Chapter 5

VALIDATION AND OPTIMIZATION OF AN ELEVATOR SIMULATION MODEL WITH MODERN SEARCH HEURISTICS

Thomas Bartz–Beielstein,[1] Mike Preuss[1] and Sandor Markon[2]

[1] *Universitität Dortmund*
D-44221 Dortmund, Germany
{tom, preuss}@LS11.cs.uni-dortmund.de

[2] *FUJITEC Co.Ltd. World Headquarters*
28-10, Shoh 1-chome, Osaka, 567-8511 Japan
markon@rd.fujitec.co.jp

Abstract: Elevator supervisory group control (ESGC) is a complex combinatorial optimization task that can be solved by modern search heuristics. To reduce its complexity and to enable a theoretical analysis, a simplified ESGC model (S-ring) is proposed. The S-ring has many desirable properties: Fast evaluation, reproducibility, scalability, and extensibility. It can be described as a Markov decision process and thus be analyzed theoretically and numerically. Algorithm based validation (ABV), as a new methodology for the validation of simulation models, is introduced. Based on ABV, we show that the S-ring is a valid ESGC model. Finally, the extensibility of the S-ring model is demonstrated.

Keywords: Elevator group control, optimization, discrete-event simulation models, validation, search heuristics, evolutionary algorithms, Markov decision processes.

5.1 INTRODUCTION

Today's urban life cannot be imagined without elevators. The central part of an elevator system, the elevator group controller, assigns elevator cars to service calls in real-time while optimizing the overall service quality, the traffic throughput, and/or the energy consumption. The elevator supervisory group control (ESGC) problem can be classified as a combinatorial optimization problem (Barney, 1986; So and

Chan, 1999; Markon and Nishikawa, 2002). It reveals the same complex behavior as many other stochastic traffic control problems, i.e. materials handling systems with automated guided vehicles (AGVs). Due to many difficulties in analysis, design, simulation, and control, the ESGC problem has been studied for a long time. First approaches were mainly based on analytical methods derived from queuing theory, whereas currently computational intelligence (CI) methods and other heuristics are accepted as state of the art (Crites and Barto, 1998; Schwefel et al., 2003).

In this article we propose a simplified ESGC system, the sequential ring (S-ring). The S-ring is constructed as a simplified model of an ESGC system using a neural network (NN) to control the elevators. Some of the NN connection weights can be modified, whereby testing different weight settings and their influence on the ESGC performance is enabled. The performance of one specific NN weight setting $\vec{x}$ is based on simulations of specific traffic situations, which automatically lead to stochastically disturbed (noisy) objective function values $\tilde{f}(\vec{x})$. Since it is difficult for an optimization algorithm to judge the fitness $f(\vec{x})$ of one ESGC configuration, the determination of the optimal weight setting $\vec{x}^*$ is not trivial. Direct search methods that rely on the direct comparison of function values face the problem of modifying the weights without generating too many infeasible solutions.

The S-ring was introduced as a benchmark problem to enable a comparison of ESGC algorithms, independently of specific elevator configurations (Markon et al., 2001; Markon and Nishikawa, 2002). Results from the S-ring, obtained with low computational costs, should be transferable to more complex ESGC models.
In the following, we will present different techniques to answer the question whether the S-ring is a simplified, but valid ESGC simulation model. We propose a new validation methodology that takes the optimization algorithm for the simulation model into account. Tuning the optimization algorithm for the simplified simulation model results in a good parameter setting of the optimization algorithm. This setting is also applicable to the complex simulation model.

It is generally assumed that the performance of an algorithm depends on the structure of the search space. (Reeves and Yamada, 1998) observed that local optima of randomly generated instances of an optimization problem are distributed in a 'big-valley' structure. For example, better local optima tend to be closer to the global optimum. This big-valley structure is convenient for many search algorithms. But do real-world problem instances – as opposed to randomly generated problem instances – possess a big-valley structure? Experiments indicate that this is *not* the case. As (Whitley et al., 2002) write: 'Local optima are generally members of large plateaus of equally-fit solutions.' This plateau-like distribution has also been observed while optimizing the S-ring and real ESGC problems alike. Additionally, knowledgeable persons have evaluated the model output for reasonableness. Thus, improved algorithm parameter settings obtained from simulation results on the S-ring should be transferable to real ESGC problems. S-ring simulations might give valuable hints for the optimization of highly complex elevator group controller optimization tasks.

The rest of this article is organized as follows: In section 5.2, we introduce the elevator group control problem. Section 5.3 discusses S-ring basics, whereas section 5.4 presents simulation and analysis techniques. Section 5.5 demonstrates the validity of this model simplification. The extensibility of the S-ring model is demonstrated in section 5.6. The final section combines a summary with an outlook.

5.2 THE ELEVATOR SUPERVISORY GROUP CONTROLLER PROBLEM

The elevator group controller determines the floors where the cars should go to. Additional elevator controllers handle the functions inside the car, such as door control, measurement of the car load, and car calls. Since the group controller is responsible for the allocation of elevators to hall calls, a control strategy to perform this task in an optimal manner is required. The main goal in designing a better controller is to minimize the time passengers have to wait until they can enter an elevator car after having requested service. This time-span is called the waiting time. The so-called service time additionally includes the time a passenger stays within the elevator car.

An important aspect is the changing traffic pattern we can observe throughout the day in an office building (Markon, 1995). There is 'up-peak' traffic in the morning when people start to work and symmetrically we observe 'down-peak' traffic in the evening. Most of the day there is 'balanced' traffic with much lower intensity than at peak times. 'Lunchtime' traffic consists of two - often overlapping - phases where people first leave the building for lunch or head for a restaurant floor, and then get back to work. The ESGC problem subsumes the following problem: How to *assign elevators to passengers* in real-time while optimizing different elevator configurations with respect to overall service quality, traffic throughput, energy consumption etc.

Fujitec, one of the world's leading elevator manufacturers, developed a controller that is trained by use of a set of fuzzy controllers. Each controller represents control strategies for different traffic situations (Markon, 1995). The NN structure and the neural weights determine a concrete control strategy. The network structure as well as many of the weights remain fixed, only some of the weights on the output layer can be modified and optimized. A discrete-event based elevator group simulator permits computing the controller's performance. This highly complex ESGC simulation model will be referred to as the 'lift model' (or simply 'lift') throughout the rest of this paper.

The identification of globally optimal NN weights is a complex optimization problem. The distribution of local optima in the search space is unstructured and there are many local minima on flat plateaus. The objective function values are stochastically disturbed due to the non-determinism of service calls, and dynamically changing with respect to traffic loads. (Arnold and Beyer, 2003) compared evolution strategies (ES) with other search methods in the presence of noise. Their results indicate that gradient based optimization techniques cannot be applied successfully to this optimization problem.

(Beielstein et al., 2003) applied evolution strategies to determine optimal NN weights. Their lift model has been implemented as follows: The objective function considers different time dependent traffic patterns as described above. Let the *handling capacity* of an elevator system be defined as the maximum number of customers an elevator system is able to serve per hour without exceeding an average waiting time. We considered handling capacities at 30, 35, and 40 seconds. This results in a multi-criteria optimization problem. The different objectives are aggregated to obtain a single-criteria optimization problem by averaging handling capacities and then subtracting from 3,000 pass./h to obtain a minimization problem. The latter value was empirically chosen as an upper bound for the given scenario. The resulting fitness function reads: $F(\vec{x}) = 3000.0 - \overline{f}_{\vec{a}}(\vec{x})$, where $\overline{f}_{\vec{a}}$ is the average handling capacity (pass./h), $\vec{a}$ is the parameter design of the evolution strategy optimization algorithm, and $\vec{x}$ is a 36 dimensional vector that specifies the NN weights. $F(\vec{x})$ is called the 'inverse handling capacity'. The computational effort for single simulator runs limits the maximum number of fitness function evaluations to the order of magnitude 10^4.

In general, ESGC research results are incomparable, since the elevator group control *per se* is not appropriate as a 'benchmark problem':
– Elevator systems have a very large number of parameters that differ widely among buildings, elevator models, manufacturers etc.
– Elevator cars have complex rules of operation, and even slight differences, e.g. in door operation or in the conditions for changing the traveling direction, can affect the system performance significantly. Even

the smallest elevator system has a very large state space, making direct solution infeasible, thus no exact solutions are available for comparison. – The sophisticated ESGC rules are usually trade secrets of the manufacturers, and cannot made commonly available for research.

In principle, the optimization practitioner can cope with the enormous complexity of the ESGC problem in two different ways: The problem can be simplified or resources can be used extensively (i.e. parallelization, see, e.g. (Beielstein et al., 2003c)). We will concentrate on the first strategy and present a simplified ESGC model. Ideally, a simplified ESGC model should comply with the following requirements: It enables fast and reproducible simulations and is applicable to different building and traffic configurations. Furthermore it must be a valid simplification of a real elevator group controller and thus enable the optimization of one specific controller policy and the comparison of different controller policies. The simplified model should be scalable to enable the simulation of different numbers of floors or servers. It should be extensible, so that new features (i.e. capacity constraints) can be added. Last but not least, the model is expected to favor a theoretical analysis. We propose a model that conforms to all these requirements in the next section.

5.3 S-RING BASICS

When passengers give a hall call, they simply press a button. Therefore, only a one bit information for each floor is sent to the ESGC. It appears intuitively correct to map the whole state of the system to a binary string. The system dynamic is represented by a state transition table and can be controlled by a policy. The sequential-ring model (S-ring model) has only a few parameters: The number of elevator cars m, the number of queues n, and the passenger arrival rate p (Markon et al., 2001). A 2-bit state (s_i, c_i) is associated with each site. The s_i bit is set to 1 if a server is present on the ith floor, to 0 otherwise. Correspondingly, the c_i bit is set to 0 or 1 if there is no waiting passenger resp. at least one waiting passenger. Figure 5.1 depicts a typical S-ring configuration. The state at time t is given as

$$x(t) := (s_0(t), c_0(t), \dots, s_{n-1}(t), c_{n-1}(t)) \in \mathbf{X} = \mathbb{B}^{2n}, \tag{5.1}$$

with $\mathbb{B} := \{0, 1\}$. A transition probability function f, a decision function δ, and a reward function r are used to determine the dynamic of the system. A look-up table as shown in table 5.1 can be used to represent f, δ, and r in a compact manner. We will give a formal definition of the S-ring in the appendix (definition 1).

The state evolution is sequential (S-ring stands for sequential ring), scanning the sites from $n - 1$ down to 0, and then again around from

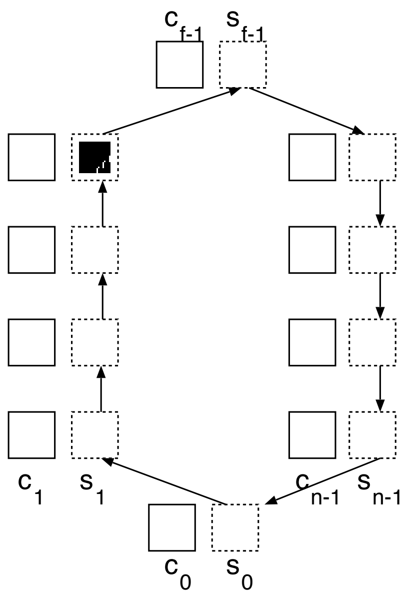
c_{f-1}
s_{f-1}
c_1
s_1
c_{n-1}
s_{n-1}
c_0
s_0

Table 5.1. The triple in the first column represents the state of the actual site: Customer waiting, server present, and server present on the next floor. The probability of a state change to the state in the fourth column is given in the second column. Columns three and five denote the decision and the reward respectively. I.e., the server has to make a decision (to 'take' or to 'pass' the customer) if there is a customer waiting (1xx), and if there is a server present on the same floor (11x) but no server on the next floor (110).

$\xi(t)$	Prob	$\pi(x)$	$\xi(t+1)$	Δr
000	$1-p$		000	0
	p		100	-1
100	1		100	0
010	$1-p$		001	0
	p	0	101	-1
		1	010	0
110	1	0	101	0
		1	010	$+1$
001	$1-p$		001	0
	p		101	-1
101	1		101	0
011	1		011	0
111	1		011	$+1$

the unobservable case. Next we will introduce some elementary policies:
– The most obvious heuristic policy is the greedy one: When given the choice, always serve the customer: $\pi^g(o) \equiv 1$. Rather counter-intuitively, this policy is not optimal, except in the heavy traffic ($p > 0.5$) case. This means that a good policy must bypass some customers occasionally. The greedy-policy does not take any information about the state of the system into account.
– The random policy is another trivial policy that leads to rather poor results. For some given $\sigma \in [0,1]$, we can define $\pi^b(o) = 0$ with probability (w. pr.) $1-\sigma$, and 1 otherwise. Actions based on the random-policy require no information about the actual system state.
– A quite good heuristic policy, that takes information about the actual state of the system into account, is the balance-policy: $\pi^b(o) = 0$, if $s_{n-1} = 1$, and 1 otherwise. The intention is to put some distance between servers by passing when there is another tailing server, letting it serve the customer: Waiting customers on the $(n-1)$th floor queue are not served by the leading server, thus a gap is created between the leading and the following server. Balancing the positioning of the servers, π^b is significantly better than π^g for medium values of p.

Finally, we present the perception policy representation: Let $\theta : \mathbb{R} \to \mathbb{B}$ define the Heaviside function (see definition 4), and $x = x(t)$ be the state at time t (see equation 5.1), and $y \in \mathbb{R}^{2n}$ be a weight vector. A linear discriminator, or perceptron, $\pi^p(x) = \theta(y^T \cdot x)$, can be used to present the policy in a compact manner. The perception presentation can be used to encode the other policies mentioned above. Variants of this policy require information on the complete state of the current system, since the state vector x is used.

5.3.1 The S-Ring Model as an Optimization Problem

The 'optimization via simulation' approach requires the definition of a performance measure for the simulation model. The long-run time-average number of customers in the system or in the queue (Q) are commonly used performance measures in queuing theory (Banks et al., 2001). Consider a simulation run of a queuing system over a period of time T. The steady-state time-average number in queue is

$$Q := \lim_{T\to\infty} \frac{\int_0^T Q(t)dt}{T} \quad \text{w.pr. 1.} \tag{5.2}$$

The basic optimal control problem is to find a policy π^* for a given S-ring configuration $S \in \mathcal{S}$ (see definition 1), so that the expected number of sites with waiting passengers Q, that is the steady-state time-average as defined in equation 5.2, in the system is minimized:

$$\pi^* = \arg\min_{\pi} Q(\pi). \tag{5.3}$$

Equivalently, a policy π is optimal, if it maximizes the expected reward. The general S-ring problem: *For a given S-ring* S, *find the optimal policy* π^*, can be modified to the

PROBLEM 1 (PERCEPTRON S-RING PROBLEM) *For a given S-ring* S, *find the weight vector* $y \in \mathbb{R}^{2n}$ *that represents the optimal policy* π^*.

The perceptron S-ring problem can serve as a benchmark problem for many optimization algorithms, since it relies on the fitness function $F : \mathbb{R}^{2n} \to \mathbb{R}$ (Markon et al., 2001; Beielstein and Markon, 2002). In general, π can be realized as a look-up table of the system state x and π^* is found by enumerating all possible π and selecting the one with the lowest Q. Since this count grows exponentially with n, the naive approach would not work for any but the smallest cases.

5.4 ANALYSIS AND SIMULATION OF THE S-RING SYSTEM

The S-ring system can be interpreted as a Markov decision process. Let $x(t) \in \mathcal{X}$ denote the state of the S-ring system at time-step t, where $\mathcal{X}$ denotes the state space of the S-ring system. A single state transition describes the changes of the system, if the kth floor queue is considered. A transition cycle describes the changes of the system, if the n sites ($k = 0, 1, \ldots, n-1$) are considered in sequence. The N different system states can be enumerated using the function $s_{\text{num}} : \mathbb{B}^{2n} \to \{0, 1, \ldots, 2^{2n} - 1\}$ defined as $s_{\text{num}}(x(t)) := \sum_{i=1}^{n} 2^{i-1}(s_i + 2^{n-1}c_i)$, and the function s_{legal}, that determines the feasibility of a state ($\sum_{i=1}^{n} s_i = m$).
If the kth floor queue is scanned, the corresponding state transition can be described by a S-ring single state transition matrix $\mathbf{P_k}$. The matrix element (p_{ij}) defines the state transition probability from state i to state j. The single state transition matrices can be multiplied to obtain the transition cycle matrix: $\mathbf{P} := \prod_{i=1}^{n} \mathbf{P_{n-i+1}}$. Based on $\mathbf{P}$, we can determine the limiting state probability distribution.

Example 1 *Even the simplest non-trivial case with $n = 3$ floors and $m = 2$ elevators requires $2^3 \cdot \binom{3}{2} = 24$ different states. Based on the limiting state probability distribution $\vec{\pi}$ and on the vector $\vec{c}$, that contains at its ith position the number of customers when the system is in the ith state, we can determine the value Q for the greedy strategy as $\vec{\pi} \cdot \vec{c} = 3 \cdot p^2/(p^4 - p^3 + 2p^2 + 1)$. E.g., if we chose $p = 0.3$, we obtain $Q = 0.2325$.*

The S-ring can be seen as a partially-observable Markov decision process (POMDP) (see definition 5). The unobservable MDP (UMDP) is a subclass of the POMDP: No information on the state of the system is available. The S-ring equipped with the random or with the greedy policy is an UMDP.
The complete state of the system is known to the observer at each time point in the fully observable Markov decision process (MDP). The perceptron S-ring is a MDP.
POMDPs can be formulated as optimization problems: I.e. for a given POMDP, the decision maker selects the policy with the maximum expected value. There exist several dynamic programming approaches for solving POMDP problems: Standard algorithms are value iteration and policy iteration (Howard, 1972). A solution by dynamic programming and by numerical methods such as Q-learning, Kiefer-Wolfowitz stochastic approximation and a (1+1)-ES is presented in (Markon and Nishikawa, 2002).

The conclusions drawn from the theoretical and numerical analysis might be complemented by simulation experiments. The S-ring can

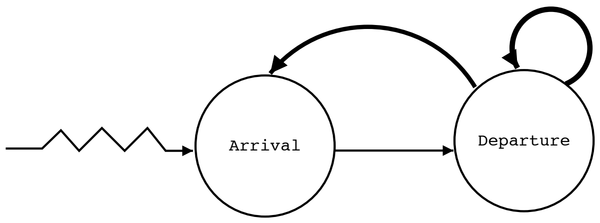

Figure 5.2. Event graph. The thin arrow represents an event at the beginning (arrival) scheduling an event at the end of the arrow (departure) *immediately.* The heavy arrows indicate that the event at the beginning of the arrow *may* schedule the event at the end of the arrow.

be treated as a discrete-event-simulation model (Banks et al., 2001). An arrival event schedules a departure event without any intervening time, whereas other events occurring at time t_i are scheduled at time $t_{i+1} := t_i + 1$. Based on the event-graph method, where each event is represented by a node, and directed arrows show how events are scheduled from other events, the S-ring can be presented as depicted in figure 5.2 (Som and Sargent, 1989). A flowchart for the departure event is shown in figure 5.3. An event-based variant of the S-ring was implemented in `simlib`. `simlib` is a C-based simulation 'language', that provides functions to accumulate and to summarize simulation data, to schedule events, and to generate random variates (Law and Kelton, 2000).

5.5 THE S-RING MODEL AS A VALID ESGC MODEL

The complete validation process requires subjective and objective comparisons of the model to the real system. Subjective comparisons are judgments of experts ('face validity'), whereas objective tests compare data generated by the model to data generated by the real system. Building a model that has a high face validity, validating the model assumptions, and comparing the model input-output transformations to corresponding input-output transformations for the real system can be seen as three widely accepted steps of the validation process (Naylor and Finger, 1967).

Important similarities of the S-ring with real elevator systems have been observed by experts: Both are found to show suboptimal performance when driven with simple greedy policies. They exhibit a characteristic instability, commonly called bunching in case of elevators.

In the following we will consider input-output transformations more detailed. The model is described by the function:

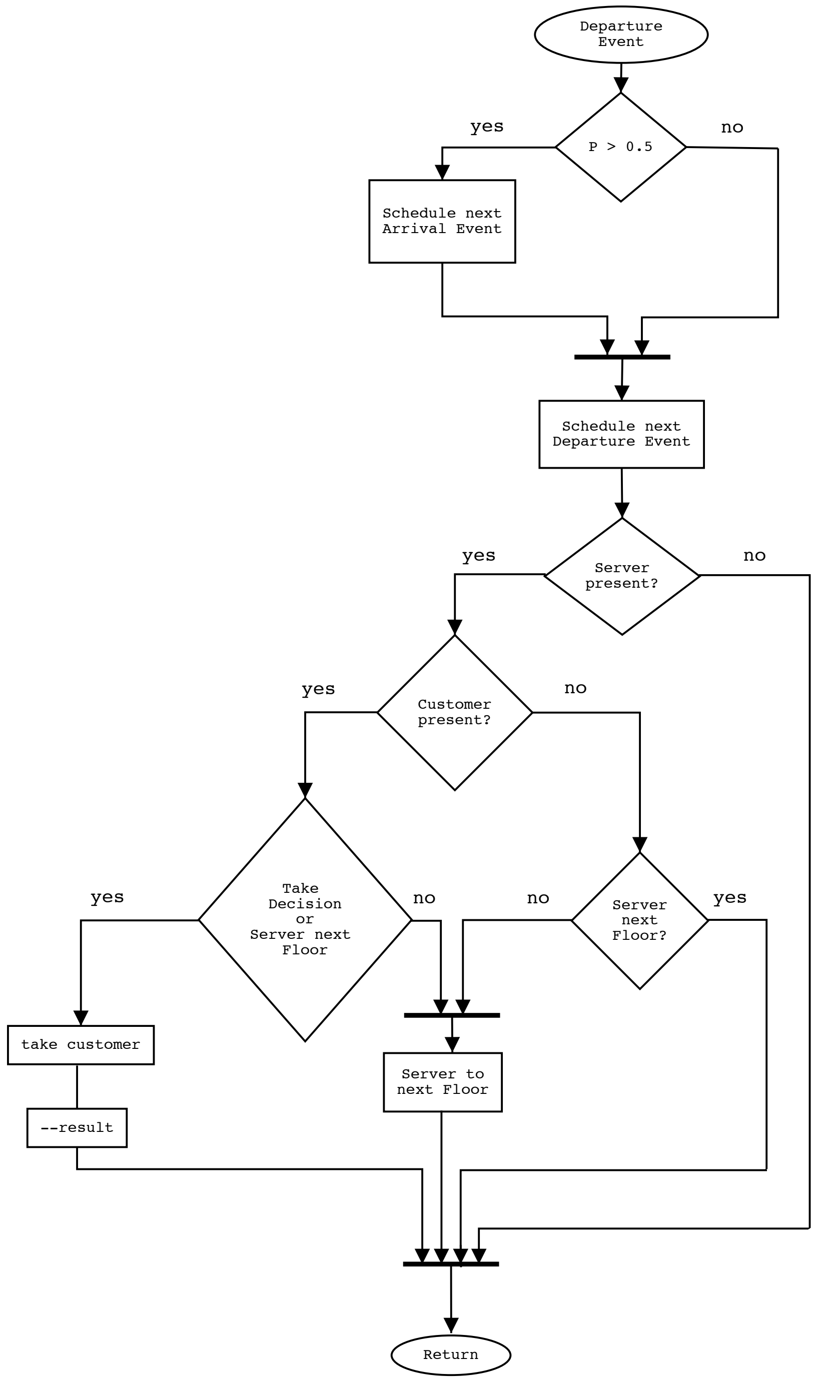

Figure 5.3. Function depart. The probability of an arrival event is set to 0.5.

$$f : (Z, D) \to Y \tag{5.4}$$

Thus values of the uncontrollable input parameters Z and values of the controllable decision variables (or of the policy) D are mapped to the output measures Y. The model can be run using generated random variates Z_i to produce the simulation-generated output measures. E.g. the S-ring model takes a policy and a system configuration and determines the expected average number of floors with waiting customers in the system using the generated random variates that determine a customer arrival event. If real system data is available, a statistical test of the null hypothesis can be conducted:

$$H_0 : \ E(Y) = \mu \text{ is tested against } H_1 : \ E(Y) \neq \mu, \tag{5.5}$$

where μ denotes the real system response and Y the simulated model response. We will extend these standard validation techniques by introducing a new approach, that takes the choice of an optimization algorithm into account. The main idea is based on the observation that the complexity of a model can only be seen in relation to the associated optimization algorithm (Naudts and Kallel, 2000): The functions $H_{n,b} : \{0,1\}^n \to \{0,1\}$ defined by $H_{n,b}(b) = 1$ and $H_{n,b}(a) = 0$ if $a \neq b$ ('needle in a haystack') can be extremely difficult for (standard) genetic algorithms, whereas they are simple for the degenerated search heuristic that generates the solution b deterministically.
We additionally assume that every problem requires a specific algorithm parameter setting $\vec{a}$ (Wolpert and Macready, 1997). $\vec{a}$ includes the exogenous parameters such as the population size in evolutionary algorithms or the starting temperature for the cooling schedule in simulated annealing. (François and Lavergne, 2001) introduce a methodology that classifies problem classes based on the parameterization of the underlying optimization algorithm. This methodology is extended in the following to 'optimization via simulation' approaches. We will give a definition first:

DEFINITION 5.1 (ALGORITHM BASED EQUIVALENCE) *Let the regression model $E(Y) = X\beta$ model the functional relationship between the algorithm A and its expected performance $E(Y)$ for the optimization problem P. α denotes a new variable that specifies the underlying optimization problem P. Two optimization problems P_1 and P_2 are* equivalent *with respect to an algorithm A ($P_1 \equiv_A P_2$), if there are no interactions between the model parameters β and the problem parameter α.*

REMARK 1 *$P_1 \equiv_A P_2$ does not require that the main effect of α is insignificant.*

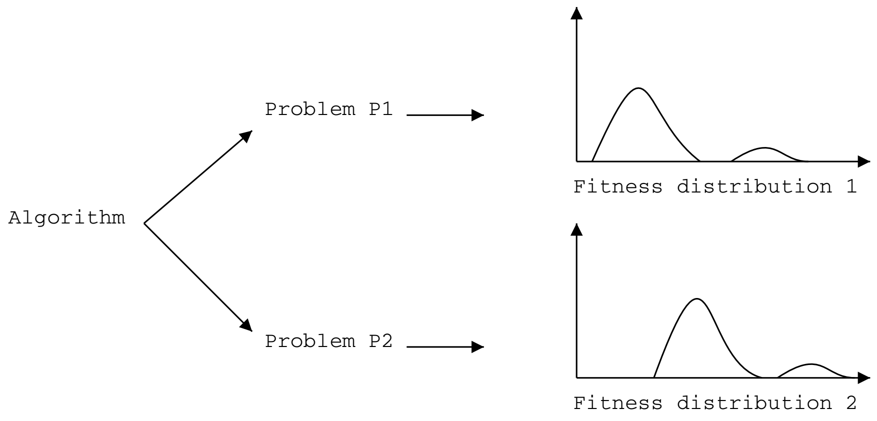

Figure 5.4. Schematic representation of algorithm based validation. A regression model describes the functional relationship between the parameter setting of an algorithm and its expected performance. Two optimization problems are equivalent, if there are no interactions between the problem and model parameters.

REMARK 2 *If $P_1 \equiv_A P_2$, then an optimization algorithm A with parametrization $\vec{a}$ shows a similar behavior on problem P_1 and problem P_2.*

Equation 5.4 can be written equivalently as $f_{P,Q} : (Z, \vec{a}) \rightarrow Y$, where Y is the performance of an algorithm with the exogenous parameter setting $\vec{a}$ for the problem P and the quality function Q.
Similar to the test from equation 5.5 the following statistical test to identify equivalent problems can be performed: The null hypothesis $H_0 : \alpha = 0$, is tested against the alternative $H_1 : \alpha \neq 0$ (P_1 and P_2 are not equivalent).

Our goal is to show that the S-ring model is a valid ESGC-model, so that we can transfer results from the S-ring to the lift model. The first step in algorithm based validation (ABV) requires the selection of an optimization algorithm. Evolution strategies, that are applicable to many optimization problems, have been chosen in the following (Beielstein et al., 2003b). Recent publications propose generalized linear models (GLMs) or methods from computational statistics such as design and analysis of computer experiments (DACE) or regression trees (Bartz-Beielstein and Markon, 2004; Bartz-Beielstein et al., 2004). GLM analysis provides a unified approach for linear and non-linear models with both normal and non-normal responses (McCullagh and Nelder, 1989; François and Lavergne, 2001). A generalized linear model that is based on the Gamma distribution and the canonical link is used in the following analysis. To model the problem, a factor L with two levels {S-ring, Lift} is introduced. We are interested to see whether or not there are interactions between algorithm parameters and L. Starting from the over fitted model we perform a backward elimination procedure.[3] The final model shows that there are no significant interactions between the problem L and other factors.[4] We can conclude that the S-ring problem and the lift problem are equivalent with respect to ES. Therefore the S-ring model can be seen as a valid simplification of the lift model. Remark 2 justifies the development of a simplified ESGC model: An improved (tuned) parameterization $\vec{a}$ of algorithm A and problem P_1 (S-ring) can be applied to optimize the complex optimization problem P_2 (lift).

5.6 S-RING MODEL EXTENSIONS

The S-ring model has been introduced as simplified elevator system model with very limited parameters and features. To improve our understanding of its applicability, it makes sense to explore two types of changes to this model. Firstly, effects of mechanisms looking inappropriate compared to a real elevator system shall be investigated. Secondly, features not existent in the S-ring model but obviously present in the real-world situation may be subsequently added to find out if they influence the design of a controller significantly.

For our experiments, we used a default NN controller with all weights set to 1.0 and a previously optimized controller which has been adapted to an arrival rate of 0.3. Experiences from previously performed optimization studies, i.e. (Beielstein et al., 2003a), recommended the following experimental setup: A (10+50)-ES performing the optimization was allowed 50,000 evaluations with 1,000 simulation steps each. 1,000 steps have been chosen to cope with the *problem of the initial transient* (Law and Kelton, 2000). We used self-adaptation with $\tau = 0.3$, $\sigma \in [0.01, 1.0]$ and re-evaluation of surviving parents.

The S-ring model has been defined in a way that favors its analysis as Markov decision process: Within a full circle, floor states are updated in sequence. Alternatively, one can imagine random order or even

[3]The model search (determination of the predictors, their orders and interactions) can be based on the Akaike Information Criterion (AIC). Backward elimination starts with an overfitted model, whereas forward selection adds a new variable to the model at each stage of the process.

[4]Another way to test for interactions between the factor L and other factors is to compare two nested models $M_2 \subset M_1$. M_1 includes interactions between L and other factors of the reduced model, whereas interactions are omitted in M_2. M_1: `Y` $\sim$ `Function * model` is compared to M_2: `Y` $\sim$ `Function + model.` The symbol '`+`' denotes independent factors and the symbol '`*`' denotes interactions between two factors. ANOVA indicates that there is no significant difference if interactions between L and the other factors are included.

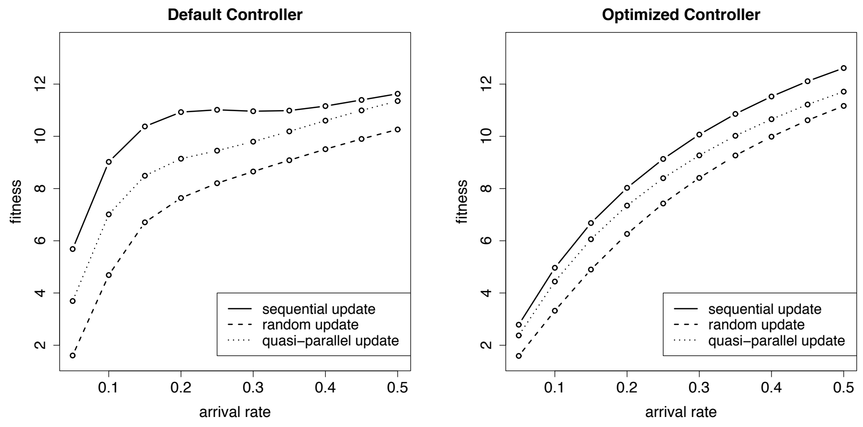

Figure 5.5. Two NN controllers dealing with different update methods (sequential, random, quasi-parallel) and arrival rates on an S-ring with 20 floors and 6 servers. Left hand: Default controller, all weights are 1.0, right hand: ES-optimized controller. Each point represents a simulation run with 1 million steps. Lower values are better.

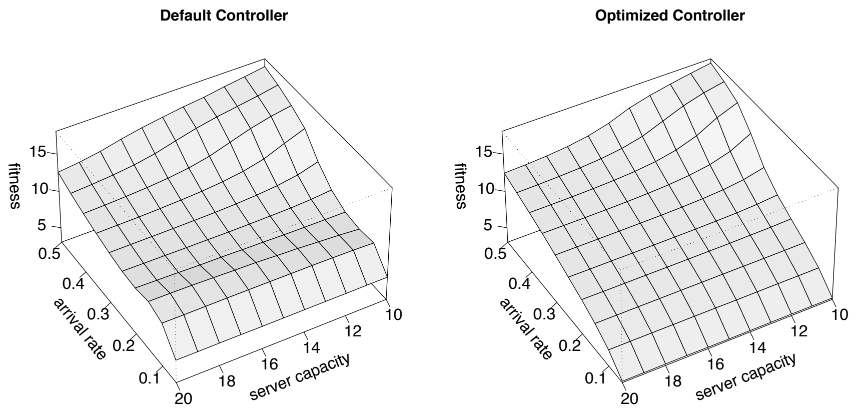

Figure 5.6. Default (left) and optimized NN controller performance on a wide range of arrival rates and server capacities on an S-ring with 20 floors and 6 servers. Each point represents a simulation run with 1 million steps. Lower values are better.

quasi-parallelism. The general behavior of our model must remain stable whatever variant is chosen because the update order of the real system we want to model is usually not known beforehand. We cannot yet prove that this is true for all possible configurations and controllers, but simulations done so far (see figure 5.5) indicate that the S-ring behavior is very similar for different update orders.

An obvious difference between real-world elevator cars and S-ring ones is that the latter have infinite capacity. Previously, we assumed that it is reasonable to neglect capacities for a wide range of parameter settings. Simulation runs depicted in figure 5.6 show that this is indeed the case. Only for extreme passenger flows congestion occurs and waiting customers sum up to large numbers. However, the turning point can be shifted upwards with a good controller policy.

5.7 SUMMARY AND OUTLOOK

We proposed a simplified elevator supervisory group controller, the so-called S-ring. The S-ring can serve as a helpful benchmark problem for evaluating and comparing search heuristics. Different techniques for its analysis and simulation were presented. A new method was developed to validate the S-ring as an ESGC model by taking an optimization algorithm into account. Furthermore, we demonstrated how new features can easily be added to the S-ring.
The current work can be extended by implementing different parallel optimization strategies. Additionally, this methodology is transferable to other traffic systems or, more general, other distributed control problems. Hence, we hope that the S-ring is an interesting real-world related optimization problem for other researchers.

ACKNOWLEDGMENTS

This work is a product of the Collaborative Research Center 531, ‘Computational Intelligence’, at the University of Dortmund and was supported by the Deutsche Forschungsgemeinschaft.

APPENDIX: DEFINITIONS

Definition 1 (S-ring) *The S-ring is the tuple*

$$\mathsf{S} = (n, m, \mathcal{X}, x_0, \mathcal{A}, \mathcal{O}, f, o, r, g),$$

where $n \in \mathbb{N}$ and $m \in \mathbb{N}$ are the number of queues and servers respectively. $\mathcal{X}$, $\mathcal{A}$, and $\mathcal{O}$ denote finite set of states, actions and observation respectively, o is an observation function, and x_0 denotes the initial state. $\mathcal{X}$ is defined as the set of binary vectors

$$x = (s_{n-1}, \dots, s_0, c_{n-1}, \dots, c_0) \in \mathbb{B}^{2n}$$

with $\sum_{i=0}^{n-1} s_i = m$. Let $g(t) : \mathbb{N}_0^+ \to \{0, 1, 2, \dots, n-1\}$ be the function that determines the number of the floor queue scanned at time step t:

$$g(t) := n - 1 - (t \bmod n). \tag{5.A.1}$$

$h : \mathcal{X} \times \mathbb{N}_0^+ \to \mathbb{B}^3$ is a helper function, that extracts three bits (customer present on the actual floor, server present on the actual floor, and server present on the next

floor) from the state vector x:

$$\begin{aligned} h(x,t) :=& (c_{g(t)}, s_{g(t)}, s_{g_{(t-1)}}) \\ =& (x_{2n-1-g(t)}, x_{n-1-g(t)}, x_{n-1-(g(t-1) \bmod n)}). \end{aligned} \tag{5.A.2}$$

The transition probability function

$$f : \mathcal{X} \times \mathcal{A} \times \mathbb{N}_0^+ \times \mathcal{X} \to [0,1] \tag{5.A.3}$$

defines probabilities for a state transition from state x to state x' depending on the action a performed. Finally,

$$r : \mathcal{X} \times \mathcal{X} \times \mathbb{N}_0^+ \to \mathbb{Z} \tag{5.A.4}$$

is the reward function. $\mathcal{S}$ denotes the set of all possible S-ring configurations.

Definition 2 (Decision rule) *A decision rule is a mapping from observations to actions:*

$$\delta : \mathcal{O} \to \mathcal{A}, \quad \delta(o) = \{0,1\}. \tag{5.A.5}$$

Definition 3 (Policy) *A sequence $(\delta_0, \delta_1, \delta_2 \ldots)$ of decision rules is called a* policy.

Definition 4 (Heaviside Function)

$$\theta(z) = \begin{cases} 0, & \textit{if } z < 0 \\ 1, & \textit{if } z \geq 0, \end{cases} \tag{5.A.6}$$

Definition 5 (POMDP) *A partially-observable Markov Decision Process (POMDP) M is a tuple $\mathsf{M} = \{\mathcal{X}, x_0, \mathcal{A}, \mathcal{O}, f, o, V\}$, where:*

- *$\mathcal{X}$ denotes a finite or countable set of states, $x_0 \in \mathcal{X}$ is the initial state, $\mathcal{A}$ denotes a set of actions, and $\mathcal{O}$ denotes a set of observations. If not mentioned otherwise, the Markov assumption is made in general: Each state has all information necessary to predict the next event and action.*
- *$f : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0,1]$ defines probabilities for a state transition from state x to state y depending on the action a performed. $f(x,a,y)$ is the probability that state y is reached from state x on action a. Therefore, each action can be represented by a state transition table of size $N \times N$, or by a state transition matrix P^k with entries (p_{ij}^k) as defined in equation 5.1. The probabilities in the transition matrix take also exogenous effects into account.*
- *$o : \mathcal{X} \to \mathcal{O}$ denotes the observation function: The corresponding set of observations $\mathcal{O}$ can be interpreted as a set of messages sent to the decision maker after an action is completed.*
- *And finally, the value function $V : \mathcal{H} \to \mathbb{R}$. If V is time-separable, then it can be written as a combination of the reward function r and the cost function c, that are defined as follows:*

$$r : \mathcal{X} \to \mathbb{R}, \quad \textit{and } c : \mathcal{X} \times \mathcal{A} \to \mathbb{R}. \tag{5.A.7}$$

REFERENCES

Arnold, D.V. and Beyer, H.-G. (2003). A comparison of evolution strategies with other direct search methods in the presence of noise. *Computational Optimization and Applications*, 24(1).

Banks, J., Carson, J. S., Nelson, B. L., and Nicol, D. M. (2001). *Discrete Event System Simulation.* Prentice Hall.

Barney, G. (1986). *Elevator Traffic Analysis, Design and Control.* Cambridge U.P.

Bartz-Beielstein, T. and Markon, S. (2004). Tuning search algorithms for real-world applications: A regression tree based approach. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 1111–1118, Portland, Oregon. IEEE Press.

Bartz-Beielstein, T., Parsopoulos, K. E., and Vrahatis, M. N. (2004). Analysis of particle swarm optimization using computational statistics. In Simos, T. E. and Tsitouras, Ch., editors, *International Conference on Numerical Analysis and Applied Mathematics 2004 (ICNAAM)*, pages 34–37. European Society of Computational Methods in Science and Engineering (ESCMSE), Wiley.

Beielstein, T., Ewald, C.-P., and Markon, S. (2003a). Optimal elevator group control by evolution strategies. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L. D., Roy, R., O'Reilly, U.-M., Beyer, Hans-Georg, et al., editors, *Proc. Genetic and Evolutionary Computation Conf. (GECCO 2003), Chicago IL, Part II*, volume 2724 of *Lecture Notes in Computer Science*, pages 1963–1974, Berlin. Springer.

Beielstein, T. and Markon, S. (2002). Threshold selection, hypothesis tests, and DOE methods. In Fogel, David B., El-Sharkawi, Mohamed A., Yao, Xin, Greenwood, Garry, Iba, Hitoshi, Marrow, Paul, and Shackleton, Mark, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 777–782. IEEE Press.

Beielstein, T., Markon, S., and Preuß, M. (2003b). Algorithm based validation of a simplified elevator group controller model. In Ibaraki, T., editor, *Proc. 5th Metaheuristics Int'l Conf. (MIC'03)*, pages 06/1–06/13 (CD–ROM), Kyoto, Japan.

Beielstein, T., Markon, S., and Preuß, M. (2003c). A parallel approach to elevator optimization based on soft computing. In Ibaraki, T., editor, *Proc. 5th Metaheuristics Int'l Conf. (MIC'03)*, pages 07/1–07/11 (CD–ROM), Kyoto, Japan.

Crites, R.H. and Barto, A.G. (1998). Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262.

François, O. and Lavergne, C. (2001). Design of evolutionary algorithms – a statistical perspective. *IEEE Transactions on Evolutionary Computation*, 5(2):129–148.

Howard, R. A. (1972). *Dynamic Programming and Markov Processes.* MIT Press, 7th edition.

Law, A.M. and Kelton, W.D. (2000). *Simulation Modelling and Analysis.* McGraw-Hill, New York, 3rd edition.

Markon, S. (1995). *Studies on Applications of Neural Networks in the Elevator System.* PhD thesis, Kyoto University.

Markon, S., Arnold, D.V., Bäck, T., Beielstein, T., and Beyer, H.-G. (2001). Thresholding – a selection operator for noisy ES. In Kim, J.-H., Zhang, B.-T., Fogel, G., and Kuscu, I., editors, *Proc. 2001 Congress on Evolutionary Computation (CEC'01)*, pages 465–472, Seoul, Korea. IEEE Press, Piscataway NJ.

Markon, S. and Nishikawa, Y. (2002). On the analysis and optimization of dynamic cellular automata with application to elevator control. The 10th Japanese-German Seminar, Nonlinear Problems in Dynamical Systems, Theory and Applications. Noto Royal Hotel, Hakui, Ishikawa, Japan.

McCullagh, P. and Nelder, J.A. (1989). *Generalized Linear Models.* Chapman and Hall, 2nd edition.

Naudts, B. and Kallel, L. (2000). A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1–15.

Naylor, T.H. and Finger, J.M. (1967). Verification of computer simulation models. *Management Science*, 2:B92–B101.

Reeves, C.R. and Yamada, T. (1998). Genetic algorithms, path relinking and the flowshop sequencing problem. *Evolutionary Computation journal (MIT press)*, 6(1):230–234.

Schwefel, H.-P., Wegener, I., and Weinert, K., editors (2003). *Advances in Computational Intelligence – Theory and Practice.* Natural Computing Series. Springer, Berlin.

So, A.T. and Chan, W.L. (1999). *Intelligent Building Systems.* Kluwer A.P.

Som, T. K. and Sargent, R. G. (1989). A formal development of event graphs as an aid to structured and efficient simulation programs. *ORSA J. Comp.*

Whitley, D., Watson, J.P., Howe, A., and Barbulescu, L. (2002). Testing, evaluation and performance of optimization and learning systems.

Technical report, The GENITOR Research Group in Genetic Algorithms and Evolutionary Computation, Colorado State University.

Wolpert, D.H. and Macready, W.G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.