Annual Reviews in Control 000 (2018) 1–14

Contents lists available at ScienceDirect

Annual Reviews in Control

journal homepage: www.elsevier.com/locate/arcontrol

Perturbation analysis: A framework for data-driven control and optimization of discrete event and hybrid systems[☆,☆☆]

Y. Wardi[a,*], C.G. Cassandras[b], X.R. Cao[c]

[a] School of Electrical and Computer Engineering, Georgia Institute of Technology, 777 Atlantic Drive, Atlanta, GA, 30332, USA
[b] Division of Systems Engineering, and Department of Electrical and Computer Engineering, Boston University, 8 St. Mary's Street, Boston, MA, 02215, USA
[c] Antai College of Economics and Management and School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, 1954 Huashan Road, Shanghai, 200030, People's Republic of China

ARTICLE INFO

Article history:
Received 10 January 2018
Revised 18 March 2018
Accepted 3 April 2018
Available online xxx

Keywords:
Perturbation analysis
Discrete-event dynamic systems
Hybrid systems
Performance optimization and control

ABSTRACT

The history of Perturbation Analysis (PA) is intimately related to that of Discrete Event Dynamic Systems (DEDS), starting with a solution of a long-standing problem in the late 1970s and continuing today with the control and optimization of Hybrid Systems and the emergence of event-driven control methods. We review the origins of the PA theory and how it became part of a broader framework for modelling, control and optimization of DEDS. We then discuss the theoretical underpinnings of Infinitesimal Perturbation Analysis (IPA) as a data-driven stochastic gradient estimation method and how it has been applied over the past few decades. We explain how IPA offers a basis for general-purpose stochastic optimization of Markovian systems through the notion of the performance potential and how it has evolved beyond DEDS and now provides a framework for control and optimization of Hybrid Systems and, more generally, event-driven methodologies.

1. The origin of perturbation analysis

In pioneering the field of Discrete Event Systems (DES) in the early 1980s, Y.C. Ho and his research group at Harvard University discovered 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. This eventually led to the development of a theory for *Perturbation Analysis* (PA) in DES (Cassandras & Lafortune, 2008; Glasserman, 1991; Ho & Cao, 1991), the most successful branch of which is *Infinitesimal Perturbation Analysis* (IPA) due to its simplicity and ease of implementation. In fact, by the early 2000s, IPA was shown to apply to all virtually arbitrary Hybrid Systems (HS) and continues to be today one of the most attractive tools for data-driven control and optimization, especially in stochastic environments where modelling random aspects of a process is prohibitively hard.

The origin of the key concepts that form the cornerstones of the PA theory are found in a long-standing problem in operations research and industrial engineering known as the *buffer allocation problem*. In its industrial engineering version, it was presented to Ho's research group by the FIAT automobile company in the late 1970s as follows. 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 kth workstation, $k = 1, \dots, N$, with B_k discrete slots where production parts can be queued. Since the space within which this transfer line operates is limited, there is an upper bound B to the total number of buffer slots that can be allocated over the N workstations so that $\sum_{k=1}^{N} B_k = B$. 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 Nth workstation. Tackling this problem in a "brute force" manner requires considering all possible buffer allocations, a num-

☆ Wardi's work was supported in part by NSF under Grant Number CNS-1239225. Cassandras's work has been supported in part by NSF under grants CNS-1239021, ECCS-1509084, and IIP-1430145, by AFOSR under grant FA9550-15-1-0471, and by a grant from the MathWorks.
☆☆ This paper extends an earlier version which appeared in the Proceedings of the 20th IFAC World Congress, (Wardi, Cassandras, and Cao (2017)).
* Corresponding author.
E-mail addresses: ywardi@ece.gatech.edu (Y. Wardi), cgc@bu.edu (C.G. Cassandras), eecao@ust.hk (X.R. Cao).

https://doi.org/10.1016/j.arcontrol.2018.04.003
1367-5788/© 2018 Published by Elsevier Ltd.

ber given by $\binom{B+N-1}{B}$. 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 early 1980s simulation technology requiring about 3 min per trial to estimate the resulting performance, the overall task would take about 250 days of CPU time.

The approach taken by Ho's group and first reported in Ho, Eyler, and Chien (1979) 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_k were changed to B_k+1?" The "brute force" way to answer this question is to first simulate the system under the nominal allocation with the value B_k and estimate the system's performance over a (sufficiently long) time period T which may be denoted by $L_T(B_k)$. Then, repeat the simulation under B_k+1 to obtain $L_T(B_k+1)$. The difference $\Delta L_T(B_k)=L_T(B_k+1)-L_T(B_k)$ provides an estimate of the system's performance sensitivity with respect to B_k. What the research team realized, however, is that this is unnecessary: indeed, the initial simulation alone yielding $L_T(B_k)$ and a simple thought experiment can deliver the value of $\Delta L_T(B_k)$. Moreover, the same thought experiment can deliver the entire vector $[\Delta L_T(B_1),\ldots,\Delta L_T(B_N)]$ with minimal extra effort.

The key observation that led to a formal procedure describing this thought experiment is the following. When B_k is replaced by B_k+1, no change in the state of the system can take place unless one of two "events" is observed at time t: (*i*) The kth buffer content, say $x_k(t)$, reaches its upper limit, i.e., $x_k(t)=B_k$ and a part is ready to leave the $(k-1)$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_k replaced by B_k+1 that would not happen and one can simply predict a buffer content perturbation $\Delta x_k(t)=1$. Moreover, one can record when this blocking occurs at time $t\equiv t_{k,B}$ and the next time that a part departs from the kth workstation, $t_{k,D}$. Then, $t_{k,D}-t_{k,B}$ 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 $t_{k,D}$, $t_{k,B}$ are directly observed along the nominal system realization. (*ii*) The $(k+1)$th buffer content reaches its lower limit, i.e., $x_{k+1}(t)=0$ and a part is ready to leave the kth workstation. In this case, if $\Delta x_k(t)=1$, i.e., the kth workstation has already gained a part from an earlier blocking event, then this gain can now propagate downstream and we can set $\Delta x_{k+1}(t)=\Delta x_k(t)=1$.

This simple observation leads to the conclusion that estimating the effect of replacing B_k by B_k+1 boils down to observing just a few events along the nominal system realization: *blocking events* (when $x_{ki}(t)=B_k$ and a part departure from $k-1$ takes place) and *idling events* (when $x_k(t)=0$ at any $k=1,\ldots,N$). This can be formalized into an "estimator" for buffer perturbations $\Delta x_k(t)$ and event timing perturbations for all part departures at workstations. More generally, this estimator transforms a given hypothetical perturbation $\Delta B_k(t)=1$ (or -1) into state perturbations, which can ultimately be used to estimate a performance perturbation $\Delta L_T(B_k)$. Most importantly, this is accomplished without ever having to implement the perturbation $\Delta B_k(t)$, since the estimator depends only on directly observable data from the nominal system realization; in particular, it suffices to observe selected events and associated event times and to perform extremely simple calculations.

This initial procedure pertaining to a very specific type of dynamic system and problem was given the name *Perturbation Analysis* (PA). It soon became clear that it could be extended to any system with a structure similar to that of the serial transfer line and to a perturbation in any system parameter. Thus, one could consider, for instance, speeding up the operation of a workstation and studying the effect of a perturbation Δr_k in the operation rate r_k of the kth workstation. The general procedure is one where some parameter perturbation $\Delta\theta$ *generates* a state perturbation $\Delta x_k(t)$ when a specific event occurs at time t. Subsequently, the system dynamics dictate how $\Delta x_k(t)$ *propagates* through the system by affecting $\Delta x_k(t)$ or $\Delta x_j(t)$ for $j\neq k$. Depending on a performance metric of interest, this ultimately yields $\Delta L_T(\Delta\theta)$, the change in performance due to $\Delta\theta$. As for the system structure amenable to this kind of efficient PA, it became obvious that it fits the general class of queueing networks.

An obvious next question was: "Does PA hold for any value of $\Delta\theta$ or do we have to restrict it to "small" $\Delta\theta$ when θ is real-valued?" There was ample empirical evidence collected over the early 1980s that $\Delta\theta$ had to be small but not necessarily "very small". In other words, the values of $\Delta L_T^{PA}(\Delta\theta)$ obtained through PA were identical to those obtained through the "brute force" finite difference $L_T(\theta+\Delta\theta)-L_T(\theta)$ for "sufficiently small" $\Delta\theta$. This led to the term *Infinitesimal Perturbation Analysis* (IPA) to capture the fact that the methodology was applicable to perturbations which were "infinitesimally" small, although a formal quantification characterizing limits for $\Delta\theta$ was lacking. Moreover, when $\Delta\theta$ became larger, it was still possible to satisfy $\Delta L_T^{PA}(\Delta\theta)=L_T(\theta+\Delta\theta)-L_T(\theta)$ at the expense of observing more "interesting events" and performing a few more calculations. For instance, in the case of the integer-valued buffer size parameter B_k, the minimal feasible perturbation is obviously either $+1$ or -1. To differentiate these cases, the term *Finite Perturbation Analysis* (FPA) was introduced. FPA reverts to IPA when parameters are real-valued and may be allowed to take "sufficiently small" values $\Delta\theta$.

To illustrate the distinction between IPA and FPA, we consider the case of a simple First-In-First-Out (FIFO) queueing system with a single server preceded by a queue. Let $\{A_i\}$ be the sequence of (generally random) arrival times, $i=1,2,\ldots,$ and $\{D_i\}$ be the corresponding sequence of departure times from the system. If S_i denotes the service time of the ith entity (customer) processed, then the Lindley equation

$$D_i=\max(A_i,D_{i-1})+S_i \tag{1}$$

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

$$\Delta D_i=\Delta S_i+\begin{cases}\Delta D_{i-1} & \text{if } I_k\le 0,\ \Delta D_{i-1}\ge I_i\\ 0 & \text{if } I_i>0,\ \Delta D_{i-1}\le I_i\\ I_i & \text{if } I_i\le 0,\ \Delta D_{i-1}\le I_i\\ \Delta D_{i-1}-I_i & \text{if } I_i>0,\ \Delta D_{i-1}\ge I_i\end{cases} \tag{2}$$

where ΔD_i can be obtained from the generated perturbations ΔS_i and directly observed data in the form of I_i. This is the FPA procedure for evaluating ΔD_i, $i=1,2,\ldots$ Observe, however, that if we select $\Delta S_i>0$ to be sufficiently small so that $\Delta D_{i-1}>0$ can never exceed the finite value of $I_i>0$, then this reduces to

$$\Delta D_i=\Delta S_i+\begin{cases}\Delta D_{i-1} & \text{if } I_i\le 0\\ 0 & \text{otherwise}\end{cases} \tag{3}$$