

Hybrid Dynamic Programming*

Wolf Kohn Jeffrey B. Rummel

Hynomics Corporation
3655 131st Avenue S.E., Suite 602
Bellevue, Washington 98006
e-mail: wk@hynomics.com jremmel@hynomics.com

Abstract

In this paper we outline the derivation of a hybrid Hamilton-Jacobi-Bellman equation for our Multiple Agent Hybrid Control Architecture (MACHA) [5, 4] and a hybrid system dynamic programming methodology.

1 Introduction

The classical discrete optimization problem is the following. Given x_0 , find $\min_{u_i \in U} \psi(X_N, n) + \sum_{k=0}^{N-1} \phi(x_k, u_k)$ where $x_{k+1} = f(x_k, u_k)$ for all k . Here U is the space of controls, X is the state space, $f : X \times U \rightarrow X$, $\psi : X \times N \rightarrow R$, and $\phi : X \times U \rightarrow R$ where R and N denote the set of real number and natural numbers respectively. One then defines the usual cost-to-go function

$$V(y, t) = \min_{u_i \in U} \left[\psi(X_N, n) + \phi(y, u_t) + \sum_{k=t+1}^{N-1} \phi(x_k, u_k) \right]. \quad (1)$$

Then $V(x_0, 0)$ is the desired optimal value and $V(y, t)$ satisfies Bellman's equation

$$V(y, t) = \min_{u_t \in U} \phi(y, u_t) + V(f(y, u_t), t + 1) \quad (2)$$

where $V(y, N) = \psi(y, N)$.

In our multiple agent hybrid control architecture we can use chattering to derive a hybrid Bellman equation. That is, corresponding to an autonomous problem, $\dot{x} = F(x, t)$, we construct a non-negative Lagrangian $L(x, \dot{x})$ where we have included time as an extra variable $t = x_{n+1}$ with the constraint that $\dot{x}_{n+1} = 1$. We assume that the Lagrangian $L(x, \dot{x})$ is positive definite in \dot{x} along an extremal tube. That is,

$$(g_{ij}(x, \dot{x}))|_* = \left(\frac{\partial^2(L(x, \dot{x}))}{\partial \dot{x}_j \partial \dot{x}_i} \right) |_*$$

*Research supported by Dept. of Commerce Agreement 70-NANB5H1164.

is a positive definite matrix for each fixed x where

(*) restricted to an extremal tube of trajectories which are solutions to $\dot{x} = F(x, t)$.

Under suitable convexity assumptions on L , we can then define a metric ground form ds on the underlying manifold M such that $ds^2 = \sum_{ij} g_{ij}(x, \dot{x})|_* dx_i dx_j$. A key result of the theory is the fact that the geodesics under the metric ground form induced by L are the extremals of the corresponding parametric calculus of variations problem which is to find an admissible curve on M which minimizes $\int_{t_0}^{t_1} L(x(t), \dot{x}(t)) dt$ satisfying given endpoint conditions. That is, the geodesics are solutions to the Euler-Lagrange equation $\frac{d}{dt} \frac{\partial L(x, \dot{x})}{\partial \dot{x}_i} - \frac{\partial L(x, \dot{x})}{\partial x_i} = 0$.

In our multiple-agent hybrid control architecture, see section 1, each agent wants to minimize the integral of a Lagrangian over trajectories on the carrier manifold M . More precisely, fix a goal x_1 and an element y_1 in the tangent space T_{x_1} . Then for any given initial condition $x(0) = x_0$, each agent wants to force the plant to follow a trajectory $x(\cdot)$ such that $\int_0^T L(x(t), \dot{x}(t), t) dt$ is within ϵ of

$$\min_{x(\cdot) \in M} \int_0^T L(x(t), \dot{x}(t), t) dt. \quad (3)$$

Define a counting variable Z_i by $Z_i = \sum_{n=0}^i \int_{n\Delta}^{(n+1)\Delta} L(x(t), \dot{x}(t), t) dt$ where $(N+1)\Delta = T$ so that

$$Z_{i+1} = Z_i + \int_{(i+1)\Delta}^{(i+2)\Delta} L(x(t), \dot{x}(t), t) dt. \quad (4)$$

If we replace the integral on the righthand side of (4) by its chattering approximation, see [3], we can define a new sequence of counting variables \tilde{Z}_i by

$$\tilde{Z}_{i+1} = \tilde{Z}_i + \sum_{k=0}^s L(x((i+1)\Delta), v_{i+1}^k, (i+1)\Delta) \alpha_{i+1}^k \Delta. \quad (5)$$

where for each i (a) v_i^k for $k = 0, \dots, s$ is in the field of extremals of the original problem (3) at $x(i\Delta)$ and (b) α_i^k are nonnegative reals such that $\sum_{k=0}^s \alpha_i^k = 1$.

Note the desired minimum of the original problem (3) is a geodesic on a manifold M so that we can obtain $x((i+1)\Delta)$ and v_{i+1}^k for $k = 0, \dots, s$ by parallel transport from $x(i\Delta)$ and v_i^k for $k = 0, \dots, s$. That is, assume for any point x_0 in the manifold, there is a unique geodesic starting at x_0 and ending at x_1 with tangent vector y_1 at x_1 . The idea is to compute with the Levi-Civita connection ∇ along an integral curve or geodesic $\alpha : [t_0, t_1] \rightarrow M$ such that $x_0 = \alpha(t_0)$ and $x_1 = \alpha(t_1)$, see [6]. For example, suppose that α lies entirely within some chart (U, ψ) where U is an open set of M and ψ is a homeomorphism of U to a region of Euclidean space. Assume that in the local coordinates of the chart, the curve α is given by $\psi(\alpha(t)) = (\alpha_1(t), \dots, \alpha_n(t))$. Let TM_x denote the tangent space of M at the point x . Now suppose that we are given a tangent vector $y_1 \in TM_{x_1}$ and we want to compute the parallel transport of the tangent vector y_1 along the geodesic α to get a tangent vector y_0 . In such a case, we

will write that $x_1 = \overrightarrow{x_0}(t_1)$ and that $y_1 = \overrightarrow{y_0}(t_1)$. This requires transporting y_1 to a vector $h(t) = \overrightarrow{y_0}(t)$ tangent to the surface at the point $\alpha(t) = \overrightarrow{x_0}(t)$ for all intermediate t . In local coordinates $h(t)$ will be of the form $h(t) = \sum h_i(t) \frac{\partial}{\partial x_i}$. The condition that $h(t)$ is parallel along $\alpha(t)$ is that the covariant derivative be zero along that path. This says that $h(t)$ satisfies

$$0 = \nabla h = \sum_{ijk} \left[\frac{dh_k}{dt} + h_i \frac{d\alpha_j}{dt} \Gamma_{ij}^k \right] \frac{\partial}{\partial x_k} \quad (6)$$

where the Γ_{ij}^k s are the Christoffel symbols.

Thus in (5), we have that $x((t+1)\Delta) = \overrightarrow{\overrightarrow{x(i\Delta)}}(\Delta)$ and $v_{i+1}^k = \overrightarrow{v_i^k}(\Delta)$ for $k = 0, \dots, s$. We can then show that the original problem (3) which is to find $\min_{x(\cdot) \in M} Z_{N+1}$ can be replaced by the relaxed problem which is to find $\inf_{\alpha_i^j} \tilde{Z}_{N+1}$ where $\min_{x(\cdot) \in M} Z_{N+1} + O(\Delta^2) \geq \inf_{\alpha_i^j} \tilde{Z}_{N+1}$.

This formulation allows us to derive a dynamic programming equation for the relaxed problem. That is, define

$$V(\tilde{Z}_i, x, i\Delta) = \inf_{\alpha_j^k} \tilde{Z}_i + \sum_{t=i+1}^N \sum_{k=0}^s L(x(t\Delta), v_t^k, t\Delta) \alpha_t^k \Delta \quad (7)$$

where $x = x(i\Delta)$ and for each $r > i$,

1. $x(r\Delta) = \overrightarrow{\overrightarrow{\overrightarrow{x((r-1)\Delta)}}}(\Delta)$ and $v_r^k = \overrightarrow{v_{r-1}^k}(\Delta)$ for $k = 0, \dots, s$,
2. v_r^k for $k = 0, \dots, s$ is in the field of extremals of the original problem (3) at $x(r\Delta)$, and
3. α_t^r are nonnegative real numbers such that $\sum_{k=0}^s \alpha_r^k = 1$. Then by Bellman's Principle of Optimality,

$$V(\tilde{Z}, x, i\Delta) = \inf_{\alpha_i^k} V(\tilde{Z}_{x, (i+1)\Delta}, x((i+1)\Delta), (i+1)\Delta) \quad (8)$$

where $\tilde{Z}_{x, (i+1)\Delta} = \tilde{Z} + \sum_{k=0}^s L(\overrightarrow{x}(\Delta), \overrightarrow{v_x^k}(\Delta), (i+1)\Delta) \alpha_{i+1}^k \Delta$ and we have the boundary condition that $V(\tilde{Z}, x, N+1) = \tilde{Z}$.

We showed in [7] that V satisfies

$$\begin{aligned} - \frac{\partial_c V}{\partial c t}(\tilde{Z}, x, i\Delta) = \\ \inf_{\alpha_i^k} \left[\left(\frac{\partial_c V}{\partial_c \tilde{Z}}(\tilde{Z}, x, i\Delta) \cdot \left(\sum_{k=0}^s L(\overrightarrow{x}(\Delta), \overrightarrow{v_x^k}(\Delta), (i+1)\Delta) \alpha_{i+1}^k \right) \right) \right. \\ \left. + \frac{\partial_c V}{\partial_c x}(\tilde{Z}, x, i\Delta) \cdot \frac{(\overrightarrow{x}(\Delta) - x)}{\Delta} + \frac{O(\Delta^2)}{\Delta} \right] \quad (9) \end{aligned}$$

Here the notation $\frac{c\partial}{c\partial u}$ stands for the covariant partial derivative with respect to u .

We can compute the terms $\overrightarrow{x}(\Delta)$ and $\overrightarrow{v}_x^k(\Delta)$ which appear on the righthand side of (9) by the following procedure. We can embed the manifold M into a Euclidean space E^n for sufficiently large n . In E^n , the geodesics are straight lines and the second derivatives along the geodesics are 0. Thus we can take the explicit equations of the geodesics and transfer them back to M . In local coordinates, we then get the following equations for the geodesics.

$$\frac{d_c^2 x^j}{d_c t^2} = - \sum_{h=1}^n \sum_{k=1}^n \Gamma_{h,k}^j \frac{d_c x^h}{d_c t} \frac{d_c x^k}{d_c t} \quad (10)$$

where the $\Gamma_{k,l}^j$ are the Christoffel coefficients for our affine connection ∇ on M and $\frac{d_c}{d_c u}$ denotes the total covariant derivative. Thus if we write $y^j = \frac{d_c x^j}{d_c t}$, then we can show that $d_c x^j = y^j \Delta + x^j$ and $d_c y^j = -(\sum_{h=1}^n \sum_{k=1}^n \Gamma_{h,k}^j y^h y^k) \Delta + y^j$ from which we can recover $\overrightarrow{x}(\Delta) - x$ via integration. Similarly we can show that the j th coordinate of the k th element of the field of extremals at x , $v_x^{k,j}$ satisfies the equation

$$v_{\overrightarrow{x}(\Delta)}^{k,j} = - \left(\sum_{s=1}^n \sum_{l=1}^n \Gamma_{s,l}^j v_x^{k,s} (y^l + x^l) \right) \Delta. \quad (11)$$

It follows that we can use these equations to give an explicit expression for $\frac{\partial_c V}{\partial_c t}$ and thus derive an explicit analogue of Jacobi-Bellman equation for Hybrid Systems.

2 Hybrid Dynamic Programming and Automata

Our solution to the hybrid dynamic programming equation described in the previous section is carried out by each agent in our Multiple Agent Hybrid Control Architecture (MAHCA). MAHCA is implemented as a distributed system composed of agents and a communication network which we call the logic communication network. The architecture realizing this system operates as an on-line distributed theorem prover. At any update time, each active agent generates control actions as side effects of proving an existentially quantified subtheorem (lemma) which encodes the model of the plant as viewed by the agent. The conjunction of lemmas at each instant of time, encodes the desired behavior of the entire network. In this section, we shall briefly describe the functionality of an agent's basic modules and the basic computational elements which is employed by an agent to solve its dynamic programming problem.

The basic architecture of an MAHCA agent is pictured Figure 1.

The agent consists of 5 modules with the following functionality:

Planner The Planner constructs and repairs the agent optimization criteria, i.e., it constructs the agent's Lagrangian. The Planner, in fact, generates

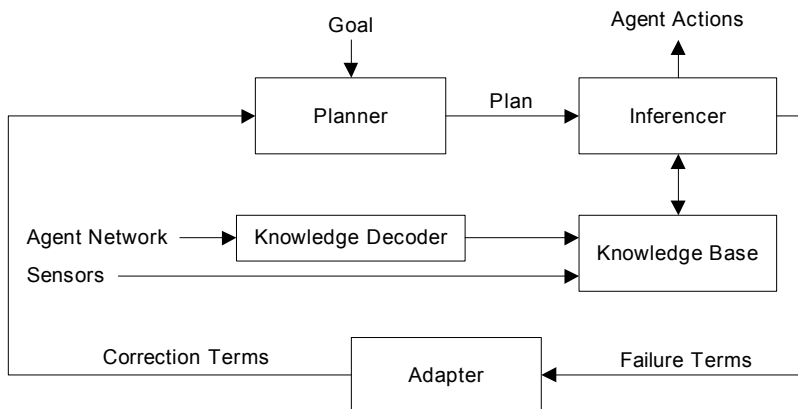


Figure 1: MAHCA Agent Architecture

an existentially quantified logic expression which encapsulates the agent’s model and optimization problem which is called the Behavior Statement.

Inferencer The Inferencer determines whether there is a near optimal solution for the agent’s relaxed variational control problem. If there is such a solution, the agent infers a near optimal solution and sends data to the other agents. Otherwise it infers failure terms and a new state for the agent and reports the failure to the other agents. In particular, the Inferencer determines whether the Behavior Statement is a theorem in the theory currently active in the Knowledge Base. If the Behavior Statement logically follows from the current status of the Knowledge Base, the inferencer generates, as a side effect of proving this Behavior Statement to be true, the current control for the plant. If the Control Statement does not logically follow from the current status of the Knowledge Base, that is, if the desired behavior is not realizable, the inferencer transmits the failed terms to the Adapter module for replacement or modification.

Adapter The Adapter repairs failure terms and constructs correction terms.

Knowledge Base The Knowledge Base stores and updates the agent’s plant model and constraints. The Knowledge Base also stores the requirements of operations or processes within the scope of the agent’s control problem. It also encodes system constraints, interagent protocols and constraints, sensory data, operational and logic principles and a set of primitive inference operations defined in the domain of equational terms.

Knowledge Decoder The Knowledge Decoder receives and translates the other agent’s data.

The agent solves the dynamic programming equation by constructing, on-line, a procedure, termed the inference automaton, that generates the solution.

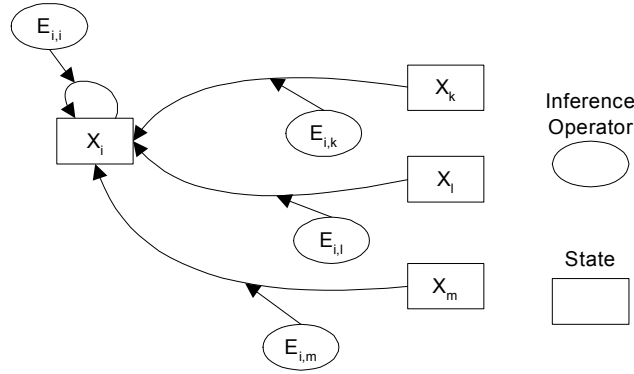


Figure 2: State Graph of Inference Automaton

A solution consists of the control actions to be sent to the process under control and the current state of the agent. The solution is generated in three steps. First the dynamic programming equation is transformed into two subproblems: goal backward propagation and current interval optimization [6]. The coordinated resolution of these two subproblems is generated by a two-level finite state machine (each level computes a solution to one of the two subproblems) referred to as the inference automaton associated with the current plan. The event at the end of this step is a canonical equation for the inference automaton.

The canonical equation of the inference automaton is always version of the Kleene-Schutzenger Equation (KSE). The generic form of a KSE is given below in (12) .

$$Q(V) = E(V) \cdot Q(V) + T(X) \quad (12)$$

where V is the space of instructions I , X is in the computation space CS ,

$$Q(V) = \begin{bmatrix} Q_1(V) \\ \vdots \\ Q_n(V) \end{bmatrix} \text{ is a vector of rules, } E(V) = \begin{bmatrix} E_{1,1}(V) & \cdots & E_{1,n}(V) \\ \vdots & \vdots & \vdots \\ E_{n,1}(V) & \cdots & E_{n,n}(V) \end{bmatrix} \text{ is}$$

the matrix of inference operators of the procedure, and $T(X) = \begin{bmatrix} T_1(X) \\ \vdots \\ T_n(X) \end{bmatrix}$ is the

vector of goals.

In (12) , each entry of the matrix E is a rational form constructed from the basis of inference operators and T is a vector of equational forms from the Knowledge Base. If the (i, j) th entry of the matrix E is a non-empty entry, this represents an edge in the finite state machine from state j to state i , see figure 2. The binary operator \cdot between $E(V)$ and $Q(V)$ represents “apply inference to” operator. This operator is also called unification and is implemented as an evaluation routine.

In the second step, the canonical equation is operated on by the deduction,

evolution and output relation inference operators to construct the inference automaton. The canonical equation is solvable if it is Lyapunov stable and its domain has quasi-regular convergence.

Finally, in the third step, the inference automaton is executed by first decomposing the inference automaton into an equivalent (i.e., same behavior) series-parallel network of simpler inference automata. This is carried out from the canonical equation by the unitary, prefix, loop decomposition, and trimmer inference operators. As the inference automaton executes, it accesses clauses in the Knowledge Base which are used to determine the control actions to be executed and the next state of the agent.

References

- [1] R. Bellman, *Dynamic Programming*, Princeton U. Press, Princeton, NJ, (1957).
- [2] R. W. R. Darling: *Differential Forms and Connections*, Cambridge U. Press, (1995).
- [3] X. Ge, W. Kohn, A. Nerode, and J. B. Remmel: Hybrid Systems: Chattering Approximations to Relaxed Controls. *Hybrid Systems III*, Lecture Notes in Computer Science 1066, Springer-Verlag (1996), 76–100.
- [4] W. Kohn and A. Nerode: A Multiple agent Hybrid Control Architecture In *Logical Methods* (J. Crossley, J. B. Remmel, R. Shore, M. Sweedler, eds.), Birkhauser, (1993) 593–623.
- [5] W. Kohn and A. Nerode: Models for hybrid systems: automata, topologies, controllability and observability, *Hybrid Systems*, Lecture notes in Computer Science 736, Springer-Verlag, (1993), 317–356.
- [6] W. Kohn, A. Nerode, and J. B. Remmel, Hybrid Systems as Finsler Manifolds: Finite State Control as Approximation to Connections, *Hybrid Systems II*, Lecture Notes in Computer Science vol. 999, Springer-Verlag (1995), 294–321.
- [7] W. Kohn, A. Nerode, and J.B. Remmel: Feedback Derivations: Near Optimal Controls for Hybrid Systems, Proc. of CESA '96 IMACS Multiconference vol. 2, 517–521.