Proceedings of Machine Learning Research vol 144:1–12, 2021

# Abstraction-based branch and bound approach to Q-learning for hybrid optimal control

**Benoît Legat** BENOIT.LEGAT@UCLOUVAIN.BE
**Raphaël M. Jungers** RAPHAEL.JUNGERS@UCLOUVAIN.BE
*ICTEAM, UCLouvain, 4 Av. G. Lemaître, 1348 Louvain-la-Neuve, Belgium*

**Jean Bouchat** JEAN.BOUCHAT@UCLOUVAIN.BE
*ELI, UCLouvain, 2 Croix du Sud, 1348 Louvain-la-Neuve, Belgium*

## Abstract

In this paper, we design a theoretical framework allowing to apply model predictive control on hybrid systems. For this, we develop a theory of approximate dynamic programming by leveraging the concept of alternating simulation. We show how to combine these notions in a branch and bound algorithm that can further refine the Q-functions using Lagrangian duality. We illustrate the approach on a numerical example.

**Keywords:** Hybrid systems, reinforcement learning, approximate dynamic programming, branch and bound

## 1. Introduction

The capability of hybrid systems to model both continuous dynamics and discrete events in the same mathematical model renders them essential in fields such as robotics, automotive control or air traffic management. However, with their ability to model such complex systems come substantial challenges for controlling them. In this work[1], we study finite time horizon optimal control problems on hybrid systems.

For a linear hybrid system, a quadratic cost function and a fixed choice of discrete control inputs, the optimal value of the continuous control inputs can be found solving a Quadratic Program (QP) Bemporad et al. (2002). However, the number of discrete control inputs typically grows exponentially with the time horizon or "size" of the system. In Bemporad and Morari (1999), the authors introduce a Mixed Integer Quadratic Program (MIQP) that simultaneously finds the optimal value of both the discrete and continuous control inputs. While the number of integer variables of the MIQP grows linearly with the time horizon or "size" of the system, MIQPs are NP-hard to solve in general hence this approach is not suitable for the online control of large-sized problems with real-time constraints.

Several approaches were proposed to enable a small horizon Model Predictive Controller (MPC) to satisfy such real-time constraints online along with the control objective. In Gol et al. (2014, 2015), the authors develop an algorithm to obtain a Lyapunov function that guarantees the MPC

1. A version of this paper containing the proofs is available in Legat et al. (2020b). The results of the numerical experiments presented in section 4 can be reproduced using the Code Ocean capsule in Legat et al. (2021). It uses the Dionysos Julia package which relies on the OSQP solver Stellato et al. (2020) for solving quadratic programs through the MathOptInterface Legat et al. (2020a).

controller to reach a target discrete state. Computing this Lyapunov function can however be prohibitive and their method is not ensured to find an optimal solution. In Bouchat et al. (2020); Menta et al. (2020), the authors show how the weak duality of the MIQP allows the refinement of an under-approximation given by a value or Q-function. This value or Q-function can be used as terminal cost of the MPC to improve the cost of the solution found.

Computing a Lyapunov function or an accurate approximation of the value or Q-function over the whole state-space is intractable for most classes of hybrid systems Blondel and Tsitsiklis (2000), hence it seems appropriate to only generate an accurate approximation along the optimal trajectory. As the optimal trajectory is unknown, Bouchat et al. (2020) alternates between 1) a search for a sub-optimal trajectory according to the current under-approximation of the value function using Model Predictive Control (MPC), which they called *forward pass*, and 2) a refinement of the under-approximation of the value function along the trajectory, called *backward pass*.

In section 2, we formalize an approach based on *simulation* relations to obtain *Lyapunov* functions and *Bellman-like* Q-functions. This generalizes the algorithm of Gol et al. (2014) for Lyapunov functions. This abstraction approach provides both a Bellman-like value function on the whole state-space as well as a Lyapunov function on some set $\mathcal{X}_f$ containing the target set.

In section 3, we show how to combine a Lyapunov and a Bellman-like Q-function in a branch and bound algorithm solving an optimal control problem. Since learning an accurate Q-function in the whole state-space is intractable, the algorithm only refines it along trajectories computed with an MPC-approach throughout the algorithm.

In section 4, we demonstrate the algorithm on an example from Gol et al. (2014); Bouchat et al. (2020) illustrated in fig. 1.

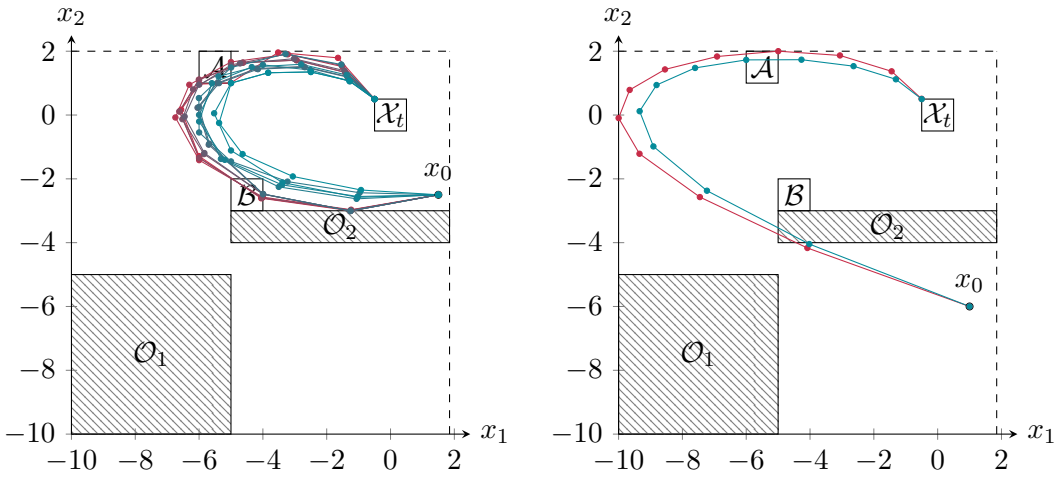


Figure 1: Feasible trajectories providing increasingly better upper bounds found by algorithm 2 on the example detailed in section 4. The first feasible trajectory found by algorithm 2 is represented in red, the last one in blue and the intermediate ones are colored accordingly. The left (resp. right) figure provides the trajectories found for instance $\mathcal{I}_1$ (resp. $\mathcal{I}_2$) with Bellman-like Q-function $Q_2$; see section 4 for the definition of $\mathcal{I}_1, \mathcal{I}_2$ and $Q_2$.

## 2. Discrete optimal control

In this section, we define *simulation relations* between discrete-time systems and show how to deduce a *Lyapunov* function for a system from a Lyapunov function for a simulated system as well as a *Bellman-like* value function for a system from a Bellman-like value function for a simulation.

We use the following notation for discrete-time control systems.

**Definition 1 (Discrete-time control system)** *A* discrete-time control system *is defined as a triple* $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$ *where* $\mathcal{U}$ *is the set of input sets and* $\rightsquigarrow$ *is the subset of transitions* $(x, u, x')$ *such that the system can reach* $x' \in \mathcal{X}$ *from* $x \in \mathcal{X}$ *with input* $u \in \mathcal{U}$*. We denote* $(x, u, x') \in \rightsquigarrow$ *as* $x \underset{u}{\rightsquigarrow} x'$*, and the set of* $x'$ *such that* $x \underset{u}{\rightsquigarrow} x'$ *as* $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x)$.

We denote the set of inputs associated with transitions departing from some state $x \in \mathcal{X}$ as $\mathcal{U}(x) = \{ u \in \mathcal{U} \mid \mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x) \neq \varnothing \}$. Also, we say that a discrete-time control hybrid system is *deterministic* if for every state $x \in \mathcal{X}$ and control input $u \in \mathcal{U}(x)$, $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x)$ is a singleton.

The simulation used in this section is commonly referred to as an *alternating simulation*.

**Definition 2 (Alternating simulation relation (Tabuada, 2009, Definition 4.19 and Definition 4.22))** *Consider discrete-time control systems* $S_1 = (\mathcal{X}_1, \mathcal{U}_1, \rightsquigarrow^1)$ *and* $S_2 = (\mathcal{X}_2, \mathcal{U}_2, \rightsquigarrow^2)$*, as defined in theorem 1. Given a relation* $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$*, consider the* extended relation $R^e$ *defined by the set of* $(x_1, x_2, u_1, u_2)$ *such that for every* $x_2' \in \mathrm{Post}_{\underset{u_2}{\overset{2}{\rightsquigarrow}}}(x_2)$*, there exists* $x_1' \in \mathrm{Post}_{\underset{u_1}{\overset{1}{\rightsquigarrow}}}(x_1)$ *such that* $(x_1', x_2') \in R$*. If for all* $(x_1, x_2) \in R$*, and for all* $u_1 \in \mathcal{U}_1(x_1)$*, there exists* $u_2 \in \mathcal{U}_2(x_2)$ *such that* $(x_1, x_2, u_1, u_2) \in R^e$ *then* $R$ *is an* alternating simulation relation*,* $R^e$ *is its associated* extended alternating simulation relation *and* $S_2$ *is an* alternating simulation *of* $S_1$.

### 2.1. Bellman-like value and Q-functions

In this section, we define *Bellman-like value functions* and *Bellman-like Q-functions*, and show how a Bellman-like value function of a system can be deduced from the Bellman-like value function of an alternating simulation. The Bellman-like value function will be used to provide lower bounds for the branch and bound algorithm in section 3.

We denote the empty tuple as $\emptyset$, the $l$-tuple $(u_i)_{i=1}^l$ as $\mathbf{u}_l$ and the concatenation of tuples $\mathbf{u}_{l_1}, \mathbf{u}'_{l_2}$ as the $(l_1 + l_2)$-tuple $(\mathbf{u}_{l_1}; \mathbf{u}'_{l_2})$. A *cost function* is a function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R} \cup \{\infty\}$ such that $c(x, u) = \infty$ if $u \notin \mathcal{U}(x)$ and a *value function* is a function $V : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$. Given a *Q-function* $Q : \mathcal{X} \times \mathcal{U}^l \to \mathbb{R} \cup \{\infty\}$ for some $l \in \mathbb{N}$ with some cost function $c$, we recursively define the value of $Q(x, \mathbf{u}_{l'})$ for $l' > l$ with the following identity for $k = l + 1, \ldots, l'$:

$$Q(x, \mathbf{u}_k) = \begin{cases} c(x, u_1) + \max_{x' \in \mathrm{Post}_{\underset{u_1}{\rightsquigarrow}}(x)} Q(x', (u_i)_{i=2}^k) & \text{if } u_1 \in \mathcal{U}(x) \\ \infty & \text{otherwise.} \end{cases} \tag{1}$$

Given a cost function $c$ and a value function $V$, $\mathcal{T}_c^Q V$ denotes the Q-function with cost function $c$ such that $\mathcal{T}_c^Q V(x, \emptyset) = V(x)$.

The *Bellman operator* $\mathcal{T}_c$ is defined as[2]:

$$\mathcal{T}_c V(x) = \min_{u \in \mathcal{U}} \mathcal{T}_c^Q V(x, u) \qquad \mathcal{T}_c Q(x, \mathbf{u}_l) = \min_{u' \in \mathcal{U}} Q(x, (\mathbf{u}_l; u')). \tag{2}$$

2. Note that we have $\mathcal{T}_c V(x) = \mathcal{T}_c \mathcal{T}_c^Q V(x, \emptyset)$.

**Definition 3 (Bellman-like value function)** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$. A value function $V$ is a* Bellman-like value function *of $S$ with cost function $c$ if $V(x) \leq \mathcal{T}_c V(x)$ for all $x \in \mathcal{X}$.*

**Definition 4 (Bellman-like Q-function)** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$. A function $Q : \mathcal{X} \times \mathcal{U}^k \to \mathbb{R} \cup \{\infty\}$ is a* Bellman-like Q-function *of $S$ with cost function $c$ if $Q(x, \mathbf{u}_k) \leq Q(x, (\mathbf{u}_k; \mathbf{u}'_l))$ for all $x \in \mathcal{X}$, $\mathbf{u}_k \in \mathcal{U}^k$ and $\mathbf{u}'_l \in \mathcal{U}^l$, where $Q(x, (\mathbf{u}_k; \mathbf{u}'_l))$ is defined from eq. (1).*

**Proposition 5** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$. If $V$ is a Bellman-like value function of $S$ with cost function $c$, then $\mathcal{T}_c^Q V$ is a Bellman-like Q-function of $S$ with cost function $c$.*

**Theorem 6** *Consider discrete-time control systems $S_1 = (\mathcal{X}_1, \mathcal{U}_1, \rightsquigarrow^1)$, $S_2 = (\mathcal{X}_2, \mathcal{U}_2, \rightsquigarrow^2)$, as defined in Definition 1, and an alternating simulation relation $R$ such that for each $x_1 \in \mathcal{X}_1$, there is exactly one $x_2 \in \mathcal{X}_2$ such that $(x_1, x_2) \in R$, which we denote by $R(x_1)$. Given a cost function $c_1$ for $S_1$, consider an associated cost function satisfying $c_2(x_2, u_2) \leq \min_{(x_1, x_2, u_1, u_2) \in R^e} c_1(x_1, u_1)$. If $V_2(x)$ is a Bellman-like value function for $S_2$ with cost function $c_2$, then $V_1(x_1) = V_2(R(x_1))$ is a Bellman-like value function for $S_1$ with cost function $c_1$.*

### 2.2. Lyapunov functions and receding horizon control

In this section, we define *Lyapunov functions* and show how a Lyapunov function of a system can be deduced from a Lyapunov function of an alternatingly simulated system. We then show how a Lyapunov function can ensure that a model predictive controller reaches a target.

**Definition 7 (Lyapunov function)** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, a set $\mathcal{X}_f \subseteq \mathcal{X}$ and a cost function $c$. A value function $L$ is a* Lyapunov function *with cost function $c$ for $S$ in $\mathcal{X}_f$ if, for all $x \in \mathcal{X} \setminus \mathcal{X}_f$, $L(x) = \infty$, and for all $x \in \mathcal{X}_f$, $L(x)$ is finite and $L(x) \geq \mathcal{T}_c L(x)$.*

**Theorem 8** *Consider discrete-time control systems $S_1 = (\mathcal{X}_1, \mathcal{U}_1, \rightsquigarrow^1)$, $S_2 = (\mathcal{X}_2, \mathcal{U}_2, \rightsquigarrow^2)$, as defined in Definition 1, and an alternating simulation relation $R$ such that for each $x_2 \in \mathcal{X}_2$, there is exactly one $x_1 \in \mathcal{X}_1$ such that $(x_1, x_2) \in R$, which we denote by $R(x_2)$. Given a cost function $c_2$ for $S_2$, consider an associated cost function satisfying $c_1(x_1, u_1) \geq \max_{(x_1, x_2, u_1, u_2) \in R^e} c_2(x_2, u_2)$. If $L_1(x)$ is a Lyapunov function for $S_1$ with cost function $c_1$, then $L_2(x_2) = L_1(R(x_2))$ is a Lyapunov function for $S_2$ with cost function $c_2$.*

Receding horizon controllers may not reach the target due to their short-sighted nature. This can be circumvented thanks to a Lyapunov function in several ways, two of which we recall in Proposition 9 and Proposition 10.

The following proposition provides a classical condition for ensuring the convergence of a model predictive controller Mayne (2001).

**Proposition 9 (Mayne (2001).)** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, a target set $\mathcal{X}_t \subseteq \mathcal{X}$, a set $\mathcal{X}_f \subseteq \mathcal{X}$ and a nonnegative cost function $c$. Let $L$ be a nonnegative Lyapunov*

**Data:** Initial state $x_0$, target set $\mathcal{X}_t$, horizon $H$, Q-functions $(Q_k(x))_{k=0}^{\infty}$.
$k \leftarrow 0$
**while** $x_k \notin \mathcal{X}_t$ **do**
  $\mathbf{u}_H^\star \in \arg\min_{\mathbf{u}_H \in \mathcal{U}^H} Q_k(x_k, \mathbf{u}_H)$
  $u_{k+1} \leftarrow u_1^\star$
  $x_{k+1} \in \mathrm{Post}_{\underset{u_{k+1}}{\rightsquigarrow}}(x_k)$
  $k \leftarrow k+1$
**end**
**return** $\mathbf{u}_k, \mathbf{x}_k$

**Algorithm 1:** Receding horizon controller algorithm for a discrete-time control system as defined in Definition 1.

*function with cost function $c$ for $S$ in $\mathcal{X}_f$. Let $v_k = \min_{\mathbf{u}_H \in \mathcal{U}^H} Q_k(x_k, \mathbf{u}_H)$. If $Q_k = \mathcal{T}_c^Q L$ for all $k \in \mathbb{N}$, and*

$$\delta = \inf_{x \in \mathcal{X} \setminus \mathcal{X}_t,\ u \in \mathcal{U}} c(x,u) > 0 \tag{3}$$

*then algorithm 1 terminates in at most $v_0/\delta$ iterations.*

The following proposition generalizes (Gol et al., 2015, Theorem 5.4) where the Lyapunov function is called "distance function". This distance function is computed from the Lyapunov function of an alternatingly simulated system that is constructed with (Gol et al., 2014, Algorithm 2).

**Proposition 10** *Consider a deterministic discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, a target set $\mathcal{X}_t \subseteq \mathcal{X}$, a set $\mathcal{X}_f \subseteq \mathcal{X}$ and the cost function $c$ such that for all $x \in \mathcal{X}$ and $u \in \mathcal{U}$, $c(x,u) = 0$ if $x \in \mathcal{X}_t$ and $c(x,u) = 1$ otherwise. Let $L$ be a Lyapunov function with cost function $c$ for $S$ in $\mathcal{X}_f$ such that $L(x) = 0$ if $x \in \mathcal{X}_t$. Suppose there is $T \in \mathbb{N}$ and value functions $(V_i)_{i=0}^T$ such that $V_i(x)$ is finite if and only if $i \geq L(x)$ and $Q_k(x, \mathbf{u}_H) = \mathcal{T}_c^Q V_{\max(0,T-k-H)}(x, \mathbf{u}_{\min(H,T-k)})$ for $k = 1, 2, \ldots, T$, $x \in \mathcal{X}$ and $\mathbf{u}_H \in \mathcal{U}^H$. Let $v_k = \min_{\mathbf{u}_H \in \mathcal{U}^H} Q_k(x_k, \mathbf{u}_H)$. If $v_0$ is finite, then algorithm 1 terminates in at most $T$ iterations.*

## 3. Branch and bound algorithm

In this section, we show how the concepts of Lyapunov functions and Bellman-like Q-functions can be exploited by a branch and bound algorithm. In section 3.3, we show that the Bellman-like Q-function can be further refined during the optimization by learning it only along the optimal trajectory as approximating it over the whole state-space is not scalable.

For this section we use the following definition of hybrid systems. As it is a special case of Definition 1, it allows to reuse the results of the previous section.

**Definition 11 (Discrete-time control hybrid system)** *A* discrete-time control hybrid system *is defined as a triple $\mathcal{S} = (\mathcal{Q} \times \mathcal{X}, \mathcal{V} \times \mathcal{U}, \rightsquigarrow)$ where $\mathcal{Q}$ is the finite set representing the discrete state-space, $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ represents the continuous state-space, $\mathcal{V}$ is the finite set of discrete control inputs, $\mathcal{U} \subseteq \mathbb{R}^{n_u}$ is the set of continuous control inputs and $\rightsquigarrow$ is the subset of transitions $((q,x),(v,u),(q',x'))$ such that the system can reach $q' \in \mathcal{Q}, x' \in \mathcal{X}$ from $q \in \mathcal{Q}, x \in \mathcal{X}$ with inputs $v \in \mathcal{V}, u \in \mathcal{U}$. We denote $((q,x),(v,u),(q',x')) \in \rightsquigarrow$ as $(q,x) \underset{v,u}{\rightsquigarrow} (q',x')$.*

The optimal control problem is formally defined as follows.

**Problem 1** *Consider a discrete-time control hybrid system $\mathcal{S}$ as defined in Definition 11. The optimal control problem for $\mathcal{S}$ with initial states $q_0 \in \mathcal{Q}, x_0 \in \mathcal{X}$, target set $\mathcal{X}_t$ and cost function $c : \mathcal{Q} \times \mathcal{X} \times \mathcal{V} \times \mathcal{U} \to \mathbb{R}$ is defined as the optimization problem:*

$$\inf_{l \in \mathbb{N},\ \mathbf{v}_l \in \mathcal{V}^l,\ \mathbf{u}_l \in \mathcal{U}^l} \mathcal{T}_c^Q V_0((q_0, x_0), (\mathbf{v}_l, \mathbf{u}_l)) \tag{4}$$

*where $V_0(q, x) = 0$ for all $(q, x) \in \mathcal{X}_t$ and $V_0(q, x) = \infty$ otherwise.*

Given a Bellman-like Q-function $Q$, we define the Q-function $\hat{Q}$ that is only parametrized by the discrete input $\mathbf{v}_k$ as:

$$\hat{Q}((q, x), \mathbf{v}_k) = \min_{\mathbf{u}_k \in \mathcal{U}^k} Q((q, x), (\mathbf{v}_k, \mathbf{u}_k)). \tag{5}$$

The following proposition shows that this is a Bellman-like Q-function as well.

**Proposition 12** *Consider a discrete-time control hybrid system $\mathcal{S}$ as defined in Definition 11. If $Q$ is a Bellman-like Q-function of $\mathcal{S}$ with cost function $c$, then the Q-function $\hat{Q}$ defined by (5) is a Bellman-like Q-function of $\mathcal{S}$ with cost function $c$.*

As shown in Bemporad and Morari (1999), if the cost function $c$ is quadratic then problem 1 can be formulated as a Mixed Integer Quadratic Program (MIQP) and then solved by generic MIQP solvers. On the other hand, we show in the remaining of this section that algorithm 2 can incorporate the information gathered in the computation of a Lyapunov and Bellman-like value functions, as well as refine these functions during the optimization. The aim is to enable the branch and bound algorithm to better exploit the structure of the problem than a generic MIQP solver.

### 3.1. Convergence and optimality

In this section is discussed the convergence and optimality of the algorithm. For optimality, we need to ensure that the condition "$\hat{Q}((q_0, x_0), \mathbf{v}_l) \leq \bar{\beta}$" in the algorithm does not exclude any optimal solution. To this end, we start by proving in Lemma 13 that $\hat{Q}((q_0, x_0), \mathbf{v}_l)$ gives a lower bound to (4), provided that $\hat{Q}$ is a Bellman-like Q-function.

**Lemma 13** *Consider problem 1 for a discrete-time control hybrid system $\mathcal{S}$ with initial states $q_0, x_0$, cost function $c$, and a Bellman-like Q-function $Q$ for $\mathcal{S}$ with cost function $c$. If $Q((q_0, x_0), (\mathbf{v}_l, \mathbf{u}_l)) \leq \mathcal{T}_c^Q V_0((q_0, x_0), (\mathbf{v}_l, \mathbf{u}_l))$ for any $\mathbf{v}_l \in \mathcal{V}^l$ and $\mathbf{u}_l \in \mathcal{U}^l$, then $\hat{Q}((q_0, x_0), \mathbf{v}_k)$ is a lower bound to (4) for any $\mathbf{v}_k \in \mathcal{V}^k$.*

To ensure the convergence of the algorithm, the following assumption excludes pathological optimal control problems that only admit arbitrarily long optimal solutions.

**Assumption 1** *There exists $L \in \mathbb{N}$ and an optimal solution $\mathbf{v}_L, \mathbf{u}_L$ of problem 1.*

**Remark 14** *Assumption 1 ensures that there exists an optimal solution of finite length. A more conservative alternative to assumption 1 would be to assume that the cost function $c$ is lower bounded by a positive number and that there is an upper bound to the optimal cost of the optimal control problem.*

**Data:** Initial states $q_0 \in \mathcal{Q}$, $x_0 \in \mathcal{X}$, a heuristic function $h$, a target set $\mathcal{X}_t$, a Bellman-like Q-function $Q$, an upper bound function $\beta$ and $L \in \mathbb{N}$ satisfying assumption 1.

$\bar{\beta} \leftarrow \infty$
$\mathcal{N} \leftarrow \{\emptyset\}$
**while** $\mathcal{N} \neq \varnothing$ **do**
  $\mathbf{v}_l \leftarrow h(\mathcal{N})$
  $\mathcal{N} \leftarrow \mathcal{N} \setminus \{\mathbf{v}_l\}$
  **if** $\hat{Q}((q_0, x_0), \mathbf{v}_l) \leq \bar{\beta}$ *and* $l < L$ **then**
    **for** $v' \in \mathcal{V}$ **do**
      $\hat{\mathbf{v}}_{\hat{l}}, \hat{\mathbf{u}}_{\hat{l}}, \hat{\beta} \leftarrow \beta(q_0, x_0, (\mathbf{v}_l; v'))$
      **if** $\hat{\beta} < \bar{\beta}$ **then** $\bar{\mathbf{v}}_{\bar{l}}, \bar{\mathbf{u}}_{\bar{l}}, \bar{\beta} \leftarrow \hat{\mathbf{v}}_{\hat{l}}, \hat{\mathbf{u}}_{\hat{l}}, \hat{\beta}$
      $\mathcal{N} \leftarrow \mathcal{N} \cup \{(\mathbf{v}_l; v')\}$
    **end**
  **end**
**end**
**return** $\bar{\mathbf{v}}_{\bar{l}}, \bar{\mathbf{u}}_{\bar{l}}$

**Algorithm 2:** Branch and bound algorithm for problem 1. The set $\mathcal{N}$ represents the set of nodes of the search tree for which subtrees still need to be explored. The heuristic function determines which node is considered next, two different heuristics are discussed in section 4. Note the difference between the notation $\emptyset$ used to denote the empty tuple of discrete control inputs and the notation $\varnothing$ used to denote the empty set.

The following result ensures both the convergence and optimality of algorithm 2.

**Theorem 15** *Consider problem 1 for a deterministic discrete-time control hybrid system $\mathcal{S}$ with initial states $q_0, x_0$, cost function $c$, a Bellman-like Q-function $Q$ for $\mathcal{S}$ with cost function $c$, and an upper bound function $\beta$. Assume that $Q((q_0, x_0), (\mathbf{v}_l, \mathbf{u}_l)) \leq \mathcal{T}_c^Q V_0((q_0, x_0), (\mathbf{v}_l, \mathbf{u}_l))$ for any $\mathbf{v}_l \in \mathcal{V}^l$ and $\mathbf{u}_l \in \mathcal{U}^l$ and that $\beta$ either returns $\emptyset, \emptyset, \infty$ or $v, u, \mathcal{T}_c^Q V_0((q_0, x_0), (\mathbf{v}_l, \mathbf{u}_l))$ with finite $\mathcal{T}_c^Q V_0((q_0, x_0), (\mathbf{v}_l, \mathbf{u}_l))$. Then algorithm 2 returns an optimal solution of problem 1.*

### 3.2. Obtaining upper bounds

Algorithm 2 is parametrized by a function $\beta$ responsible to provide upper bounds. As discussed in section 3.1, the only necessary condition on $\beta$ for the convergence and optimality of algorithm 2 is that $\beta$ should either return nothing or a feasible solution of problem 1. However, a good algorithm for $\beta$, i.e. that provides a feasible solution of low cost, can have a dramatic impact on the efficiency of algorithm 2 as it allows it to prune significant parts of the search tree.

In this section, we introduce a candidate for $\beta$ as algorithm 3. Algorithm 3 searches for feasible trajectories given a fixed prefix of discrete inputs. Its ability to return feasible solutions highly depends on the size of the set $\mathcal{X}_f$ and the horizon $H$. However, computing a Lyapunov function with a larger set $\mathcal{X}_f$ requires more offline computation, while a larger horizon $H$ requires more online computation in algorithm 3. This increase in computational effort might result in better pruning for algorithm 2, hence there is a compromise to reach between a computationally cheap $\beta$ function that often provides a costly feasible solution or no feasible solution at all, and a computationally expensive $\beta$ function that will quickly find good feasible solutions by pruning large parts of the search tree.

**Data:** A deterministic discrete-time control hybrid system $\mathcal{S}$, initial states $q_0 \in \mathcal{Q}, x_0 \in \mathcal{X}$, target set $\mathcal{X}_t$, discrete input $\mathbf{v}_k$, horizon $H$, deadline $T$ and Q-functions $(Q_k)_{k=0}^T$.
Let $Z$ be the value function such that $Z(q,x) = 0$ for all $q \in \mathcal{Q}, x \in \mathcal{X}$
$\mathbf{u}_k \leftarrow \arg\min_{\mathbf{u}_k \in \mathcal{U}^k} \mathcal{T}_c^Q Z((q_0,x_0),(\mathbf{v}_k,\mathbf{u}_k))$
**if** $\mathcal{T}_c^Q Z((q_0,x_0),(\mathbf{v}_k,\mathbf{u}_k)) = \infty$ **then return** $\emptyset, \emptyset, \infty$
Let $(q_k, x_k)$ be the unique pair such that $(q_0,x_0) \underset{\mathbf{v}_k,\mathbf{u}_k}{\rightsquigarrow} (q_k,x_k)$
**if** $\min_{\mathbf{v}_H \in \mathcal{V}^H, \mathbf{u}_H \in \mathcal{U}^H} Q_k((q_k,x_k),(\mathbf{v}_H,\mathbf{u}_H))$ *is finite* **then**
  $(\mathbf{v}'_{l'},\mathbf{u}'_{l'}),(\mathbf{q}'_{l'},\mathbf{x}'_{l'}) \leftarrow$ Algorithm 1 with $(q_k,x_k)$, $H$, $\mathcal{X}_t$,, $(Q_i)_{i=k}^T$
  **return** $(\mathbf{v}_k;\mathbf{v}'_{l'}),(\mathbf{u}_k;\mathbf{u}'_{l'}),\mathcal{T}_c^Q V_0((q_0,x_0),((\mathbf{v}_k;\mathbf{v}'_{l'}),(\mathbf{u}_k;\mathbf{u}'_{l'})))$
**else return** $\emptyset, \emptyset, \infty$

**Algorithm 3:** Upper bound algorithm that can be used as $\beta$ function for algorithm 2.

**Proposition 16** *Consider problem 1 for a deterministic discrete-time control hybrid system $\mathcal{S}$ with initial states $q_0, x_0$ and cost function c. If the Q-functions $(Q_k(x))_{k=0}^T$ satisfy either the assumptions of Proposition 9 or Proposition 10, then algorithm 3 either returns $\emptyset, \emptyset, \infty$ or $\mathbf{v}_l, \mathbf{u}_l, \mathcal{T}_c^Q V_0((q_0,x_0),(\mathbf{v}_l,\mathbf{u}_l))$ with finite $\mathcal{T}_c^Q V_0((q_0,x_0),(\mathbf{v}_l,\mathbf{u}_l))$.*

### 3.3. Q-learning

The difference between the value of the Q-function provided to algorithm 2 with the actual minimal cost of problem 1 has a significant impact on the performance of algorithm 2. As discussed in Bouchat et al. (2020); Menta et al. (2020), the function providing this minimal cost is in general nonlinear and nonconvex. As a matter of fact, a Q-function with a small such difference over the whole state space may not be computable in a reasonable amount of time.

To circumvent this issue, Bouchat et al. (2020) suggests a reinforcement learning approach to generate an approximation of the Q-function that is close to the actual minimal cost near the optimal trajectory of the problem. As the optimal trajectory is unknown, the approach employed by Bouchat et al. (2020), which is classical in Stochastic Programming Birge and Louveaux (2011), consists in alternating between a *forward pass* and a *backward pass*. The forward pass computes a feasible trajectory with a receding horizon controller using the current Q-function as terminal cost. Starting from the end of the trajectory, the backward pass generates new cuts for the Q-function corresponding to each transition using the state at each step of the trajectory.

This backward pass can be used to refine the Q-function along feasible trajectories found by $\beta$ in algorithm 2. Given such trajectory $(\hat{q}_0,\hat{x}_0) \underset{\hat{v}_1,\hat{u}_1}{\rightsquigarrow} (\hat{q}_1,\hat{x}_1) \underset{\hat{v}_2,\hat{x}_2}{\rightsquigarrow} \cdots \underset{\hat{v}_l,\hat{u}_l}{\rightsquigarrow} (\hat{q}_l,\hat{x}_l)$ and a Q-function $Q((q,x),(\mathbf{v}_l,\mathbf{u}_l))$ defined for $l \geq k$ with (1), this learning phase consists in replacing $Q$ by $\max(Q,Q'_j)$ for $j = l-k, l-k-1, \ldots, 1$ where $Q'_j$ is computed as follows. The affine lower approximation $Q'_j$ of the function $\mathcal{T}_c Q((q,x),(\mathbf{v}_k,\mathbf{u}_k))$ is obtained using a feasible solution of the dual of the problem $\min_{v',u'} Q((\hat{q}_j,\hat{x}_j),(((\hat{v}_i)_{i=j+1}^{j+k};v'),((\hat{u}_i)_{i=j+1}^{j+k};u')))$. As shown in the following proposition, the set of Bellman-like Q-functions is invariant under this operation.

**Proposition 17** *Consider problem 1 for a deterministic discrete-time control hybrid system $\mathcal{S}$ with cost function c and a Bellman-like Q-function Q for $\mathcal{S}$ with c. If $Q'((q,x),(\mathbf{v}_k,\mathbf{u}_k)) \leq \mathcal{T}_c Q((q,x),(\mathbf{v}_k,\mathbf{u}_k))$, then $\max(Q,Q')$ is a Bellman-like Q-function.*

## 4. Numerical example

In this section, we illustrate the algorithms developed in this paper on the double integrator dynamics example introduced in (Gol et al., 2014, Example VIII.A). The deterministic discrete-time hybrid control automaton $(\mathcal{Q} \times [-10, 1.85]^2, \mathcal{Q} \times [-2, 2], \rightsquigarrow)$ is such that a transition $(q, x) \underset{q',u}{\rightsquigarrow} (q', x')$ occurs if $x' = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0.5 \\ 1.0 \end{bmatrix} u$, in addition to logical constraints ensuring that a feasible trajectory goes through either square $\mathcal{A}$ or $\mathcal{B}$ before reaching the target square $\mathcal{X}_t$. The squares are represented in Figure 1. See (Gol et al., 2014, Example VIII.A) for more details on the definition of the system. Finally, let the cost

$$c((q, x), (q', u)) = \begin{cases} \infty & \text{if } (q', u) \notin \mathcal{U}((q, x)), \\ u_1^2 & \text{if } (q', u) \in \mathcal{U}((q, x)) \text{ and } (q, x) \in \mathcal{X}_t, \\ u_1^2 + 1 & \text{if } (q', u) \in \mathcal{U}((q, x)) \text{ and } (q, x) \notin \mathcal{X}_t. \end{cases}$$

We benchmark the number of iterations of the branch and bound algorithm with no Lyapunov function and horizon $H = 0$, i.e. upper bounds are only obtained when the candidate $\mathbf{v}_l$ of algorithm 2 is such that $x_k \in \mathcal{X}_t$ in algorithm 3. The heuristic $h$ is a depth-first heuristic, i.e., it selects the candidate $\mathbf{v}_l$ with the largest $k$ and breaks ties by selecting the one with the smallest lower bound $\hat{Q}((q_0, x_0), \mathbf{v}_l)$. Note that, as we have no Lyapunov function and a horizon $H = 0$, a breadth-first heuristic would not be able to prune much of the search tree as $\bar{\beta}$ would remain infinite for most of the iterations.

In order to study the generalization of the Q-function we consider two instances of the optimal control problem: $\mathcal{I}_1$ with $x_0 = [1.5, -2.5], L = 9$ and $\mathcal{I}_2$ with $x_0 = [1, -6], L = 11$.

We analyze the behavior of algorithm 2 with two different Bellman-like Q-functions: $Q_1$ and $Q_2$. The first one is the trivial $Q_1(x, u) = -\infty$ that corresponds to no lower bound hence no pruning in the branch and bound algorithm. The second one, $Q_2$, is obtained by applying theorem 6 to the extended alternating simulation relation $R^e = \{ ((q, x), q, (q', u), q') \mid (q', u) \in \mathcal{U}((q, x)) \}$ and the alternating simulation $(\mathcal{Q}, \mathcal{Q}, \rightsquigarrow')$ such that $q \underset{q'}{\rightsquigarrow'} q'$ if there exists $x, x' \in [-10, 1.85]^2$, $u \in [-2, 2]$ such that $(q, x) \underset{q',u}{\rightsquigarrow} (q', x')$. The number of iterations for different choices of Bellman-like Q-function is given in table 1. Feasible trajectories found by $\beta$ are given in fig. 1.

## 5. Conclusion

The size of abstractions that can simulate the behavior of a hybrid system with enough accuracy in the whole state-space typically grows exponentially with the dimension of the systems. However, as we show in section 2, any abstraction can provide a Lyapunov function over some set $\mathcal{X}_f$ or a Bellman-like value function. Of course, the coarser the abstraction, the smaller the set $\mathcal{X}_f$, and the larger the gap between the value of the Q-function provided to algorithm 2 and the actual minimal cost of problem 1. Nevertheless, we show in section 3 that this information can be leveraged by a branch and bound algorithm. Moreover, as illustrated by our numerical example in section 4, even the Bellman-like value function obtained from a rather coarse abstraction allows drastic pruning of the search tree of the branch and bound algorithm.

Our algorithm is parametrized by a solver for the sub-problems of small horizon that needs to be solved in algorithm 1. For quadratic objectives, this solver can for instance be miOSQP Stellato et al.

| Bellman-like Q-function | Number of iterations for $\mathcal{I}_1$ | Number of iterations for $\mathcal{I}_2$ |
|---|---|---|
| $Q_1$ | 197 234 | 9 388 410 |
| $Q_2$ | 1076 | 96 |
| $Q_2$ with learning | 789 | 75 |
| $Q_2$ learning on $\mathcal{I}_1$ | 782 | 74 |
| $Q_2$ learning on $\mathcal{I}_2$ | 800 | 74 |

Table 1: Number of iterations of algorithm 2 with input parameters described in section 4 for the instances $\mathcal{I}_1$ and $\mathcal{I}_2$ with different Bellman-like Q-functions. We observe that the number of iterations for $\mathcal{I}_2$ is drastically reduced. When using the abstraction-based Q-function $Q_2$, it is divided by appropriately $10^5$. It is also significantly reduced for $\mathcal{I}_1$, by more than $10^3$, and it is further decreased thanks to the learning approach detailed in section 3.3. We also note that the Bellman-like Q-function learned on $\mathcal{I}_2$ generalize well for $\mathcal{I}_1$, with only 800 iterations, and the Q-function learned on the same instance prunes even more nodes, as there is only 782 iterations.

(2018). While such solver could be used directly to solve problem 1, embedding it in algorithm 2 allows to additionally exploit control theoretical concepts such as Lyapunov or Bellman-like Q-functions that are either known for by the control engineer or computed offline via an abstraction as detailed in section 2. Refinements for these functions can then be learned throughout the algorithm and reused to solve variations of the same control problem. Future work should include both a computational and performance-oriented analysis on these benefits.

While the computation of a global Lyapunov function or a good approximation of the minimal cost of problem 1 is not tractable for hybrid systems in general, we can still aim at computing a local Lyapunov function and a Bellman-like value function that are sufficient for an effective pruning of the search tree. For this purpose, it seems appropriate to guide the improvements of these functions using feasible trajectories found during the algorithm. Often, this will in practice enhance the refinement of these functions in the appropriate regions of the state-space. As we show in section 4, the refinement of the Bellman-like value function obtained after algorithm 2 can be reused for solving similar optimal control problem.

Several key research directions of this approach are left as future work. This includes a detailed complexity analysis of the algorithm with, in particular, the complexity of computing the cut in section 3.3. A second line of research is the iterative refinement of the abstractions used to compute the Lyapunov and Bellman-like value functions throughout the algorithms.

## Acknowledgments

RJ is a FNRS honorary Research Associate. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 864017 - L2C. RJ is also supported by the Walloon Region, the Innoviris Foundation, and the FNRS (Chist-Era Druid-net).

## References

Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

Vincent D Blondel and John N Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

Jean Bouchat, Benoît Legat, and Raphaël M Jungers. Reinforcement learning for the optimal control of hybrid systems. Master's thesis, UCLouvain, June 2020.

Ebru Aydin Gol, Mircea Lazar, and Calin Belta. Language-guided controller synthesis for linear systems. *IEEE Transactions on Automatic Control*, 59(5):1163–1176, may 2014. doi: 10.1109/tac.2013.2295664.

Ebru Aydin Gol, Mircea Lazar, and Calin Belta. Temporal logic model predictive control. *Automatica*, 56:78–85, 2015.

Benoît Legat, Oscar Dowson, Joaquim Dias Garcia, and Miles Lubin. MathOptInterface: a data structure for mathematical optimization problems, 2020a. Second round of reviews.

Benoît Legat, Raphaël M. Jungers, and Jean Bouchat. Abstraction-based branch and bound approach to q-learning for hybrid optimal control, 2020b. URL https://arxiv.org/abs/2011.11029.

Benoît Legat, Jean Bouchat, and Raphaël M. Jungers. Abstraction-based branch and bound approach to Q-learning for hybrid optimal control. https://www.codeocean.com/, April 2021. URL https://doi.org/10.24433/CO.6650697.v1.

David Q Mayne. Control of constrained dynamic systems. *European Journal of Control*, 7(2-3): 87–99, 2001.

Sandeep Menta, Joseph Warrington, John Lygeros, and Manfred Morari. Learning solutions to hybrid control problems using benders cuts. In *Learning for Dynamics and Control (L4DC) 2020*, 2020.

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. doi: 10.1007/s12532-020-00179-2. URL https://doi.org/10.1007/s12532-020-00179-2.

Bartolomeo Stellato, Vihangkumar V. Naik, Alberto Bemporad, Paul Goulart, and Stephen Boyd. Embedded mixed-integer quadratic optimization using the OSQP solver. In *2018 European Control Conference (ECC)*. IEEE, jun 2018. doi: 10.23919/ecc.2018.8550136.

Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.