A Web-Based Platform for the Simulation-Optimization of Industrial Problems

Anna Syberfeldt\textsuperscript{a}\textsuperscript{*}, Ingemar Karlsson\textsuperscript{a}, Amos Ng\textsuperscript{a}, Joakim Svantesson\textsuperscript{b}, Torgny Almgren\textsuperscript{b}

\textsuperscript{a}University of Skövde, Skövde, Sweden
\textsuperscript{b}Volvo Aero Corporation, Trollhättan, Sweden

Abstract

This study presents a platform for industrial, real-world simulation-optimization based on web techniques. The design of the platform is intended to be generic and flexible and thereby make it possible to apply the platform in various problem domains. In the implementation of the platform, modern web techniques, such as Ajax, JavaScript, GWT, and ProtoBuf, are used. The platform is tested and evaluated on a real industrial problem of production optimization at Volvo Aero Corporation, a company that develops and manufactures high-technology components for aircraft and gas turbine engines. The results of the evaluation show that while the platform has several benefits, implementing a web-based system is not completely straightforward. At the end of the paper, possible pitfalls are discussed and some recommendations for future implementations are outlined.

Keywords: Simulation; Optimization; Web; Industrial Case Study.

1. Introduction

The use of a simulation in combination with an optimization algorithm, so called simulation-optimization, is a commonly used technique in industry today, to find the best parameter values for a system or a process. Simulation applications have traditionally been provided as desktop systems, but during the last few years the interest for web-based simulation has grown. “Web-based simulation” does not have an exact meaning, but is a broad term including various approaches to integrate the web with the field of simulation. Byrne (2010) states that web-based simulation “... can be defined as the use of resources and technologies offered by the World-Wide-Web (WWW) for interaction with client and server modelling and simulation tools”. Page et al. (2000) provide another definition, stating that the web-based simulations “... denote the invocation of simulation programs over the Internet, specifically through a web browser”.
Compared to classical desktop systems, several advantages with web-based systems can be identified:

- **Accessibility** - a web-based system is accessible from anywhere with an internet connection, and not only from the specific computer that has the simulation system installed. This also means that a web-based system allows access outside normal business hours.

- **Scalability** - web-based systems allow for the dynamic provision of computing resources (Rabinovich and Spatscheck, 2002) and are thereby able to handle an increasing number of simulation/optimization requests without performance degradation.

- **Portability** - a web-based system can be run in any web browser on any operating system, without requiring recompilation (Suh, 2005). It is not only limited to a traditional computer, but can be run on any device that has a web browser (e.g., a mobile phone or an iPad).

- **Maintenance** – the maintenance of web-based systems is easier, since they do not have to be installed in each client’s computer. Updates are made through a server and reach the clients instantly, eliminating virtually all on-site maintenance and allowing for a frequent update scheme.

- **Controlled access** - through user logins, a web-based system allows for the configuration of user groups with different privileges based on work tasks. The privileges can be easily changed on the server, instead of updating the client computer.

- **Licensing** - simulation software licenses are usually very expensive, and a traditional simulation system is often required to have one license installed in each computer that runs the system. With a web-based system, the number of licenses can be significantly reduced, since the simulation is located in a common infrastructure.

Although web-based simulation has obvious advantages and is a concept that has been discussed in the simulation research community for 15 years, the field is still in its infancy (Fishwick, 1996; Byrne, 2010). While the field has started to grow during the last few years, there are still only a small number of real-world applications of web-based simulation (Byrne, 2010). There are even fewer studies of integrating optimization with a web-based simulation system; less than a handful can be found in the literature. This study aims to add to this number, by presenting a new web-based platform for simulation-optimization. Contrary to existing, similar platforms, the one proposed specifically targets industrial, real-world problems. Since the design of the platform is generic and flexible, it is thereby possible to apply to a wide range of problem domains. In the implementation of the platform, modern web techniques, such as Ajax, JavaScript, GWT, and ProtoBuf, are used. The platform has been tested and evaluated on a real industrial problem of production optimization at Volvo Aero Corporation, a company that develops and manufactures high-technology components for aircraft and gas turbine engines. The results of the evaluation show that while the platform has several benefits, implementing a web-based system is not completely straightforward.

The remainder of the paper is organized as follows. Chapter 2 presents existing web-based platforms for simulation-optimization that have been used as reference and inspiration. Chapter 3 provides a description of the design and implementation of the new platform. The application and evaluation of the platform on the industrial optimization problem at Volvo Aero Corporation is presented in Chapter 4, which is followed by an analysis in Chapter 5. Finally, Chapter 6 summarizes the conclusions from the study.

2. Related work

This chapter presents existing web-based platforms for simulation-optimization, described in the literature, and discusses their strengths and weaknesses.

2.1 Existing web-based platforms for simulation-optimization

A review of the literature reveals only a few papers that discuss web-based simulation-optimization. An early study on the subject was presented by Luo and his colleagues in 2000 (Luo et al., 2000) and describes a general platform for web-based simulation-optimization that is based on Java. The platform consists of three parts: a web browser, a web server, and a number of parallel simulation clients (Figure 1, freely from Luo et al., 2000).
The user interface is the Java program started by a web-browser. Through the user interface, it is possible to set up, run, and view simulation-optimization experiments. Optimizations are performed using an algorithm called Sequential Algorithm for Optimal Computing Budget Allocation (OCBA) and run by an optimization manager. The optimization manager, which is located at a web server, is also responsible for distributing simulation jobs among the clients and collecting the results.

Luo et al. evaluate their platform on a number of theoretical queuing problems. There are two main objectives of the evaluation: a) to study the reduction in execution time accomplished by the distributed client, and b) to study the network delay caused by communication with the clients. The results of the evaluation show that the total execution time decreases with an increased number of clients, but the network traffic significantly impedes the results when the clients are distributed over the internet.

A more recent study on web-based simulation-optimization is presented by Yoo et al. (2009). The design of this platform, which is implemented partly using Java and partly using the .NET framework, is shown in Figure 2 (freely from Yoo et al., 2009). The platform is similar to the one proposed by Luo et al. (2000) in that both implement an OCBA-based optimization algorithm and a distributed architecture for parallel simulation evaluations. Two main differences are: a) Luo et al. use a web page as user interface instead of a Java program, and b) Lou et al. utilize a database for storing and retrieving optimization results, which is not done by Yoo et al. (at least not described in the paper).

The optimization manager holds an intelligent algorithm that evaluates feasible regions within the search space, chooses the most promising region, and partitions the search space. The results are stored in a central database and made available to the simulation clients. Contrary to the platform described by Lou et al (2000), the simulation clients have a certain degree of intelligence by holding an OCBA algorithm. The OCBA algorithm performs experiments at the simulation client, based on the information provided by the optimization manager. The results from the simulation clients are stored in the central database, which becomes the single point of interaction between the optimization manager and the simulation clients. The database has a web interface which allows communication over the internet.
Yoo et al. (2009) evaluate their proposed platform using a theoretical AGV control problem. However, the evaluation is only performed with respect to optimization algorithm behavior, and only a general discussion of the web-based aspects of the platform is provided. In this discussion, the authors speculate that the performance and robustness of the platform might be affected by factors such as network speed and network reliability.

While both the platform described by Yoo et al. and the one described by Luo et al. have been developed for theoretical test problems, Andersson et al. (2007) present a web-based platform for the simulation-optimization of a real industrial problem (Figure 3, freely from Andersson et al., 2007). This platform is completely implemented using the .NET framework and, compared to the former ones, mainly differs in that the user interface is in the form of a Windows application. However, similar to the former platforms, Andersson et al. utilize distributed simulation clients for parallel evaluations. In contrast to Yoo et al., but similar to Luo et al, the simulation clients are slaves without intelligence. All optimization functions are held by an optimization manager, which performs optimizations by running a genetic algorithm. Solutions to be evaluated are sent to the simulation manager, which distributes jobs among the clients and notifies the optimization manager of the results. A database also stores the results which are accessed by the user interface.

Andersson et al. evaluate their platform on a scheduling problem within the manufacturing industry where the production line of 34 machines to be optimized produces cam shafts. The evaluation solely focuses on the optimization algorithm and there is no discussion of the web-based aspects.

2.2 Discussion

Both similarities and differences can be found in a comparison of the three existing platforms. The most important features identified are listed and briefly analyzed below. The objective is to re-use good ideas from the existing platforms in the design of the new platform for web-based simulation-optimization.

- **User interface** – using a web page inside a web browser as user interface, which Yoo et al., 2009 adopt, is advantageous with regard to accessibility and portability. Although neither Lou et al. nor Andersson et al. use this concept, it seems to be a natural choice for a web-based platform.

- **Simulation clients** – all three existing platforms implement multiple, distributed simulation clients to perform evaluations in parallel. This significantly reduces the total execution time and is, without doubt, necessary for efficient optimizations. Indeed, Yoo et al. state that distributed clients cause network traffic delays, but considering the improvements in technology since their study was published (more than ten years ago), the problem is no longer expected to be as great.

- **Database** – using a database for storing optimization results, as Lou et al. and Andersson et al. employ, supports the convenient preservation and management of data. The use of a database is easily motivated, especially with large amounts of data.
Computers & Industrial Engineering

- **Operating system** – the platforms described by Yoo et al. and Andersson et al. are both based on .NET and only run in Windows environments. However, the platform Lou et al. present is based on Java and therefore runs on multiple operating systems. Since operating system independence enables a broader adoption and eliminates lock-in, this should be strived for.

- **Modularity** – the platform presented by Andersson et al. is divided into five separate modules, while Lou et al. and Yoo et al. use a simpler design with only three modules. Since the separation of a system into modules increases the manageability of complex software, a high degree of modularity is advantageous.

- **Optimization algorithm** – the platforms described by Lou et al. and Yoo et al. implement the OCBA algorithm for performing optimizations, while the platform described by Andersson et al., implements metaheuristic algorithms. Since specific algorithms are suitable only to specific problems, supporting only one algorithm, or class of algorithms, is a limitation considering both generality and efficiency.

Besides the features discussed above, two additional aspects have been identified as important:

- **Security** – web-based platforms send data over networks and it is therefore of critical importance that this data is protected from third party access, especially in industrial applications.

- **Users** – in real-world scenarios, multiple users must be a consideration, and it is therefore important for the platform to be able to handle users, their accounts, and their privileges.

Bearing in mind the features and aspects discussed above, a new web-based simulation-optimization platform has been developed in this study and is described in the next chapter. The new platform is specifically intended for industrial adoption and real-world optimization.

### 3. A new web-based platform for industrial simulation-optimization

This chapter presents a description and discussion of the design and technical implementation of the proposed web-based simulation-optimization platform. A reference implementation of the platform, complete with source code, is provided at [http://sourceforge.net/projects/iwsop/](http://sourceforge.net/projects/iwsop/).

#### 3.1 Design

The overall design of the platform is presented in Figure 4. In order to achieve a high cohesion and manageable complexity of the software, the platform is divided into four components: a) user interface, b) optimization interface, c) optimization system, and d) simulation system.

![Fig. 4. Design of proposed web-based platform for simulation-optimization.](image)

---

---
The user interacts with the platform through a web page displayed in a web browser (e.g., Internet Explorer, Chrome, Mozilla Firefox, or any other browser). Using a web page as user interface enables access to the platform from any hardware (e.g., PC, mobile phone, tablet, etc.) at any site, as long as a web browser and an internet connection are available. This is especially advantageous in industrial environments, since employees often have limited (or no) possibilities of installing, running, and maintaining software at their work stations. Furthermore, it also enables access to the web page from outside the company network.

The platform is designed to serve multiple users simultaneously, in order to support an industrial context in which different employees perform various tasks at the same time. Such tasks might, for example, include initiating optimizations, controlling optimization progress, viewing optimization results, or analyzing previous optimization runs. Access to the various functions is controlled and a command can only be executed if the user has the privileges to do so. A manager, for example, may be allowed to execute all commands, while a machine operator is only allowed to view the latest optimization result. To distinguish between different users, the platform implements a system that authorizes users by user names and passwords. The authorization system also prevents unauthorized access, which is of critical importance since the platform holds sensitive company data.

User input is processed by the optimization interface, which is the second component of the proposed platform. A command from the user interface is received by a web application, which controls user privileges in an authentication system before reacting to the command. The authentication system retrieves privileges from a database that stores user accounts and information about which user (or user group) is authorized to execute which commands. When a command is sent by a logged-in user with the correct privileges, it is forwarded from the web server to the optimization system, which is the third component of the platform.

The optimization system is usually co-located with the optimization interface (although it is possible to separate them, the authors see no reason in doing so). A request from the web application to start an optimization, or to retrieve information about previous optimizations, is processed by the optimization system which performs an optimization by executing an optimization algorithm. The platform supports the parallel execution of multiple algorithms, in order to support the fact that many industrial scenarios need to be able to run several optimizations simultaneously (e.g., for short- and long-term planning, or to compare the effect of different input configurations). No specific algorithm is defined in the platform, but any algorithm suitable for the problem at hand is possible to implement. All optimization results are stored in a database from which they can be retrieved over time. In an industrial environment, viewing historical results is often of interest with regard to analyzing the past and making predictions for the future. Historical results can also be used by the optimization algorithm to speed up the optimization process, for example, by discovering common patterns in the data.

The simulation system, as the fourth component of the proposed platform, is responsible for evaluating solutions generated by the optimization algorithms. Evaluation requests sent to the simulation system are received by a simulation server, which distributes evaluation jobs to a number of parallel simulation clients. Performing evaluations in parallel is an important means of achieving an acceptable execution time, especially with regard to industrial optimization problems. These often require hundreds or thousands of evaluations before an acceptable solution is found and one single evaluation may take a couple of minutes of computing time. In order to further enhance an increase in efficiency, the simulation manager always assigns a job to the available simulation client that has the most computing power. When a job has been completed, the simulation client notifies the simulation server, which returns the result to the optimization system.

**3.2 Implementation**

The software techniques used to realize the platform are presented in Figure 5. As the figure illustrates, a web browser connects to a web server through HTTPS (Hypertext Transfer Protocol Secure), to retrieve the web page that constitutes the user interface. In order to prevent eavesdropping, all communication between the web browser and the web server is encrypted using Secure Sockets Layer (SSL). SSL also provides a certification mechanism that is implemented in the platform, to ensure that the web page shown to a user is genuine. Encryption and certification, in combination with a login system, provides the platform with a safeguard against tampering and unauthorized access by a third party.
The web server software Apache Tomcat\(^a\) has been chosen mainly because it is possible to run the software on multiple operating systems and using it is free of charge. The web server is configured to accept requests from the web browser only if the user is logged in and approved by the authorization system. To avoid the need to approve users by user name and password at every request, the authorization system creates a unique hash sum when a user logs in that is used to directly verify the user’s following requests. The hash sum is kept in the web browser as a session cookie and automatically included in every request to the web server.

A web page requested by the web browser is dynamically generated by the web server to enable an interactive user interface. The dynamic generation of web pages is implemented using a Java servlet, which is basically a Java class run in a Java servlet container. The Java servlet is based on the Google Web Toolkit (GWT)\(^b\), which is an open source framework for creating complex web pages for those without expertise in JavaScript or browser technology. With GWT, the web page is written in common Java code that is automatically translated into Java script and HTML adjusted to the specific web browser requesting the web page. To enable interaction, GWT utilizes Ajax (Asynchronous JavaScript and XML). Instead of reloading the whole web page when a partial change occurs, Ajax supports the reloading of the changed parts only, by asynchronous information exchange between the web browser and the web server. This significantly reduces the response time of the web page and enables frequent user interaction and frequent updates.

The exact content of the web page is adjusted according to the specific user being logged in. As previously mentioned, work tasks and privileges vary between different user groups and the web page is adjusted accordingly. All information about users and user groups is stored in a PostgreSQL database\(^c\), from which it is retrieved by the authentication system. PostgreSQL was chosen on the basis of its well-known stability and the fact that the use of the software is free of charge. Three database tables are used to store user information, as shown in Figure 6. The users’ passwords are encrypted using the SHA-2 hash function and stored in this format to ensure that they are unusable if stolen.

\(^a\) http://tomcat.apache.org/
\(^b\) http://code.google.com/webtoolkit/
\(^c\) http://www.postgresql.org/
Fig. 6. Database scheme.

A user’s privileges, in combination with the type of request, determine how to populate the web page. A request can be either to control (i.e., start or stop) an optimization, or to retrieve optimization results. All requests are forwarded from the web application to the optimization server. The communication between the web application and the optimization server (and also between some of the other components of the platform, as shown in Figure 5) is implemented by sending messages that are structured using Google Protocol Buffers (ProtoBuf). ProtoBuf is a serialization format for information exchange that supports multiple programming languages, thus enabling messages to be sent between applications written in different programming languages. This feature makes it possible to implement the various components in the platform using different programming languages and to run them on different operating systems.

At the optimization server, an optimization manager determines how to act upon a request from the web application. If a new optimization is to be started, the optimization manager initiates an optimization algorithm. The platform supports the parallel execution of multiple, potentially heterogeneous, algorithms by multi-threading the optimization server. Simulation evaluations needed in the optimization process are sent to a simulation manager, which utilizes a job distributor to assign simulation jobs to the simulation clients. The job distributor monitors the simulation clients’ attributes (i.e., location, computing power, installed software, etc.) and the platform supports the use of heterogeneous clients with different hardware and/or software installed. For increased efficiency, the job distributor makes sure that a job is only assigned to a client with the necessary software installed and that a job is always assigned to the available client with the most computing power. When a job has been assigned to a client, a simulation controller starts the simulation software and initiates a simulation according to the job description. When the simulation has been completed, the simulation controller reports the result back to the simulation manager. If a result has not been received within a reasonable time period, the simulation manager assumes that the simulation has crashed. The simulation software is then restarted and the job is sent once again. This functionality ensures that the simulation manager does not wait forever for a result to return and that simulation clients are never locked due to malfunctioning software. As soon as a simulation has been completed, the simulation manager notifies the optimization server. At the optimization server, the optimization algorithm uses the result to proceed in the process of generating iteratively improved solutions. The result is also stored in a PostgreSQL database, to be retrievable over time.

When an optimization is finished, the results are sent to the web application and the web page is updated. Results cannot only be shown for terminated optimizations, but also for those currently running. Using the dynamic behavior enabled by the Ajax technology makes it possible to continuously show optimization progress to the user without needing to reload the web page.

In the next section, the characteristics of the proposed platform are compared to existing web-based platforms for simulation-optimization.

3.3 Comparison with existing platforms

A number of differences both in the design and in the technical implementation can be found if comparing the proposed platform with the three existing web-based platforms for simulation-optimization presented in Chapter 2. One difference is that the proposed platform defines an architecture in which the optimization and the simulation are separated on different networks, instead of being co-located. The motivation behind the separation is an increased flexibility and the possibility to effectively utilize computing resources at different physical locations, for example...
by using the cloud computing paradigm. Furthermore, a separation also supports a distributed replication of the simulation and the optimization that improves fault-tolerance and accessibility.

Another difference between the proposed platform and the existing platforms is that no specific optimization algorithm(s) is implemented in the former, but instead any algorithm suitable for the specific problem can be used. The open choice of algorithm makes the proposed platform generally applicable to all types of optimization problems and allows the best possible optimization procedure to be implemented in each unique scenario.

A further difference is related to information security. The proposed platform implements encryption and security certificates to protect eavesdropping and hostile acts, and such preventive security measures are not discussed in any of the existing platforms. Security is an important issue in all industrial scenarios and especially crucial in web-based systems since these usually send information over public networks.

Another difference between the proposed platform and the existing ones of significance in industrial scenarios is the cost. Many companies have tight economic margins and can allocate only small amounts of money for installing and using new software. Contrary to the existing platforms, the proposed platform is explicitly designed with this in mind and only includes software that is free of charge. For example, the web server utilized in the proposed platform is based on Apache Tomcat and the database on PostgreSQL – open source software available without cost also for companies. Source code for the complete platform is also publically available on the internet in order to minimize the development cost. Such reference implementation is not provided by any of the existing platforms.

4. Industrial case study

In order to evaluate the proposed platform for web-based simulation-optimization, it has been implemented and tested on a real industrial problem of simulation-optimization at Volvo Aero Corporation. This chapter describes how this has been realized.

4.1 Case study overview

Volvo Aero develops and manufactures high-technology components for aircraft and gas turbine engines. Today, more than 80 percent of all new commercial aircraft with a capacity of more than 100 passengers are equipped with engine components from Volvo Aero. Components manufactured at Volvo Aero can also be found in military fighter aircraft, such as the F/A-18 E/F Super Hornet. As a partner in the European space program, Volvo Aero is also the primary supplier of nozzles and fuel pump turbines for the Vulcain rocket engine. Volvo Aero’s facilities, located both in Scandinavia and in the US, have a total of approximately 3 200 employees. In this work, a factory located at the headquarters of Volvo Aero Corporation in Sweden is the subject under study.

More precisely, the focus of the study is on an advanced, automated manufacturing cell comprising five multi-task machines, two burring stations, and three manual stations for work such as arranging components in fixtures. The sequence of operations performed, as well as operation time and resource demands, in a machine or at a station vary for different components. Instructions and tools are automatically set up in a machine for the component that arrives, which means that several different components can easily be processed in the cell at the same time.

The machines in the cell, as well as the components being processed, are capital-intensive and it is therefore important to achieve high utilization of the cell. However, finding an efficient processing schedule is not trivial, due to the considerable complexity of the cell in combination with a fluctuating inflow and resource constraints. This fact has raised a need to perform automatic simulation-optimizations and has motivated an evaluation of the proposed web-based platform. Details which describe how this has been realized are presented in the following subsections, starting with a presentation of the user interface.

4.2 User interface

As described in Chapter 3.1, the user platform interface is in the form of a web page based on HTML, JavaScript, and Ajax. The content of the web page is not, and cannot be, generic in the platform, but must be adjusted according to each unique application scenario. In the Volvo Aero case, the content was developed during an iterative process undertaken in close cooperation between the company and the software developers. A screenshot of one part of the resulting web page is presented in Figure 7 (for integrity reasons, company specific information was deleted before the screenshot was taken). The panel to the left in the screenshot constitutes the menu from which the user accesses
various functions in the platform. The content of the menu is adjusted according to the user group of the specific user being logged in: a) operator, b) shift leader, or c) manager. In the screenshot, a manager is being logged in and all functions are therefore accessible via the menu. If, instead, a shift leader is logged in, the “ADMIN” alternative will not be visible, since this user group is not allowed to administrate users. An operator has the least number of privileges and does not see anything except the “RESULTS” alternative. This alternative (the one chosen in the screenshot) shows the output from a simulation-optimization run, namely, a production schedule derived from a prioritization of all components to be processed during a specific time period. The aim of the production schedule is to support the operators of the manufacturing cell by specifying which component should be processed in which machine at each point in time.

Before being able to generate a production schedule (i.e., to start a simulation-optimization), the user must first manually specify the current status and configuration of the manufacturing cell. This is done using the “STATUS” menu alternative. The required data includes shift period, status of components currently under process, list of components entering, availability of fixtures, availability of operators, and scheduled maintenance. The simulation model needs all this data to be able to mimic the operations of the cell as close to reality as possible. When the data has been specified, the simulation-optimization is started using the OPTIMIZATION alternative.

4.3 Simulation-optimization

Like the user interface, neither the simulation nor the optimization are generic in the platform, but must be implemented according to the specific requirements of the problem to be solved. In the case study at Volvo Aero, the problem is to find the best prioritization of the different engine components being simultaneously processed in the manufacturing cell. The prioritization is used to determine which specific component has precedence when two
or more components are available at a machine or station. Based on the priorities, it is possible to create a schedule showing which component should be processed in which machine at each point in time.

The aim of the optimization is to set the priority numbers of the components in a way that 1) minimizes tardiness (overdue components) and 2) simultaneously minimizes lead times (the time between a component entering and exiting the manufacturing cell). The company identifies these two optimization objectives as being important in order to achieve high utilization of the manufacturing cell.

To optimize priorities, an iterative simulation-optimization process is being used (Figure 8). In this process, an optimization algorithm run at the optimization server generates a set of priority values (input parameters) and feeds them to a simulation model run at the simulation server. The simulation model computes tardiness and lead times of components in the manufacturing cell, and sends these objective values back to the optimization server. Based on the evaluation feedback obtained from the simulation model, the optimization algorithm generates a new set of priority values and the generation-evaluation process continues until the maximum run time is reached. In the case study, the company has determined the maximum runtime to 10 minutes. This relatively short runtime is due to the need of running a new optimization at each shift change.

![Simulation-optimization process](image).

Fig. 8. Simulation-optimization process.

The simulation model of the manufacturing cell is based on the discrete-event paradigm and has been developed by simulation experts at the company using the SIMUL8 software package. Validity tests indicate that the simulation model represents reality well, and the model is generally accepted among operators working in the manufacturing cell.

To run the simulation model in the platform, all simulation clients are equipped with the SIMUL8 software. The simulation controller, whose purpose is to start the simulation and collect results, is implemented using the application programming interface (API) provided by SIMUL8. This API enables the simulation model to be controlled programmatically. When the simulation controller receives a job (i.e., a set of priority values to be evaluated) from the job distributor, the simulation controller invokes the simulation model with a “Run” command and provides the priority values. When the simulation has been completed (after about two seconds), the simulation controller collects the results and sends them back to the simulation server.

In the next chapter, details of the optimization are presented.

5. Optimization

While traditional, analytical optimization methods have been unable to cope with the challenges imposed by many simulation-optimization problems in an efficient way, such as multimodality, non-separability and high dimensionality, so called metaheuristic algorithms have been shown to be applicable to this type of problem (Boesel et al., 2001; Laguna and Marti, 2002). Metaheuristic algorithms are powerful stochastic search algorithms with mechanisms inspired from natural science. A metaheuristic algorithm optimizes a problem by iteratively improving

---

www.simul8.com
Computers & Industrial Engineering

one or several candidate solution(s) with regard to a given objective function. The algorithms are not guaranteed to find the optimal solution for a given problem, but are instead computationally fast and make no explicit assumptions about the underlying structure of the function to be optimized. These properties have motivated their usage for solving the optimization problem at Volvo Aero, as the maximum run time in this case is short (10 minutes) and the simulation model is a black-box from an optimization point of view.

5.1 Algorithms

Three different metaheuristic algorithms are being considered for solving the optimization problem at Volvo Aero. The first algorithm is Hill Climbing, which is selected since it is well-known and simple to implement. The second algorithm is NSGA-II, which is selected since it is a state-of-the-art benchmark algorithm for problems involving multiple optimization algorithms (Deb et al., 2004; Coello Coello et al., 2007; Tsuchida et al., 2009). The third algorithm is Cuckoo Search, which is selected since it is a newcomer which has shown successful results in several research studies (Yang and Deb, 2010; Tein and Ramli, 2010; Dhivya et al., 2011). The three algorithms are presented in detail in the following of this chapter.

5.1.1 Hill Climbing

Hill Climbing is an iterative algorithm that belongs to the family of local search algorithms (Russell and Norvig, 2003). The algorithm starts with a random solution to the problem and attempts to find a better solution by mutating (changing) the solution. If the mutation produces a better solution, it is kept and the procedure is repeated until no further improvements can be made.

In the basic version, Hill Climbing only allows a single objective to be optimized. However, in the case study, there were two optimization objectives and the basic algorithm was therefore modified to take this into consideration. This is done by optimizing one objective at a time, without deteriorating the other objective. More precisely, the algorithm starts by optimizing objective 1 (tardiness) and when 15 mutations have been performed without finding any better solution with respect this objective, the algorithm instead focuses on objective 2 (lead times). The number 15 was found through extensive trial-and-error experiments and this number was shown to produce the best results. It should be noted that the optimization of objective 2 is constrained in the sense that a solution improving objective 2 is only accepted if it does not deteriorate objective 1. The whole procedure is described with pseudo code in Figure 9.

```plaintext
obj = 1
obj1Value = undef
numTries = 0
current = random solution
EvaluateObjectiveValue(current)

loop until the user-defined time is up
    candidate = Mutate(current)
    EvaluateObjectiveValue(candidate)
    if obj is 1 AND candidate < current
        current = candidate;
    else if obj is 2 AND candidate < current AND obj1Value is preserved
        current = candidate
    else
        numTries = numTries+1
    end if
    if numTries > 15
        obj = 2;
        obj1Value = candidate
    end if
end loop
```

Fig. 9. Pseudo code for Hill Climbing algorithm.
5.1.2 NSGA-II

NSGA-II (elitist non-dominated sorting genetic algorithm) is a population-based evolutionary algorithm explicitly supporting multiple optimization objectives (Deb et al., 2002). Instead of only seeking a single optimum, NSGA-II maintains a set of optimal trade-offs, called Pareto-optimal solutions. The algorithm starts with a random population of solutions, and during each generation, a proportion of the best solutions in the population is selected to breed offspring for the next generation of the population. The selection of solutions to continue to the next generation of the population is done from a set $R$, which is the union of a parent population and an offspring population (both of size $N$) (Deb et al., 2000). Non-dominated sorting is applied to $R$ to identify Pareto fronts, and the next generation of the population is formed by selecting solutions from one of the fronts at a time. The selection starts with solutions in the best Pareto front, then continues with solutions in the second best front, and so on, until $N$ solutions have been selected. If there are more solutions in the last front than there are remaining to be selected, niching is applied to determine which solutions should be chosen. In other words, the highest ranked solutions located in the least crowded areas are the ones chosen. All the remaining solutions are discarded. The selection procedure is illustrated in Figure 10 (adopted from Deb 2004).

![Fig. 10. Selection procedure in NSGA-II.](image)

Pseudo-code for the NSGA-II algorithm can be found in Figure 11 and for details about the algorithm the reader is referred to Deb et al. (2002).

```pseudo
population = random solutions

loop until the user-defined time is up
    EvaluateObjectiveValues(population)
    parents = Select(population)
    children = Breed(parents)
    Mutate(children)
    combinedPopulation = Merge(population + children)
    NonDominatedSorting(combinedPopulation)
    population = SelectBest(combinedPopulation)
end loop
```

Fig. 11. Pseudo code for NSGA-II algorithm.

5.1.3 Cuckoo Search

Cuckoo Search is a recently proposed population-based evolutionary optimization algorithm (Yang, 2010). The algorithm is inspired by the parasitic breeding of cuckoos and utilizes the behavior of these birds to traverse a search space. Sets of nests with an “egg” (candidate solution) inside are placed at random locations in the search space, and a number of “cuckoos” traverse the search space and records the highest objective values for different encountered candidate solutions. When traversing the search space, the cuckoos utilize a search pattern called Lévy flight. This pattern is encountered in real birds, insects, grazing animals and fish (Viswanathan et al., 1999). Pseudo-code for the
Cuckoo Search algorithm can be found in Figure 12 and for details about the algorithm the reader is referred to Yang (2010).

Since Cuckoo Search is a single-objective algorithm, it has to be slightly modified for the case study in order to be able to handle two optimization objectives. Basically, the same procedure as in the Hill Climbing algorithm is being used in which one optimization objective at a time is optimized. Objective 1 (tardiness) is optimized until 25 Lévy flights (tries) have been performed without finding any better solution with respect this objective. When this happens, the algorithm instead focuses on objective 2 (lead times). The number of tries (25) was found through trial-and-error experiments.

```
population = random host nests
obj = 1
obj1Value = undef
numTries = 0

loop until the user-defined time is up
    candidate = PerformLévyFlight
    EvaluateObjectiveValue(candidate)
    current = SelectOneNestRandomly(population)
    if obj is 1 AND candidate < current
        current = candidate;
    else if obj is 2 AND candidate < current AND obj1Value is preserved
        current = candidate
    else
        numTries = numTries+1
    end if
    if numTries > 25
        obj = 2;
        obj1Value = candidate
    end if
    DeleteWorstNest(population)
    BuildNewNests
end loop
```

Fig. 12. Pseudo code for Cuckoo Search algorithm.

5.2 Comparison of algorithms

The three algorithms are applied to Volvo Aero’s optimization problem and their performance is compared. The runtime allocated for the simulation-optimization procedure is 10 minutes (specified by the company). In both NSGA-II and Cuckoo Search the population size is set to 20. Furthermore, NSGA-II is implemented with a partially mapped crossover operator, swap range mutation, and a mutation probability of 10%. Cuckoo Search is implemented with a swap operator utilizing Lévy flights to determine the indices to swap, and the percentage of abandoned nests is set to 40%. All parameter settings have been found through extensive trial-and-error experiments.

As baseline for comparing the three algorithms, a basic scheduling function defined by the company is being used. In this function, a Critical Ratio (CR) value of each component is calculated to determine how well it is on schedule. The CR value is derived by dividing the time to due date (i.e. scheduled completion) by the time expected to finish the component, according to Equation 1.
In this Equation, due is the due date of the component (i.e. deadline), now is the current time, and TRPT is the theoretical total remaining processing time (the active operation time including set-up times and movements between machines/stations). A component with a CR value of 1.0 is "according to schedule", while it is behind if the value is less than 1.0 and ahead if the value is larger than 1.0. In case of a race condition, the component with the lowest CR value has precedence.

The optimization results achieved by the three algorithms are presented in Table 1 and stated in relation to the CR function. More specifically, the percentage numbers given in the table represent the relative improvement achieved by each algorithm when compared to the result of the CR function. No real numbers can be presented due to company restrictions.

### Table 1: Results with CR function as baseline (average of 20 replications).

<table>
<thead>
<tr>
<th></th>
<th>Lead times (improvement)</th>
<th>Tardiness (improvement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSGA-II</td>
<td>18%</td>
<td>15%</td>
</tr>
<tr>
<td>Cuckoo Search</td>
<td>12%</td>
<td>9%</td>
</tr>
<tr>
<td>Hill Climbing</td>
<td>9%</td>
<td>7%</td>
</tr>
</tbody>
</table>

As shown in Table 1, the results clearly indicate that NSGA-II outperforms both Hill Climbing and Cuckoo Search. Motivated by the superior results, NSGA-II is the algorithm selected for further usage in the case study at Volvo Aero. In the next chapter, evaluation outcomes from the case study are presented.

### 6. Evaluation and discussion

After some fine-tuning and adjustments to case specific requirements, the platform was ready to be evaluated by a test group at Volvo Aero. This test group included eight employees who represented all three user groups supported by the platform (operator, shift leader, and manager), as well as logistics engineers at the company. University representatives first demonstrated the platform on site at the company, after which it was made available to the test group. They tried the platform out for a couple of weeks, during which their feedback was collected and compiled.

After the test period, the evaluation feedback was discussed and analyzed. A positive aspect of the platform, identified in the feedback, was high usability; all the test group members considered that the user interface was easy to access, understand, and use. The feedback also highlighted the great potential the platform demonstrated with regard to improving production schedules, especially during periods of heavy workloads. Initial tests of the optimization performed before the actual evaluation period started indicated a productivity improvement of 18% in lead times and 15% in tardiness (see Chapter 5.4), and equivalent results were achieved also during the evaluation tests. At maximum work load, the productivity improvement gained by using the platform was most significant and reached over 30%.

Besides improving the production schedules, using the platform was also considered a way of significantly reducing the human effort currently associated with creating the schedules. A further advantage raised by the test group was the possibility of easily sharing optimized production schedules in real-time among stakeholders. Since the platform is web-based, it allows all users to easily view results simultaneously, which stands in contrast to traditional desktop applications where results must be printed or e-mailed.
A negative aspect of the platform raised, more or less, by everyone in the test group was the need to manually specify the current status and configuration of the manufacturing cell before each optimization. This work moment can be time-consuming and should preferably be eliminated so that the optimization can be fully automatic and run frequently, or even constantly. To realize this, the platform can be integrated with the computerized automation system of the manufacturing cell. This system, delivered by Fastems, includes advanced control software able to schedule production based on production orders, to manage numerical control programs, and to run unmanned production. The system also holds information about components’ status and localization. Integrating the platform with this system is, however, non-trivial as there exist no public software interface, nor any documentation on how to access data in the system. Realizing an integration therefore requires, for the first, vendor acceptance and, for the second, a customized solution developed by the vendor. This involves an investment for the company, but considering the potential production improvements obtained with a fully automated web-based optimization system this investment is motivated.

Besides criticizing the manual efforts required before an optimization begins, the test group only complained about minor details in the user interface. Such details included, for example, labels or color shades which were easily rectified. In general, however, creating the graphical user interface was more complicated than expected. During the development process, it became obvious that creating advanced graphical features for a web page is considerably more difficult than for a traditional desktop program. There was a growing realization that achieving a “perfect” graphical look could be very time consuming, or even impossible, for a web-based system. However, there is an expectation that the strong trend in web-based systems seen in the software development community will probably reduce this problem over time.

Another problem with using a web page as user interface identified during the development was the fact that the same web page may have a different appearance in the various web browsers (e.g., Internet Explorer, Mozilla Firefox, or Chrome) and even in different versions of the same browser. This problem originates in the different interpretations and implementations of web standards made in the various web browsers. Consequently, a web page must be tested and possibly adapted to all the common web browsers and their versions not only once, but every time a new version of a web browser is released. In general, this means that developing and maintaining the web-based user interface is more time consuming (and thereby more expensive) than a traditional desktop program.

A final aspect, identified in the evaluation, worth mentioning concerns the network delay that can be expected in a web-based platform which frequently communicates data over the internet. This drawback has been discussed in the study by Lou et al. (2000) and has not been explicitly disproved in any of the subsequent studies on web-based platforms for simulation-optimization. It was therefore of interest to find out how great an impact network delay has on the overall performance of the platform. In order to study this, the platform was used from different geographical sites, both in Sweden and from distant countries. In none of the cases was any network delay noticeable. Of course, in practice a delay must actually exist when the data is conveyed from one site to another, however, during the platform tests the delay was so small that a human was not aware of it. Although a human should actually notice a delay of a few milliseconds (or even seconds, at worst), it can be argued to be negligible considering that the total simulation-optimization process takes several minutes.

7. Conclusions

This chapter presents a summary of the study, provides general recommendations for web-based platforms, and outlines future work.

7.1 Summary

This paper presents a web-based platform for real-world, industrial simulation-optimization. The design of the platform is generic and meant to meet common requirements in industrial scenarios. In the implementation of the platform, modern web techniques, such as Ajax, JavaScript, GWT, and ProtoBuf, are used. The user interacts with the platform through a web page, which means that the platform is accessible from anywhere with an internet connection.

---

a www.fastems.com
connection. To prevent data sent over the internet to be accessed by a third party, the platform implements encryption and security certificates. The platform serves multiple users simultaneously, in order to support an industrial context in which different employees perform various tasks at the same time. Through modularization, the platform provides possibility to effectively utilize computing resources at different physical locations, and also to distribute replication in order to improve fault-tolerance and accessibility. To minimize the cost associated with introducing the platform, it is explicitly designed to only include software that is free of charge. A reference implementation of the platform with source code is also available in order to reduce development efforts.

In the paper, the platform is evaluated on a real industrial problem of production optimization within the Volvo Aero Corporation. The optimization problem considered is about finding the best prioritization of different engine components being processed in one of the company’s manufacturing cells. For solving the problem, three different optimization algorithms are being evaluated: Hill Climbing, NSGA-II, and Cuckoo Search. Of these algorithms, NSGA-II shows superior results and a significant productivity improvement is achieved using this algorithm, especially at high workloads in the manufacturing cell.

7.2 Recommendations

During the presented study, a number of general features of a successful web-based platform for industrial simulation-optimization were identified. These features, presented below, are recommended to consider in the development of future web-based platforms.

- **Generic** – the system architecture should be generally applicable to various simulation problems and not tailor-made for a specific scenario.
- **Modular** – the system should be designed on the basis of independent components, in order to promote easier implementation, testing, and maintainability.
- **Heterogeneous** – it should be possible to implement the different components of the system using different programming languages and it should also be possible to run them on different operating systems.
- **Low cost** – the use of all third-party software included in the system (e.g., for the web server and/or the databases) should be free of charge, in order to reduce the total cost.
- **Interactive** – developers should strive for a high degree of interactivity in the user interface of the system, which can be realized using Java, GWT, and/or Ajax.
- **Multi-user** – the system should support multiple users with different privileges and enable them to utilize the platform simultaneously.
- **Parallel** – for increased efficiency, the system should be able to execute parallel, multiple optimization algorithms and simulation evaluations.
- **Flexible** – the design of the system should be flexible and, for example, make it possible to vary the number of simulation clients or change the simulation software.
- **Secure** – encryption and security certificates should be implemented in the system, in order to protect it from hostile acts and influences.

Besides the features discussed above, for the successful realization of a web-based platform it is also recommended to carefully considering the graphical design and layout of the web page that constitutes the user interface. During the case study at Volvo Aero, it was clear that the user interface was the single most important factor in gaining user acceptance of the platform. In general, software development more easily focuses on algorithms and technical details, but putting at least as great an effort into the graphical design is recommended. It is especially important to keep usability in mind when developing systems for industrial adoption, since the success of a system is dependent on its acceptance by its users (Rogers et al. 2007).

Another important learning from the case study at Volvo Aero was the fact that manual specification of input data is not reasonable, but the optimization process must be fully automatic. This requirement is not unique for Volvo Aero, but can be expected to exist in most organizations. A fully automatic optimization, however, usually requires integration with existing software systems and can in many cases be expected to be both non-trivial and expensive. A recommendation is therefore to consider this issue already in the initial discussion of introducing of a
new platform in an organization, and also make sure that the associated cost is approved early in the process.

A final learning from the case study at Volvo Aero lifted as a recommendation is related to the company’s IT security policy. At Volvo Aero, as at most companies, public network traffic is strictly controlled in order to prevent unauthorized data access and malicious attacks. This caused unexpected problems when the platform was deployed as data must frequently be sent over the internet. The recommendation is therefore that is important to be aware of the company’s IT security policy in advance, and that it is also necessary to be able to prove that the system has the required level of security.

7.3 Future work

Without doubt, web-based systems have several advantages compared to classical desktop systems, including, for example, accessibility, scalability, portability, and maintenance. Despite these advantages, desktop systems are still the prevailing approach in simulation-optimization and the adoption of web techniques is very limited, especially in industrial contexts. One reason for this is probably the limited expertise available regarding how to best utilize web techniques in simulation-optimization. Another reason could be the lack of implementations and evaluations of web-based simulation-optimization, especially on real-world problems. More research regarding both theoretical knowledge and practical case studies is therefore important, in order to bring the field of simulation-optimization into the web-based generation – a step that has already been taken by many other types of software, such as word processors, mail programs, and image editors.
References


