Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

ON-LINE INSTRUMENTATION FOR SIMULATION-BASED OPTIMIZATION

Anna Persson
Henrik Grimm
Amos Ng

University of Skövde
Box 408, 541 48, SWEDEN

ABSTRACT

Traditionally, a simulation-based optimization (SO) system is designed as a black-box in which the internal details of the optimization process is hidden from the user and only the final optimization solutions are presented. As the complexity of the SO systems and the optimization problems to be solved increases, *instrumentation* – a technique for monitoring and controlling the SO processes – is becoming more important. This paper proposes a white-box approach by advocating the use of instrumentation components in SO systems, based on a component-based architecture. This paper argues that a number of advantages, including efficiency enhancement, gaining insight from the optimization trajectories and higher controllability of the SO processes, can be brought out by an on-line instrumentation approach. This argument is supported by the illustration of an instrumentation component developed for a SO system designed for solving real-world multi-objective operation scheduling problems.

1 INTRODUCTION

One of the most important and challenging subjects in the simulation field today is simulation-based optimization (Buchholz 2005). It has shown to be a powerful technique for systems improvement and has been successfully applied to address a wide range of real-world industrial optimization problems (April et al. 2004). The general problem in simulation-based optimization (SO) is to find a setting of decision variables that maximize or minimize a given objective function, assuming that the objective function is not available directly but must be estimated through simulation. In a general SO system (Figure 1), an optimization procedure feeds input values into a simulation, which measures the performance of the input. Based on the evaluation feedback, possibly in combination with previous evaluations, the optimization procedure generates a new set of input values. The generation-evaluation process then iterates until a stopping criterion is satisfied, usually based on the user's preference for the amount of time to be spent (April et al. 2004).

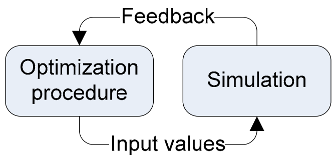


Figure 1: Simulation-Based Optimization

From the outside, a traditional SO system is an abstract black-box in which the internal processes are hidden behind an input-output interface, as shown in Figure 2. The user inputs initial system parameters and then waits for the final results to be presented, unaware of the internal system execution.

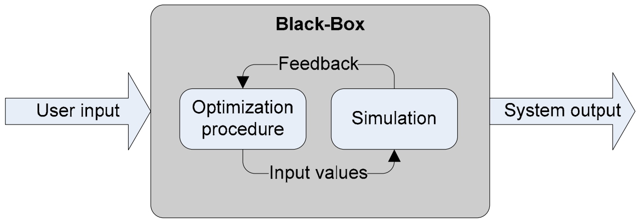


Figure 2: Traditional Black-Box Simulation-Based Optimization

Without a user-friendly user interface that presents the details of the optimization progress, it is difficult for the user to gain insight in the optimization process and the system under study. Also, with a black-box approach, efficiency might be easily lost if the SO process does not converge, for example, with the chosen optimization parameters. The use of more and more complex SO systems gives rise to a need of insight in the how and why of the system execution. In order to understand the SO system and improve its performance, monitoring of the system is critical. For this aim the SO system cannot simply be designed as a black-box, as the internal dynamic behavior of the system must be made visible from the outside. An important way to get insight is *instrumentation*, i.e. monitoring and control of the system. For SO, instrumentation is essential for system analysis tasks and performance evaluations.

Building on top of a component-based architecture for SO (Persson et al. 2006), this paper presents and describes an on-line instrumentation component for a SO system (Figure 3). This instrumentation component provides run-time visual monitoring, control, and analysis of the SO process.

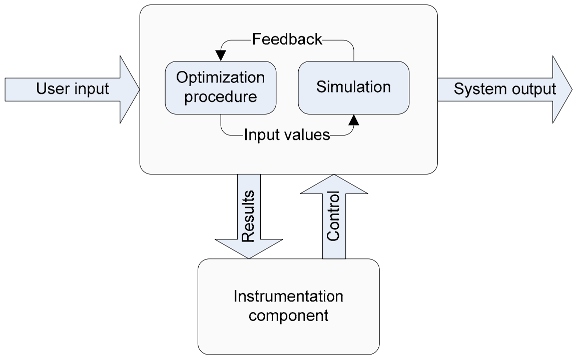


Figure 3: Instrumentation Component

In the next section, we introduce the topic of system instrumentation. In Section 3, we describe the concepts of the instrumentation component and some of its advantages. Section 4 presents a case implementation of an instrumentation component where it is connected to a SO system for solving a complex operation scheduling problem. This section also includes some ideas of how to design the graphical user interface for the component. In Section 5, we discuss some implications of adopting the instrumentation component.

2 BACKGROUND

Instrumentation in general is the art and science of measurement and control. In software engineering, instrumentation is a general term denoting techniques used to modify an existing program in order to collect data during the program's execution (Maebe et al. 2002). The basic principle of program instrumentation is to aggregate instrumentation code with the original program code, as shown in Figure 4 (adapted from Maebe et al. 2002). The instrumentation code is executed together with the program code during the execution.

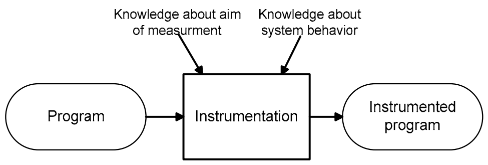

Knowledge about aim of measurment
Knowledge about system behavior
Program
Instrumentation
Instrumented program

Figure 4: Program Instrumentation

Program instrumentation can be performed on various abstraction levels, such as hardware level, source code level, or executable level (Maebe et al. 2002). Instrumentation can also be performed at various stages, such as compile-time or run-time (Luk et al. 2005). In this paper we consider run-time instrumentation of an executable program.

A great number of instrumentation applications have been built for a wide range of different systems, e.g. in the area of parallel and distributed systems, and in operating system kernels. Although a lot of work have been done in the field of program instrumentation, our review of the literature revealed no applications of instrumentation for SO systems.

3 AN ON-LINE INSTRUMENTATION COMPONENT

This section describes the concepts and functionality of the proposed on-line SO instrumentation component. The component supports monitoring and control of both the simulation and the optimization. However, there is slightly more focus on the latter as it is the optimization procedure that brings the SO process forward.

In short, the instrumentation component allows the user to perform the following:

- Visually observe and monitor the optimization process and its progress
- Keeping track of and view the most promising solutions
- Analyze solutions on-line by collecting and generating statistical data from both the simulation and optimization components
- Interactively control the optimization

The instrumentation component is designed to be used by non-specialist users and requires no familiarity with the underlying simulation/optimization technology. It can be linked to a wide range of various optimization problems and it is not tightened to or constrained by any specific simulation software or optimization procedure.

3.1 Architectural Design

Monitoring of software systems is either time-driven or event-driven (Klar et al. 1992). In the former, samples of the program's behavior are collected over time and used for statistical analysis of the program behavior, while in the latter events are represented by program activities and the dynamic functionality of the system is studied. The instrumentation component proposed in this paper is event-driven; when an event happen in the SO, a database is updated with information of the current status of the simulation and the optimization procedure. As soon as the database is updated, the instrumentation component reads the information and presents the intermittent results of the SO system to the user. The overall architecture of a SO system with an instrumentation component is shown in Figure 4.

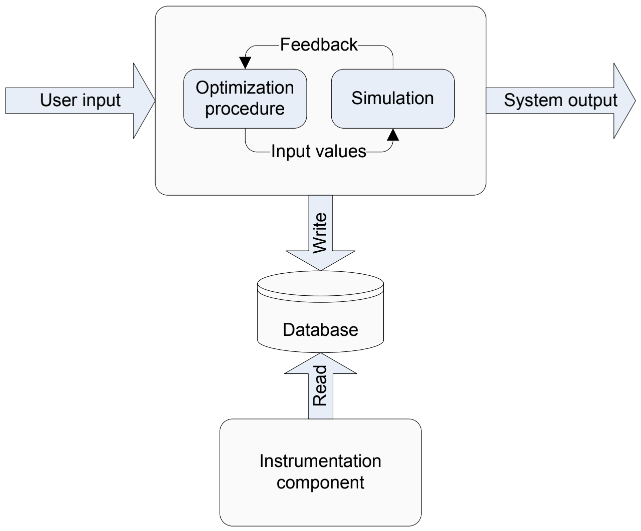


Figure 4: Architectural Design

An advantage of using a database, instead of obtaining the results directly from the SO system, is that the steadily growing amount of information does not need to be held in volatile memory but are saved in the database. This reduces memory use and facilitates efficient system recovery in case of a system crash. The database query language also supports complex questions to be asked about the data in a convenient way. Furthermore, multiple instrumentation components can connect to the database.

3.2 Functionality

This section describes the general functionality of the instrumentation component. The component comprises three modules for the support of on-line visual monitoring, control, and analysis of the SO system, namely, a solution analysis module, a statistical analysis module, and a control module (Figure 5).

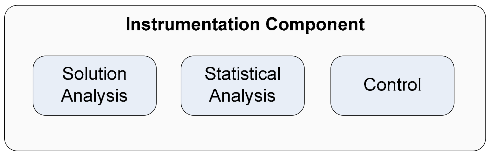


Figure 5: Instrumentation Component Modules

3.2.1 Solution Analysis Module

The solution analysis module provides functionality for analyzing individual solutions, according to Figure 6. The solutions to analyze are displayed in a list (a). To avoid overwhelming the user in the analysis process, the list of solutions only contains the most promising solutions. Still, the list can potentially be extensive. In order to reduce the number of solutions to analyze, filtering rules can be applied so that only solutions with certain properties are displayed in the list (b). The filtering rules are problem dependent. The list of solutions can also be sorted according to the achievement level of different objectives, in order to ease the user's analysis (b). When the user selects a solution from the list (c) for evaluation, this solution is visualized graphically together with relevant information such as the achievement level of each optimization objective (d). In an optimal buffer allocation problem, for example, all buffer capacities are graphically presented in a diagram in which the Y-axis corresponds to overall throughput of the system, which can represent the optimization objective. Two solutions can also be compared with each other graphically, to analyze similarities and differences between them (e). To facilitate comparison, the user chooses two solutions from the list and a visualization of these results are displayed next to each other.

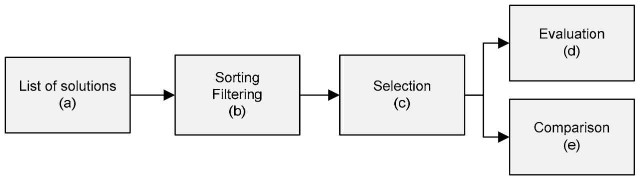


Figure 6: Analysis of Solutions

3.2.2 Statistical Analysis Module

The Statistical Analysis Module supports prognoses and analysis of the overall SO performance. The status of the SO process is plotted continually as new information are read from the database. These progress curves are a valuable tool for the user to determine the expected running length of the SO process. When the curves begin to level out it is a signal that no further improvements are to be expected and hence the process can be terminated or the settings can be modified to guide the optimization.

3.2.3 Control Module

The control module allows the user to change the settings of the simulation and the optimization on-line, in order to test if improved results can be achieved with a modified configuration. Example of variable settings are objective preferences and optimization algorithm parameters.

3.3 Advantages

This section discusses some advantages of connecting an on-line instrumentation component to a SO system.

3.3.1 Improved Understanding

The instrumentation component provides dynamic visualization of the SO system execution and a continuous view of its performance. This white-box approach allows for a better understanding of the SO and how the system's performance can be improved, in comparison with the traditional approach in which the performance of the SO system is evaluated only after the process has terminated.

3.3.2 No Explicit Stopping Criterion

The instrumentation component allows running the SO system without an explicit stopping criterion; instead the process is executed until the user is satisfied with the results. In the traditional approach when designing the SO system as a black-box, a stopping criterion (e.g., number of optimization iterations or maximum time consumption) must be specified to determine the SO run length. As there are no general heuristics for deriving stopping criteria, the length of running a SO process becomes an arbitrary decision. This decision is unlikely to be optimal; either the stopping criterion may be too tight and hinder optimal solutions to be found, or too generous so that a lot of time is wasted.

3.3.3 Fast Feedback of Objective Trade-offs Results

In most situations, it is not obvious how tradeoff information should be assigned to the various optimization objectives in order to obtain the requested solution, and there is a need for an agile analysis of the early results which may give some insight. The graphical visualizations of the SO progress provided in the instrumentation component supports this feature and allows the user to quickly acquire insight of the direction of the SO performance for a specific tradeoff setting. The user does not need to wait for the entire experiment to be finished, but may stop the process whenever noticing that the tradeoff information assigned will not result in a satisfactory solution and then may try with another tradeoff setting.

3.3.4 Interactive Control

The instrumentation component allows the user to evaluate different SO parameter settings in a convenient way by enabling run-time parameter modifications and immediate monitoring of their consequences. Since the result of the modifications are presented on-line there is no need to restart the system between each modification, like the traditional approach wherein results are presented only after the SO process has been terminated.

4 CASE IMPLEMENTATION

This section describes how the concepts of the instrumentation component presented in the previous chapter has been implemented to monitor and control a SO system solving a complex real-world machine scheduling problem. In short, the problem is to schedule mail batches on sorting machines at the Swedish Postal Services (see Persson et al. 2006 for more information).

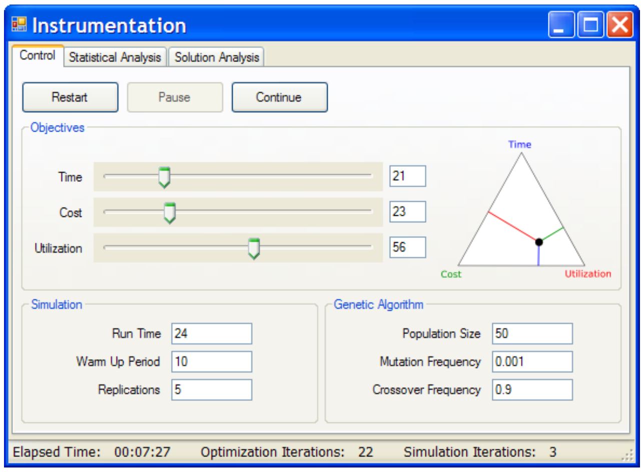
The SO system in this case implementation comprises a simulation model built in the Arena simulation software and an optimization procedure based on a Genetic Algorithm. There are three conflicting optimization objectives; minimization of cost, maximization of slack time, and maximization of even machine utilization. As it is not possible to obtain solutions which maximize performance of all objectives at the same time, the user must express tradeoff preferences between the objectives, specifying their relative importance.

Building on top of a component-based architecture specifically designed for SO (Persson et al. 2006), the online monitoring component that connects to the SO system is implemented in C++ using Microsoft Visual Studio .NET. Spreadsheet applications like Microsoft Excel that is commonly used for data input and output in simulation modeling may be used to develop the user interface of the instrumentation component. However, to increase the flexibility and controllability of the software development during our research study, we have chosen to develop our own Graphical User Interface (GUI). Currently, full implementation of the instrumentation component is underway, but it is sufficiently built to illustrate the concepts and ideas presented in this paper.

4.1.1 Control Module

The Control Module (Figure 7) allows the user to modify optimization and simulation settings while running the SO system, in order to test if improved results can be achieved with a different configuration. The settings that can be modified include objectives tradeoffs, simulation parameters, and parameters for the specific optimization strategy.

The user expresses tradeoff preferences regarding the various objectives by assigning a weight value to each objective specifying its relative importance – a higher value indicates that the objective is considered more important. The decision maker allots each objective a percentage value and the total weighting assigned must sum up to 100%. The objective tradeoff weightings can be changed using either slide bars or a graphical distance triangle, as shown in Figure 8. There is one slide bar for each of the objectives and the weighted importance of an objective is



Objectives

Time	21
Cost	23
Utilization	56

Simulation

Run Time	24
Warm Up Period	10
Replications	5

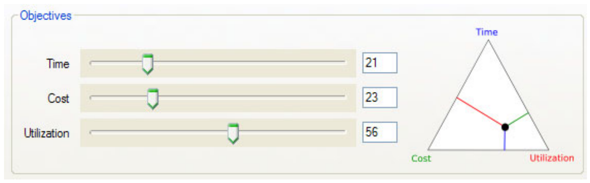
Genetic Algorithm

Population Size	50
Mutation Frequency	0.001
Crossover Frequency	0.9

Elapsed Time: 00:07:27 Optimization Iterations: 22 Simulation Iterations: 3

Figure 7: Control Module

shown on the right-hand side of its corresponding slider. Using the graphical distance triangle, the user changes objective tradeoff information by dragging a handle. Each corner of the triangle represent an objective, and the closer the point is to a corner the more important is the corresponding objective considered. If the point in the distance triangle is changed, the slide bars are changed accordingly and vice versa.

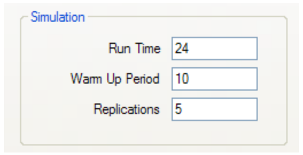


Objectives

Time	21
Cost	23
Utilization	56

Figure 8: Objectives Preferences

The simulation parameters that the user can modify run-time include simulation run time period (e.g., 24 hours or one week), simulation warm up period, and number of replications, as shown in Figure 9.

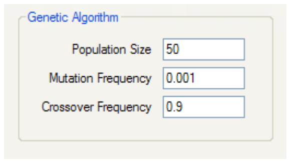


Simulation

Run Time	24
Warm Up Period	10
Replications	5

Figure 9: Simulation Settings

In this application example, since a Genetic Algorithm is used as the optimization algorithm, the variable parameters are therefore population size, mutation rate, and crossover frequency (Figure 10) that are specific for GA as well as other evolutionary strategies.



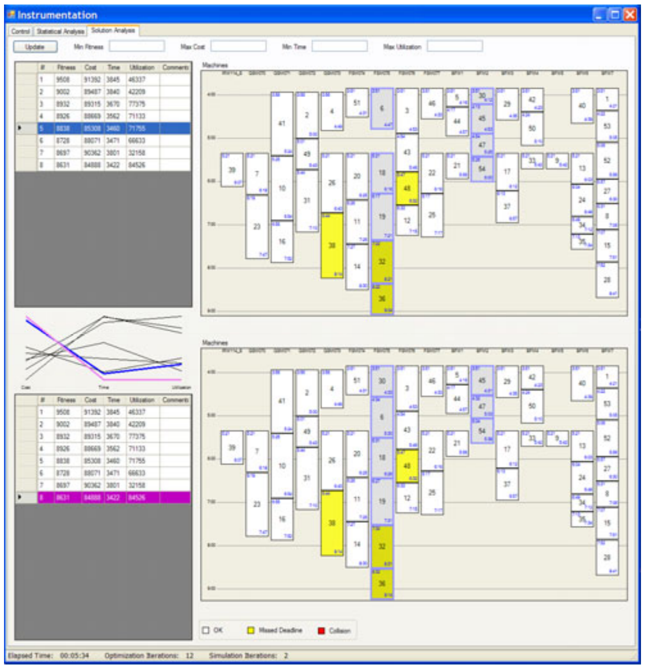
Genetic Algorithm

Population Size	50
Mutation Frequency	0.001
Crossover Frequency	0.9

Figure 10: Genetic Algorithm Settings

4.1.2 Solution Analysis Module

In the Solution Analysis Module (Figure 11), the user can perform visual on-line analysis of promising solutions. In this case, promising solutions are considered to be the Pareto optimal set. Pareto optimal solutions are solutions superior to the other solutions considering all objectives but possibly inferior to other solutions considering one or several objectives (Srinivas and Deb 1995).



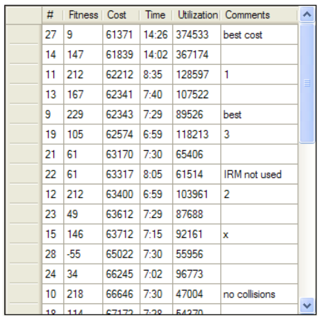
#	Fitness	Cost	Time	Utilization	Comments
1	9508	91392	3845	46337	
2	9002	89487	3840	42209	
3	8932	89315	3670	77375	
4	8926	88669	3562	71133	
5	8838	85308	3460	71755	
6	8728	88071	3471	66633	
7	8697	90362	3801	32158	
8	8631	84888	3422	84526	

Elapsed Time: 00:05:34 Optimization Iterations: 12 Simulation Iterations: 2

Figure 11: Solution Analysis Module

The Pareto optimal set of solutions is displayed in a list (Figure 12), where each list entry corresponds to an individual solution. Each solution has a unique identifier and is presented in the list together with its achievement values of the various optimization objectives. To ease the user's analysis, the list can be sorted according to the achievement values, as in Figure 12 where solutions are sorted based on cost. To aid the user in the evaluation of solutions, there is also a possibility to write a comment next to each solution.

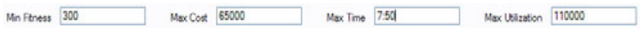
The list of solutions is updated on the user's initiative when pushing the "Update" button. This button is only enabled when new Pareto optimal solutions have been found, and in that way the user knows when there are new solutions available. The reason for not updating the list automatically as soon as a new Pareto optimal solution is found is to avoid, from a user-perspective, confusing unexpected changes in the list.



#	Fitness	Cost	Time	Utilization	Comments
27	9	61371	14:26	374533	best cost
14	147	61839	14:02	367174	
11	212	62212	8:35	128597	1
13	167	62341	7:40	107522	
9	229	62343	7:29	89526	best
19	105	62574	6:59	118213	3
21	61	63170	7:30	65406	
22	61	63317	8:05	61514	IRM not used
12	212	63400	6:59	103961	2
23	49	63612	7:29	87688	
15	146	63712	7:15	92161	x
28	-55	65022	7:30	55956	
24	34	66245	7:02	96773	
10	218	66646	7:30	47004	no collisions
18	114	67172	7:28	54370	

Figure 12: List of Pareto Optimal Solutions

The number of solutions to analyze can be reduced by applying filtering rules (Figure 13), which specify that only solutions that fulfill certain constraints should be displayed in the list. Filtering can be done according to minimum fitness, maximum cost, maximum latest stop time, and maximum machine utilization.



Min Fitness: 300 Max Cost: 65000 Max Time: 7:50 Max Utilization: 110000

Figure 13: Filtering Rules

The Pareto optimal solutions are also presented in a trade-off graph (Figure 14). In this graph, the relationships between objectives are visually presented and conflicting objectives are indicated with crossing lines (Fonseca and Fleming 1993). The high-dimensional (in this case, 3-dimensional) space is reduced to only two dimensions, comprehensible for a human analyst. The horizontal axis represents the different objectives and the vertical axis indicates the normalized performance of each of the objectives. A line in the graph is highlighted when the user selects a solution in one of the lists of Pareto optimal solutions, with different highlighting colors for the two lists.

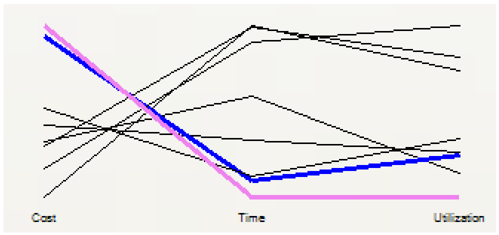


Figure 14: Graph of Pareto-optimal Solutions

When the user selects a solution, either in one of the lists or in the trade-off graph, the solution is visualized in a graphical schedule diagram as shown in Figure 15. The X-axis of the diagram represents machines (there are sixteen machines in this case) and the Y-axis represents time. A box in the diagram corresponds to a job and for each job its identification number is shown together with its start and stop time.

To ease the evaluation of solutions, important properties of a schedule are color encoded. For the problem considered in this implementation there are two properties to highlight, namely missed deadlines for jobs and job collisions (i.e. when more then one job is scheduled on the same machine in the same time). In the schedule shown in Figure 15, there are three jobs with missed deadlines and one job collision.

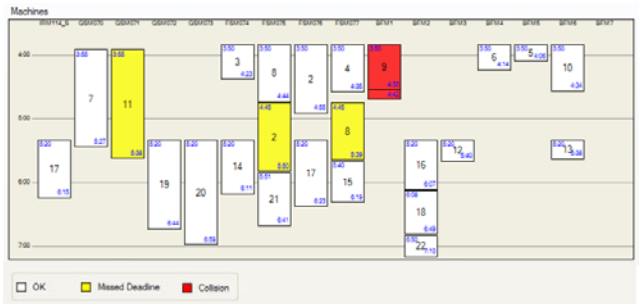
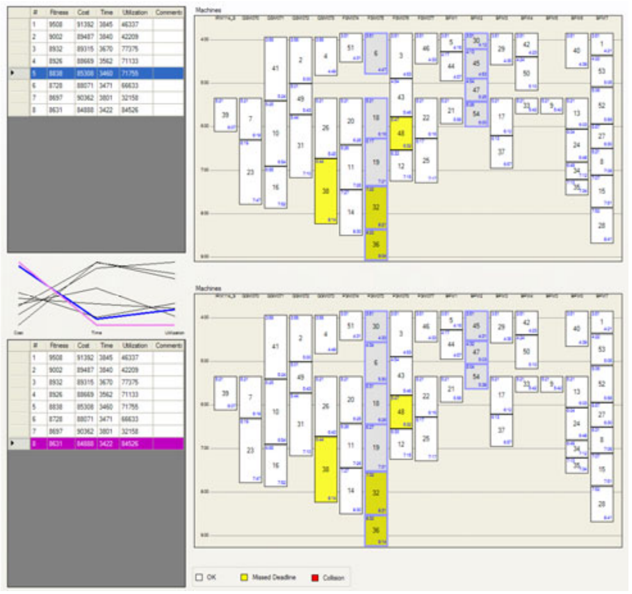


Figure 15: Graphical Visualization of a Solution

Two solutions can be compared with each other graphically to analyze similarities and differences between them, as shown in Figure 16. To facilitate a convenient comparison, differences of the solutions are highlighted in the schedule diagrams.



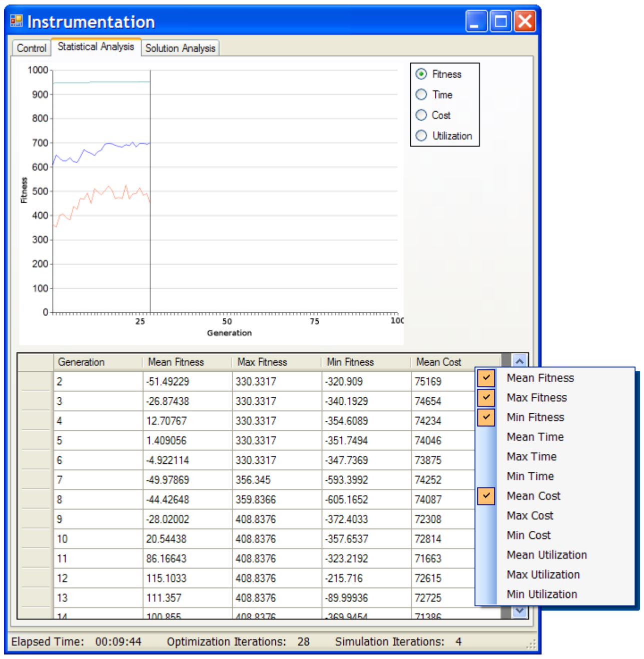
#	Fitness	Cost	Time	Utilization	Comments
1	9508	91392	3845	46337	
2	9002	89487	3840	42209	
3	8932	89315	3670	77375	
4	8926	88669	3562	71133	
5	8838	85308	3460	71755	
6	8728	88071	3471	66633	
7	8697	90362	3801	32158	
8	8631	84888	3422	84526	

#	Fitness	Cost	Time	Utilization	Comments
1	9508	91392	3845	46337	
2	9002	89487	3840	42209	
3	8932	89315	3670	77375	
4	8926	88669	3562	71133	
5	8838	85308	3460	71755	
6	8728	88071	3471	66633	
7	8697	90362	3801	32158	
8	8631	84888	3422	84526	

Figure 16: Graphical Comparison of Solutions

4.1.3 Statistical Analysis Module

The Statistical Analysis Module (Figure 17) supports prognoses and analysis of the optimization performance. In this module, status information of the Genetic Algorithm fitness value is continuously presented, as well as the achievement level of each of the three optimization objectives.



Generation	Mean Fitness	Max Fitness	Min Fitness	Mean Cost
2	-51.49229	330.3317	-320.909	75169
3	-26.87438	330.3317	-340.1929	74654
4	12.70767	330.3317	-354.6089	74234
5	1.409056	330.3317	-351.7494	74046
6	-4.922114	330.3317	-347.7369	73875
7	-49.97869	356.345	-593.3992	74252
8	-44.42648	359.8366	-605.1652	74087
9	-28.02002	408.8376	-372.4033	72308
10	20.54438	408.8376	-357.6537	72814
11	86.16643	408.8376	-323.2192	71663
12	115.1033	408.8376	-215.716	72615
13	111.357	408.8376	-89.99936	72725
14	100.855	408.8376	-369.9454	71386

Elapsed Time: 00:09:44 Optimization Iterations: 28 Simulation Iterations: 4

Figure 17: Statistical Analysis Module

To facilitate analysis of optimization progress, curves representing fitness value and achievement levels of the optimization objectives are displayed, showing minimum, mean, and maximum value for each simulation-optimization iteration (Figure 18).

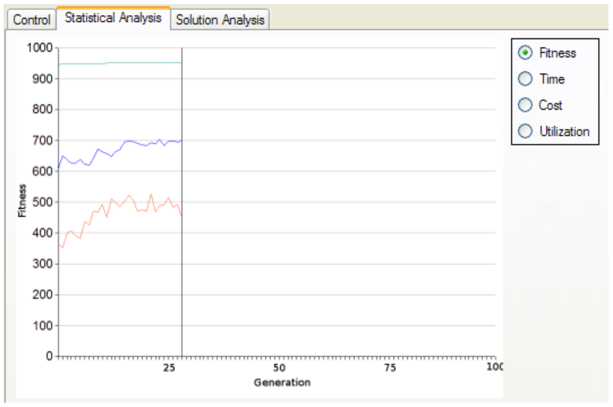
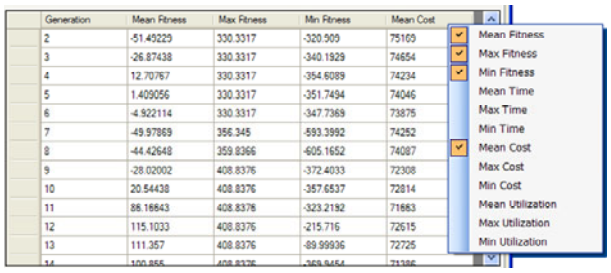


Figure 18: Time Plotted Progress Curves

To allow for a detailed analysis of the plotted curves, continuous status values are also displayed in a table. The user chooses what columns to include in the table using a selection menu, as shown in Figure 19.



Generation	Mean Fitness	Max Fitness	Min Fitness	Mean Cost
2	-51.49229	330.3317	-320.909	75169
3	-26.87438	330.3317	-340.1929	74654
4	12.70767	330.3317	-354.6089	74234
5	1.409056	330.3317	-351.7494	74046
6	-4.922114	330.3317	-347.7369	73875
7	-49.97869	356.345	-593.3992	74252
8	-44.42648	359.8366	-605.1652	74087
9	-28.02002	408.8376	-372.4033	72308
10	20.54438	408.8376	-357.6537	72814
11	86.16643	408.8376	-323.2192	71663
12	115.1033	408.8376	-215.716	72615
13	111.357	408.8376	-89.99936	72725
14	100.855	408.8376	-369.9454	71386

Figure 19: Table of Status Values

5 CONCLUSIONS AND FUTURE WORK

A major problem of SO for practical applications is that it is computationally time consuming (Dengiz et al. 2006; Boesel et al. 2001; Fu et al. 2005). By introducing instrumentation to SO systems, as proposed in this paper, there is a potential for alleviating this problem. There are mainly three features of the SO instrumentation that enables system efficiency enhancement:

- The instrumentation enables run-time system performance analysis and evaluations, essential in order to gain the understanding necessary to accomplish efficiency improvements of the system implementation.
- The instrumentation allows for the SO system to be run for the exact time needed for the user to be satisfied with the results.
- The instrumentation provides possibility to interrupt the running SO system and reconfigure or restart it if its progress is not satisfying.

However, it should be remarked that the instrumentation is not a pure efficiency enhancement tool – it also introduce some overhead to the SO process. The SO system status is written to a database, or exposed in some other way, frequently and each such command causes a certain performance degradation. It is important that this instrumentation overhead is limited and future work includes studying how this can be achieved.

As argued in this paper, a white-box approach to SO systems have a number of advantages, such as for example higher controllability of the SO processes and insight into the system internals. However, it also has a drawback in that the instrumentation component implementation, to some degree, becomes dependent on the implementation of the simulation and the optimization strategy – although the general concepts of using instrumentation component are generic irrespective of underlying SO technology. For example, changing the optimization strategy to another strategy has the consequence that the Control Module must be changed accordingly, and probably also the Statistical Analysis Module. In future work, we intend to study how a generic instrumentation component interface can be designed to facilitate the loose coupling between different components in the SO system. In this aspect, the traditional black-box approach has an advantage as it allows for system internals to be changed without influencing the use of the system.

One of our current research focuses which is related to this work is the design of an Internet-based component architecture called OPTIMISE (OPTIMization using Intelligent Simulation and Experimentation). This architecture has emerged from the observation that many simulation-optimization applications, for example, the automatic generation of optimized operation schedules described here, can be rapidly developed by making use of a common architectural design. With a common component-based architecture, many system components can be reused and optimization components can be customized rapidly for new applications. The design of OPTIMISE is now underway and integrating the proposed instrumentation components to the OPTIMISE platform will be one of the major tasks in our future work.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the Knowledge Foundation (KK Stiftelsen), Sweden, for the provision of research funding and Posten AB for their collaborative inputs to this work.

REFERENCES

April, J, Better, M., Glover, F. and Kelly, J. 2004. New advances for marrying simulation and optimization. In *Proceedings of the 2004 Winter Simulation Conference*, 80-86.

Boesel, J., Bowden, R.O., Glover, F., Kelly, J.P. and Westwig, E. 2001. Future of simulation optimization. In *Proceedings of the 2001 Winter Simulation Conference*, 1466-1469.

Buchholz, P. and Thümmler, A. 2005. Enhancing evolutionary algorithms with statistical selection procedures for simulation optimization. In *Proceedings of the 2005 Winter Simulation Conference*, 842-852.

Dengiz, B., Bektas, T. and Ultanir, E. 2006. Simulation optimization based DSS application: A diamond tool production line in industry. *Simulation Modelling Practice and Theory* 14: 296–312.

Fonseca, C. M. and Fleming, P. J. 1993. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, 416–423.

Fu, M.C., Glover, F. and April, J. 2005. Simulation optimization: a review, New Developments, and Applications. In *Proceedings of the 2005 Winter Simulation Conference*, 83-95.

Klar, R. Quick, A. and Sotz, F. 1992 Tools for a model–driven instrumentation for monitoring. In *Proceedings of the 5th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, 165–180.

Luk, C. K, Cohn, R, Muth, R. Patil, H., Klauser, A., Lowney, G., Wallace, S., Reddi, V. J. and Hazelwood, K. 2005. Pin: Building customized program analysis tools with dynamic instrumentation. In *Programming Language Design and Implementation*, Chicago, IL, June 2005.

Maebe, J., Ronsse, M. and De Bosschere, K. 2002. DIOTA: Dynamic instrumentation, optimization and transformation of applications. In *Proceedings of the 4th Workshop on Binary Translation*.

Persson, A., Ng. A., Grimm, H., Karlsson, T., Ekberg, J., Falk, S. and Stablum, P. 2006. Simulation-based multi-objective optimization of a real-world operation scheduling problem. *To appear in Winter Simulation Conference 2006*.

Srinivas, N. and Deb, K. (1995) Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

AUTHOR BIOGRAPHIES

ANNA PERSSON is a PhD candidate at University of Skövde, Sweden and De Montfort University, U.K. She holds a Master's degree in Computer Science from University of Skövde. Her research interests include artificial intelligence, simulation-based optimization, multi-objective optimization and efficiency enhancement techniques for simulation-based optimization. Her e-mail address is <anna.persson@his.se>.

HENRIK GRIMM is a Systems Developer at the University of Skövde, Sweden. He received his BSc and MSc degrees in Computer Science from University of Skövde. His research interests include computer simulation, artificial intelligence, and distributed systems. His e-mail address is <henrik.grimm@his.se>.

AMOS H.C. NG is a Senior Lecturer at the University of Skövde, Sweden. He holds a B.Eng. degree and a M.Phil. degree, both in Manufacturing Engineering from the City University of Hong Kong and a Ph.D. degree in Computing Sciences and Engineering from De Montfort University, Leicester, U.K. He is a member of the IEE and a Chartered Engineer in the U.K. His research interests include agent-based machine control systems, virtual engineering for manufacturing machinery and machine systems as well as simulation-based optimization. His e-mail address is <amos.ng@his.se>.