Journal of Control and Decision, 2014
Vol. 1, No. 1, 3–17, http://dx.doi.org/10.1080/23307706.2014.885288

Taylor & Francis
Taylor & Francis Group

The event-driven paradigm for control, communication and optimization

Christos G. Cassandras*

Division of Systems Engineering and Center for Information and Systems Engineering, Boston University, Boston, MA, USA

(*Received 31 December 2013; accepted 17 January 2014*)

The event-driven paradigm offers an alternative to the time-driven paradigm for modelling, sampling, estimation, control and optimization. This has come about largely as a consequence of systems being increasingly networked, wireless and consisting of distributed communicating components. The key idea is that control actions need not be dictated by time steps taken by a "clock"; rather, an action should be triggered by an "event" which may be a well-defined condition on the system state, including the possibility of a simple time step, or a random state transition. We provide an overview of recent developments in event-driven approaches and focus on two areas to illustrate their value. First, in distributed systems, we describe how event-driven, rather than synchronous, communication can guarantee convergence in cooperative distributed optimization while provably maintaining optimality. Second, in hybrid systems where events naturally decompose state trajectories into different discrete states (modes), we review the theory of infinitesimal perturbation analysis (IPA) which offers an event-driven "IPA calculus" for evaluating (or estimating in the case of stochastic systems) gradients of performance metrics, thus facilitating the solution of a large class of control and optimization problems.

Keywords: event-driven control; distributed systems; hybrid systems; infinitesimal perturbation analysis

1. Introduction

The time-driven paradigm for the modelling and analysis of dynamic systems is founded on the centuries-old theoretical framework provided by differential (or difference) equations. In this paradigm, time is an independent variable and, as it evolves, so does the state of the system. Conceptually, there is an underlying "clock" and with every "clock tick" a state update is performed, including the case where no change in the state occurs. Methodologies for sampling, estimation, communication, control and optimization are also founded on the same time-driven principle. The digital technological advances of the 1970s and beyond have facilitated the implementation of this paradigm with digital clocks embedded in hardware and used to drive the collection of data or the actuation of devices employed to control various processes.

In a world increasingly networked, wireless and involving large-scale distributed systems, the universal value of this point of view has understandably come to question. While it is always possible to postulate an underlying clock with time steps dictating

*Email: cgc@bu.edu

state transitions, it may not be feasible to guarantee the synchronisation of all components of a distributed system to such a clock; nor is it efficient to trigger actions with every time step when such actions may be unnecessary. The *event-driven* paradigm offers an alternative, complementary look at control, communication and optimization. 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 random state transition. Note that such an event could be defined to be the occurrence of a "clock tick", so 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.

This alternative event-driven view is well-motivated. For starters, there are many natural discrete event systems (DES) where the only changes in their state are dictated by event occurrences. The Internet is a prime example, where "events" are defined by packet transmissions and receptions at various nodes, causing changes in the contents of various queues. For such systems, a time-driven modelling approach may not only be inefficient, but also potentially erroneous, as it cannot deal with events designed to occur concurrently in time. Beyond these systems, there are many more which are "hybrid" in nature, i.e. at least some of their state transitions are caused by (possibly controllable) events. The recent emergence of cyber-physical systems (CPS) is an example of this hybrid structure, where physical processes are modelled through time-driven dynamics, while various embedded sensing and actuating devices interfacing with these processes operate in event-driven mode. Last but not least, many systems of interest are networked and spatially distributed. In such settings, especially when energy-constrained wireless devices are involved, frequent communication among system components can be inefficient, unnecessary and sometimes infeasible. Thus, rather than imposing a time-driven communication mechanism, it is reasonable to seek instead to define specific events which dictate when a particular node in a network needs to exchange information with one or more other nodes. When, in addition, the environment is stochastic, significant changes in the operation of a system are the result of random event occurrences, so that, once again, understanding the implication of such events and reacting to them is crucial.

Besides their modelling potential, it is important to note that event-driven approaches to fundamental processes such as sampling, estimation and control possess important properties related to variance reduction and robustness of control policies to modelling uncertainties. These properties render them attractive, compared to time-driven alternatives.

The importance of event-driven behaviour in dynamic systems was recognised and studied in the early 1980s with the emergence of DES, e.g. see Cassandras and Lafortune (2008) and references therein. In recent years, however, there have been significant developments in applying event-driven methods (also referred to as "event-based" and "event-triggered") to classical feedback control systems; see Arzen (2002), Heemels, Sandee, and Bosch (2008), Lunze and Lehmann (2010), Tabuada (2007), Anta and Tabuada (2010) and references therein. For example, in Heemels et al. (2008) a controller for a linear system is designed to update control values only when a specific error measure (e.g. for tracking or stabilisation purposes) exceeds a given threshold, while refraining from any updates otherwise. It is also shown how such controllers may be tuned and how bounds may be computed in conjunction with known techniques from linear system theory. As another example, in Anta and Tabuada

(2010) an event-driven approach termed "self-triggered control" determines instants when the state should be sampled and control actions taken for some classes of non-linear control systems. Benefits of event-driven mechanisms for estimation purposes are considered in Astrom and Bernhardsson (2002) and Shima, Rasmussen, and Chandler (2007). In Astrom and Bernhardsson (2002), for instance, an event-based sampling mechanism is studied where a signal is sampled only when measurements exceeded a certain threshold and it is shown that this approach outperforms a classical periodic sampling process at least in the case of some simple systems. Finally, in distributed systems, event-driven mechanisms have the advantage of significantly reducing communication among networked components without affecting desired performance objectives; see Wang and Lemmon (2011), Zhong and Cassandras (2010), Heemels, Donkers, and Teel (2013). In Heemels et al. (2013), a control scheme combining periodic (time-driven) and event-driven control is used for linear systems to update and communicate sensor and actuation data only when necessary in the sense of maintaining a satisfactory closed-loop performance. It is shown that this goal is attainable with a substantial reduction in communication over the underlying network. In Zhong and Cassandras (2010), on the other hand, the goal is for a system of networked components to cooperatively maximise (or minimise) a given objective; it is shown that an event-driven scheme can still achieve the optimization objective while drastically reducing communication (hence, prolong the lifetime of a wireless network.)

In the remainder of this paper, we limit ourselves to discussing how the event-driven paradigm is applied in two areas. First, in distributed systems, we review the framework in Zhong and Cassandras (2010) illustrating how event-driven, rather than synchronous, communication can guarantee convergence in cooperative distributed optimization while provably maintaining optimality. This can significantly reduce the energy consumption of wireless devices used in distributed systems (sensor networks, multi-agent systems, etc.) Second, in hybrid systems (HSs), we consider a general-purpose control and optimization framework where controllers are parameterised and the parameters are adaptively tuned on line based on observable data. This process is carried out based on gradients of given performance measures with respect to these parameters, so as to iteratively adjust their values. When the environment is stochastic, this entails generating gradient estimates with desirable properties such as unbiasedness. This gradient estimation approach is based on the Infinitesimal Perturbation Analysis (IPA) theory (Cassandras & Lafortune, 2008; Ho & Cao, 1991) adapted to HSs and results in an "IPA calculus" developed in Cassandras, Wardi, Panayiotou, and Yao (2010) which amounts to a set of simple event-driven iterative equations. This IPA calculus may be used to obtain state sensitivity estimates on line and, ultimately, to solve a large class of optimization problems with little or no knowledge of the noise or random processes affecting their dynamics. As we will see, the event-driven nature of these equations provides a scalable setting for addressing potentially very complex control and optimization problems.

2. Event-driven control and optimization in distributed systems

Distributed control and optimization arise in settings which involve multiple controllable system components cooperating toward a common objective without a central controller to coordinate their actions. The cooperating components define a dynamic system which may be thought of as a network with each component corresponding to a node maintaining its own state s_i, $i = 1, \ldots, N$. The goal of each node is to control its

state so as to optimize some system-wide objective expressed as a function $H(\mathbf{s})$ of $\mathbf{s} = [s_1, \ldots, s_N]$ and possibly the state of the environment.

To achieve such a goal in a dynamic and uncertain environment, the nodes must share, at least partially, their state information. However, this may require a large amount of information flow and becomes a critical issue when the system consists of wireless communicating nodes which are often small, inexpensive devices with limited resources (e.g. a sensor network). Aside from energy required to move (if nodes are mobile), communication is known to be by far the largest consumer of the limited energy of a node (Shnayder, Hempstead, Chen, Allen, & Welsh, 2004), compared to other functions such as sensing and computation. Moreover, every communication among nodes offers an opportunity for corruption or loss of information due to random effects or adversarial action. Therefore, it is crucial to reduce communication between nodes to the minimum possible. This in turn imposes a constraint on the optimization task performed by each node, since it requires that actions be taken without full knowledge of other nodes' states. Standard synchronisation schemes require that nodes exchange state information frequently, usually periodically, which can clearly be inefficient and, in fact, often unnecessary since it is possible that: (i) system inactivity makes the periodic (purely time-driven) exchange of information unnecessary, (ii) occasional state information is adequate for control and/or optimization mechanisms which do not require perfect accuracy at all times and (iii) the state information of other nodes can often be estimated reasonably well and explicit communication is thus redundant. This motivates the need for asynchronous optimization mechanisms in which a node communicates with others only when it considers it indispensable; in other words, each node tries to reduce the cost of communication by transmitting state information only under certain conditions and only as a last resort. The conditions under which a node takes such communication actions defines "events" which in turn drive an underlying optimization scheme.

This general setting applies to a variety of application domains. If the nodes are vehicles, one of the objectives may be to control their locations so as to maintain some desirable formation (Lawton, Young, & Beard, 2000; Ögren, Fiorelli, & Leonard, 2004) while following a given trajectory. In a sensor network, nodes must be placed so as to achieve objectives such as maximising the probability of detecting events in a given region or maintaining a desired distance from data sources that ensures high-quality monitoring (Cassandras & Li, 2005; Cortés et al., 2004; Ganguli, Cortés, & Bullo, 2006; Hokayem, Stipanovic, & Spong, 2007; Hussein & Stipanovic, 2007; Meguerdichian, Koushanfar, Potkonjak, & Srivastava, 2001; Mihaylova, Lefebvre, Bruyninckx, & Gadeyne, 2002; Zhong & Cassandras, 2011; Zou & Chakrabarty, 2003); this is often referred to as a "coverage control" problem. Related to coverage control is the "persistent monitoring" problem where nodes must monitor a dynamically changing environment which cannot be fully covered by a stationary team of nodes; in this case, all areas of a mission space must be visited infinitely often (Pasqualetti, Franchi, & Bullo, 2012; Smith, Schwager, & Rus, 2012; Cassandras, Lin, & Ding, 2013). In some cases, the state of a node may not be its location but rather its perception of the environment which changes based on data directly collected by that node or communicated to it by other nodes; consensus problems fall in this category (DeGroot, 1974; Jadbabaie, Lin, & Morse, 2003; Moreau, 2005).

We consider a distributed system viewed as a network of N cooperating nodes. The system's goal is to minimise an objective function $H(\mathbf{s})$ known to all nodes with every

node controlling its individual state $s_i \in \mathbb{R}^{n_i}$, $i = 1, \ldots, N$. The state update scheme employed by the ith node is of the general form

$$s_i(k+1) = s_i(k) + \alpha_i d_i(\mathbf{s}(k)), \quad k = 0, 1, \ldots \tag{1}$$

where α_i is a constant positive step size and $d_i(\mathbf{s}(k))$ is an *update direction* evaluated at the kth *update event* (see also (Bertsekas & Tsitsiklis, 1997)). We normally use

$$d_i(\mathbf{s}(k)) = -\nabla_i H(\mathbf{s}(k))$$

where $\nabla H(\mathbf{s}(k))$ is the gradient of $H(\mathbf{s}(k))$ and $\nabla_i H(\mathbf{s}(k)) \in \mathbb{R}^{n_i}$. In general, each state is characterised by dynamics of the form $\dot{s}_i(t) = f_i(s_i, u_i, t)$ where $u_i \in \mathbb{R}^l$ is a control vector; for our purposes, however, we treat s_i as a directly controllable vector. Thus, in (1) we view $s_i(k+1)$ as the *desired* state determined at the kth update event and assume that the control u_i is capable of reaching $s_i(k+1)$ from $s_i(k)$ within a time interval shorter than the time between update events.

A fundamental difficulty in (1) is that $\mathbf{s}(k)$ is in fact not fully known to node i. Thus, $d_i(\mathbf{s}(k))$ has to be evaluated by synchronising all nodes to provide their states to node i at the time its kth update event takes place. This is extremely costly in terms of communication; it also assumes no delays so that the state information can be accurate. Alternatively, node i can evaluate $d_i(\mathbf{s}(k))$ using estimates of s_j for all $j \neq i$ relying on prior information from node j and possibly knowledge of its dynamics. This gives rise to the state update scheme

$$s_i(k+1) = s_i(k) + \alpha d_i(\mathbf{s}^i(k)), k = 0, 1, \ldots \tag{2}$$

used by nodes $i = 1, \ldots, N$, where $d_i(\mathbf{s}^i(k))$ is an update direction such that $\mathbf{s}^i(k)$ is the state *estimate* vector evaluated by node i at the kth update event. There are various ways for node i to estimate the state of some $j \neq i$. As an example, the simplest estimate is to use the most recent state information received from node j.

To be more precise, let t_k, $k = 1, 2, \ldots$, denote the time when any one node performs a state update, i.e. it takes an action based on (2). We impose no constraint on when such an update event occurs at a node and only assume that every node performs an update with sufficient frequency relative to the updates of other nodes (this assumption is formally stated in Zhong & Cassandras, 2010). Next, let us discuss the state communication process which is, in general, decoupled from the state update process. Let τ_n^j be the nth time when node j broadcasts its true state to its neighbouring nodes, $n = 1, 2, \ldots$ and set $\tau_0^j = 0$. We assume that at all communication event times, the state information broadcasted by node j can reach any other node with bounded delay, i.e. we assume that the underlying network is connected (if connectivity is lost by some node, then a related problem is to ensure that the node, if mobile, can control its state so as to re-establish such connectivity). We are interested in the most recent communication event from a node $j \neq i$ and define

$$\tau^j(k) = \max\{\tau_n^j : \tau_n^j \leqslant t_k, n = 0, 1, 2, \ldots\} \tag{3}$$

as the time of the most recent communication event at node j up to a state update event at t_k. In order to differentiate between a node state at any time t and its value at the specific update times t_k, $k = 0, 1, \ldots$, we use $x_i(t)$ to denote the former and observe that $s_i(k) = x_i(t_k)$. Returning to the issue of how node i may estimate the state of some $j \neq i$, the simplest is to use the most recent state information received at time $\tau^j(k)$, i.e.

$$s_j^i(k) = x_j(\tau^j(k)) \tag{4}$$

Alternatively, node i may use a dynamic linear estimate of the form

$$s_j^i(k) = x_j(\tau^j(k)) + \frac{t_k - \tau^j(k)}{\Delta_j} \cdot \alpha_j \cdot d_{j,\tau^j(k)} \tag{5}$$

where Δ_j is an estimate of the average time between state updates at node j (e.g. a known constant if node j performs periodic updates) and $d_{j,\tau^j(k)}$ is the update direction communicated by node j at time $\tau^j(k)$ along with its state. Note that $[t_k - \tau^j(k)]/\Delta_j$ is an estimate of the number of state updates at j since its last communication event. Observe that under the assumption that the precise local decision-making process of j is known to i, then i can evaluate $s_j^i(k)$ using this information with initial condition $x_j(\tau^j(k))$. In this case, the estimate is error-free except for noise that may have affected the actual state evolution of node j in the interval $[\tau^j(k), t_k]$. In general, the value of an estimate $s_j^i(k)$ used by node i to estimate node j's state depends on t_k, the most recent communication event time $\tau^j(k)$, and the actual state $x_j(\tau^j(k))$ of node j at that time. In practice, we have found that the simple estimate in (4) performs very well in implementing the event-driven scheme that follows.

We now concentrate on the key issue related to an event-driven communication scheme: determining instants when a node j may communicate its state to other nodes through *communication events*. Such communication events occur at different times for each node, as do each node's state update events, so that the resulting mechanism is fully asynchronous. The scheme developed and analysed in Zhong and Cassandras (2010) is one where a node j maintains an *error function* $g(x_j(t), x_j^i(t))$ of its actual state $x_j(t)$ relative to its state as estimated by other nodes $i \neq j$, $x_j^i(t)$ (which node j can evaluate). This function measures the quality of the state estimate of node i with the requirement that

$$g(x_i(t), x_i^j(t)) = 0 \text{ if } x_i(t) = x_i^j(t) \tag{6}$$

Examples of $g(x_i(t), x_i^j(t))$ include $\| x_i(t) - x_i^j(t) \|_1$ and $\| x_i(t) - x_i^j(t) \|_2$. If different nodes use different means to estimate i's state, then generally $x_i^j(t) \neq x_i^k(t)$ for nodes $j \neq k$ and communication may be limited to a node-to-node process. Let $\delta_i(k)$ be an error *threshold*, determined by node i after the kth state update event. Let $\tilde{k}_t^i$ be the index of the most recent state update time of node i up to t, and τ_n^{ij} be the nth time when node i sends its true state to node j. Let us also set $\tau_0^{ij} = 0$ for all i, j. Then, the communication event policy at node i with respect to node j is specified by

$$\tau_n^{ij} = \inf\{t : g(x_i(t), x_i{}^j(t)) \geqslant \delta_i(\tilde{k}_t^i), t > \tau_{n-1}^{ij}\} \tag{7}$$

When a communication event is triggered by (7) at τ_n^{ij}, assuming negligible communication delay, x_i^j is instantaneously set to $x_i(\tau_n^{ij})$, i.e. $x_i{}^j((\tau_n^{ij})^+) = x_i(\tau_n^{ij})$. Therefore, the error measure is reset to zero, i.e. $g(x_i((\tau_n^{ij})^+), x_i{}^j((\tau_n^{ij})^+)) = 0$. In other words, a node does not incur any communication cost unless it detects that the deviation of its state from the other nodes' estimate of its state becomes too large; this may happen due to the normal state update (2) accumulating noise, imperfect state estimation or through unexpected state changes (e.g. if a mobile node encounters an obstacle).

Regarding the choice of threshold $\delta_i(k)$, the basic idea is to use a large value at the initial stages of the optimization process and later reduce it to ultimately ensure convergence. The approach we follow is to control $\delta_i(k)$ in a manner which is proportional to $\|d_i(\mathbf{s}^i(k))\|_2$, the Euclidean norm of the update direction at the kth update event henceforth denoted by $\|\cdot\|$. Thus, let

$$\delta_i(k) = \begin{cases} K_\delta \|d_i(\mathbf{s}^i(k))\| & \text{if a state update occurs at } t_k \\ \delta_i(k-1) & \text{otherwise} \end{cases} \tag{8}$$

where K_δ is a positive constant. We also impose an initial condition such that

$$\delta_i(0) = K_\delta \|d_i(\mathbf{s}^i(0))\|, \quad i = 1, \ldots, N \tag{9}$$

and

$$s^i_j(0) = x_j(0) \tag{10}$$

Note that (10) can be readily enforced by requiring all nodes to share their initial states at the beginning of the optimization process, i.e. $\tau^{ij}_0 = 0$ for all i, j; since generally $g(x_i(0), x^j_i(0)) \geqslant \delta_i(0)$, this triggers (7) and results in $g(x_i((0)^+), x^j_i((0)^+)) = 0$. for all $i = 1, \ldots, N$. Also note that since $\mathbf{s}^i(k)$ is node i's local estimate of $\mathbf{s}(k)$ at t_k, the computation in (8) requires only local information.

First, assuming negligible communication delays, it is proved in Zhong and Cassandras (2010) that under rather mild technical conditions the resulting optimization scheme converges and leads to a minimum of $H(\mathbf{s})$; this minimum may be local or global depending on the nature of the objective function. The analysis is based on the distributed optimization framework in Bertsekas and Tsitsiklis (1997), but the emphasis here is on controlling the asynchronous occurrence of communication events through the threshold-based scheme outlined above in a way that may drastically reduce the number of such events while still guaranteeing convergence. In what follows, we state the assumptions and the theorem found in Zhong and Cassandras (2010).

Assumption 1. There exists a positive integer B such that for every $i = 1, \ldots, N$ and $k \geq 0$ at least one of the elements of the set $\{k-B+1, k-B+2, \ldots, k\}$ is such that a state update occurs at t_k.

Assumption 2. The objective function $H(\mathbf{s})$, where $\mathbf{s} \in \mathbb{R}^m$, $m = \sum_{i=1}^N n_i$, satisfies the following:

(a) $H(\mathbf{s}) \geqslant 0$ for all $\mathbf{s} \in \mathbb{R}^m$
(b) $H(\cdot)$ is continuously differentiable and $\nabla H(\cdot)$ is Lipschitz continuous, i.e. there exists a constant K_1 such that for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$, $\|\nabla H(\mathbf{x}) - \nabla H(\mathbf{y})\| \leqslant K_1 \|\mathbf{x} - \mathbf{y}\|$.

Assumption 3. There exist positive constants K_2 and K_3 such that for all $i = 1, \ldots, N$ and $k \in \mathcal{C}^i$, we have

(a) $d_i(k)' \nabla_i H(\mathbf{s}^i(k)) \leqslant -\|d_i(k)\|^2 / K_3$
(b) $K_2 \|\nabla_i H(\mathbf{s}^i(k))\| \leqslant \|d_i(k)\|$

Assumption 4. The error function $g(x_i(t), x^j_i(t))$ satisfies the following:

(a) There exists a positive constant K_4 such that $\| x_i(t) - x^j_i(t) \| \leqslant K_4 g(x_i(t), x^j_i(t))$ for all i, j, t
(b) $g(x_i(t), x^j_i(t)) \leqslant \delta_i(\tilde{k}^i_t)$ where $\tilde{k}^i_t$ was defined earlier as the index of the most recent state update time of node i up to t.

Theorem 1: Under Assumptions 1–4, the communication event policy (7), and the state update scheme (2), if the error threshold $\delta_i(k)$ controlling communication events is set by (8)–(9), then there exist positive constants α and K_δ such that $\lim_{k\to\infty}\nabla H(\mathbf{s}(k)) = 0$.

When an explicit noise term is included in (2), this analysis still leads to a similar convergence result under some additional conditions bounding this noise term. Finally, we also allow communication delays to be non-negligible, as long as there exists an upper bound in the number of state update events that occur between the time of a communication event initiated by a node and the time when all nodes receive the communicated message. This requires a simple modification in how communication events are generated. However, the resulting optimization mechanism is shown to still converge to a minimum of $H(\mathbf{s})$ and a formal result similar to Theorem 1 can be found in Zhong and Cassandras (2010).

This event-driven distributed optimization approach has been applied to coverage control problems in which a scheme based on (2) is used in order to deploy sensor nodes in a region (possibly containing polygonal obstacles) so as to maximise the probability of detecting events (e.g. unknown data sources) in this region. Compared to the work in Cassandras and Li (2005), where it was assumed that all nodes have perfect state information by synchronising update events with communication events, this synchronisation requirement is relaxed and communication events are limited to occur according to the event-driven policy outlined above, leading to convergence to the optimum.

3. Event-driven perturbation analysis and optimization in HSs

A HS consists of both time-driven and event-driven components (Cassandras & Lygeros, 2007). The modelling, control and optimization of these systems is quite challenging. In particular, the performance of a *Stochastic* Hybrid System (SHS) is generally hard to estimate because of the absence of closed-form expressions capturing the dependence of interesting performance metrics on design or control parameters. Most approaches rely on approximations and/or using computationally taxing methods, often involving dynamic programming techniques. The inherent computational complexity of these approaches makes them unsuitable for online optimization. In the case of parametric optimization, on the other hand, application of IPA (Cassandras, Wardi, Melamed, Sun, & Panayiotou, 2002; Cassandras et al., 2010) to SHS has been very successful in online applications. This general-purpose framework is depicted in Figure 1 where an observed sample performance metric is denoted by $L(\theta)$ and its gradient, as evaluated by IPA, is denoted by $\nabla L(\theta)$. This is used by a standard gradient-based scheme to iterate on the value of θ which characterises a control or design policy subsequently affecting the performance of the HS. Note that $\nabla L(\theta)$ is evaluated based on readily available data observed from a *single* sample path of the SHS. More importantly, as discussed in the remainder of this section, this gradient estimate is obtained in event-driven fashion and possesses several attractive properties.

As indicated in Figure 1, the ultimate goal of the iterative process shown is to maximise $E_\omega[L(\theta,\omega)]$, where we use ω to emphasise dependence on a sample path ω of the SHS. This is possible under standard ergodicity conditions imposed on the underlying stochastic processes, as well as the assumption that a single global optimum exists; otherwise, the gradient-based approach is simply continuously attempting to improve the observed performance $L(\theta,\omega)$. Thus, we are interested in estimating the gradient

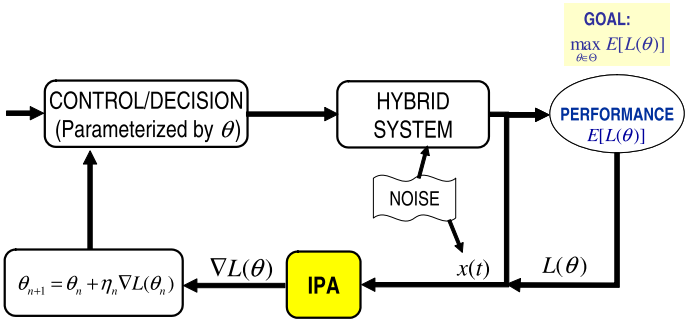

Figure 1. Online gradient-based optimization framework for SHSs.

$$\frac{dJ(\theta)}{d\theta} = \frac{dE_\omega[L(\theta,\omega)]}{d\theta}$$

by evaluating $\frac{dL(\theta,\omega)}{d\theta}$ based on *directly observed data*. We obtain θ^* (under the conditions mentioned above) by optimizing $J(\theta)$ through an iterative scheme of the form

$$\theta_{n+1} = \theta_n - \eta_n H(\theta_n, \omega_n), \quad n = 0, 1, \ldots \tag{11}$$

where $\{\eta_n\}$ is a stepsize sequence and $H(\theta_n, \omega_n)$ is the estimate of $\frac{dJ(\theta)}{d\theta}$ at $\theta = \theta_n$ (in IPA, this is the sample derivative $\frac{dL(\theta,\omega)}{d\theta}$). The conditions under which such algorithms converge are well-known (e.g. see (Kushner and Yin, 1997)). Moreover, it can be shown that such gradient estimates are independent of the probability laws of the stochastic processes involved, require minimal information from the observed sample path, and they are unbiased under very mild technical conditions (Cassandras et al., 2010).

In order to formally apply IPA and optimization methods to SHS, we need to establish a general modelling framework. We use a standard definition of a hybrid automaton (Cassandras & Lygeros, 2007). Thus, let $q \in Q$ (a countable set) denote the discrete state (or mode) and $x \in X \subseteq \mathbb{R}^n$ denote the continuous state. Let $\upsilon \in \Upsilon$ (a countable set) denote a discrete control input and $u \in U \subseteq \mathbb{R}^m$ a continuous control input. Similarly, let $\delta \in \Delta$ (a countable set) denote a discrete disturbance input and $d \in D \subseteq \mathbb{R}^p$ a continuous disturbance input. The state evolution is determined by means of (*i*) a vector field $f: Q \times X \times U \times D \to X$, (*ii*) an invariant (or domain) set *Inv*: $Q \times \Upsilon \times \Delta \to 2^X$, (*iii*) a guard set *Guard*: $Q \times Q \times \Upsilon \times \Delta \to 2^X$ and (*iv*) a reset function $r: Q \times Q \times X \times \Upsilon \times \Delta \to X$. The system remains at a discrete state q as long as the continuous (time-driven) state x does not leave the set $Inv(q, \upsilon, \delta)$. If x reaches a set $Guard(q, q', \upsilon, \delta)$ for some $q' \in Q$, a discrete transition can take place. If this transition does take place, the state instantaneously resets to (q', x') where x' is determined by the reset map $r(q, q', x, \upsilon, \delta)$. Changes in υ and δ are discrete events that either *enable* a transition from q to q' by making sure $x \in Guard(q, q', \upsilon, \delta)$ or *force* a transition out of q by making sure $x \notin Inv(q, \upsilon, \delta)$. We will classify all events that cause discrete state transitions in a manner that suits the purposes of IPA. In what follows, we limit ourselves to an overview of the "IPA calculus" and refer the reader to Cassandras et al. (2010) and Kebarighotbi and Cassandras (2012) for more details and application examples.

The purpose of IPA is to study the behaviour of a HS state as a function of a parameter vector $\theta \in \Theta$ for a given compact, convex set $\Theta \subset \mathbb{R}^l$. Let $\{\tau_k(\theta)\}$, $k = 1, \ldots, K$, denote the occurrence times of all events in the state trajectory. For

convenience, we set $\tau_0 = 0$ and $\tau_{K+1} = T$. Over an interval $[\tau_k(\theta), \tau_{k+1}(\theta))$, the system is at some mode during which the time-driven state satisfies $\dot{x} = f_k(x, \theta, t)$. An event at τ_k is classified as

(i) *Exogenous* if it causes a discrete state transition independent of θ and satisfies $\frac{d\tau_k}{d\theta} = 0$;
(ii) *Endogenous*, if there exists a continuously differentiable function $g_k : \mathbb{R}^n \times \Theta \to \mathbb{R}$ such that $\tau_k = \min\{t > \tau_{k-1} : g_k(x(\theta, t), \theta) = 0\}$; and
(iii) *Induced* if it is triggered by the occurrence of another event at time $\tau_m \leqslant \tau_k$.

IPA specifies how changes in θ influence the state $x(\theta, t)$ and the event times $\tau_k(\theta)$ and, ultimately, how they influence interesting performance metrics which are generally expressed in terms of these variables. Given $\theta = [\theta_1, \ldots, \theta_\Gamma]^{\mathrm{T}}$, we use the Jacobian matrix notation:

$$x'(\theta, t) \equiv \frac{\partial x(\theta, t)}{\partial \theta}, \ \tau'_k \equiv \frac{\partial \tau_k(\theta)}{\partial \theta}, \ k = 1, \ldots, K$$

for all state and event time derivatives. For simplicity of notation, we will omit θ from the arguments of the functions above unless it is essential to stress this dependence. It is shown in Cassandras et al. (2010) that $x'(t)$ satisfies:

$$\frac{d}{dt}x'(t) = \frac{\partial f_k(t)}{\partial x}x'(t) + \frac{\partial f_k(t)}{\partial \theta} \tag{12}$$

for $t \in (\tau_k, \tau_{k+1})$ with boundary condition:

$$x'(\tau_k^+) = x'(\tau_k^-) + [f_{k-1}(\tau_k^-) - f_k(\tau_k^+)]\tau'_k \tag{13}$$

for $k = 0, \ldots, K$. We note that whereas $x(t)$ is often continuous in t, $x'(t)$ may be discontinuous in t at the event times τ_k, hence the left and right limits above are generally different. If $x(t)$ is not continuous in t at $t = \tau_k$, the value of $x(\tau_k^+)$ is determined by the reset function $r(q, q', x, \upsilon, \delta)$ discussed earlier and

$$x'(\tau_k^+) = \frac{dr(q, q', x, \upsilon, \delta)}{d\theta} \tag{14}$$

Furthermore, once the initial condition $x'(\tau_k^+)$ is given, the linearised state trajectory $\{x'(t)\}$ can be computed in the interval $t \in \tau_k(\theta), \tau_{k+1}(\theta))$ by solving (12) to obtain:

$$x'(t) = e^{\int_{\tau_k}^{t} \frac{\partial f_k(u)}{\partial x} du} \left[\int_{\tau_k}^{t} \frac{\partial f_k(v)}{\partial \theta} e^{-\int_{\tau_k}^{t} \frac{\partial f_k(u)}{\partial x} du} dv + \xi_k \right] \tag{15}$$

with the constant ξ_k determined from $x'(\tau_k^+)$ in either (13) or (14).

In order to complete the evaluation of $x'(\tau_k^+)$ in (13), we need to also determine τ'_k. Based on the event classification above, $\tau'_k = 0$ if the event at τ_k is exogenous and

$$\tau'_k = -\left[\frac{\partial g_k}{\partial x} f_k(\tau_k^-)\right]^{-1} \left(\frac{\partial g_k}{\partial \theta} + \frac{\partial g_k}{\partial x} x'(\tau_k^-)\right) \tag{16}$$

if the event at τ_k is endogenous (i.e. $g_k(x(\theta, \tau_k), \theta) = 0.$), defined as long as $\frac{\partial g_k}{\partial x} f_k(\tau_k^-) \neq 0$ (details may be found in Cassandras et al., 2010) Finally, if an induced event occurs at $t = \tau_k$ and is triggered by an event at $\tau_m \leq \tau_k$, the value of τ'_k depends on the derivative τ'_m. The event induced at τ_m will occur at some time $\tau_m + w(\tau_m)$, where

$w(\tau_m)$ is a (generally random) variable which is dependent on the continuous and discrete states $x(\tau_m)$ and $q(\tau_m)$, respectively. This implies the need for additional state variables, denoted by $y_m(\theta, t)$, $m = 1, 2, \ldots$, associated with events occurring at times τ_m, $m = 1, 2 \ldots$ The role of each such state variable is to provide a "timer" activated when a triggering event occurs. Triggering events are identified as belonging to a set $\mathcal{E}_1 \subseteq \mathcal{E}$ and let e_k denote the event occurring at τ_k. Then, define $F_k = \{m : e_m \in \mathcal{E}_1, m \leq k\}$ to be the set of all indices with corresponding triggering events up to τ_k. Omitting the dependence on θ for simplicity, the dynamics of $y_m(t)$ are then given by

$$\dot{y}_m(t) = \begin{cases} -C(t) & \tau_m \leq t < \tau_m + w(\tau_m), m \in F_m \\ 0 & \text{otherwise} \end{cases}$$
$$y_m(\tau_m^+) = \begin{cases} y_0 & y_m(\tau_m^-) = 0, m \in F_m \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where y_0 is an initial value for the timer $y_m(t)$ which decreases at a "clock rate" $C(t) > 0$ until $y_m(\tau_m + w(\tau_m)) = 0$ and the associated induced event takes place. Clearly, these state variables are only used for induced events, so that $y_m(t) = 0$ unless $m \in F_m$. The value of y_0 may depend on θ or on the continuous and discrete states $x(\tau_m)$ and $q(\tau_m)$, while the clock rate $C(t)$ may depend on $x(t)$ and $q(t)$ in general, and possibly θ. However, in most simple cases where we are interested in modelling an induced event to occur at time $\tau_m + w(\tau_m)$, we have $y_0 = w(\tau_m)$ and $C(t) = 1$, i.e. the timer simply counts down for a total of $w(\tau_m)$ time units until the induced event takes place. Henceforth, we will consider $y_m(t)$, $m = 1, 2, \ldots$, as part of the continuous state of the SHS and we set

$$y'_m(t) \equiv \frac{\partial y_m(t)}{\partial \theta}, \; m = 1, \ldots, N. \tag{18}$$

For the common case where y_0 is independent of θ and $C(t)$ is a constant $c > 0$ in (17), the following lemma facilitates the computation of τ'_k for an induced event occurring at τ_k. Its proof is given in Cassandras et al. (2010).

Lemma 1. If in (17) y_0 is independent of θ and $C(t) = c > 0$ (constant), then $\tau'_k = \tau'_m$.

With the inclusion of the state variables $y_m(t)$, $m = 1, \ldots, N$, the derivatives $x'(t)$, τ'_k, and $y'_m(t)$ can be evaluated through (12)–(17) and this set of equations is what we refer to as the "IPA calculus." In general, this evaluation is recursive over the event (mode switching) index $k = 0, 1, \ldots$ In other words, the IPA estimation process in entirely event driven.

For a large class of problems, the SHS of interest does not involve induced events and the state does not experience discontinuities when a mode-switching event occurs. In this case, the IPA calculus reduces to the application of three equations:

(i) (12), which describes how the state derivative $x'(t)$ evolves over $[\tau_k(\theta), \tau_{k+1}(\theta))$,
(ii) (13), which specifies the initial condition ξ_k in (12), and
(iii) either $\tau'_k = 0$ or (16) depending on the event type at $\tau_k(\theta)$, which specifies the event time derivative present in (13).

Since a performance metric $L(\theta)$ is a function of the system state and event times, $\frac{dL(\theta)}{d\theta}$ can subsequently be obtained and used as shown in (11). Using the notation $L'_k(x, t, \theta) \equiv \frac{\partial L_k(x,t,\theta)}{\partial \theta}$, we can rewrite $\frac{dL(\theta)}{d\theta}$ as

$$\frac{dL(\theta)}{d\theta} = \sum_k \left[[\tau'_{k+1} \cdot L_k(\tau^+_{k+1}) - \tau'_k \cdot L_k(\tau^+_k)] + \int_{\tau_k}^{\tau_{k+1}} \left[\frac{\partial L_k(x,\theta,t)}{\partial x} x'(t) + \frac{\partial L_k(x,\theta,t)}{\partial \theta} \right] dt \right] \tag{19}$$

where as already mentioned, $\{\tau_k(\theta)\}$, $k=1, \ldots, K$, is the sequence of all event occurrence times in the state trajectory. In the right-hand-side above, $x'(t)$ and τ'_k are determined through (12), (13) and (16). What makes IPA appealing is the simple form the right-hand-side above often assumes. In fact, it is often the case that the integral term vanishes, leaving *only* a dependence on information related to the event times τ_k, τ_{k+1}, resulting in an IPA estimator which is not only simple to implement, but also independent of the time-driven dynamics $f_k(x,\theta,t)$, hence also any associated noise process Therefore, in such cases the sensitivity of a performance metric with respect to θ is essentially dependent only on the event times and the system behaviour in a neighbourhood of these times. This is formalised in the following result from Yao and Cassandras (2011):

Theorem 2. If either condition (*i*) or (*ii*) below holds, then $\frac{dL(\theta)}{d\theta}$ depends only on information available at event times $\{\tau_k\}$, $k=0, 1, \ldots$

(i) $L_k(x, t, \theta)$ is independent of t over $[\tau_k, \tau_{k+1})$ for all $k=0, 1, \ldots$

(ii) $L_k(x, t, \theta)$ is only a function of x and the following condition holds for all $t \in [\tau_k, \tau_{k+1})$, $k=0, 1, \ldots$:

$$\frac{d}{dt}\frac{\partial L_k}{\partial x} = \frac{d}{dt}\frac{\partial f_k}{\partial x} = \frac{d}{dt}\frac{\partial f_k}{\partial \theta} = 0$$

This provides sufficient conditions under which $\frac{dL(\theta)}{d\theta}$ is independent of t and involves only the event time derivatives τ'_k, τ'_{k+1} and the "local" performance $L_k(\tau^+_{k+1})$, $L_k(\tau^+_k)$ which is obviously easy to observe. In other words, (19) reduces to

$$\frac{dL(\theta)}{d\theta} = \sum_k [\tau'_{k+1} \cdot L_k(\tau^+_{k+1}) - \tau'_k \cdot L_k(\tau^+_k)]$$

Condition (*ii*) of Theorem 2 in particular is satisfied for a large class of systems and objective functions $L(\theta)$. This is an instance of event-driven behaviour for performance sensitivity estimates where the pertinent information is captured by the events taking place in the HS while the state evolution in between events plays a minor or no role in the sensitivity estimation process. Specific detailed examples illustrating this behaviour are given in Cassandras et al. (2010). An additional attractive feature of the "IPA calculus" which can be seen from (19) is that it scales with events in the system. In other words, the steps required to evaluate $\frac{dL(\theta)}{d\theta}$ involve *only* events; in particular, while the complexity of the system may increase prohibitively with its state dimensionality, the complexity of performance sensitivity estimates increases only with the number of events defined over sample paths of the system.

We conclude this overview of the "IPA calculus" with a comment on the unbiasedness of the IPA derivative $dL/d\theta$. This IPA derivative is statistically unbiased (Cassandras & Lafortune, 2008; Ho & Cao, 1991) if, for every $\theta \in \Theta$,

$$E\left[\frac{dL(\theta)}{d\theta}\right] = \frac{d}{d\theta}E[L(\theta)] = \frac{dJ(\theta)}{d\theta}.$$

An important property of IPA in the SHS setting is that it yields unbiased performance derivatives for a large class of systems and performance metrics compared to the traditional DES setting. The following conditions have been established as sufficient for the unbiasedness of IPA (Cassandras et al., 2010):

Theorem 3. Suppose that the following conditions are in force: (*i*) for every $\theta \in \Theta$, the derivative $\frac{dL(\theta)}{d\theta}$ exists w.p.1. (*ii*) w.p.1, the function $L(\theta)$ is Lipschitz continuous on Θ, and the Lipschitz constant has a finite first moment. Fix $\theta \in \Theta$. Then, the derivative $\frac{dJ(\theta)}{d\theta}$ exists, and the IPA derivative $\frac{dL(\theta)}{d\theta}$ is unbiased.

The crucial assumption above is the continuity of the sample performance function $L(\theta)$, which in many SHS is guaranteed in a straightforward manner. Differentiability w.p. 1 at a given $\theta \in \Theta$ often follows from mild technical assumptions on the probability law underlying the system, such as the exclusion of co-occurrence of multiple events. Lipschitz continuity of $L(\theta)$ generally follows from upper boundedness of $|\frac{dL(\theta))}{d\theta}|$ by an absolutely integrable random variable, generally a weak assumption. In light of these observations, the proofs of unbiasedness of IPA have become standardised and the assumptions in Theorem 3 can be verified fairly easily from the context of a particular problem.

4. Conclusions

Glancing into the future of systems and control theory, the main challenges one sees involve larger and ever more distributed wireless networked structures in application areas spanning cooperative multi-agent systems, energy allocation and management and transportation among many others. Barring any unexpected dramatic developments in battery technology, limited energy resources in wireless settings will have to largely dictate how control strategies are designed and implemented so as to carefully optimize this limitation. Taking this point of view, the event-driven paradigm offers an alternative to the time-driven paradigm for modelling, sampling, estimation, control and optimization, not to supplant it but rather complement it. As we have seen in two separate areas which call for such alternatives, control actions need not always be dictated by time steps taken by a "clock", but they can often be triggered by "events" defined in a versatile manner suitable for different applications. In distributed systems, we have described how event-driven communication can guarantee convergence of optimization schemes while provably maintaining optimality. Doing so, can drastically reduce the energy associated with communication processes, therefore contributing to prolonging a wireless network's operational lifetime or limiting the need for frequent battery maintenance. In HSs, we have reviewed the theory of IPA which offers a general-purpose event-driven process for evaluating or estimating (in the case of stochastic systems) gradients of performance metrics. Such information can then be used on line so as to maintain a desirable system performance and, under appropriate conditions, lead to the solution of optimization problems.

Funding

This work was supported in part by National Science Foundation [grant number CNS-1139021]; Air Force Office of Scientific Research [grant number FA9550-12-1-0113]; Office of Naval Research [grant number N00014-09-1-1051]; Army Research Office [grant number W911NF-11-1-0227].

References

Anta, A., & Tabuada, P. (2010). To sample or not to sample: Self-triggered control for nonlinear systems. *IEEE Transactions Automatic Control, 55*, 2030–2042.

Arzen, K. E. (2002). A simple event based PID controller. *Proceedings of 14th IFAC World Congress* (pp. 423–428). Barcelona, Spain.

Astrom, K. J., & Bernhardsson, B. M. (2002). Comparison of Riemann and Lebesgue sampling for first order stochastic systems. *Proceedings of 41st IEEE Conference on Decision and Control* (pp. 2011–2016). Las Vegas, NV.

Bertsekas, D. P., & Tsitsiklis, J. N. (1997). *Parallel and distributed computation: Numerical methods*. Boston, MA: Athena Scientific.

Cassandras, C. G., & Lafortune, S. (2008). *Introduction to discrete event systems* (2nd ed.). Boston, MA: Springer.

Cassandras, C. G., & Li, W. (2005). Sensor networks and cooperative control. *European Journal of Control, 11*, 436–463.

Cassandras, C. G., Lin, X., & Ding, X. C. (2013). An optimal control approach to the multi-agent persistent monitoring problem. *IEEE Transactions on Automatic Control, 58*, 947–961.

Cassandras, C. G., & Lygeros, J. (Eds.). (2007). *Stochastic hybrid systems*. Boca Raton, FL: CRC Press.

Cassandras, C. G., Wardi, Y., Melamed, B., Sun, G., & Panayiotou, C. G. (2002). Perturbation analysis for on-line control and optimization of stochastic fluid models. *IEEE Transactions on Automatic Control, 47*, 1234–1248.

Cassandras, C. G., Wardi, Y., Panayiotou, C. G., & Yao, C. (2010). Perturbation analysis and optimization of stochastic hybrid systems. *European Journal of Control, 16*, 642–664.

Cortés, J., Martínez, S., Karatas, T., & Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation, 20*, 243–255.

DeGroot, M. (1974). Reaching a consensus. *Journal of the American Statistical Association, 69*, 118–121.

Ganguli, A., Cortés, J., & Bullo., F. (2006). Distributed deployment of asynchronous guards in art galleries. *Proceedings of the American Control Conference* (pp. 1416–1421). Minneapolis, MN.

Heemels, W. P., Sandee, J. H., & Bosch, P. P. (2008). Analysis of event-driven controllers for linear systems. *International Journal of Control, 81*, 571–590.

Heemels, W. P. M. H., Donkers, M. C. F., & Teel, A. R. (2013). Periodic event-triggered control for linear systems. *IEEE Transactions on Automatic Control, 58*, 847–861.

Ho, Y. C., & Cao, X. (1991). *Perturbation analysis of discrete event dynamic systems*. Dordrecht: Kluwer Academic.

Hokayem, P. F., Stipanovic, D., & Spong, M. W. (2007). Dynamic coverage control with limited communication. *Proceedings of the 2007 American Control Conference* (pp. 4878–4883). New York, NY.

Hussein, I., & Stipanovic, D. (2007). Effective coverage control using dynamic sensor networks with flocking and guaranteed collision avoidance. *Proceedings of the American Control Conference* (pp. 3420–3425).

Jadbabaie, A., Lin, J., & Morse, S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions Automatic Control, 48*, 988–1001.

Kebarighotbi, A., & Cassandras, C. G. (2012). A general framework for modeling and online optimization of stochastic hybrid systems. *Proceedings of 4th IFAC Conference on Analysis and Design of Hybrid Systems*. Eindhoven, The Netherlands.

Kushner, H. J., & Yin, G. G. (1997). *Stochastic approximation algorithms and applications*. New York, NY: Springer-Verlag.

Lawton, J., Young, B., & Beard, R. (2000). A decentralized approach to elementary formation maneuvers. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation* (pp. 2728–2733).

Lunze, J., & Lehmann, D. (2010). A state-feedback approach to event-based control. *Automatica, 46*, 211–215.

Meguerdichian, S., Koushanfar, F., Potkonjak, M., & Srivastava, M. B. (2001). Coverage problems in wireless ad-hoc sensor networks. *Proceedings of IEEE INFOCOM* (pp. 1380–1387). Anchorage, AK.

Mihaylova, L., Lefebvre, T., Bruyninckx, H., & Gadeyne, K. (2002). Active sensing for robotics-A survey. *Proceedings of the 5th International Conference on Numerical Methods and Applications*. Borovets, Bulgaria.

Moreau, L. (2005). Stability of multi-agent systems with time-dependent communication links. *IEEE Trans-actions Automatic Control, 50*, 169–182.

Ögren, P., Fiorelli, E., & Leonard, N. E. (2004). Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions Automatic Control, 49*, 1292–1302.

Pasqualetti, F., Franchi, A., & Bullo, F. (2012). On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. *IEEE Transactions on Robotics, 28*, 592–606.

Shima, T., Rasmussen, S., & Chandler, P. (2007). UAV team decision and control using efficient collaborative estimation. *ASME Journal of Dynamic Systems, Measurement, and Control, 129*, 609–619.

Shnayder, V., Hempstead, M., Chen, B. R., Allen, G. W., & Welsh, M. (2004). Simulating the power consumption of large-scale sensor network applications. *Proceedings of the International Conference on Embedded Networked Sensor Systems* (pp. 188–200). Los Angeles, CA: ACM Press.

Smith, S. L., Schwager, M., & Rus, D. (2012). Persistent robotic tasks: Monitoring and sweeping in changing environments. *IEEE Transactions on Robotics, 28*, 410–426.

Tabuada, P. (2007). Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions Automatic Control, 52*, 1680–1685.

Wang, X., & Lemmon, M. (2011). Event-triggering in distributed networked control systems. *IEEE Trans-actions Automatic Control, 56*, 586–601.

Yao, C., & Cassandras, C. G. (2011). Perturbation analysis of stochastic hybrid system and applications to resource contention games. *Frontiers of Electrical and Electronic Engineering in China, 6*, 453–467.

Zhong, M., & Cassandras, C. G. (2010). Asynchronous distributed optimization with event-driven communication. *IEEE Transactions Automatic Control, 55*, 2735–2750.

Zhong, M., & Cassandras, C. G. (2011). Distributed coverage control and data collection with mobile sensor networks. *IEEE Transactions Automatic Control, 56*, 2445–2455.

Zou, Y., & Chakrabarty, K. (2003). Sensor deployment and target localization based on virtual forces. *Proceedings of IEEE INFOCOM* (pp. 1293–1303).

Notes on contributor

Christos G. Cassandras received the BS degree from Yale University, the MSEE degree from Stanford University, and the SM and PhD degrees from Harvard University. From 1982 to 1984, he was with ITP Boston, Inc. where he worked on the design of automated manufacturing systems. From 1984 to 1996, he was a faculty member at the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. Currently, he is head of the Division of Systems Engineering and professor of Electrical and Computer Engineering at Boston University, Boston, MA and a founding member of the Center for Information and Systems Engineering (CISE). He specialises in the areas of discrete event and HSs, cooperative control, stochastic optimisation and computer simulation, with applications to computer and sensor networks, manufacturing systems and transportation systems. He has published over 300 papers in these areas, and five books. He was editor-in-chief of the *IEEE Transactions on Automatic Control* from 1998 through 2009 and has served on several editorial boards and as guest editor for various journals. He was the 2012 president of the IEEE Control Systems Society and the recipient of several awards, including the 2011 IEEE Control Systems Technology Award, the Distinguished Member Award of the IEEE Control Systems Society (2006), the 1999 Harold Chestnut Prize (IFAC Best Control Engineering Textbook) for *Discrete Event Systems: Modeling and Performance Analysis*, a 2011 prize for the IBM/IEEE Smarter Planet Challenge competition, a 2012 Kern Fellowship and a 1991 Lilly Fellowship. He is a member of Phi Beta Kappa and Tau Beta Pi, a fellow of the IEEE and a fellow of the IFAC.