Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-18 (2016) 462–467

A Nonsmooth Calculus for Solving Some Graph-Theoretic Control Problems ⋆

Isaac M. Ross * **Mark Karpenko** * **Ronald J. Proulx** *

* *Naval Postgraduate School, Monterey, CA 93943 USA*
{imross, mkarpenk, rjproulx}@ nps.edu

Abstract: Motivated by the needs of real-time tasking of a nonlinear controlled dynamical system, we develop the notion of a real-valued label space to represent a complete graph. A walk in the graph maps to a controlled trajectory in label space. We use the indicator function as our primary tool to formulate such problems within a calculus-based setting. Urysohn's Lemma forms the key bridge for approximating the resulting nonsmooth optimal control problem to a smooth computational framework. An illustrative uninhabited-aerial-vehicle collection planning and and scheduling problem is solved to illustrate the viability of the entire framework.

Keywords: Artificial intelligence, Graph theory, Nonlinear control systems, Optimal control.

1. INTRODUCTION

Planning and scheduling problems are typically considered to be in the realm of artificial intelligence (AI) [Russell and Norvig (2010)]. The operations of many intelligence, surveillance and reconnaissance (ISR) assets, such as an uninhabited aerial vehicle (UAV) or an orbiting spacecraft, involve some form of electromagnetic collection planning and scheduling. In such problems, the vertices of the planning and scheduling graph are ISR tasks and the weights are functionals of the trajectories of the dynamical system. The introduction of controlled dynamics in such graph problems implies the following: in addition to the generation of a task schedule, a classic graph-theoretic problem, the weights of the graph are also part of the unknowns. Thus, a mere construction of the graph requires solving $O(N_v^2)$ control problems, where N_v are the number of vertices. Furthermore, because the weights are not constants, $O(N_v^2)$ control problems need to be solved for each instance of time. Given such seemingly insurmountable issues, it is not surprising that the control problems are relegated to "inner loops" of a larger AI problem. Consequently, the outer AI loop dominates the operational performance characteristics of ISR assets [Ross et al. (2016)].

In this paper, we show that the combined AI-control problem can be addressed at the same loop level. From an AI perspective, we do not generate the weights of the graph; instead, we consider the "law of weights" as determined implicitly by the nonlinear dynamics $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ associated with the ISR asset. This concept is in sharp contrast to the existing literature in that we do not compute the weights explicitly. From a control-theoretic perspective, we consider the AI problem to be part of an optimal control framework that involves the vertex labels as (an indirect) part of the state space. Through the use of an indicator function, the full problem can then be posed as a nonsmooth optimal control problem [Vinter (2000), Clarke et al. (1998)]. Urysohn's Lemma [Tao (2011), Rudin (1991)] forms the key bridge between the nonsmooth optimal control problem formulation and the computational framework. The entire process is illustrated by a sample numerical problem that involves a UAV with a gimballed sensor package mounted on its underside.

⋆ This research was sponsored by the U.S. Navy.

2. DEVELOPMENT OF A REAL-VALUED LABEL SPACE

A collection planning and scheduling graph consists of a finite collection of labeled vertices $N_v \in \mathbb{N}$, where each vertex is called a task. An example of a task is to point a camera to the location of a specific object for the purposes of imaging. Let each task, and hence each vertex of the graph, have an intrinsic nonnegative value, $v_i \in \mathbb{R}_+, i = 1, \ldots, N_v$. The weights of the edges of the graph are controllable by way of an ordinary nonlinear differential equation,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \quad \boldsymbol{u} \in \mathbb{U}(\boldsymbol{x}, t) \subseteq \mathbb{R}^{N_u} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^{N_x}$ is the state vector, $\boldsymbol{u}$ is a control vector that is constrained to a state-dependent set $\mathbb{U}(\boldsymbol{x}, t)$ of $\mathbb{R}^{N_u}$, and t is time. The multifunction $\mathbb{U}$ is not used merely for the sake of greater generality. It turns out that a large number of practical problems involve state-dependent control sets either because of physics-based constraints or because of the choice of the coordinate system [Ross (2015)].

A ***dynamically feasible walk*** is a collection of scheduled tasks that can be accomplished by the ISR asset per some schedule. The value of the walk is given by a payoff function that depends upon the values assigned to the tasks.

10.1016/j.ifacol.2016.10.208

The problem is to find a dynamically feasible walk within a finite time window $[t_0, t_f]$ and clock time constraints that maximizes value of a given payoff function. Additional feasibility constraints are imposed on the problem, and these will be discussed later.

We associate each vertex $i = 1, \ldots, N_v$ of the graph to distinct real-valued sets $\mathbb{L}_i$ in $\mathbb{R}^{N_l}$, $N_l \in \mathbb{N}$; see Fig. 1. Each $\mathbb{L}_i$ may be a point or a continuous subset in $\mathbb{R}^{N_l}$. A vector $\boldsymbol{l} \in \mathbb{R}^{N_l}$ is a collection of real values that represent the labels associated with the vertices of the graph. To execute task i, the condition $\boldsymbol{l} \in \mathbb{L}_i$ must be satisfied. Thus $\mathbb{L}_i$ represents the entire collection of constraints that

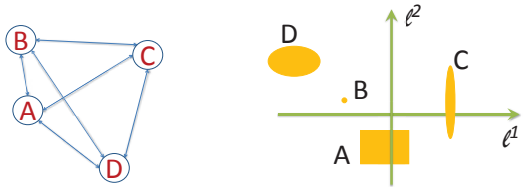

Fig. 1. The vertices of a collection planning graph (left) are mapped to sets $\mathbb{L}_A, \mathbb{L}_B, \mathbb{L}_C$ and $\mathbb{L}_D$ in real-valued label space (right).

must be satisfied to successfully execute task i.

A dynamically feasible walk in the "graph space" generates a trajectory $s \mapsto \boldsymbol{l}$ in "label space." The label-space trajectory passes through the appropriate regions $\mathbb{L}_i, i = 1, \ldots, N_v$ in $\mathbb{R}^{N_l}$. The label space trajectory is implicity governed by (1) by way of some function

$$\boldsymbol{L} : \boldsymbol{x} \mapsto \boldsymbol{l} \tag{2}$$

The "shape" of the label-space trajectory determines the weights of the edges in graph-space and vice versa; see Fig. 2.

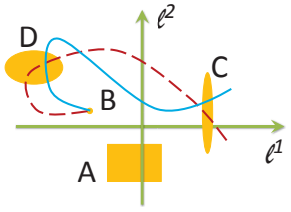

Fig. 2. Illustration of two dynamically feasible label-space trajectories corresponding to the same walk BDC but with different weights.

3. DEVELOPMENT OF A PAYOFF FUNCTIONAL

Let $\mathcal{I}(\cdot, \mathbb{L}_i)$ denote the Kronecker indicator function of $\mathbb{L}_i$ defined by [1]

$$\mathcal{I}(\boldsymbol{l}, \mathbb{L}_i) := \begin{cases} 1 \text{ if } \boldsymbol{l} \in \mathbb{L}_i \\ 0 \text{ if } \boldsymbol{l} \notin \mathbb{L}_i \end{cases} \tag{3}$$

Let $t \mapsto \boldsymbol{x}$ be any given state trajectory. Then the functionals $T_i : \boldsymbol{x}(\cdot) \mapsto \mathbb{R}_+$ defined by

$$T_i[\boldsymbol{x}(\cdot)] := \int_{t_0}^{t_f} \mathcal{I}\big(\boldsymbol{L}(\boldsymbol{x}(t)), \mathbb{L}_i\big)\, dt, \quad i = 1, \ldots, N_v \tag{4}$$

generate the dwell times or ***task times*** over each vertex. Let $\Delta t_i \in \mathbb{R}_+$ denote the value of the dwell time computed by (4); that is,

$$\Delta t_i = T_i[\boldsymbol{x}(\cdot)], \quad i = 1, \ldots, N_v \tag{5}$$

Then, a payoff functional can be modeled through the use of a value function

$$V : (\Delta t_1, \Delta t_2, \ldots, \Delta t_{N_v}) \mapsto \mathbb{R} \tag{6}$$

that associates a value for the set of tasks performed. For instance, one basic model for evaluating a payoff functional that is based solely on the binary condition of instantaneous tasking can be written as,

$$V^0(\Delta t_1, \ldots, \Delta t_{N_v}) := \sum_{i=1}^{N_v} v_i \ \big(1 - \mathcal{I}\big(\Delta t_i, 0\big)\big) \tag{7}$$

In many instances, a minimum amount of task time $\Delta t_i^* > 0$ may be necessary to execute task i. In this case, a payoff functional may be evaluated using the value function

$$V^*(\Delta t_1, \Delta t_2, \ldots, \Delta t_{N_v}) := \sum_{i=1}^{N_v} v_i \ \mathrm{step}\big(\Delta t_i - \Delta t_i^*\big) \tag{8}$$

where, $\mathrm{step}(\xi)$ is defined by

$$\mathrm{step}(\xi) := \begin{cases} 1 \text{ if } \xi \geq 0 \\ 0 \text{ if } \xi < 0 \end{cases}$$

4. DETECTING AND CONSTRAINING AUTONOMOUS RETASKING

The value of the dwell time Δt_i generated by (4) is agnostic to the countable additivity of measuring time intervals. That is, (4) can produce the same Δt_i in countable infinite ways as indicated in Fig. 3. Hence, it is necessary to

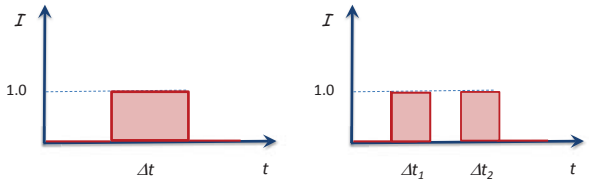

Fig. 3. Schematic of two different functions $t \mapsto \mathcal{I}\big(\boldsymbol{L}(\boldsymbol{x}(t)), \boldsymbol{l}_i\big)$ that generate the same $\Delta t_i = \Delta t = \Delta t_1 + \Delta t_2$.

detect and prevent the same task being performed more times than stipulated. We incorporate such constraints as follows. The derivative of the function $t \mapsto \mathcal{I}\big(\boldsymbol{L}(\boldsymbol{x}(t)), \mathbb{L}_i\big)$, denoted by $d_t\mathcal{I}_i$ is given by,

[1] The indicator function in nonsmooth calculus is used differently [Vinter (2000); Clarke et al. (1998)]; hence, we distinguish our use of the term by using Kronecker as the qualifying adjective at its first usage.