Data-Driven Simulation-Optimization (DSO): An Efficient Approach to Optimize Simulation Models with Databases

Mohammad Dehghanimohammadabadi(✉)

Mechanical and Industrial Engineering Department, Northeastern University, Boston, MA, USA
m.dehghani@northeastern.edu

Abstract. Simulation-optimization is instrumental to solve stochastic problems with complexity. Over the past half-century, simulation-optimization methods have progressed theoretically and methodologically across different disciplines. The majority of commercial simulation packages - to some degree - offer an optimizer that allows decision-makers to conveniently determine an optimal or near-optimal system design. With the latest advancements in simulation techniques, such as data-driven modeling and Digital Twins, optimizer platforms need a redesign to include new capabilities. This paper proposes a Data-driven Simulation-Optimization (DSO) platform to narrow this gap. By considering data-tables as a decision variable (control), DSO can systematically generate new tables, run experiments, and determine the best table entries to optimize the model. To implement DSO, three software packages (MATLAB, Simio, and MS Excel) are integrated via a customized coded interface, called Simio-API. The applicability of this Simulation-optimization tool is tested in two experimental settings to evaluate its effectiveness and provide some insights for future extensions. The DSO initial results are promising and should stimulate further research in academia and industry.

Keywords: Data-generated modeling · Digital twins · Simheuristics · Intelligent simulation · Simio · Data driven models · Industrial 4.0

1 Introduction

One of the major goals of using simulation modeling is to obtain the ideal configuration of a system. This is achievable by integrating the simulation model with an optimization module. In this approach, the optimizer explores the solution space in order to find the best input values to optimize the model design and its performance measures. A general algebraic form of the simulation optimization (SO) problem can be defined as,

B. Dorronsoro et al. (Eds.): OLA 2022, CCIS 1684, pp. 117–132, 2022.
https://doi.org/10.1007/978-3-031-22039-5_10

$$\min_{x,y,v} \quad E_v[f(x,y,v)] \tag{1}$$

$$\text{subject to:} \quad E_v[g(x,y,v)] \preccurlyeq 0 \tag{2}$$

$$h\,(x,y) \preccurlyeq 0 \tag{3}$$

$$x_l < x < x_u \tag{4}$$

$$y_l < y < y_u \tag{5}$$

$$x \in \mathbb{R}^n, y \in \mathbb{D}^m \tag{6}$$

where Eq. (1) refers to a real-valued objective function f evaluation or the model output, which without loss of generality, is minimized. The performance measure of the model is calculated based on the expected output value with respect to input variables. Discrete and continuous input parameters of the simulation model are defined by x and y, respectively. Vector v is the realization of the associated random variables in the model. Equation (2) refers to stochastic constraints where g is a real vector-valued function of stochastic constraints of the model to account for uncertainty. Equation (3) represents a real vector-valued for deterministic constraints h that are not affected by the uncertain parameters. Equations (4) and (5) lower and upper boundaries for discrete and continuous input parameters. Equations (6) defines real and integer decision variables.

The above-mentioned formulation is general enough to represent all kinds of SO problems. Choosing the appropriate simulation and optimization design is crucial for practical applications and highly depends on the problem characteristics [1]. Commercial simulation software packages often use provably convergent algorithms proposed by the research community with metaheuristics to help their users deploy SO for their models [2]. In order to use these optimization algorithms embodied in simulation packages, a user needs to create the preliminary design of the optimization problem. This design includes simulation inputs (controls), the objective functions (responses), and constraints. The optimizer explores a series of simulation configurations by changing the model controls and tries to obtain the optimum or close-to-optimum set of input parameters.

1.1 Research Motivation

Nowadays, using data-table inputs is very essential in developing Discrete-event Simulation (DES) models with large amounts of data. Using data-tables makes simulation model development, execution, and experimentation efficient and easy to implement. Instead of defining an abundant number of parameters and variables, all required data for the simulation modeling can be stored in a data-table format. The data-table values can be manually entered by the user or bound to a data structure framework such as MS Excel, databases, or Enterprise Resource Planning (ERP) systems. This feature provides a highly flexible approach to handling large data inputs for modeling needs. Data-tables could efficiently include any type of model's information with a large number of entries. This could

include entities' information (e.g. entity types, arrival times, processing time(s), sequence and priorities, etc.), resources' data (e.g. schedules, maintenance plans, locations on the layout, etc.), or even transportation networks. Based on the new advancement in DES commercial packages, these data can be accessed sequentially, randomly, directly, and even automatically [3]. This emphasizes the importance of using data-tables with the simulation model creation and experimental analysis.

1.2 Novelty and Main Contributions of the Paper

Although enhancing a simulation model with data-table inputs simplifies the model development, there is not a trivial way to optimize the simulation models based on data-table inputs. Existing commercial SO tools are designed to include a limited number of numerical controls (binary, integer, or real) for optimization purposes and are incapable of including data-table inputs to the optimization process. This becomes more challenging when data-tables are non-numerical, e.g. categorical, Date/Time, etc. Therefore, this paper aims to remedy this lack and introduce Data-Table Simulation-Optimization (DSO) platform.

This platform that allows simulation users to optimize the model's performance by deploying simulation experiments with respect to data-table inputs with no data format restrictions. For instance, a user can optimize "Patient's Arrival Table" (Data/Time format) in a healthcare system, "Product Mix Table" (categorical format) in a manufacturing setting, or "Destinations/Nodes Sequence Table" (integer format) for a set of transporters/vehicles in a supply chain network. This platform benefits simulation model extensibility, scenario creation, and experiment repeatability. So, the original form of the SO problem can be modified as follows:

$$\min_{x,y,t,v} \quad E_v[f(x,y,t,v)] \tag{7}$$

$$\text{subject to:} \quad E_v[g(x,y,t,v)] \preccurlyeq 0 \tag{8}$$

$$h\,(x,y,t) \preccurlyeq 0 \tag{9}$$

$$x_l < x < x_u \tag{10}$$

$$y_l < y < y_u \tag{11}$$

$$x \in \mathbb{R}^n, y \in \mathbb{D}^m \tag{12}$$

$$t \in \mathbb{T} \tag{13}$$

where Eqs. 7–11 are equivalent of Eqs. 1–5. The only difference is adding a new term t to represent data-table inputs used towards simulation optimization. The values stored in the table could have any format and structure, where all of these formats are represented by T (Eq. 13). The proposed DSO platform introduces the following capabilities:

- **Data-driven:** the ability to include data-table inputs as a decision variable (control) in the optimization process. The optimizer automatically generates

different values across the entire table, and gradually evolves its values to optimality.
- **Generalized:** the ability to handle multiple data types for optimization purposes. This includes but not limited to dates (e.g. order dates, dues dates, lead times), strings (e.g. dispatching rules name), numeric (e.g. resource capacities, order quantities, entity priority), objects list (e.g. list of nodes, list of tasks, list of transporters).
- **Scalable:** optimize a large set of parameters simultaneously. For instance, to optimize priority values of 100 jobs with 2 servers, the existing solution approaches require an exhaustive list of parameters (in this cases $(100)(2) = 200$) which is which is neither practical nor scalable. By considering the table as one-single input, DSO reduces the need of defining a large number of individual parameters. This makes the optimization process scalable, hassle free, and easy to implement.
- **Real-time:** leveraging data-tables enables the simulation model to be connected with a stream of data extracted from ERP systems. This allows DSO to be more aware of the changes in the real system and provide dynamic and reliable results.

1.3 Organization and Structure of the Paper

The rest of this paper is organized as follows. Section 2 provides a brief introduction of SO and highlights the motivation of this work. In Sect. 3, the DSO platform is explained in detail and its implementation aspects are discussed. To demonstrate the applicability of the proposed framework, two experiments are designed and analyzed in Sect. 4. This work is wrapped up in Sect. 5, and future work directions are presented in the end.

2 Literature Review and Background

The desirability of seeking better solutions is the main driver for developing SO techniques. As a definition, SO is a systematic search process to find the best configuration of a stochastic system in order to optimize objective function(s). This search needs to be efficient to minimize the resources spent while maximizing the information obtained in a simulation experiment [4]. With a long and illustrious history, SO is arguably the ultimate aim of most simulationists [5]. With a huge advancement in both research and practice, SO is considered as one of the main streams of simulation studies and has received considerable attention from both simulation researchers and practitioners [6]. Many studies applied SO to address problems in healthcare [7–9], manufacturing [10–12], and supply chain [13–16].

Nowadays, SO is a vibrant field and various sub-disciplines are evolved from different communities such as systems and control, statistics and design-of-experiments, math programming, and even computer science [5]. With the advancements in the SO literature, many of the simulation vendors provide some

sort of automatic experimental generators or optimization tools. Based on a survey conducted by [17], 40 out of 55 software packages are featured with SO tools. Most of these SO tools such as OptQuest [18] and SimRunner [19] are designed based on the Simheuristics structure. In Simheuristics, a Metaheuristic algorithm is coupled with the simulation environment [20] to perform an iterative search through parameter values to obtain improved responses. These tools do not offer guarantees of optimality, but the provided solutions are near-to-optimal and realistic.

Almost in all the existing SO tools, the decision variables are restricted to numerical parameters without considering other important elements of the model. For instance, in a healthcare system, one can easily evaluate a hospital model based on different numbers of resources (i.e. physicians, nurses, beds, etc.) and determine the best numeric combination of these values. However, within the same model, it is not trivial to systematically change the layout of the hospital, the schedule of physicians/nurses, or even individual patients' arrival times. To evaluate each of the above-mentioned scenarios, a tremendous amount of effort is required to manually make changes and customize the simulation model. The same concept relates to a manufacturing setting. For example, finding the best number of workers, transporters, servers, etc. is easily attainable using the existing commercial or non-commercial SO tools, while optimizing workers' schedules, transporters' network, and servers' location (layouts) is not a straightforward task.

The proposed DSO in this article is a perfect alternative in which numerical and data-table inputs can be optimized simultaneously. This platform is general enough to include any type of table entries with multiple data formats. Therefore, DSO is a promising SO framework and can introduce a significant opportunity for the simulation community to solve problems efficiently on a larger scale.

3 The Proposed DSO Platform

To obtain a practical and ideal DSO model, an integrated framework is developed using three modules, namely (i) optimization, (ii) simulation, and (iii) data exchange. As illustrated in Fig. 1, both simulation and optimization modules are bound to an external data source. In this iterative scheme, the optimization module provides a new solution and updates the data-table input(s) in each iteration, and then, the simulation module runs the model based on the newly provided data settings. Using this new framework, users can easily build a simulation model with data-tables and optimize it in an efficient manner.

The main goal is to make this framework general enough to be used in any simulation setting with different applications. To implement the proposed DSO framework, three software packages are linked together. The Optimization Module is deployed in MATLAB, and the Simulation Module is designed in Simio. These two modules are linked together via MS Excel for data-table input exchange.

MATLAB is a powerful tool and is adequate for addressing heavy computing needs such as optimization problems. MATLAB can be easily linked with

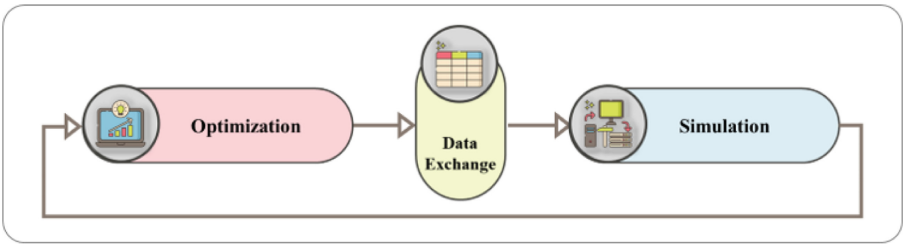


Fig. 1. The proposed DSO platform structure.

external software for advanced calculations and its programmable environment allows users to design and customize their algorithms. It has enhanced optimization capabilities and allows its user to choose between the existing optimization libraries or their developed algorithms. As a result, the applied optimization algorithm in this work is coded in MATLAB.

Simulation models often require large amounts of data to define different elements of the model such as entities, objects, networks, schedules, etc. Simio can represent data in simple tables and allows users to match the data schema for the manufacturing data (e.g. an ERP system) [21]. This simulation software is flexible enough to model complex systems with different operational needs. Another major benefit of Simio is its API capability which helps developers to extend Simio's access to external software packages. By taking advantage of this feature, a customized API is coded in C# to assist with the TDSO idea. This API connects MATLAB with Simio and provides a scheme to exchange data between the two. This enables users to connect Simio with other programming languages such as Python, R, or Julia.

The third component of this framework is MS Excel which is compatible with both MATLAB and Simio. This Data Exchange Module is the central piece of the framework and facilitates the data transfer between simulation and optimization packages. In Simio, data-tables can be bound to MS Excel and be accessed sequentially, randomly, directly, and even automatically. This important feature makes the development of DSO a feasible and effective intervention. A schematic illustration of DSO is shown in Figure 2 and its pseudocode is provided in Algorithm 1.

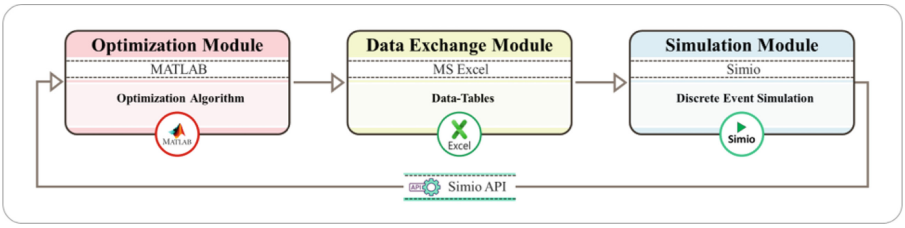


Fig. 2. Implementation components of the DTSO framework using MATLAB, Simio, and Excel.

Algorithm 1: Data-driven Simulation Optimization (DSO)

Data: Define the simulation model parameters: $x \in \mathbb{R}^n$, $y \in \mathbb{D}^m$
Data: Define the simulation model data-table inputs $t \in T$

initialization;
while *not the end of optimization algorithm* **do**
  Generate a new data-table solution
  Update data with the new data-table solution
  Trigger the simulation model and run experiments
  for $r \leftarrow 1$ **to** $MaxReplications$ **do**
    Replicate the simulation model
    Calculate the expected value of simulation responses $\mathbb{E}_v[f(x, y, t, v)]$
    Update objective function values

Result: Optimal/near-to-optimal data-table input

4 Experimental Analysis: DSO for Job Scheduling and Sequencing

Using data-tables can significantly facilitate simulation modeling development, execution, and improvement. Data can be imported, exported, and even bound to external resources. While reading and writing disk files interactively during a run can reduce the execution speed, tables hold their data in memory and so execute very quickly [22]. Applying DSO can harness the advantages of data-tables and place more emphasis on their use. To reveal some insights into the present and future works, this section demonstrates the applicability of DSO in two experimental settings.

To maintain the focus of the paper on the DSO advantages, the following experiments in this section introduce typical manufacturing settings with nominal operations. However, without loss of generality, DSO can be utilized in any simulation models in Simio with different levels of complexities. Also, the applied optimization algorithm is Particle Swarm Optimization (PSO) which is manually coded in MATLAB to optimize data-table entries. Again, this does

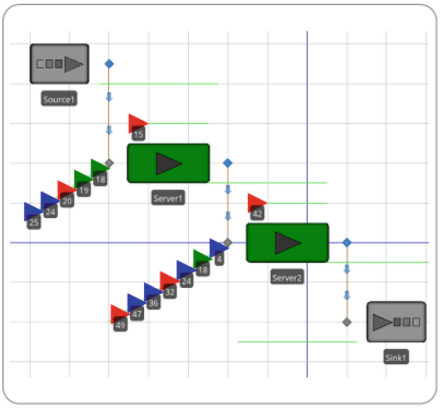
not limit the applicability of DSO, and different users can leverage a variety of optimization tools and algorithms to solve their problems. These experiments are discussed as follows. To maintain the paper's flow and readability, details of PSO, its operations, and pseudocode are provided in Appendix A.

4.1 Experiment 1: Job Scheduling with DSO

This study considers a flow shop model where 50 jobs (entities) are processed sequentially by two (2) servers. This model includes three (3) types of jobs that randomly arrive in the system in batches of five (5). The model assumptions are:

- All machines are ready to be scheduled in time zero.
- Preemption of operations of each job is not allowed.
- Different job types have different distributions for processing time and due dates.
- Setup time is job dependent (setup time varies from one job type to another on each server.)
- Each machine can process only one operation at a time.

Figure 3-a depicts the simulation environment where the flow shop model is developed based on the assumptions provided above. This model has two objective functions: i) minimizing the average Time In the System (TIS), and ii) minimizing the Total Tardiness Cost (TTC) of all jobs. The goal is to find the best prioritization of jobs in both servers in such a way that all objective functions are optimized. This model is simulated in Simio and a data-table is created to implement its operational logic.



(a) Simio model

Bound to Excel: C:\Paper2\Case1.xlsx, Worksheet: Table2

	Arrival Time	Entity Type	Sr1Priority	Sr2Priority
1	10/26/2018 8:00:00 AM	Type2	44	40
2	10/26/2018 8:00:00 AM	Type3	4	27
3	10/26/2018 8:00:00 AM	Type1	7	20
4	10/26/2018 8:00:00 AM	Type2	42	47
5	10/26/2018 8:00:00 AM	Type2	46	21
6	10/26/2018 9:35:00 AM	Type1	11	13
7	10/26/2018 9:35:00 AM	Type3	16	42
8	10/26/2018 9:35:00 AM	Type1	19	33
9	10/26/2018 9:35:00 AM	Type3	33	50
10	10/26/2018 9:35:00 AM	Type2	49	48
11	10/26/2018 10:20:00...	Type1	27	11
12	10/26/2018 10:20:00...	Type1	5	18
13	10/26/2018 10:20:00...	Type2	25	46
14	10/26/2018 10:20:00...	Type3	28	23

(b) Jobs' priority in Server1 and Server2

Fig. 3. Job Scheduling simulation model and the data-table input structure.

Figure 3-b depicts a snapshot of the data-table entry which stores entities' information such as arrival time, entity type, and priority numbers in Servers 1

and 2. The highlighted columns (Sr1Priority and Sr2Priority) indicate the priority of jobs on each server. This Optimization Module (i.e. PSO algorithm) in DSO treats this table as a decision variable (control) and improves its entries sequentially until the desired solution is obtained. In every iteration, the optimization module in DSO changes job priorities (values in Sr1Priority and Sr2Priority columns). Then, the simulation is triggered to simulate the model based on new data-table entries and runs replications to calculate the expected value of objective functions. These results are transferred to the Optimization Module to generate new solutions. This cycle repeats until the stopping criteria (which are usually set by the user) are met.

To analyze the performance of DSO, its results are compared with two heuristic methods available in the literature for solving flow shop scheduling.

- **Heuristic 1- SPT:** Shortest Processing Time or SPT has shown superior performance for job scheduling in many research investigations [23]. By ranking jobs based on the ascending order of their processing times, SPT minimizes the total completion time.
- **Heuristic 2- EDD:** The second heuristic is EDD (Earliest Due Date) which arranges job orders to minimize the total tardiness cost of jobs [24].

The applied PSO algorithm in DSO uses a weighted average of both objective functions to solve the problem (Eq. 14).

$$\min_{x,y,t,v} \quad w_1 \times \mathbb{E}_v \left[TIS\left(x, y, t,\ v\right)\right] + w_2 \times \mathbb{E}_v \left[TTC\left(x, y, t,\ v\right)\right] \tag{14}$$

The performance of the calculated optimal solution provided by DSO is compared with heuristic results provided by SPT and EDD. Each of these solutions is replicated 200 times and the ultimate results are plotted in Fig. 4. These results suggest the superiority of DSO over SPT and EDD in terms of both objective functions (time in the system and tardiness cost). With a smooth and straightforward implementation, DSO could efficiently improve the prioritization of jobs and provide competitive results.

Insight 1: The applicability of DSO can be easily extended to a flow shop model with more servers. To include a new server in the simulation model, a new column needs to be added to the data-table to represent jobs' priority on that server (i.e. Sr3Priorirty). In the optimization algorithm, the size of decision variables ($nVar$) directly depends on the number of jobs (n) and the number of servers m in the model ($nVar = n \times m$). So, adding a new server or more jobs just needs a slight change in the optimization algorithm and updating $nVar$ parameter.

4.2 Experiment 2: Job Sequencing with DSO

The second experiments demonstrate the applicability of DSO in solving a job sequencing problem in a multi-stage flow shop system (known as assembly flow

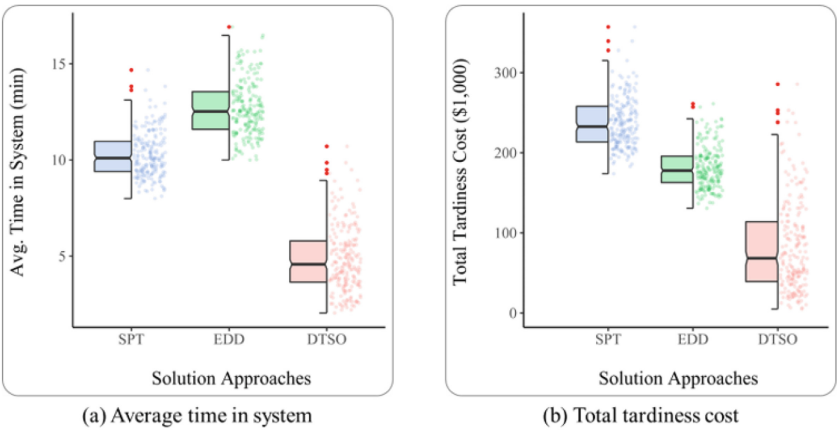


Fig. 4. Experimental results of the flow shop scheduling study.

shop). In this case, 90 jobs are released to the floor with 3 stages and 5 servers in each. The model assumptions are:

- Each machine can process only one operation at a time.
- Assembly or post-processing stages begin readily after all previous stage operations are completed.
- All machines are ready to be scheduled in time zero.
- Preemption of operations of each job is not allowed.
- The setup time is zero.

Figure 5 depicts the simulation environment where this assembly shop is developed. Objective functions are to minimize both i) average Time In the System (TIS), and ii) Total Tardiness Cost (TTC) of all jobs. In all stages, the processing time of jobs is set to $normal(20, 2)$ min, and due dates are uniformly distributed using $uniform(50, 250)$ min. There are five (5) homogenous servers

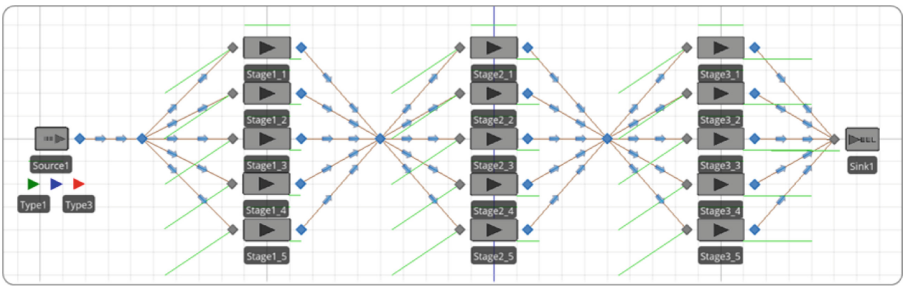


Fig. 5. Multi-stage flow shop system developed in Simio.

in each stage and each job has to follow a sequence of tasks to complete the assembly.

The goal is to find the best sequence for each job in each stage. In other words, the ideal solution should determine which server is selected in each stage to process a given job. To deploy this, a data-table is utilized in the simulation model to set up the sequencing logic. Figure 6 shows a screenshot of this table where each row represents a given job arrival time and its sequence in different stages. Three highlighted columns (Stage1Sr, Stage2Sr, and Stage3Sr) indicate the server IDs (1 to 5) that each job has to go through sequentially from stage 1 to 3 before leaving the system.

Bound to Excel: Case2.xlsx, Worksheet: Table2
Data has not been imported or is of an unknown age

	Arrival Time	Stage1Sr	Stage2Sr	Stage3Sr
▸ 1	10/26/2018 8:00:00 AM	3	3	5
2	10/26/2018 8:00:00 AM	1	3	2
3	10/26/2018 8:00:00 AM	2	3	3
4	10/26/2018 8:00:46 AM	4	5	5
⋮	⋮	⋮	⋮	⋮
89	10/26/2018 1:31:27 PM	3	5	3
90	10/26/2018 1:32:37 PM	1	4	4

Fig. 6. Data-table input structure for the job sequencing problem.

To optimize this problem, the DSO platform explores different combinations of sequences for all jobs and provides a solution that minimizes objective functions. Two heuristics rules are considered to evaluate the quality of DSO results. These heuristics are:

- **Heuristic 1- Cyclic:** In each stage, this rule selects servers cyclically to carry out new jobs.
- **Heuristic 2- LLS:** This routing rule, selects a server with the lowest service load (LLS) upon a new job arrival to the stage.

The obtained solutions of these three approaches are simulated with 200 replications to estimate the expected value of objective functions, TIS, and TTC. The jitter-boxplots of these results are presented in Fig. 7, where DSO performance is adequately better than heuristics. The optimization algorithm in DSO could find a solution with higher quality and less variability. The significance of this difference is tested using one-way ANOVA for TIS and TTC objectives (Table 1 and Table 2). The p-value of both tests is low ($p < 0.001$), which appears that DSO's superiority is statistically significant.

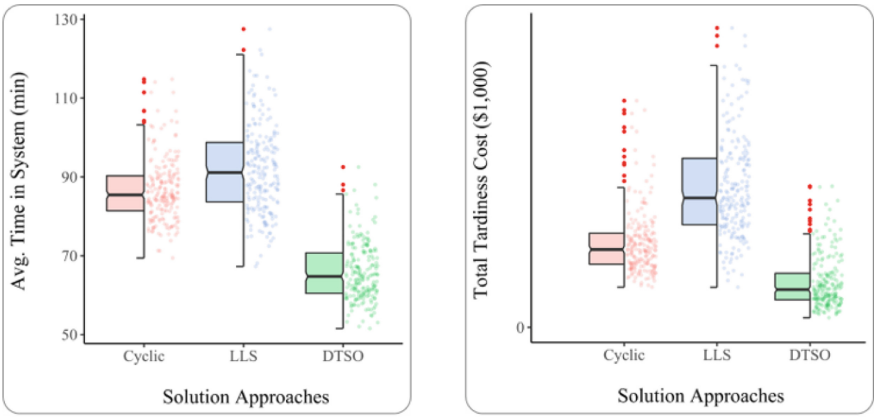


Fig. 7. Data-table input structure for the job sequencing problem.

Table 1. One-way ANOVA results for the average time in the system results.

Source of variation	SS	df	MS	F	P-value	F critical
Between groups	76580.58	2	38290.29	474.10	5.2E−124	3.010815
Within groups	48215.91	597	80.76			
Total	124796.5	599				

Table 2. One-way ANOVA results for the total tardiness cost.

Source of variation	SS	df	MS	F	P-value	F critical
Between groups	2.63E+08	2	1.32E+08	309.3843	6.31E−93	3.010815
Within groups	2.54E+08	597	425708.2			
Total	5.18E+08	599				

Insight 2: In this experiment, DSO solved the problem of sequencing for 90 jobs in 3 stages ($nVar = 270$). To solve this problem, OptQuest requires at least 180 controls (properties) to find the solution; whereas, DSO takes the data-table as a decision variable and evolves its values until the desired solution is achieved.

5 Conclusion and Future Works

For more than fifty years, simulation has been extensively applied by researchers and developers. Traditionally, it is appealing to equip simulation with optimization to tackle stochasticity and the complexity of problems. Despite tremendous advancements in SO techniques, designing well-established models is still demanding by today's standards [1]. In addition, simulation software packages

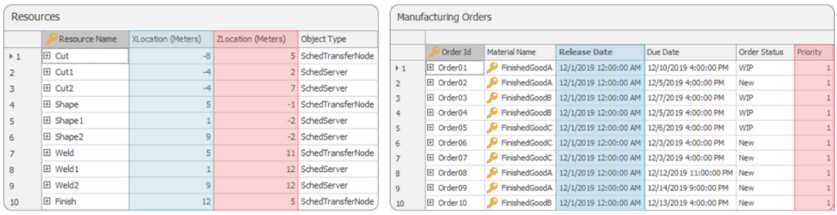
fail to incorporate human decision-making analysis or highly computational support tools [25].

This article introduced an innovative interaction between simulation and optimization to carry out the decision-making process based on table-table inputs. Unlike the existing commercial software packages, the proposed DSO framework can efficiently solve problems with a large set of parameters and data-tables. By developing an application programming interface, called Simio-API, this framework connects three modules, simulation, optimization, and data exchange. This integration can run simulation experiments with multiple data-table settings, evolve their values, and achieve satisfactory results.

The usefulness of the proposed framework is demonstrated in two experimental scenarios, 1) job scheduling in a flow shop system, and 2) Job sequencing in a multi-stage flow shop system. These experiments demonstrated DSO's applicability and efficiency. More importantly, some insights are provided to show how the new model can be implemented and extended to new works.

Introducing DSO offers new opportunities to the community and paves a new avenue of research for theory and practice. Nowadays, many simulation software developers value data-driven models. With the emergence of new simulation techniques such as data-generated modeling and Digital Twins, the usefulness of DSO can become more obvious. The concept of data-generated modeling is based on creating a simulation model automatically using data-tables. This populates a complete simulation model from scratch by adding objects to the environment and mapping them to the tables. These tables can include all of the modeling needs such as resources' information, entities, networks, transports, schedules, tasks, etc. Lately, Simio announced a newly added feature to its software to build data-generated models [26]. By leveraging DSO, one can optimize different components of the model systematically without the need for manual changes. For instance, optimizing system layouts (i.e. hospitals, manufacturing systems, etc.) are traditionally limited to a few scenarios suggested by layout designers. By taking advantage of data-generated modeling in Simio and DSO, one can easily change the layout, make simulation models instantly, and experiment with an abundance of layouts. As shown in Fig. 8-a, this requires defining object coordinates as decision variables (XLocation and ZLocation columns) for DSO and let it solve the problem. Another example could be optimizing manufacturing orders where orders' release date and priority need to be optimized (Fig. 8-b).

Another future research direction is to explore other optimization techniques in the model. To keep the focus of the paper on the platform, just one algorithm (PSO) is used in experiments. However, there are plenty of SO techniques that can be borrowed and tested within DSO design. Multiple metaheuristic algorithms can be utilized to develop data-table Simheuristics and solve the problems. By increasing the size of data-tables (and decision variables), neural networks can be added to the optimizer and make the model computationally efficient by approximating objective functions.



Resources

	Resource Name	XLocation (Meters)	ZLocation (Meters)	Object Type
1	Cut	-8	5	SchedTransferNode
2	Cut1	-4	2	SchedServer
3	Cut2	-4	7	SchedServer
4	Shape	5	-1	SchedTransferNode
5	Shape1	1	-2	SchedServer
6	Shape2	9	-2	SchedServer
7	Weld	5	11	SchedTransferNode
8	Weld1	1	12	SchedServer
9	Weld2	9	12	SchedServer
10	Finish	12	5	SchedTransferNode

(a) Resources data-table

Manufacturing Orders

	Order Id	Material Name	Release Date	Due Date	Order Status	Priority
1	Order01	FinishedGoodA	12/1/2019 12:00:00 AM	12/10/2019 4:00:00 PM	WIP	1
2	Order02	FinishedGoodA	12/1/2019 12:00:00 AM	12/5/2019 4:00:00 PM	New	1
3	Order03	FinishedGoodB	12/1/2019 12:00:00 AM	12/7/2019 4:00:00 PM	WIP	1
4	Order04	FinishedGoodB	12/1/2019 12:00:00 AM	12/5/2019 4:00:00 PM	WIP	1
5	Order05	FinishedGoodC	12/1/2019 12:00:00 AM	12/6/2019 4:00:00 PM	WIP	1
6	Order06	FinishedGoodC	12/1/2019 12:00:00 AM	12/3/2019 4:00:00 PM	New	1
7	Order07	FinishedGoodC	12/1/2019 12:00:00 AM	12/3/2019 4:00:00 PM	New	1
8	Order08	FinishedGoodA	12/1/2019 12:00:00 AM	12/12/2019 11:00:00 PM	New	1
9	Order09	FinishedGoodA	12/1/2019 12:00:00 AM	12/14/2019 9:00:00 PM	New	1
10	Order 10	FinishedGoodB	12/1/2019 12:00:00 AM	12/13/2019 4:00:00 PM	New	1

(b) Manufacturing orders data-table

Fig. 8. Examples of data-generated modeling in Simio.

A Appendix A

Particle Swarm Optimization or PSO is a population-based Metaheuristic algorithm developed by Kennedy and Eberhart in 1995 [27] PSO is a swarm-based algorithm and by moving particles in a specific exploration field [28]. Due to its effective balancing of exploration and exploitation [29], PSO has been widely used in the development of Simheuristic models and in solving SO problems. Recent examples include using PSO to deal with stochastic models in supply chain management [30], healthcare systems [31], and manufacturing [32]. The general pseudocode of PSO is shown in Algorithm 2.

Algorithm 2: Pseudo-code of Particle Swarm Optimization (PSO)

1: // **Initialization** ▷ Generate particles
   for $i = 1 : N_s$ **do**
2:
   Initialize $s_i(t = 0)$
3: Initialize $v_i(t = 0)$
4: $P_i^{best} \leftarrow s_i$

5: // **PSO loop**
6: $G^{best} = 0$ **for** $t = 1 : Max_{it}$ **do**
   for $i = 1 : N_s$ **do**
7:
   $v_i(t+1) = wv_i(t) + c_1 r_1[P_i^{best} - s_i(t)] + c_2 r_2[G^{best} - s_i(t)]$
8: $s_i(t+1) \leftarrow s_i(t) + v_i(t+1)$
9: Evaluate $fitness s_i(t+1)$
   if $fitness(P_i^{best}) < fitness(s_i(t+1))$ **then**
10:
   $P_i^{best} \leftarrow s_i(t+1)$ ▷ Update Personal best **if** $fitness(G^{best}) < fitness(P_i^{best})$ **then**
11:
   $G^{best} \leftarrow P_i^{best}$ ▷ Update Global best
12: $t \leftarrow t + 1$

Notations:
N_s : Swarm size, $i = 1, 2, ..., N_s$: Particles index
s_i : Solution (particle), v_i : Velocity, w : Inertia weight
P_i^{best} : Personal best solution, c_1 : Personal learning factor
G^{best} Global best, c_2 : Global learning factor
r_1, r_2 : Random numbers $\sim u(0, 1)$
MAX_{it} : Max number of iterations, $t : 0, 1, 2, ..., MAX_{it}$: Iteration index

References

1. Figueira, G., Almada-Lobo, B.: Hybrid simulation-optimization methods: a taxonomy and discussion. Simul. Modell. Pract. Theory Simul.-Optim. Complex Syst.: Methods Appl. **46**, 118–134 (2014). ISSN 1569-190X. https://doi.org/10.1016/j.simpat.2014.03.007, http://www.sciencedirect.com/science/article/pii/S1569190X14000458. Accessed 29 May 2016
2. Amaran, S., Sahinidis, N.V., Sharda, B., Bury, S.J.: Simulation optimization: a review of algorithms and applications. 4OR **12**(4), 301–333 (2014). http://link.springer.com/article/10.1007/s10288-014-0275-2. Accessed 01 May 2017
3. Smith, J.S., Sturrock, D.T., Kelton, W.D.: Simio and Simulation: Modeling, Analysis, Applications: 4th Edition - Economy, English, 4 edn. CreateSpace Independent Publishing Platform (2017). ISBN 978-1-5464-6192-0
4. Carson, Y., Maria, A.: Simulation optimization: methods and applications. In: Conference Proceedings, pp. 118–126. IEEE Computer Society (1997)
5. Fu, M.C., Henderson, S.G.: History of seeking better solutions, AKA simulation optimization. In: 2017 Winter Simulation Conference (WSC), pp. 131–157. IEEE (2017)
6. Ólafsson, S., Kim, J.: Simulation optimization. In: Proceedings of the Winter Simulation Conference, vol. 1, pp. 79–84. IEEE (2002)
7. Dehghanimohammadabadi, M., Kabadayi, N.: A two-stage AHP multi- objective simulation optimization approach in healthcare. Int. J. Anal. Hierarchy Process **12**(1), 117–135 (2020)
8. Azadeh, A., Ahvazi, M.P., Haghighii, S.M., Keramati, A.: Simulation optimization of an emergency department by modeling human errors. Simul. Modell. Pract. Theory **67**, 117–136 (2016)
9. Rezaeiahari, M., Khasawneh, M.T.: Simulation optimization approach for patient scheduling at destination medical centers. Expert Syst. Appl. **140**, 112 881 (2020)
10. Seif, J., Dehghanimohammadabadi, M., Yu, A.J.: Integrated preventive maintenance and flow shop scheduling under uncertainty. Flex. Serv. Manuf. J. **32**, 852–887 (2020). https://doi.org/10.1007/s10696-019-09357-4
11. Aiassi, R., Sajadi, S.M., Molana, S.M.H., Babgohari, A.Z.: Designing a stochastic multi-objective simulation-based optimization model for sales and operations planning in built-to-order environment with uncertain distant outsourcing. Simul. Modell. Pract. Theory **104**, 102103 (2020)
12. Amiri, F., Shirazi, B., Tajdin, A.: Multi-objective simulation optimization for uncertain resource assignment and job sequence in automated flexible job shop. Appl. Soft Comput. **75**, 190–202 (2019)
13. Drenovac, D., Vidović, M., Bjelić, N.: Optimization and simulation approach to optimal scheduling of deteriorating goods collection vehicles respecting stochastic service and transport times. Simul. Modell. Pract. Theory **103**, 102 097 (2020)
14. Kabadayi, N., Dehghanimohammadabadi, M.: Multi-objective supplier selection process: a simulation-optimization framework integrated with MCDM. Ann. Oper. Res. **319**, 1607–1629 (2022). https://doi.org/10.1007/s10479-021-04424-2
15. Vieira, A.A., Dias, L., Santos, M.Y., Pereira, G.A., Oliveira, J.: Are simulation tools ready for big data? Computational experiments with supply chain models developed in Simio. Proc. Manuf. **42**, 125–131 (2020)
16. Goodarzian, F., Hosseini-Nasab, H., Muñuzuri, J., Fakhrzad, M.-B.: A multi-objective pharmaceutical supply chain network based on a robust fuzzy model: a comparison of meta-heuristics. Appl. Soft Comput. **92**, 106 331 (2020)

17. Swain, J.J.: Simulated worlds. OR/MS Today **42**(5), 36–49 (2015)
18. Laguna, M.: Optimization of Complex Systems with OptQuest. A White Paper from OptTek Systems Inc. (1997)
19. Hein, D.L., Harrell, C.R.: Simulation modeling and optimization using ProModel. In: 1998 Winter Simulation Conference. Proceedings (Cat. No. 98CH36274), vol. 1, pp. 191–197. IEEE (1998)
20. Juan, A.A., Faulin, J., Grasman, S.E., Rabe, M., Figueira, G.: A review of simheuristics: extending metaheuristics to deal with stochastic combinatorial optimization problems. Oper. Res. Perspect. **2**, 62–72 (2015)
21. Xu, J., Huang, E., Hsieh, L., Lee, L.H., Jia, Q.-S., Chen, C.-H.: Simulation optimization in the era of industrial 4.0 and the industrial internet. J. Simul. **10**(4), 310–320 (2016)
22. Jian, N., Freund, D., Wiberg, H.M., Henderson, S.G.: Simulation optimization for a large-scale bike-sharing system. In: 2016 Winter Simulation Conference (WSC), pp. 602–613. IEEE (2016)
23. Pegden, C.D.: Introduction to SIMIO. In: 2008 Winter Simulation Conference, pp. 229–235. IEEE (2008)
24. Sturrock, D.T.: Traditional simulation applications in industry 4.0. In: Gunal, M.M. (ed.) Simulation for Industry 4.0. SSAM, pp. 39–54. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-04137-3_3
25. Jules, G., Saadat, M., Saeidlou, S.: Holonic goal-driven scheduling model for manufacturing networks. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics, pp. 1235–1240. IEEE (2013)
26. Dehghanimohammadabadi, M.: Iterative optimization-based simulation (IOS) with Predictable and unpredictable trigger events in simulated time. Ph.D. thesis, Western New England University (2016). http://gradworks.umi.com/10/03/10032181.html. Accessed 30 May 2016
27. Dehghanimohammadabadi, M., Keyser, T.K.: Intelligent simulation: integration of SIMIO and MATLAB to deploy decision support systems to simulation environment. Simul. Modell. Pract. Theory **71**, 45–60 (2017). http://www.sciencedirect.com/science/article/pii/S1569190X16301356. Accessed 17 Dec 2016
28. Sturrock, D.T.: Using commercial software to create a digital twin. In: Gunal, M.M. (ed.) Simulation for Industry 4.0. SSAM, pp. 191–210. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-04137-3_12
29. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995-International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
30. Shaheen, M.A., Hasanien, H.M., Alkuhayli, A.: A novel hybrid GWOPSO optimization technique for optimal reactive power dispatch problem solution. Ain Shams Eng. J. **12**, 621–630 (2020)
31. Usman, M., Pang, W., Coghill, G.M.: Inferring structure and parameters of dynamic system models simultaneously using swarm intelligence approaches. Memetic Comput. **12**(3), 267–282 (2020)
32. Park, K.: A heuristic simulation-optimization approach to information sharing in supply chains. Symmetry **12**(8), 1319 (2020)