

Complexity Made Simple (at a Small Price)

Christos G. Cassandras^(✉)

Division of Systems Engineering, Boston University, Boston 02215, USA
cgc@bu.edu

Abstract. Fundamental complexity limits prevent us from directly solving many design, control, and optimization problems. Yet, there are ways to solve such hard problems by exploiting their specific structure, by asking the “right” questions, and by challenging some conventional approaches. First, slow, inefficient, and intrusive trial-and-error techniques can sometimes be bypassed through simple thought experiments constructed at a “small price.” In particular, for Discrete Event Systems this can be systematically accomplished through the theory of Perturbation Analysis. Second, decomposition and abstraction methods can often provide accurate solutions or significantly simplify a hard problem at the “small price” of some loss of accuracy. Finally, conventional time-driven methods for sampling, control, and communication can be replaced by event-driven techniques with the proper events triggering these actions.

1 Complexity Limits

There are a number of problems which are well known for their “complexity.” The sources of such complexity may be in the physical nature of the processes involved or in the operational rules imposed to control an engineered system. They may also originate in the stochastic nature of the problem or in its mere dimensionality which limits our computational ability to solve it despite the fastest computational engines at our disposal. In fact, there are at least three fundamental limits that involve complex dynamic systems. First is the well-known $1/\sqrt{T}$ (or $1/\sqrt{N}$) limit best illustrated through the central limit theorem: we cannot learn from data faster than $1/\sqrt{T}$ (or $1/\sqrt{N}$) where T is the time spent to collect data or to simulate a process (equivalently, N is the amount of data collected). Second is the NP-hard limit, i.e., the simple combinatorial fact that the size of a strategy space to be explored is given by N^M where N is the size of the decision space and M is the size of the information space. Third is the so-called “No-Free-Lunch” limit expressed in [9] as a No-Free-Lunch Theorem quantifying the tradeoff between generality and efficiency of an algorithm. This is illustrated, for example, by Linear Programming algorithms which can very efficiently solve a very small (albeit important) class of optimization problems. Still, there are ways to solve many hard

C.G. Cassandras—Supported in part by NSF under grants CNS-1239021, ECCS-1509084, and IIP-1430145, by AFOSR under grant FA9550-15-1-0471, and by The MathWorks.

problems by exploiting their specific structure, by asking the “right” questions, and by challenging some conventional approaches.

2 Perturbation Analysis for Discrete Event Systems

Trial-and-error techniques are often used to systematically learn and predict the behavior of a complex system. These are invariably slow, inefficient, and intrusive. However, this learning can sometimes be accomplished at a fraction of the usual brute-force trial-and-error process through simple “thought experiments” constructed at a “small price.” This is particularly true for Discrete Event Systems (DES), regardless of the modeling framework (automata, Petri Nets, or other) one chooses to employ. In particular, the theory of *Perturbation Analysis* (PA) [2, 5, 6] has established that event-driven dynamics give rise to state trajectories (sample paths) from which one can very efficiently and nonintrusively extract sensitivities of state variables (therefore, various performance metrics as well) with respect to at least certain types of design or control parameters. The origin of the key concepts that form the cornerstones of the PA theory are found in a long-standing problem known as the *buffer allocation problem*. In its industrial engineering version, a typical serial transfer line consists of N workstations in tandem, each with different characteristics in terms of its production rate, failure rate and repair time when failing. In order to accommodate this inhomogeneous behavior, a buffer is placed before the i th workstation, $i = 1, \dots, N$, with B_i discrete slots where production parts can be queued and $\sum_{i=1}^N B_i = B$. The problem is to allocate these B buffer slots, i.e., determine a vector $[B_1 \dots B_N]$, so as to maximize the throughput of the transfer line while also maintaining a low overall average delay of the parts moving from an entry point before the first workstation to an exit point following the N th workstation. Tackling this problem in a “brute force” manner requires considering all possible buffer allocations, an enormous number (e.g., for a reasonably small problem such as $B = 24$ and $N = 6$, this gives 118,755 possible solutions.) A direct trial-and-error approach where one is allowed to test each allocation for about a week would require about 2300 years. If one were to reduce the initial solution space to only 1000 “good guesses” and use the fastest possible simulation software, the overall task would still require several hours of CPU time.

The approach first taken in [7] was to study the serial transfer line as a dynamic system whose state includes the integer-valued buffer contents along with real-valued “clocks” associated with each workstation as it processes a part. The question then posed was: “what would happen if in a given allocation a specific value B_i were changed to $B_i + 1$?” When part arrival and processing time distributions are unknown, there is no analytical expression relating the system throughput to the parameter vector $[B_1 \dots B_N]$. Thus, the “brute force” way to answer this question is to first simulate the system under the nominal allocation with the value B_i and estimate the system’s performance over a (sufficiently long) time period T which may be denoted by $L_T(B_i)$. Then, repeat the simulation under $B_i + 1$ to obtain $L_T(B_i + 1)$. The difference $\Delta L_T(B_i) = L_T(B_i + 1) - L_T(B_i)$

provides an estimate of the system's performance sensitivity with respect to B_i . However, this is unnecessary: indeed, the initial simulation alone yielding $L_T(B_i)$ and a simple thought experiment can deliver the value of $\Delta L_T(B_i)$. Moreover, the same thought experiment can deliver the entire vector $[\Delta L_T(B_1), \dots, \Delta L_T(B_N)]$ with minimal extra effort. The key observation is that when B_i is replaced by $B_i + 1$, no change in the state of the system can take place unless one of two "events" is observed at time t : (i) The i th buffer content, say $x_i(t)$, reaches its upper limit, i.e., $x_i(t) = B_i$ and a part is ready to leave the $(i - 1)$ th workstation. In this case, this upstream workstation is "blocked" since there is no place for the departing part to go. However, in a perturbed system with B_i replaced by $B_i + 1$ that would not happen and one can simply predict a buffer content perturbation $\Delta x_i(t) = 1$. Moreover, one can record when this blocking occurs at time $t \equiv t_{i,B}$ and the next time that a part departs from the i th workstation, $t_{i,D}$. Then, $t_{i,D} - t_{i,B}$ is the amount of time that would be gained (i.e., no blocking would have occurred) in a perturbed system realization. The important observation here is that $t_{i,D}$, $t_{i,B}$ are directly observed along the nominal system realization. (ii) The $(i + 1)$ th buffer content reaches its lower limit, i.e., $x_{i+1}(t) = 0$ and a part is ready to leave the i th workstation. In this case, if $\Delta x_i(t) = 1$, i.e., the i th workstation has already gained a part from an earlier blocking event, then this gain can now propagate downstream and we can set $\Delta x_{i+1}(t) = \Delta x_i(t) = 1$. Thus, estimating the effect of replacing B_i by $B_i + 1$ boils down to observing just a few events along the nominal system realization: *blocking events* (when $x_i(t) = B_i$ and a part departure from $i - 1$ takes place) and *idling events* (when $x_i(t) = 0$ at any $i = 1, \dots, N$.)

This basic idea can be extended to a large class of DES. The general procedure is one where some parameter perturbation $\Delta\theta$ *generates* a state perturbation $\Delta x_i(t)$ when a specific event occurs at time t . Subsequently, the system dynamics dictate how $\Delta x_i(t)$ *propagates* through the system by affecting $\Delta x_i(t)$ or $\Delta x_j(t)$ for $j \neq i$. Depending on a performance metric of interest, this ultimately yields $\Delta L_T(\Delta\theta)$, the estimated change in performance due to $\Delta\theta$. To illustrate this process, we consider the case of a simple First-In-First-Out (FIFO) queuing system with a single server preceded by a queue. Let $\{A_k\}$ be the sequence of (generally random) arrival times, $k = 1, 2, \dots$, and $\{D_k\}$ be the corresponding sequence of departure times from the system. If Z_k denotes the service time of the k th entity (customer) processed, then the Lindley equation

$$D_k = \max(A_k, D_{k-1}) + Z_k \quad (1)$$

describes the departure time dynamics with $k = 1, 2, \dots$. Suppose that all (or just some selected subset) of the service times are perturbed by ΔZ_k , $k = 1, 2, \dots$. Let $I_k = A_k - D_{k-1}$ and observe that when $I_k > 0$ it captures an idle period (since the server must wait until $A_k > D_{k-1}$ to become busy again) and when $I_k < 0$ it captures the waiting time $D_{k-1} - A_k$ of the k th arriving entity in the system. It is easy to obtain from (1) the following departure time perturbation equation:

$$\Delta D_k = \Delta Z_k + \begin{cases} \Delta D_{k-1} & \text{if } I_k \leq 0, \Delta D_{k-1} \geq I_k \\ 0 & \text{if } I_k > 0, \Delta D_{k-1} \leq I_k \\ I_k & \text{if } I_k \leq 0, \Delta D_{k-1} \leq I_k \\ \Delta D_{k-1} - I_k & \text{if } I_k > 0, \Delta D_{k-1} \geq I_k \end{cases} \quad (2)$$

where ΔD_k can be obtained from the generated perturbations ΔZ_k and directly observed data in the form of I_k . Next, observe that if we select $\Delta Z_k > 0$ to be sufficiently small so that $\Delta D_{k-1} > 0$ can never exceed the finite value of $I_k > 0$, then this reduces to

$$\Delta D_k = \Delta Z_k + \begin{cases} \Delta D_{k-1} & \text{if } I_k \leq 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

which is much simpler, requiring only the detection of an idling interval when $I_k > 0$, at which time the departure time perturbation is reset to $\Delta D_k = \Delta Z_k$. Naturally, the question is: “How small do perturbations need to be before the IPA equation can be used?” In a stochastic system, the answer to this question is generally dependent on the specific realization based on which ΔD_k is evaluated. Thus, it is logical to extend the question of estimating $\Delta D_k(\Delta\theta)$, $k = 1, 2, \dots$, the departure time perturbations, to estimating the derivative $\frac{dD_k(\theta)}{d\theta}$ by allowing $\Delta\theta \rightarrow 0$. When $\Delta\theta \rightarrow 0$, this leads to *Infinitesimal Perturbation Analysis* (IPA) and one obtains derivatives (sensitivities) of the form $\frac{dL(\theta)}{d\theta}$ for performance metrics which can be expressed as a function of the departure times, i.e., $L(D_1(\theta), D_2(\theta), \dots)$. These derivatives can be combined with standard gradient-based optimization techniques to solve hard stochastic optimization problems exploiting the fact that $dL(\theta)/d\theta$ can be shown under certain conditions to be an unbiased estimate of $dE[L(\theta)]/d\theta$, the derivative of an objective function $E[L(\theta)]$. When these conditions do not hold, several PA methods have been developed to still obtain unbiased estimates of $dE[L(\theta)]/d\theta$ at the expense of additional observed sample path data [2].

3 Decomposition and Abstraction

These are complementary approaches that can often provide accurate solutions or significantly simplify a hard problem at the “small price” of some loss of accuracy. Decomposition naturally arises in networks consisting of multiple nodes (often referred to as “agents”) that cooperate to control their individual state so as to optimize a common objective. To accomplish this, the nodes need to communicate with each other to exchange state information. Since communication costs can be significant and reliance on a single central coordinator is highly complex, we seek conditions under which the optimization process can be decomposed and distributed over the nodes: each node performs its own optimization with communication carried out only when absolutely necessary and limited to its neighbors. The “absolutely necessary” condition is specified by events defined to occur when some state estimation error function at a node exceeds a threshold, thus drastically reducing the overall communication cost. One such scheme

is based on node j maintaining an estimate of every neighboring node i 's state, denoted by x_j^i . Assuming node i knows this estimate, it notifies node j of its true state x_i only when $\|x_i - x_j^i\| > \theta_i(t)$ where $\theta_i(t)$ is some threshold (possibly time-varying) and $\|\cdot\|$ is an appropriately defined norm that measures this state estimation error. This type of asynchronous, event-driven scheme can be shown to still guarantee convergence to an optimum even in the presence of communication delays as long as they are bounded [10].

On the abstraction side, the key idea is to replace a complex model by a simpler one that retains the salient features of the problem that one wishes to address. In the case of DES, this is possible by preserving only “significant” events and replacing the system’s activity in between these events by continuous time-driven surrogates. A good example of this approach arises when a queueing system is abstracted by a *Stochastic Flow Model* (SFM) [3, 4] where the “significant” events occur when the queue either becomes empty or it ceases to be empty following an idle period. The SFM framework essentially consists of fluid queues which forego the notion of the individual customer and focus instead on the aggregate flow. In such a fluid queue, traffic and service processes are characterized by instantaneous flow rates as opposed to the arrival, departure, and service times of discrete customers. The SFM qualifies as a stochastic hybrid system with bi-layer dynamics: event-driven dynamics at the upper layer and time-driven dynamics at the lower layer. The discrete events are associated with changes in traffic-flow processes, such as the reaching or leaving the boundaries of busy periods at the queues. In contrast, the time-driven dynamics describe the continuous evolution of flow rates between successive discrete events, usually through differential equations. Performance metrics that are natural to SFMs typically reflect quantitative measures of flow rates, like average throughput, buffer workload, and loss.

It is important to point out that the abstraction process serves the purpose of control and optimization rather than performance analysis. In this case, the “right” question to ask is “what is the value of a parameter that delivers the optimal performance?” and not “what is the optimal performance?” The SFM abstraction captures only those features of the underlying “real” system that are needed to lead to the correct answer to this question, even though the corresponding optimal performance may not be accurately estimated. Even if the exact solution cannot be obtained by such lower-resolution models, one can still obtain near-optimal points that exhibit robustness with respect to certain aspects of the model they are based on.

4 Event-Driven vs. Time-Driven Methods

The emergence of DES has brought to the forefront an alternative viewpoint to the traditional *time-driven* paradigm in which time is an independent variable and, as it evolves, so does the state of a dynamic system. The *event-driven* paradigm offers an alternative, complementary look at modeling, control, communication, and optimization [1, 8]. The key idea is that a clock should not

be assumed to dictate actions simply because a time step is taken; rather, an action should be triggered by an “event” specified as a well-defined condition on the system state or as a consequence of environmental uncertainties that result in random state transitions. Observing that such an event could actually be defined to be the occurrence of a “clock tick,” it follows that this framework may in fact incorporate time-driven methods as well. On the other hand, defining the proper events requires more sophisticated techniques compared to simply reacting to time steps. In the development of DES, such events are seen as the natural means to drive the dynamics of a large class of systems (e.g., computer networks, manufacturing systems, supply chains.) More generally, however, it is evident that most interesting dynamic systems are in fact “hybrid” in nature, i.e., at least some of their state transitions are caused by (possibly controllable) events. *Stochastic hybrid systems* form the broadest class of dynamic systems one can envision (the SFM framework mentioned above is a special class of such systems.) As an example, a *hybrid* Petri net is one whose transition firing times are dependent on time-driven processes describing the evolution of continuous state variables associated with places: when a state variable exceeds a given threshold, this contributes to enabling one or more transitions in the output set of that place.

The IPA theory mentioned earlier for DES can be extended to hybrid systems, giving rise to a general-purpose *IPA Calculus* [4] which is the foundation of a gradient estimation methodology: it is based on a set of equations which provide state sensitivities of the form $dx(t; \theta)/d\theta$ for any model parameter θ (more generally, the gradient $\nabla x(t; \theta)$ with respect to a parameter vector θ .) For any performance metric $E[L(\theta)]$, this also provides sensitivities of the form $dE[L(\theta)]/d\theta$ through the sample path derivatives $dL(\theta)/d\theta$ which in turn depend on the stochastic processes $\{dx(t; \theta)/d\theta\}$. As in the case of DES, these can be obtained on line in an efficient non-intrusive manner. Moreover, they can be shown to be unbiased under very mild technical conditions and they possess a number of practically important properties. One of these properties stems from the fact that since the gradient estimation procedure is entirely event-driven, it is *scalable in the size of the event space* as opposed to the generally much larger state space of a system. Another important property is that the resulting estimators are independent of the random processes involved; at worst, they require only minimal effort to estimate simple statistics in the neighborhood of certain observable events.

References

1. Cassandras, C.G.: The event-driven paradigm for control, communication, and optimization. *J. Control Decis.* **1**(1), 3–17 (2014)
2. Cassandras, C.G., Lafortune, S.: *Introduction to Discrete Event Systems*, 2nd edn. Springer, US (2008)
3. Cassandras, C.G., Wardi, Y., Melamed, B., Sun, G., Panayiotou, C.G.: Perturbation analysis for on-line control and optimization of stochastic fluid models. *IEEE Trans. Autom. Control* **47**(8), 1234–1248 (2002)

4. Cassandras, C.G., Wardi, Y., Panayiotou, C.G., Yao, C.: Perturbation analysis and optimization of stochastic hybrid systems. *Eur. J. Control* **16**(6), 642–664 (2010)
5. Glasserman, P.: *Gradient Estimation via Perturbation Analysis*. Kluwer Academic Publishers, Dordrecht (1991)
6. Ho, Y.C., Cao, X.R.: *Perturbation Analysis of Discrete Event Dynamic Systems*. Kluwer Academic Publishers, Dordrecht (1991)
7. Ho, Y.C., Eyler, A., Chien, D.T.: A gradient technique for general buffer storage design in a serial production line. *Int. J. Prod. Res.* **17**, 557–580 (1979)
8. Miskowicz, M. (ed.): *Event-Based Control and Signal Processing*. CRC Press/Taylor and Francis, Boca Raton (2015)
9. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1**(1), 67–82 (1997)
10. Zhong, M., Cassandras, C.G.: Asynchronous distributed optimization with event-driven communication. *IEEE Trans. Autom. Control* **55**(12), 2735–2750 (2010)

Application and Theory of Petri Nets and Concurrency
38th International Conference, PETRI NETS 2017,
Zaragoza, Spain, June 25–30, 2017, Proceedings
van der Aalst, W.; Best, E. (Eds.)
2017, XIV, 351 p. 136 illus., Softcover
ISBN: 978-3-319-57860-6