(20), 5887-5905. DEL: doi:10.1080/00207543.2011.636389

Smith, S. F. (2005). Is Scheduling a Solved Problem? In G. Kendall, E. K. Burke, S. Petrovic, & M. Gendreau (Eds.), *Multidisciplinary Scheduling: Theory and Applications* (pp.3-17). Springer US. Retrieved from http://link.springer.com/chapter/10.1007/0-387-27744-7_1

Son, Y. J., & Wysk, R. A. (2001). Automatic simulation model generation for simulation-based, real-time shop floor control. *Computers in Industry,* 45(3), 291-308. DEL: doi:10.1016/S0166-3615(01)00086-0

Vakharia, A., & Chang, Y. (1990). A Simulated Annealing Approach to Scheduling a Manufacturing Cell. *Naval Research Logistics,* 37(4), 559-577. DEL: doi:10.1002/15206750(199008)37:4<559::AID-NAV3220370409>3.0.CO;2-8

Widmer, M. (1991). Job Shop Scheduling with Tooling Constraints - a Tabu Search Approach. *Journal of the Operational Research Society,* 42(1), 75-82.

Wu, S., & Wysk, R. (1989). An Application of Discrete-Event Simulation to Online Control and Scheduling in Flexible Manufacturing. *International Journal of Pro-*

duction Research, 27(9), 1603-1623. DEL: doi:10.1080/00207548908942642

Xia, F., Tian, Y.-C., Sun, Y., & Dong, J. (2008). Neural Feedback Scheduling of Real-Time Control Tasks. *International Journal of Innovative Computing Information and Control,* 4(11), 2965-2975.