UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXI

Graph-based optimization algorithms with applications to trajectory planning

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutore:
Chiar.mo Prof. Luca Consolini

Dottorando: Mattia Laurini

Anni 2015/2018

“Now I’m hoping some one will care
Living on the breath of a hope to be shared.”

David Bowie
We Are the Dead, Diamond Dogs (1974)

Contents

Chapter 1

Introduction

In this work, we address a class of specially structured problems of form

$$\begin{aligned} &\max_x f(x) \\ &\text{subject to } \; a \leq x \leq g(x), \end{aligned} \tag{1.1}$$

where $x \in \mathbb{R}^n$, $a \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$ is a continuous function, strictly monotone increasing with respect to each component and $g = (g_1, g_2, \dots, g_n)^T : \mathbb{R}^n \to \mathbb{R}^n$, is a continuous function such that, for $i \in \{1, \dots, n\}$, g_i is monotone not decreasing with respect to all variables and constant with respect to x_i. As we will see, this specific class of optimization problems has relevant applications in control engineering.

The structure of this work is the following: in Section 1.2 we introduce the linear subclass (1.3) of class (1.1), in Sections 1.3 and 1.4 we introduce respectively, the graphs and the multigraphs associated to problems in subclass (1.3). In Section 1.5 we provide the notation used in the rest of this work. In Chapter 2 we introduce the basic concepts and mathematical tools needed in the rest of the work. In Chapter 3 we justify the interest in Problem class (1.1) and, in particular, its subclass (1.3), by presenting some problems in control, such as speed planning for autonomous vehicles and robotic manipulators, but also in telecommunications, which can be reformulated as optimization problems within subclass (1.3). In Chapter 4 we focus on another problem of interest, the switching Hamilton-Jacobi-Bellman equation, which can be formulated in such a way that is falls into subclass (1.3) and present an application of it to path planning along with some numerical experiments. In Chapter 5 we derive some theoretical results about Problem (1.1) and a class of iterative algorithms for its solution. In Section 5.2 we do the same for the subclass (1.3). In Section 5.3 we discuss some theoretical and practical issues about convergence speed of the algorithms and we present some numerical experiments. In Chapter 6 we introduce a parallel algorithm for the solution of Problem (1.3). Finally, in Chapter 7 we address a speed planning problem, for which there exists a discretized version falling into subclass (1.3), in the continuous time domain.

1.1 Statement of contribution

The main contributions of this work are the following ones:

- We develop a new iterative procedure (Algorithm 2) for the solution of Problem (1.1) and a more specific one (Algorithm 3) for its linear subclass (1.3). We prove the correctness of these solution methods.
- With numerical experiments, we show that the proposed algorithms outperform generic commercial solvers in the solution of linear problem (1.3).
- We design a new parallel procedure (Algorithm 4) for the solution of Problem (1.3), we show that the procedure is convergent and provide a linear upper bound for the numerical error.
- With numerical experiments, we highlight the benefits of using the new algorithm when solving a linear problem (1.3).
- We introduce a preconditioning technique that allows to increase the convergence speed of the newly introduced algorithms.
- We extend the results of a speed planning problem falling in class (1.3) to the continuous time domain providing a necessary and sufficient condition for its feasibility and an operator for computing its solution.

1.2 Class of problems considered in this work

Let us also assume that there exists a real constant vector U such that

$$(\forall x \in \mathbb{R}^n)\ a \leq x \leq g(x) \Rightarrow g(x) \leq U. \tag{1.2}$$

A Problem related to (1.1) that is relevant in applications is the following one

$$\begin{aligned} &\max_x f(x) \\ &\text{subject to } \ 0 \leq x \leq \bigwedge_{\ell \in \mathcal{L}} \{A_\ell x + b_\ell\},\ x \leq U, \end{aligned} \tag{1.3}$$

where, for each $\ell \in \mathcal{L} = \{1, \ldots, L\}$, with $L \in \mathbb{N}$, A_ℓ is a nonnegative matrix and b_ℓ is a nonnegative vector.

Note that the expression $\bigwedge_{\ell \in \mathcal{L}}$, on the right hand side of (1.3), denotes the greatest lower bound of L vectors. It corresponds to the component-wise minimum of vectors $A_\ell x + b_\ell$, where a different value of $\ell \in \mathcal{L}$ can be chosen for each component. We will show that Problem (1.3) is actually a subclass of (1.1) after a suitable definition of function g in (1.1).

We will also show that the solution of Problems (1.1) and (1.3) is independent on the specific choice of f. Hence, Problem (1.3) is equivalent to the following linear one

$$\begin{aligned} &\max_x \sum_{i=1}^{n} [x]_i \\ \text{subject to } &\ 0 \le x,\ Cx + d \le 0,\ x \le U, \end{aligned} \tag{1.4}$$

where C is a matrix such that every row contains one and only one positive entry and d is a nonpositive vector.

1.3 Graph associated to Problem (1.1)

It is natural to associate to Problem (1.1) a directed graph $\mathbb{G} = (V, E)$, where the nodes correspond to the n components of x and of constraint g, namely $V = \mathcal{V} \cup \mathcal{C}$, with $\mathcal{V} = \{v_1, \ldots, v_n\}$, $\mathcal{C} = \bigcup_{i=1}^{n} \{c_{ij}\}_{j \in J_i}$, with J_i set of indices dependent on i, for $i \in \{1, \ldots, n\}$, where v_i is the variable node associated to $[x]_i$ and $\{c_{ij}\}_{j \in J_i}$ are the constraint nodes associated to g_i. In case a set of indices J_i is such that $|J_i| = 1$, then we will denote $\{c_{ij}\}_{j \in J_i}$ simply with c_i. The edge set $E \subseteq V \times V$ is defined according to the rules:

- for $i \in \{1, \ldots, n\}$ and $j \in J_i$, there is a directed edge from c_{ij} to v_i,
- for $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, n\}$, there is a directed edge from v_i to at least one node in $\{c_{jk}\}_{k \in J_j}$ if g_j depends on x_i,
- no other edges are present in E.

For instance, for $x \in \mathbb{R}^3$ consider problem

$$\begin{aligned} &\max_x f(x) \\ \text{subject to } &\ 0 \le x_1 \le g_1(x_2, x_3) \\ &\ 0 \le x_2 \le g_2(x_1, x_3) = g_{21}(x_1) \wedge g_{22}(x_1, x_3) \\ &\ 0 \le x_3 \le g_3(x_1, x_2). \end{aligned} \tag{1.5}$$

The associated graph, with $\mathcal{V} = \{v_1, v_2, v_3\}$, $\mathcal{C} = \{c_1, c_{21}, c_{22}, c_3\}$, is given in Figure 1.1, where constraint nodes are represented by diamonds and variable nodes by circles.

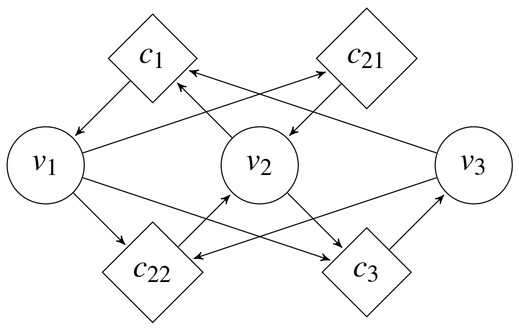


Figure 1.1: Representation of the graph associated to Problem (1.5)

We define the set of neighbors of node $i \in \mathcal{V}$ as

$$\mathcal{N}(i) := \{j \in \mathcal{V} \mid (\exists c \in \mathcal{C})\ (i,c),(c,j) \in E\},$$

namely, a node $j \in \mathcal{V}$ is a neighbor of i if there exists a directed path of length two that connects i to j. For instance, in the previous example, $v_1 \in \mathcal{N}(v_3)$ and $v_2 \notin \mathcal{N}(v_3)$. In other words, $v_j \in \mathcal{N}(v_i)$ if constraint g_j depends on x_i.

1.4 Multigraph associated to Problem (1.1)

Another convenient representation of Problem (1.1) is given by directed multigraphs.

A directed multigraph $\mathbb{G}$ is a tuple $(\mathcal{V},\mathcal{E},\mathcal{L},\varphi,\psi)$, where $\mathcal{V}$ is a finite set whose elements are called nodes (or vertices), $\mathcal{E}$ is a finite multiset [7] of ordered pairs of nodes of $\mathcal{V}$ called edges, $\mathcal{L}$ is a finite set of labels, $\varphi : \mathcal{E} \to \mathcal{V}^2$ is a mapping assigning to every edge its ends and $\psi : \mathcal{E} \to \mathcal{L}$ is a mapping assigning to every edge its label. Note that in directed multigraphs it may be that $e_1, e_2 \in \mathcal{E}$ are such that $e_1 \neq e_2$ and $\varphi(e_1) = \varphi(e_2)$, in this case, we say that e_1 and e_2 are part of the same multiedge and the set of all nodes satisfying this condition constitutes a multiedge. It may also exist $e \in \mathcal{E}$ such that $(\exists i \in \mathcal{V})\ \varphi(e) = (i,i)$, that is, self-loops (edges whose ends coincide with a single vertex) are allowed.

Edges are associated with elements from a set of labels $\mathcal{L}$ (which, for visual aid, may be thought as a set of colors or, as we will see in Chapter 4, as a set of controls) and if a pair of nodes has multiple edges connecting each other in the same direction, then these edges cannot have the same label. Formally, this means that

$$(\forall e_1, e_2 \in \mathcal{E})\ (e_1 \neq e_2 \text{ and } \varphi(e_1) = \varphi(e_2)) \;\Rightarrow\; \psi(e_1) \neq \psi(e_2).$$

A multigraph can be represented by a set of adjacency matrices; the number of matrices is given by the cardinality of the set of labels, each matrix represents the edges sharing the same label.

Given a directed multigraph $\mathbb{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \varphi, \psi)$, we define the set of neighbors of node $i \in \mathcal{V}$ as

$$\mathcal{N}(i) = \{ j \in \mathcal{V} \mid (\exists e \in \mathcal{E})\ \varphi(e) = (i, j) \}. \tag{1.6}$$

If we, again, consider Problem (1.5), the associated multigraph $\mathbb{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \varphi, \psi)$, with $\mathcal{V} = \{v_1, v_2, v_3\}$, $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, $\mathcal{L} = \{\text{black}, \text{red}\}$, $\varphi(e_1) = (v_1, v_2)$, $\varphi(e_2) = (v_1, v_2)$, $\varphi(e_3) = (v_1, v_3)$, $\varphi(e_4) = (v_2, v_1)$, $\varphi(e_5) = (v_2, v_3)$, $\varphi(e_6) = (v_3, v_1)$, $\varphi(e_7) = (v_3, v_2)$, $(\forall e \in \mathcal{E})\ e \neq e_2 \Rightarrow \psi(e) = \text{black}$ and $\psi(e_2) = \text{red}$, is given in Figure 1.2.

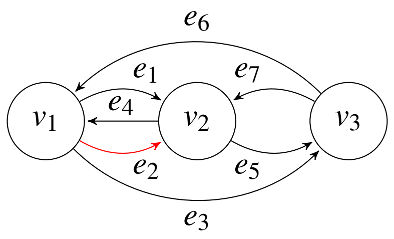


Figure 1.2: Representation of the multigraph associated to Problem (1.5)

Note that set $\{e_1, e_2\}$ constitutes a multiedge, in fact $\varphi(e_1) = \varphi(e_2) = (v_1, v_2)$ and $\psi(e_1) = \text{black}$, $\psi(e_2) = \text{red}$.

1.5 Notation

The set of nonnegative real numbers is denoted by $\mathbb{R}_+ := [0, +\infty)$ and $\underline{0}$ denotes the zero vector of $\mathbb{R}^n$.

Given $n, m \in \mathbb{N}$, let $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times m}$, for $i \in \{1, \ldots, n\}$, we denote the i-th component of x with $[x]_i$ and the i-th row of A with $[A]_{i*}$; further, for $j \in \{1, \ldots, m\}$ we denote the j-th column of A with $[A]_{*j}$ and the ij-th element of A with $[A]_{ij}$. Given matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, the Kronecker product of A and B is denoted by $A \otimes B$ and is given by the following block matrix in $\mathbb{R}^{mp\times nq}$:

$$A \otimes B := \begin{bmatrix} [A]_{11}B & \cdots & [A]_{1n}B \\ \vdots & \ddots & \vdots \\ [A]_{m1}B & \cdots & [A]_{mn}B \end{bmatrix}. \tag{1.7}$$

Function $\|\cdot\|_\infty : \mathbb{R}^n \to \mathbb{R}_+$ is the infinity norm, namely the maximum norm, of $\mathbb{R}^n$ (i.e., $(\forall x \in \mathbb{R}^n)\ \|x\|_\infty = \max_{i\in\{1,\ldots,n\}} |[x]_i|$); $\|\cdot\|_\infty$ is also used to denote the induced matrix norm.

Given a finite set S, the cardinality of S is denoted by $|S|$, the powerset of S is denoted by $\wp(S)$, the complement of S is denoted by S^c, symbol $\times$ denotes the cartesian product, symbol $\setminus$ denotes the set difference and symbol $\varnothing$ denotes the empty set.

We will use symbol $\lightning$ at the end of a proof to state that a contradiction has been reached.

Given $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}^n$ and $x \in \mathbb{R}^n$, we denote the monomial $[x]_1^{\alpha_1} \cdots [x]_n^{\alpha_n}$ with x^α and if $D_i := \frac{\partial}{\partial x_i}$, for $i \in \{1, \dots, n\}$, then the differential operation $D_1^{\alpha_1} \cdots D_n^{\alpha_n}$ with order $|\alpha| := \sum_{i=1}^{n} \alpha_i$ is denoted by D^α.

Finally, given a nonempty set $\mathcal{V}$ let us define a priority queue Q as a finite subset of $\mathcal{Q} := \mathcal{V} \times \mathbb{R}$ such that, if $(v,q) \in Q$, then, no other element $(\bar{v},\bar{q}) \in Q$ can satisfy that $\bar{v} = v$. Let us also define two operations on priority queues: $\text{Enqueue} : \wp(\mathcal{Q}) \times \mathcal{Q} \to \mathcal{Q}$, which, given $Q \in \wp(\mathcal{Q})$ and $(v,q) \in \mathcal{Q}$, if Q does not contain elements of the form (v,p), with $p \geq q$, then Enqueue adds (v,q) to the priority queue Q and removes any other element of the form (v,p), with $p < q$, if previously present. The second operation we need on priority queues is $\text{Dequeue} : \wp(\mathcal{Q}) \to \wp(\mathcal{Q}) \times \mathcal{V}$ which extracts from a priority queue Q the pair (v,q) with highest priority (i.e., it extracts $(v,q) \in Q$ such that $\forall (\bar{v},\bar{q}) \in Q, q \geq \bar{q}$) and returns element v.

Chapter 2

Background

In this chapter we introduce the basic concepts used in the following chapters. In particular, the basic definitions and properties of posets and lattices presented in Section 2.1 will be key for the understanding of Chapters 5 and 7. Section 2.2 introduces finite state automata used in Chapter 4, whilst Sections 2.3 and 2.4 are devoted to introducing Sobolev spaces and differential equations with discontinuous right-hand side involved in Chapter 7.

2.1 Posets and lattices

This section provides a brief summary of the essential concepts and properties of ordered sets and lattice theory needed in Chapter 5 and Chapter 7. The notions stated in the following are taken from [17, 45], to which we refer the reader for a more in-depth and comprehensive discussion.

Definition 2.1. Given sets A and B, a subset $R \in \wp(A \times B)$ of $A \times B$ is called a *binary relation*.

Definition 2.2. A binary relation $\sqsubseteq$ on a set A (that is, on $A \times A$) that satisfies the reflexive property: $(\forall a \in A)\ (a,a) \in \sqsubseteq$, and the transitive property: $(\forall a,b,c \in A)\ ((a,b) \in \sqsubseteq \text{ and } (b,c) \in \sqsubseteq) \Rightarrow (a,c) \in \sqsubseteq$, is called a *pre-order*.

Definition 2.3. A pre-order $\sqsubseteq$ on a set P that satisfies the antisymmetric property: $(\forall p,q \in P)\ (p \sqsubseteq q \wedge q \sqsubseteq p) \Rightarrow p = q$, is called a *partial order* and $P(\sqsubseteq)$ is said to be a *partially ordered set* (or *poset*).

Consider the binary relation $\leq$ defined on $\mathbb{R}^n$, with $n \in \mathbb{N}$, as follows

$$(\forall x,y \in \mathbb{R}^n)\ x \leq y \iff y - x \in \mathbb{R}^n_+. \tag{2.1}$$

It is easy to verify that $\leq$ is a *partial order* of $\mathbb{R}^n$.

Definition 2.4. The *converse* $\reflectbox{$R$}$ of a binary relation R is itself a binary relation such that $(b,a) \in \reflectbox{$R$} \Leftrightarrow (a,b) \in R$.

Theorem 2.5 (*Duality Principle*)**.** *The converse of any partial order is itself a partial order.*

Definition 2.6. Let $P(\sqsubseteq)$ be a poset. u is an *upper bound* of $X \subseteq P$ if and only if $(\forall x \in X)\ x \sqsubseteq u$. The *least upper bound* $\tilde{u}$ of $X \subseteq P$ is such that $(\forall x \in X)\ x \sqsubseteq \tilde{u}$ and $(\forall u \in P)\ (\forall x \in X)\ x \sqsubseteq u \Rightarrow \tilde{u} \sqsubseteq u$, and it is denoted by $\bigsqcup X$.

Observation 1. If such an element exists, it is unique. For if $\tilde{u}$ and $\tilde{v}$ are two least upper bound, it holds that $\tilde{v} \sqsubseteq \tilde{u}$ and $\tilde{u} \sqsubseteq \tilde{v}$, so, by antisymmetry $\tilde{u} = \tilde{v}$.

Definition 2.7. The *lower bound* and *greatest lower bound* of $X \subseteq P$ are *dual* to the concepts of upper bound and least upper bound respectively. This means that their definition can be obtained by the one of upper bound and least upper bound substituting $\sqsubseteq$ with its converse $\sqsupseteq$.
The greatest lower bound is denoted by $\bigsqcap X$.

Definition 2.8. Given a poset $P(\sqsubseteq)$, if it holds that

$$(\forall p,q \in P)\ p \sqsubseteq q \text{ or } p \sqsupseteq q,$$

then $\sqsubseteq$ is said to be *linear* and $P(\sqsubseteq)$ is said to be a *totally ordered set* or a *chain*.

Definition 2.9. A poset $L(\sqsubseteq)$ such that any two elements $a,b \in L$ have a greatest lower bound (called *meet* and denoted by $a \sqcap b$) and a least upper bound (called *join* and denoted by $a \sqcup b$) is called a *lattice* and it is denoted by $L(\sqsubseteq,\sqcap,\sqcup)$.
A lattice $L(\sqsubseteq,\sqcap,\sqcup)$ is *complete* if $(\forall X \subseteq L)\ \exists \bigsqcup X \in L$ and $\exists \bigsqcap X \in L$ and it is denoted by $L(\sqsubseteq,\top,\bot,\sqcap,\sqcup)$. Such a lattice has two distinctive elements: the *infimum* $\bot := \bigsqcup \varnothing = \bigsqcap L$, which is the smallest element of L, and the *supremum* $\top := \bigsqcap \varnothing = \bigsqcup L$, which is the biggest one.

Note that $\mathbb{R}^n(\leq)$ with $\leq$ defined as in (2.1) is a lattice in which the meet is denoted by $\wedge$ and the joint by $\vee$. Moreover, given any compact subset $P \subseteq \mathbb{R}^n$, it is easy to see that $P(\leq,\wedge,\vee)$ is a complete lattice in which, for any $S \subseteq P$, the greatest lower bound of S is denoted by $\bigwedge S$ and the least upper bound of S by $\bigvee S$.

Theorem 2.10. *Given a non-empty poset $P(\sqsubseteq)$, the following statements are equivalent:*

i. P is a complete lattice,

ii. for every subset S of P, $\bigsqcap S$ exists in P, *iii. P has a top element $\top$ and for every subset S of P, $\bigsqcap S$ exists in P.*

Observation 2. Given a lattice $L(\sqsubseteq,\sqcap,\sqcup)$, it follows immediately from the Duality Principle 2.5 and the lattice's Definition 2.9, that its dual $L(\sqsubseteq,\sqcap,\sqcup)' := L(\sqsupseteq,\sqcup,\sqcap)$ is a lattice itself and, moreover, if L is complete, L' is also complete.

Definition 2.11. Given a lattice $L(\sqsubseteq,\sqcap,\sqcup)$ and $a,b \in L$, the *interval* $[a,b]$ with endpoints a and b is defined as follows: $[a,b] := \{x \in L \mid a \sqsubseteq x \sqsubseteq b\}$.

Observation 3. $[a,b](\sqsubseteq)$ is obviously a lattice and if $L(\sqsubseteq,\sqcap,\sqcup)$ is complete, $[a,b](\sqsubseteq,\sqcap,\sqcup)$ is also complete.

Definition 2.12. Given posets $P(\sqsubseteq)$ and $Q(\leq)$, a function $\varphi \in P \to Q$ is *meet-preserving* if

$$(\forall p,p' \in P)\ \varphi(p \sqcap p') = \varphi(p) \wedge \varphi(p').$$

Definition 2.13. Given posets $P(\sqsubseteq)$ and $Q(\leq)$, a function $\varphi \in P \to Q$ is *order-preserving* if

$$(\forall p,p' \in P)\ p \sqsubseteq p' \Rightarrow \varphi(p) \leq \varphi(p').$$

Proposition 2.14. *Given lattices $P(\sqsubseteq,\sqcap,\sqcup)$ and $Q(\leq,\wedge,\vee)$ and a function $\varphi : P \to Q$, the following statements are equivalent:*

i. φ is order-preserving,

ii. $(\forall p,p' \in P)\ \varphi(p \sqcup p') \geq \varphi(p) \vee \varphi(p')$,

iii. $(\forall p,p' \in P)\ \varphi(p \sqcap p') \leq \varphi(p) \wedge \varphi(p')$.

Definition 2.15. Given a poset $P(\sqsubseteq)$ and a function $\varphi \in P \to P$, a point $x \in P$ is a *fixed point* of φ if $\varphi(x) = x$.

We can now state the Knaster-Tarski Fixpoint Theorem.

Theorem 2.16 (*Knaster-Tarski Fixpoint Theorem*)**.** *Given a complete lattice $L(\sqsubseteq,\top,\bot,\sqcap,\sqcup)$ and an order-preserving function $\varphi : L \to L$, then*

$$\{x \in L \mid \varphi(x) = x\} \neq \varnothing.$$

Moreover, the element

$$\sqcap\varphi := \bigsqcap\{x \in L \mid \varphi(x) \sqsubseteq x\},$$

is the greatest fixed point of φ. Dually, it also holds that

$$\sqcup\varphi := \bigsqcup\{x \in L \mid \varphi(x) \sqsupseteq x\},$$

is the least fixed point of φ.

2.2 Finite state automata

This section introduces finite state automata and regular languages and is based on [42, 27].

Definition 2.17. A finite state automaton is a tuple $\mathbb{A} = (\mathcal{I}, \Sigma, \rho, i_0, \mathcal{I}_F)$ with

- $\mathcal{I} \neq \varnothing$ is a finite non-empty set whose elements are called states,
- $\Sigma \neq \varnothing$ is a finite non-empty set called alphabet and whose elements are called symbols,
- $\rho : \mathcal{I} \times \Sigma \to \mathcal{I}$ is the transition function,
- $i_0 \in \mathcal{I}$ is the initial state,
- $\mathcal{I}_{\mathrm{F}} \subseteq \mathcal{I}$ is a subset of states called the final states.

Given an alphabet Σ, let us define the Kleene closure Σ^* of Σ, that is, the set of all strings over alphabet Σ, including the empty string denoted by ε, as follows

$$\begin{aligned} \Sigma^0 &:= \{\varepsilon\}, \\ \Sigma^1 &:= \Sigma, \\ (\forall i \in \mathbb{N})\ \Sigma^{i+1} &:= \{\sigma\tau \mid \sigma \in \Sigma^i \text{ and } \tau \in \Sigma^i\}, \\ \Sigma^* &:= \bigcup_{i \in \mathbb{N}} \Sigma^i. \end{aligned} \tag{2.2}$$

A language over Σ is a subset of Σ^*.

Given a finite state automaton $\mathbb{A} = (\mathcal{I}, \Sigma, \rho, i_0, \mathcal{I}_F)$, let us extend the domain of ρ as follows: $\rho : \mathcal{I} \times \Sigma^* \to \mathcal{I}$

$$\begin{aligned} &(\forall i \in \mathcal{I})\ \rho(i, \varepsilon) = i, \\ &(\forall i \in \mathcal{I})\ (\forall \tau \in \Sigma^*)\ (\forall \sigma \in \Sigma)\ \rho(i, s\sigma) = \rho(\rho(i, s), \sigma), \end{aligned}$$

where $s\sigma$ is the concatenation of s and σ. We define the language accepted by $\mathbb{A}$ as follows

$$L(\mathbb{A}) := \{s \in \Sigma^* \mid (\forall i \in \mathcal{I})\ \rho(i, s) \in \mathcal{I}_{\mathrm{F}}\}.$$

2.3 Measures and Sobolev spaces

This section provides the fundamental notions needed for the definition of Sobolev spaces over a certain domain. Functions belonging to such spaces will be used in Chapter 7. This section is based on [1] to which we refer the reader for further details and insights.

Definition 2.18. Given a vector space X, a function $\|\cdot\| : X \to \mathbb{R}_+$, $x \mapsto \|x\|$, that satisfies the following conditions

- $(\forall x \in X)\ \|x\| \geq 0$ and $\|x\| = 0 \Leftrightarrow x = 0$,
- $(\forall x \in X)\ (\forall c \in \mathbb{R})\ \|cx\| = |c|\|x\|$,
- $(\forall x, y \in X)\ \|x+y\| \leq \|x\| + \|y\|$,

is called a norm on X. A vector space X equipped with a norm is called a normed space.

Given a vector space X of finite dimension $n \in \mathbb{N}$, function $\|\cdot\|_\infty : X \to \mathbb{R}_+$ defined as follows

$$(\forall x \in X)\ \|x\|_\infty := \max_{i \in \{1,\dots,n\}} \left|[x]_j\right|,$$

is called the infinity norm on X.

Definition 2.19. Given a normed space X, a subset $S \subseteq X$ such that for every sequence $\{s_i\}_{i\in I} \subseteq S$, with $I \subseteq \mathbb{N}$ set of indices, $\{s_i\}_{i\in I}$ admits a subsequence $\{s_{i_k}\}_{k\in K}$, with $K \subseteq I$, converging in X to an element of S, is said to be sequentially compact (or compact).

Definition 2.20. Given a collection $\Sigma := \{S \mid S \subseteq \mathbb{R}^n\}$ of subsets of $\mathbb{R}^n$, if Σ satisfies the following conditions

- $\mathbb{R}^n \in \Sigma$,
- $S \in \Sigma \Rightarrow S^c \in \Sigma$,
- given I a countable set of indices, $\{S_i\}_{i\in I} \subseteq \Sigma \Rightarrow \bigcup_{i\in I} S_i \in \Sigma$,

is called a σ-algebra.

Definition 2.21. Given a σ-algebra Σ, a function $\mu : \Sigma \to \mathbb{R}_+ \cup \{+\infty\}$ which is countably additive, that is, which satisfies

$$\mu\left(\bigcup_{i\in I} S_i\right) = \sum_{i\in I} \mu(S_i),$$

for any $\{S_i\}_{i\in I} \subseteq \Sigma$, with I countable set of indices, in which $S_i \cap S_j = \varnothing$ if $i \neq j$, is called a measure on Σ.

Theorem 2.22. *There exists a σ-algebra Σ and a measure μ on Σ satisfying the following properties*

- *every open subset of $\mathbb{R}^n$ belongs to Σ,*
- *for any $A \subseteq \mathbb{R}^n$ for which there exists $B \in \Sigma$ such that $A \subseteq B$ and $\mu(B) = 0$, then $A \in \Sigma$ and $\mu(A) = 0$,*
- *if $A := \{x \in \mathbb{R}^n \mid a_i \leq [x]_i \leq b_i,$ with $a_i, b_i \in \mathbb{R},\ a_i \leq b_i,$ for $i \in \{1, \dots, n\}\}$, then $A \in \Sigma$ and $\mu(A) = \prod_{i \in \{1,\dots,n\}} (b_i - a_i)$*
- *μ is translation invariant, that is, for any $x \in \mathbb{R}^n$ and $A \in \Sigma$, set $x + A := \{x + y \mid y \in A\} \in \Sigma$ and $\mu(x + A) = \mu(A)$.*

The elements of such a σ-algebra are called (Lebesgue) measurable subsets of $\mathbb{R}^n$ and μ is called the (Lebesgue) measure in $\mathbb{R}^n$.

Definition 2.23. Given $A, B \subseteq \mathbb{R}^n$, if $B \subseteq A$ and $\mu(B) = 0$, then any condition that holds on $A \setminus B$ is said to hold almost everywhere (a.e.) in A. Note that any countable subset of $\mathbb{R}^n$ has measure zero.

Definition 2.24. Given a measurable set A, a function $f : A \to \mathbb{R} \cup \{+\infty, -\infty\}$ that satisfies the following condition

$$(\forall a \in \mathbb{R})\ \{x \in A \mid f(x) > a\} \text{ is measurable,}$$

is said to be measurable.

Given a domain $\Omega \subseteq \mathbb{R}^n$, we refer the reader to [1] for the formal definitions of the space of test functions $\mathcal{D}(\Omega)$, the space of Schwartz distributions $\mathcal{D}'(\Omega)$ (which is the dual space of $\mathcal{D}(\Omega)$) and the space of locally integrable functions $L^1_{\text{loc}}(\Omega)$ on Ω and remind that every $u \in L^1_{\text{loc}}(\Omega)$ corresponds to a distribution $T_u \in \mathcal{D}'(\Omega)$.

Definition 2.25. Given a domain $\Omega \subseteq \mathbb{R}^n$ and a distribution $T \in \mathcal{D}'(\Omega)$, let us define its derivative $D^\alpha T$ as follows

$$(\forall \phi \in \mathcal{D}(\Omega))\ (D^\alpha T)(\phi) := (-1)^{|\alpha|} T(D^\alpha(\phi)).$$

Definition 2.26. Given a domain $\Omega \subseteq \mathbb{R}^n$ and a function $u \in L^1_{\text{loc}}(\Omega)$, if there exists a function $v_\alpha \in L^1_{\text{loc}}(\Omega)$ such that

$$T_{v_\alpha} = D^\alpha T_u,$$

in $\mathcal{D}'(\Omega)$, is called the weak partial derivative of u and is denoted by $D^\alpha u$.

Note that the previous definition is well posed since if such a function $D^\alpha u$ exists, then it can be proved that this function is unique up to sets of measure zero.

Definition 2.27. Given a domain $\Omega \subseteq \mathbb{R}^n$, a measurable function u on Ω is said to be essentially bounded on Ω if there exists a constant C such that $|u(x)| \leq C$ for almost every $x \in \Omega$. The greatest lower bound of such set of constants is called the essential supremum of $|u|$ on Ω and is denoted by $\operatorname{ess\,sup}_{x\in\Omega} |u(x)|$. The vector space of all functions u which are essentially bounded on Ω, with functions being identified if they are equal a.e. on Ω, is denoted by $L^\infty(\Omega)$.

Function $\|\cdot\|_\infty : \Omega \to \mathbb{R}_+ \cup \{+\infty\}$, defined as follows

$$\|u\|_\infty := \operatorname*{ess\,sup}_{x\in\Omega} |u(x)|,$$

is a norm on $L^\infty(\Omega)$ called the infinity norm on $L^\infty(\Omega)$.

Definition 2.28. Given a domain $\Omega \subseteq \mathbb{R}^n$ and $m \in \mathbb{N}$, consider function $\|\cdot\|_{m,\infty}$: $L^\infty(\Omega) \to \mathbb{R}_+ \cup \{+\infty\}$, defined as follows

$$\|u\|_{m,\infty} := \max_{0\leq|\alpha|\leq m} \|D^\alpha u\|_\infty.$$

Let us consider the following vector space on which $\|\cdot\|_{m,\infty}$ is a norm,

$$W^{m,\infty} := \{u \in L^\infty(\Omega) \mid D^\alpha u \in L^\infty(\Omega) \text{ for } 0 \leq |\alpha| \leq m\}.$$

$W^{m,\infty}(\Omega)$ is called a Sobolev space over Ω.

2.4 Differential equations with discontinuous right-hand side

In this section we give a glimpse of a possible definition of solution of a differential equation with discontinuous right-hand side along with a sufficient condition for the uniqueness of such a solution first introduced in [18]. The contents of this section are taken from [19] to which we refer the reader for further details.

Given an a.e. continuous function $f : \Omega \to \mathbb{R}^n$, with $\Omega \subseteq (\mathbb{R} \times \mathbb{R}^n)$ and the set $M \subseteq \Omega$ of measure zero of points of discontinuity of f, consider the equation

$$\frac{dx}{dt} = f(t,x). \tag{2.3}$$

Definition 2.29. For each point $(t,x) \in \Omega$, let us define $F : \Omega \to \wp(\mathbb{R}^n)$ at point (t,x) as the smallest convex closed set containing all the limit values of function $f(t,x^*)$, with $f(t,x^*) \notin M$, $x^* \to x$ and t constant.

Observe that, given any point $(t,x) \in \Omega$ at which function f is continuous, set $F(t,x)$ corresponding to this point reduces to a singleton given by $\{f(t,x)\}$ and the solution $x(t)$ satisfies equation (2.3) in the classical sense. Whereas over points $(t,x) \in M$, set $F(t,x)$ constitutes a polytope.

Definition 2.30. A solution of equation (2.3) is a solution of the differential inclusion

$$\frac{dx}{dt} \in F(t,x),$$

that is, a function $x : I \to \mathbb{R}^n$, with $I \subseteq \mathbb{R}$ interval, such that $\frac{dx}{dt} \in F(t,x(t))$ a.e. on I.

Definition 2.31. Given $(t_0,x_0) \in \Omega$, if there exists $t_1 > t_0$ such that each two solutions of equation (2.3) satisfying the condition $x(t_0) = x_0$, coincide on the interval $[t_0,t_1]$ or on the part of this interval where they are both defined, we say that it holds right uniqueness at point $(t_0,x_0) \in \Omega$ for equation (2.3). If right uniqueness holds for every $(t,x) \in \Omega$, then we say that right uniqueness holds in domain Ω for equation (2.3).

Theorem 2.32. *Given a function $f : \Omega \to \mathbb{R}^n$ over a domain $\Omega \subseteq (\mathbb{R} \times \mathbb{R}^n)$, discontinuous only on a set M of measure zero, and a function $l \in L^1(\mathbb{R})$ such that $(\forall (t,x) \in \Omega)\ f(t,x) \leq l(t)$ a.e., and such that $(\forall (t,x),(t,y) \in \Omega)\ (\exists \varepsilon > 0)$*

$$|x-y| < \varepsilon \Rightarrow (x-y)\cdot(f(t,x) - f(t,y)) \leq l(t)|x-y|^2 \ \textit{a.e.},$$

then under Definitions 2.29 and 2.30, equation (2.3) *has right uniqueness in domain Ω.*

Chapter 3

Applications

As mentioned earlier, there are problems of interest in control engineering which fall into class (1.3). In this chapter, we will present in details some significant applications, and show how their formulation can be reduced to form (1.3).

It would be interesting to find some real world applications that can be represented as non linear problems of class (1.1). However, all the applications that we encountered so far, mentioned in what follows, fall into the linear subcase (1.3).

3.1 Speed planning for autonomous vehicles

The first problem we introduce is a motion planning one. Namely, we compute the minimum-time trajectory of a car-like vehicle from an initial configuration to a target one. The planning of the trajectory must take into account the presence of obstacles, so that collision avoidance is ensured, and satisfy kinematic, dynamic and mechanical constraints, such as velocity, acceleration and maximal steering angle constraints.

There are two main approaches for addressing this problem:

1. It can be treated as a minimum-time trajectory planning in which both the path and the vehicle's velocity, that is, the timing law along this path, are designed all at once;

2. it can be split into two distinct problems: one which consists in finding a purely geometric path connecting the initial and the target configurations; and a second one which is a minimum-time velocity planning on the previously computed path (see for instance [28]).

In this section, we follow the second approach. Therefore, we assume that we are given a path joining the initial and target configurations which is consistent with

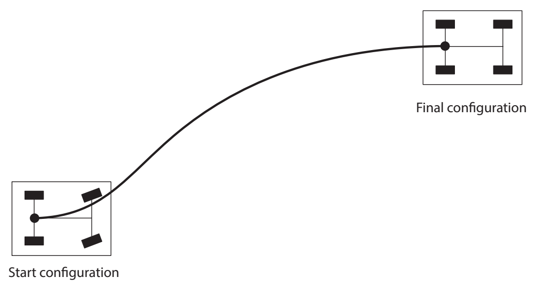


Figure 3.1: A path to be followed by an autonomous car-like vehicle.

the maximum curvature constraint for the car-like vehicle; and our goal is to find the time-optimal speed law that satisfies the kinematic and dynamic constraints of the problem. In Chapter 4 we will also address the first problem of computing a geometric path required in this approach.

In literature there are works addressing the problem in time domain. As an example, [46] obtains a semi-analytical solution by exploring necessary optimality conditions. In [21] the authors assume that the vehicle is given an assigned clothoid to move along and suggest a semi-analytical solution for the optimal profile of the longitudinal acceleration. Some other works like [36, 44, 48, 11]introduce special speed profiles for ensuring that kinematic and dynamic constraints are satisfied. In other approaches (presented, for instance, in [8, 47, 43, 33, 37]) the speed law is represented as a function v of the arc-length position s and not as a function of time.

What follows is based on [16] and we refer the reader to this reference for further detail. We consider a speed planning problem for a mobile vehicle (see Figure 3.1). We assume that the path that joins the initial and the final configuration is assigned and we aim at finding the time-optimal speed law that satisfies some kinematic and dynamic constraints. Namely, we consider the following problem

$$\min_{v \in C^1([0,s_f],\mathbb{R})} \int_0^{s_f} v^{-1}(s)ds \tag{3.1a}$$

$$\text{subject to } v(0) = 0, v(s_f) = 0 \tag{3.1b}$$

$$0 < v(s) \leq \bar{v}, \qquad s \in (0, s_f), \tag{3.1c}$$

$$|2v'(s)v(s)| \leq A_T, \qquad s \in [0, s_f], \tag{3.1d}$$

$$|k(s)|v(s)^2 \leq A_N, \qquad s \in [0, s_f], \tag{3.1e}$$

where $\bar{v}$, A_T, A_N are upper bounds for the velocity, the tangential acceleration and the normal acceleration, respectively. Here, s_f is the length of the path (that is assumed to be parameterized according to its arc length) and k is its scalar curvature (i.e., a function whose absolute value is the inverse of the radius of the circle that locally approximates the trajectory).

The objective function (3.1a) is the total maneuver time, constraints (3.1b) are the initial and final interpolation conditions and constraints (3.1c), (3.1d), (3.1e) limit velocity and tangential and normal components of acceleration.

After the change of variable $w = v^2$, the problem can be rewritten as

$$\begin{aligned} \min_{w\in C^1([0,s_f],\mathbb{R})} \ & \int_0^{s_f} w(s)^{-1/2} ds && && (3.2a)\\ \text{subject to} \ & w(0) = 0, w(s_f) = 0, && \\ & 0 < w(s) \le \bar{v}^2, && s \in (0,s_f), \\ & |w'(s)| \le A_T, && s \in [0,s_f], \\ & |k(s)|w(s) \le A_N, && s \in [0,s_f]. \end{aligned}$$

For $i \in \{1 \dots, n\}$, set $w_i = w((i-1)h)$, with $h = \frac{s_f}{n-1}$, then Problem (3.2) can be approximated with

$$\begin{aligned} \min_{w\in\mathbb{R}^n} \ & \phi(w) && \\ \text{subject to} \ & w_1 = 0, w_n = 0, && \\ & 0 < w_i \le \bar{v}^2, && i \in \{2,\dots,n-1\}, \\ & |w_{i+1} - w_i| \le hA_T, && i \in \{1,\dots,n-1\}, && (3.3a)\\ & |k(h(i-1))|w_i \le A_N, && i \in \{2,\dots,n-1\}, \end{aligned}$$

where the total time to travel the complete path is approximated by

$$\phi(w) = \sum_{i=1}^{n-1} t_i = 2h \sum_{i=1}^{n-1} \frac{1}{\sqrt{w_i} + \sqrt{w_{i+1}}}. \qquad (3.4)$$

Note that conditions (3.3a) is obtained by Euler approximation of $w'(hi)$. Similarly, the objective function (3.4) is a discrete approximation of the integral appearing in (3.2a). By setting $f(w) = \phi(w)$, $a = 0$, $g_1(w) = 0$, $g_n(w) = 0$ and, for $i \in \{2,\dots,n-1\}$,

$$g_i(w) = \bigwedge \left\{ \bar{v}^2, \ \frac{A_N}{|k(h(i-1))|}, \ hA_T + w_{i-1}, \ hA_T + w_{i+1} \right\},$$

Problem (3.3) takes on the form of Problem (1.1) and, since g is linear with respect to w, it also belongs to the more specific class (1.3). We remark that, with respect to the problem class (1.3), we minimize a decreasing function which is equivalent to maximizing an increasing function.

References [15, 16] present an algorithm, with linear-time computational complexity with respect to the number of variables n, that provides an optimal solution of Problem (3.3). This algorithm is a specialization of the algorithms proposed in Chapter 5 which exploits some specific features of Problem (3.3). In particular, the key property of Problem (3.3), which strongly simplifies its solution, is that functions g_i fulfill the so-called *superiority condition*

$$g_i(w_{i-1}, w_{i+1}) \geq w_{i-1} \text{ and } g_i(w_{i-1}, w_{i+1}) \geq w_{i+1},$$

(i.e., the value of function g_i is not lower than each one of its arguments).

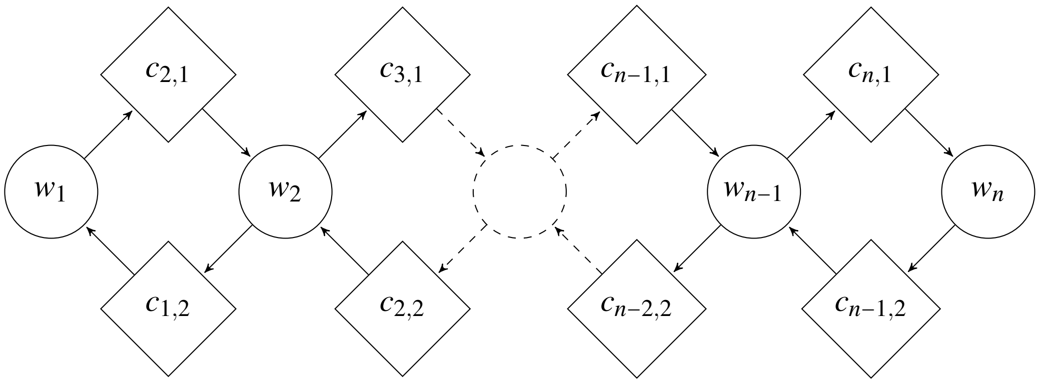


Figure 3.2: Graph associated to Problem (3.3)

The special structure of Problem (3.3) is made even more explicit by the graph associated to it. Recalling Section 1.3, the graph nodes sets are the following: $\mathcal{V} = \{w_1, \dots, w_n\}$, $\mathcal{C} = \{c_{1,2},\ c_{n,1},\ c_{i,j} \mid i \in \{2, \dots, n-1\},\ j \in \{1,2\}\}$ and the graph is given by Figure 3.2. A constraint node labeled with $c_{i,1}$, with $i \in \{2, \dots, n\}$ represents the dependence of g_i on w_{i-1}, whilst a constraint node labeled with $c_{i,2}$, with $i \in \{1, \dots, n-1\}$ represents the dependence of g_i on w_{i+1}. For ease of representation, here instead of considering exactly one constraint nodes for each constraint, we decided to split them into more constraint nodes, one for every dependence of a given constraint on a certain variable. As it can be seen in Figure 3.2, we can line up all the variable and notice how the variable nodes form a chain in which for each pair of successive variable nodes there are exactly two constraint nodes connecting the two variable nodes forward and backward. The algorithm presented in [15, 16] corresponds to a specific traversal of the graph shown in Figure 3.2: starting from node w_1, the algorithm visits all the successive variable nodes going "forward" through the constraint nodes $c_{i,1}$, with $i \in \{2, \dots, n\}$, and then, once it reaches the variable node w_n, it goes backwards until it reaches the variable node w_1 through the constraint nodes $c_{i,2}$, with $i \in \{1, \dots, n-1\}$.

The multigraph associated to Problem (3.3), which, in this case, is actually a graph, is the one give in Figure 3.3.

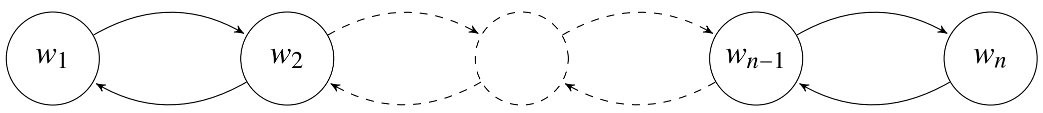


Figure 3.3: Multigraph associated to Problem (3.3)

In Chapter 7 we will address this problem in the continuous time domain and show how this algorithm generalizes. The main idea is that the forward and backward traversal the graph associated to the discretized problem turns into solving forward and backward a pair of differential equations.

3.2 Speed planning for robotic manipulators

Another important field of application of motion planning in robotics deals with robotics manipulators. In order to improve performances and productivity one wishes to compute the minimum-time motion of a robotic manipulator from a starting configuration to a desired final one while satisfying some constraints like those mentioned in Section 3.1 and saturation of the actuators. The two possible approaches seen in Section 3.1 also apply in this case and, again, the second approach, in which we assume the path planning phase has been completed, is followed, and the aim is finding the optimal speed-law that allows the manipulator to track the given path in minimum-time satisfying the aforementioned constraints. The technical details of this second applications are more involved and we refer the reader to [14] for the complete discussion.

Let $\mathbb{R}^p$ be the configuration space of a robotic manipulator with p-degrees of freedom. The coordinate vector q of a trajectory in U satisfies the dynamic equation

$$D(\mathrm{q})\ddot{\mathrm{q}}+C(\mathrm{q},\dot{\mathrm{q}})\dot{\mathrm{q}}+\ell(\mathrm{q})=\tau, \tag{3.5}$$

where $\mathrm{q}\in\mathbb{R}^p$ is the generalized position vector, $\tau\in\mathbb{R}^p$ is the generalized force vector, $D(\mathrm{q})$ is the mass matrix, $C(\mathrm{q},\dot{\mathrm{q}})$ is the matrix accounting for centrifugal and Coriolis effects (assumed to be linear in $\dot{\mathrm{q}}$) and $\ell(\mathrm{q})$ is the vector accounting for joints position dependent forces, including gravity. Note that we do not consider Coulomb friction forces.

Let $\gamma\in C^2([0,s_f],\mathbb{R}^p)$ be a function such that $(\forall\lambda\in[0,s_f])\ \|\gamma'(\lambda)\|=1$. The image set $\gamma([0,s_f])$ represents the coordinates of the elements of a reference path. In particular, $\gamma(0)$ and $\gamma(s_f)$ are the coordinates of the initial and final configurations. Define t_f as the time when the robot reaches the end of the path. Let $\lambda:[0,t_f]\to[0,s_f]$ be a differentiable monotone increasing function that represents the position of the robot as a function of time and let $v:[0,s_f]\to[0,+\infty]$ be such that $\big(\forall t\in[0,t_f]\big)\ \dot{\lambda}(t)=v(\lambda(t))$. Namely, $v(s)$ is the velocity of the robot at position s. We impose $(\forall s \in [0, s_f])\ v(s) \geq 0$. For any $t \in [0, t_f]$, using the chain rule, we obtain

$$\begin{aligned}
&\mathbf{q}(t) = \gamma(\lambda(t)), \\
&\dot{\mathbf{q}}(t) = \gamma'(\lambda(t))v(\lambda(t)), \\
&\ddot{\mathbf{q}}(t) = \gamma'(\lambda(t))v'(\lambda(t))v(\lambda(t)) + \gamma''(\lambda(t))v(\lambda(t))^2.
\end{aligned} \tag{3.6}$$

Substituting (3.6) into the dynamic equations (3.5) and setting $s = \lambda(t)$, we rewrite the dynamic equation (3.5) as follows:

$$\mathbf{d}(s)v'(s)v(s) + \mathbf{c}(s)v(s)^2 + \mathbf{g}(s) = \tau(s), \tag{3.7}$$

where the parameters in (3.7) are defined as

$$\begin{aligned}
&\mathbf{d}(s) = D(\gamma(s))\gamma'(s), \\
&\mathbf{c}(s) = D(\gamma(s))\gamma''(s) + C(\gamma(s), \gamma'(s))\gamma'(s), \\
&\mathbf{g}(s) = \ell(\gamma(s)).
\end{aligned}$$

The objective function is given by the overall travel time t_f defined as

$$t_f = \int_0^{t_f} 1\, dt = \int_0^{s_f} v(s)^{-1} ds.$$

Let $\mu, \psi, \alpha : [0, s_f] \to \mathbb{R}_+^p$ be assigned bounded functions and consider the following minimum time problem:

$$\begin{aligned}
&\min_{v \in C^1, \tau \in C^0} \int_0^{s_f} v(s)^{-1} ds, \\
&\text{subject to} \quad (\forall s \in [0, s_f]) \\
&\quad \mathbf{d}(s)v'(s)v(s) + \mathbf{c}(s)v(s)^2 + \mathbf{g}(s) = \tau(s), && \text{(3.8a)} \\
&\quad \gamma'(s)v(s) = \dot{\mathbf{q}}(s), && \text{(3.8b)} \\
&\quad \gamma'(s)v'(s)v(s) + \gamma''(s)v(s)^2 = \ddot{\mathbf{q}}(s), && \text{(3.8c)} \\
&\quad |\tau(s)| \leq \mu(s), && \text{(3.8d)} \\
&\quad |\dot{\mathbf{q}}(s)| \leq \psi(s), && \text{(3.8e)} \\
&\quad |\ddot{\mathbf{q}}(s)| \leq \alpha(s), && \text{(3.8f)} \\
&\quad v(s) \geq 0, \\
&\quad v(0) = 0,\ v(s_f) = 0, && \text{(3.8g)}
\end{aligned}$$

where (3.8a) represents the robot dynamics, (3.8b)–(3.8c) represent the relation between the path γ and the generalized position $\mathbf{q}$ shown in (3.6), (3.8d) represents the bounds on generalized forces, (3.8e) and (3.8f) represent the bounds on joints velocity and acceleration, respectively. Constraints (3.8g) specify the interpolation conditions at the beginning and at the end of the path.

After some manipulation and using a carefully chosen finite dimensional approximation (again, see [14] for the details), Problem (3.8) can be reduced to form (see Proposition 8 of [14]).

$$\begin{aligned}
&\min_w \phi(w) \\
\text{subject to } \; & w_i \leq f_{j,i} w_{i+1} + c_{j,i} \quad i \in \{1,\dots,n-1\}, \quad j \in \{1,\dots,p\}, \\
& w_{i+1} \leq b_{k,i} w_i + d_{k,i} \quad i \in \{1,\dots,n-1\}, \quad k \in \{1,\dots,p\}, \\
& 0 \leq w_i \leq u_i \qquad\qquad\qquad\qquad\qquad i \in \{1,\dots,n\},
\end{aligned} \tag{3.9}$$

where, ϕ is defined as in (3.4) and $w = (w_1,\dots,w_n)^T$. For $i \in \{1,\dots,n\}$, $w_i = v((i-1)h)^2$, $h = \frac{s_f}{n-1}$, is the squared manipulator speed at configuration $\gamma((i-1)h)$. Moreover u_i, $f_{j,i}$, $c_{j,i}$, $b_{k,i}$, $d_{k,i}$ are nonnegative constant terms depending on problem data. By setting for $k \in \{1,\dots,p\}$,

$$g_1^k(w) = \bigwedge \left\{\bar{u}_1,\ f_{k,1} w_2 + c_{k,1}\right\},$$
$$g_n^k(w) = \bigwedge \left\{\bar{u}_n,\ b_{k,n-1} w_{n-1} + d_{k,n-1}\right\},$$

and, for $i \in \{2,\dots,n-1\}$,

$$g_i^k(w) = \bigwedge \left\{\bar{u}_i, f_{k,i} w_{i+1} + c_{k,i}, b_{k,i-1} w_{i-1} + d_{k,i-1}\right\},$$

Problem (3.9) belongs to classes (1.1) and (1.3). Also in this case, the performance of the algorithms proposed in Chapters 5 and 6 can be enhanced by exploiting some further specific features of Problem (3.9). In particular, in [14], the authors were able to develop a version of the algorithm with optimal time-complexity $O(np)$.

Again, the graph associated to Problem (3.9) clarifies the specific structure of the problem. Recalling Section 1.3, the graph nodes sets are the following: $\mathcal{V} = \{w_1,\dots,w_n\}$, $\mathcal{C} = \{c_{1,2}^k,\ c_{n,1}^k,\ c_{i,j}^k \mid i \in \{2,\dots,n-1\},\ j \in \{1,2\},\ k \in \{1,\dots,p\}\}$ and the graph can be seen in Figure 3.4. For $k \in \{1,\dots,p\}$, a constraint node labeled with $c_{i,1}^k$, with $i \in \{2,\dots,n\}$ represents the dependence of g_i^k on w_{i-1}, whilst a constraint node labeled with $c_{i,2}^k$, with $i \in \{1,\dots,n-1\}$ represents the dependence of g_i^k on w_{i+1}. As we can see in Figure 3.4, the structure of the graph is a generalization of the one seen in Figure 3.2; here, for every pair of variable nodes (w_i, w_{i+1}), with $i \in \{1,\dots,n-1\}$, there are exactly p constraint nodes connecting w_i to w_{i+1} and other p connecting w_{i+1} to w_i. So Problem 3.3 can be seen as a special case of Problem 3.9 with $p = 1$.

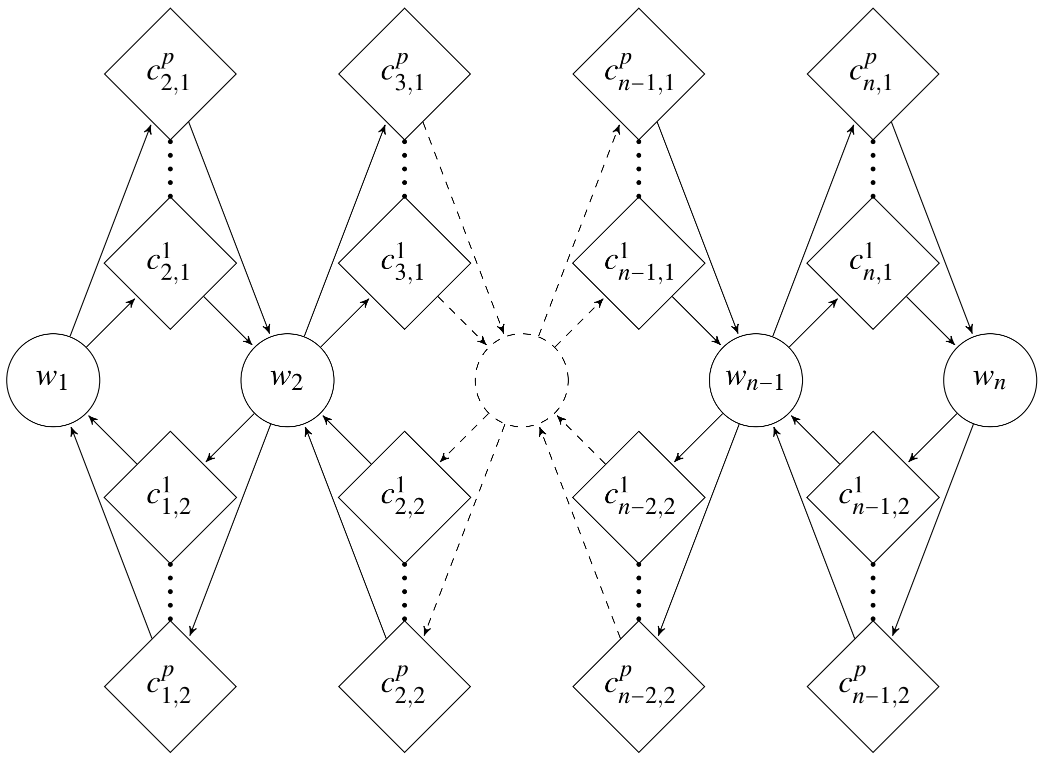


Figure 3.4: Graph associated to Problem (3.9)

The algorithm presented in [14] operates similarly to the one proposed in [15, 16] but, since in this case the superiority condition is not ensured, the forward step is more involved. Here, the forward step operates as follows: starting from pair (w_1, w_2), the algorithm isolates the subgraph involving pair (w_i, w_{i+1}), and those constraint nodes which involve all and only variable nodes w_i and w_{i+1}; it solves the subproblem associated to this subgraph and, then, moves on to the successive pair (w_{i+1}, w_{i+2}) until it reaches pair (w_{n-1}, w_n). The backward step is more similar to the one seen in Section 3.1: starting from variable node w_n down to variable node w_1, the algorithm moves backward through the constraint nodes $c_{i,2}^k$, with $k \in \{1, \dots, p\}$ and $i \in \{1, \dots, n-1\}$.

The multigraph associated to Problem (3.9) is the one give in Figure 3.5 in which, for any pair of successive nodes, we have a multiedge formed by p edges connecting them forward and another one connecting them backwards.

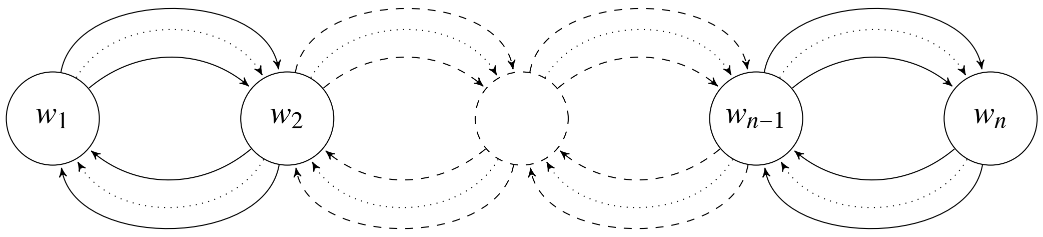


Figure 3.5: Multigraph associated to Problem (3.9)

3.3 Wireless communication systems

We can find problems that take form (1.3) also in the field of communication systems. In particular, in wireless communication systems there exists the so called uplink power control problem (see [51]), in which each user of a given communication system has to be provided enough power for establishing acceptable connections with other users of the system, while avoiding or reducing as much as possible interferences with other users not involved in the communication. The users' signal to interference ratio requirement can be represented with a set of constraints of the form

$$\mathbf{p} \geq \mathbf{I}(\mathbf{p}), \tag{3.10}$$

with $\mathbf{p} = [p_1, \ldots, p_n]'$ and $\mathbf{I}(\mathbf{p}) = [\mathbf{I}(p_1), \ldots, \mathbf{I}(p_n)]'$, where p_i denotes the power of the transmitter of user i and $\mathbf{I}(p_i)$ denotes the effective interference due to other users that user i has to surmount, for $i \in \{1, \ldots, n\}$. A power vector $\mathbf{p} \in \mathbb{R}^n_+$ is said to be feasible if it satisfies constraints (3.10), and an interference function $\mathbf{I} : \mathbb{R}^n_+ \to \mathbb{R}^n_+$ is said to be feasible if constraints (3.10) admit a feasible power vector. Moreover, if the interference function $\mathbf{I}$ satisfies the following conditions:

- positivity: $(\forall \mathbf{p} \in \mathbb{R}^n_+)\ \mathbf{I}(\mathbf{p}) > 0$,
- monotonicity: $(\forall \mathbf{p}, \mathbf{p}' \in \mathbb{R}^n_+)\ \mathbf{p} \geq \mathbf{p}' \Rightarrow \mathbf{I}(\mathbf{p}) \geq \mathbf{I}(\mathbf{p}')$,
- scalability: $(\mathbf{p} \in \mathbb{R}^n_+)\ (\alpha > 1)\ \alpha \mathbf{I}(\mathbf{p}) > \mathbf{I}(\alpha \mathbf{p})$,

then $\mathbf{I}$ is said to be standard.

Assuming that function $\mathbf{I}$ is standard and feasible, it is possible to define the following iterative power control algorithm

$$\mathbf{p}(t+1) = \mathbf{I}(\mathbf{p}(t)), \tag{3.11}$$

and show that starting from $\underline{0} \in \mathbb{R}^n_+$, iteration (3.11) generates a monotone increasing sequence converging to a fixed point of $\mathbf{I}$.

Now, given n users, m base station and a common radio channel, denote with h_{ji} the gain of user i to base j. At base j, quantity $h_{j_i}p_i$ corresponds to the received signal power of user i, whilst the interference seen by user i at base j is

$$\sum_{\substack{k=1,\\ k\neq i}}^{n} h_{jk}p_k + \sigma_j,$$

where σ_j is the receiver noise power at base station j. Thus, given power vector $\mathbf{p}$ the signal interference ratio of user i at base station j is $p_i\mu_{ji}(\mathbf{p})$, with

$$\mu_{ji}(\mathbf{p}) := \frac{h_{ji}}{\sum_{\substack{k=1,\\ k\neq i}}^{n} h_{jk}p_k + \sigma_j}$$

Let us now consider the interference function associated to the minimum power assignment. In this case, at each step of iteration (3.11), user i is assigned to the base station which minimizes its signal interference ratio. Some works analyzed the convergence of the iteration associated to the minimum power assumption both for continuous power adjustments (see for instance [50]) and for discrete power adjustments (see [24]). The constraint associated to the signal interference ratio γ_i of user i is $\max_{j\in\{1,\dots,m\}} p_i\mu_{ji}(\mathbf{p}) \geq \gamma_i$, and it can be expressed as follows

$$(\forall i \in \{1,\dots,n\})\ p_i \geq I_i(\mathbf{p}) := \min_{j\in\{1,\dots,m\}} \left\{\frac{\gamma_i}{\mu_{ji}(\mathbf{p})}\right\}.$$

Under the minimum power assignment, in iteration (3.11), user i is associated to base station j where minimum power is needed to achieve the desired signal interference ratio γ_i, assuming that the other users keep their power constant. That is: $(\forall i \in \{1,\dots,n\})$

$$\begin{aligned} p_i \geq \min_{j\in\{1,\dots,m\}} \left\{\frac{\gamma_i}{\mu_{ji}(\mathbf{p})}\right\} &\Leftrightarrow p_i \geq \min_{j\in\{1,\dots,m\}} \left\{\frac{\gamma_i}{h_{ji}}\left(\sum_{\substack{k=1,\\ k\neq i}}^{n} h_{jk}p_k + \sigma_j\right)\right\} \\ &\Leftrightarrow p_i \geq \min_{j\in\{1,\dots,m\}} \left\{\frac{\gamma_i}{h_{ji}}\sum_{\substack{k=1,\\ k\neq i}}^{n} h_{jk}p_k + \frac{\gamma_i}{h_{ji}}\sigma_j\right\}. \end{aligned}$$

If we write the previous constraints in matrix form we obtain

$$\mathbf{p} \geq \bigwedge_{j\in\{1,\dots,m\}} \left\{H_j\mathbf{p} + \Sigma_j\right\},$$

with $(\forall j \in \{1,\ldots,m\})$ $(\forall i,k \in \{1,\ldots,n\})$ $\left[H_j\right]_{ik} := \frac{\gamma_i}{h_{ji}} h_{jk}$ and $\left[\Sigma_j\right]_i := \frac{\gamma_i}{h_{ji}} \sigma_j$. So, the optimization problem we want to solve is the following one

$$\begin{aligned} &\min_{\mathbf{p}} \sum_{i=1}^{n} [\mathbf{p}]_i \\ &\text{subject to } \ \mathbf{p} \geq \bigwedge_{j\in\{1,\ldots,m\}} \left\{H_j \mathbf{p} + \Sigma_j\right\}, \ \mathbf{p} \geq 0, \end{aligned}$$

which can be rewritten in the following form

$$\begin{aligned} &\max_{\mathbf{p}} - \sum_{i=1}^{n} [\mathbf{p}]_i \\ &\text{subject to } \ 0 \leq \mathbf{p} \leq \bigwedge_{j\in\{1,\ldots,m\}} \left\{H_j \mathbf{p} + \Sigma_j\right\}, \end{aligned}$$

which falls in class (1.3).

Chapter 4

Dynamic programming

In this chapter we present another problem in control engineering which falls into class (1.3); given its importance we dedicate a whole chapter to it. Here, we introduce a continuous-time version of the dynamic programming principle, which takes the name of Hamilton-Jacobi-Bellman (HJB) equation, for the synthesis of optimal control for a switching system (see [10, 52]). After having introduced the problem, we will show how it is possible to discretize it in such a way that it falls into class (1.3). In the second part, we apply this framework to path planning of parking maneuvers for autonomous vehicles, presenting different possible models and conclude the chapter with some numerical experiments. Part of the problem derivation in Section 4.1 follows the one presented in [52].

4.1 Switching Hamilon-Jacobi-Bellman equation

Given a finite state automaton $\mathbb{A} = (\mathcal{I}, \Sigma, \rho, i_0, \mathcal{I}_{\mathrm{F}})$, let us consider a switching control system of continuous time subsystems defined by the following differential equations in $\mathbb{R}^n$: $(\forall i \in \mathcal{I})$

$$\dot{x}(t) = f(x(t), i, u(t)), \tag{4.1}$$

where $f : \mathbb{R}^n \times \mathcal{I} \times U \to \mathbb{R}^n$ is the function representing the dynamic, $x = x(t) \in \mathbb{R}^n$ is the continuous state, index $i \in \mathcal{I}$ represents the automaton state, $u = u(t) \in U$ is the continuous control input and $U \subset \mathbb{R}^m$ is a compact set of admissible controls. Note that, each state of automaton $\mathbb{A}$ is associated to a subsystem (4.1). We represent the switchings of the system by the following sequence

$$\sigma_t = \{(\tau_1, i_1), (\tau_2, i_2), \ldots\},$$

with $t \le \tau_1 \le \tau_2 \le \ldots$ representing the switching times sequence and $i_1, i_2, \ldots \in \mathcal{I}$ the switching subsystems sequence such that system (4.1) switches at time τ_{p+1}

from subsystem i_p to subsystem i_{p+1}, with $p \in \mathbb{N}$, provided that there exists a string $\sigma_1\sigma_2\cdots \in L(\mathbb{A})$ such that $\rho(i_j,\sigma_j) = i_{j+1}$, for $j \in \mathbb{N}$, and such that the last state of the sequence belongs to $\mathcal{I}_\mathrm{F}$. In other words, we are asking the subsystems sequence to correspond to an admissible sequence of states of finite state automaton $\mathbb{A}$ associated to a string of the language $L(\mathbb{A})$ it accepts; moreover, the last state of the sequence needs to correspond to a final state of $\mathbb{A}$. The control signal for the switching system (4.1) is given by

$$v_t = (u_t, \sigma_t),$$

with $u_t : (t,+\infty) \to U \in \mathcal{U}_t := \{f : (t,+\infty) \to U \mid f \text{ is measurable}\}$ measurable continuous control input. Let us denote with σ_t^i a sequence such that $i_1 \neq i$ and with $\mathfrak{V}_t^i = \{v_t^i \mid v_t^i = (u_t,\sigma_t^i)\}$ the switching control signals sequence. Now, let us consider an infinite horizon cost functional with discount factor defined as follows

$$J(x,i,v) = \int_0^\infty \mathfrak{l}(x(t),i(t),u(t))e^{-\lambda t}dt + \sum_{k=1}^{|\sigma_0^i|} \kappa(i_{k-1},i_k)e^{-\lambda\tau_k},$$

where $v \in \mathfrak{V}_0^i$ is the switching control signal, $\mathfrak{l} : \mathbb{R}^n \times \mathcal{I} \times U \to \mathbb{R}_+$ is a continuous function representing the running cost, uniformly bounded and Lipschitz-continuous on $\mathbb{R}^n$, $\lambda > 0$ represents the discount factor and partial function $\kappa : \mathcal{I} \times \mathcal{I} \to \mathbb{R}_+$ represents the switching cost satisfying the following properties: for all $i,j,k \in \mathcal{I}$ for which κ is defined, it holds that

$$\kappa(i,j) > 0, \quad \kappa(i,i) = 0 \quad \text{and} \quad \kappa(i,j) \leq \kappa(i,k) + \kappa(k,j).$$

In other words, transitioning from a state to another always has a positive cost, maintaining the same state incurs in no transitioning cost and transitioning to a certain state through an intermediate one is always more expensive then transitioning directly to that state. Let us define the value function $V : \mathbb{R}^n \times \mathcal{I} \to \mathbb{R}_+$ as follows

$$V(x,i) := \inf_{v \in \mathfrak{V}_0^i} J(x,i,v).$$

As shown in [10], the value function V is the unique viscosity solution of the switching HJB equation:

$$\max\left\{\begin{array}{c} V(x,i) - \min\limits_{\sigma\in\Sigma}\{\kappa(i,\rho(i,\sigma)) + V(x,\rho(i,\sigma))\}, \\ \lambda V(x,i) + \sup\limits_{u \in U}\{-\nabla_x V(x,i)\cdot f(x,i,u) - \mathfrak{l}(x,i,u)\} \end{array}\right\} = 0, \tag{4.2}$$

where $\nabla_x V$ represents the gradient of V with respect to x.

Since (4.2) involves a non-linear partial differential equation, in general there is no closed form solution for such equation. Some works such as [2, 5, 35, 49] develop various numerical procedures for computing approximate solutions.

In particular, [5] presents an approximation scheme based on a finite approximation of state and control spaces and a discretization in time. Roughly speaking, in (4.2) one can approximate $\nabla v(x) f(x,u) \simeq h^{-1}(v(x+hf(x,u)) - v(x))$, where h is a small positive real number that represents an integration time. In this way, (4.2) becomes

$$\max\left\{\begin{array}{c} V(x,i) - \min\limits_{\sigma\in\Sigma}\{\kappa(i,\rho(i,\sigma)) + V(x,\rho(i,\sigma))\}, \\ \lambda V(x,i) + \sup\limits_{u\in U}\{-\frac{1}{h}[V(x+hf(x,i,u),i) - V(x,i)] - \mathfrak{l}(x,i,u)\} \end{array}\right\} = 0,$$

and, by approximating $(1+\lambda h)^{-1} \simeq (1-\lambda h)$, $(1+\lambda h)^{-1}h \simeq h$, one obtains the following switching HJB equation in discrete time

$$V_h(x,i) = \min\left\{\begin{array}{c} \min\limits_{\sigma\in\Sigma}\{\kappa(i,\rho(i,\sigma)) + V_h(x,\rho(i,\sigma))\}, \\ \min\limits_{u\in U}\{(1-\lambda h)V_h(x+hf(x,i,u),i) + h\mathfrak{l}(x,i,u)\} \end{array}\right\}. \qquad (4.3)$$

For a more rigorous derivation of (4.3), again, see [5].

Let us now discretize the state space considering a grid computed on a finite set of vertices $\mathfrak{S} = \{x_s\}_{s\in\mathcal{V}} \subset \mathbb{R}^n$, with $\mathcal{V} \subseteq \mathbb{N}$ and $|\mathcal{V}| = N$. As an example consider the triangulation shown in Figure 4.1.

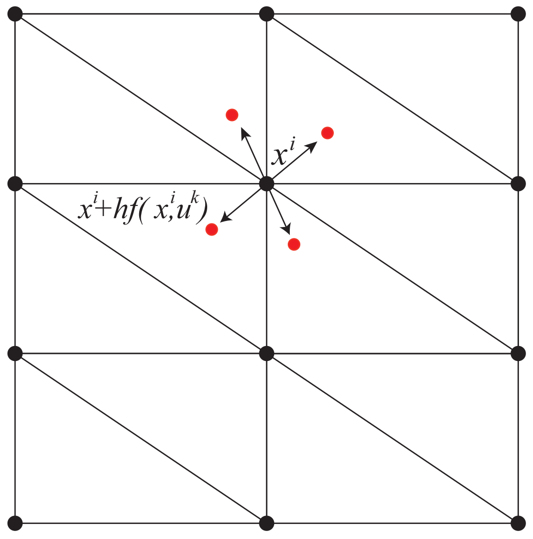


Figure 4.1: Approximation of the HJB equation on a triangulation with four controls.

Evaluating (4.3) on $\mathfrak{S}$, we get $(\forall s \in \mathcal{V})$

$$V_h(x_s, i) = \min \left\{ \begin{array}{c} \min_{\sigma\in\Sigma} \{\kappa(i, \rho(i,\sigma)) + V_h(x_s, \rho(i,\sigma))\}, \\ \min_{u\in U} \{(1-\lambda h) V_h(x + hf(x_s, i, u), i) + h\mathfrak{l}(x_s, i, u)\} \end{array} \right\}. \tag{4.4}$$

To further simplify (4.4), for each state $i \in \mathcal{I}$ of automaton $\mathbb{A}$ it is possible to discretize the control space, substituting U with a finite set of controls $\{u_\ell\}_{\ell\,\in\mathcal{L}(i)}$, so that we can rewrite (4.4) as follows

$$V_h(x_s, i) = \min \left\{ \begin{array}{c} \min_{\sigma\in\Sigma} \{\kappa(i, \rho(i,\sigma)) + V_h(x_s, \rho(i,\sigma))\}, \\ \min_{\ell\in\mathcal{L}(i)} \{(1-\lambda h) V_h(x + hf(x_s, i, u_\ell), i) + h\mathfrak{l}(x_s, i, u_\ell)\} \end{array} \right\}. \tag{4.5}$$

For each state $i \in \mathcal{I}$, set vector $z(i) := [z_1(i), \ldots, z_N(i)]^T = [V_h(x_1, i), \ldots, V_h(x_N, i)]^T$, in this way $w \in \mathbb{R}^N$ represents the value of the value cost function $V(\cdot, i)$ on the grid vertices. Note that, for each $s \in \mathcal{V}$, $\ell \in \mathcal{L}(i)$, the right-hand side of (4.5) is affine with respect to $z(i)$, so that Problem (4.5) can be rewritten as

$$z(i) = \bigwedge \left\{ \bigwedge_{\sigma\in\Sigma} \{Iz(\rho(i,\sigma)) + \kappa(i, \rho(i,\sigma)\}, \bigwedge_{\ell\in\mathcal{L}(i)} \{\mathfrak{A}_\ell z(i) + \mathfrak{b}_\ell\} \right\}, \tag{4.6}$$

where $I \in \mathbb{R}^{N\times N}$ denotes the identity matrix and for each $\ell \in \mathcal{L}(i)$, $\mathfrak{A}_\ell \in \mathbb{R}_+^{N\times N}$ are suitable non-negative matrices and $\mathfrak{b}_\ell \in \mathbb{R}_+^N$ are suitable non-negative vectors.

Let us now define vector $z := [z(1)^T, \ldots, z(|\mathcal{I}|)^T]^T$ and represent problems (4.6) for $i \in \mathcal{I}$, all at once as follows

$$z = \bigwedge_{\ell\in\mathcal{L}} \{A_\ell z + b_\ell\}, \tag{4.7}$$

where $\mathcal{L} := \bigcup_{i\in\mathcal{I}} \{\mathcal{L}(i)\} \cup \Sigma$. Further, if $\ell \in \mathcal{L}(i)$ for some $i \in \mathcal{I}$, recalling that $\otimes$ represents the Kronecker product (1.7), then $A_\ell := \mathbb{1}_{ii} \otimes \mathfrak{A}_\ell$, with $\mathbb{1}_{ii} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ representing the all zero matrix with the ii-th element equal to 1, and

$$[b_\ell]_k := \begin{cases} \left[\mathfrak{b}_{\ell(i)}\right]_p, & \text{if } k = iN + p, \text{ with } p \in \{1, \ldots, N\} \\ \frac{1}{\lambda}, & \text{otherwise.} \end{cases} \tag{4.8}$$

Otherwise, if $\ell \in \Sigma$, then, for each $i \in \mathcal{I}$, $j := \rho(i,\sigma) \in \mathcal{I}$, $A_\ell := \mathbb{1}_{ij} \otimes I$, with $\mathbb{1}_{ij} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ representing the all zero matrix with the ij-th element equal to 1, and

$$[b_\ell]_k := \begin{cases} \kappa(i, \ell), & \text{if } k \in \{iN, \ldots, i(N+1) - 1\} \\ \frac{1}{\lambda}, & \text{otherwise.} \end{cases} \tag{4.9}$$

In this way, the problem of solving the discretized version (4.5) of (4.2) is equivalent to the following optimization problem

$$\begin{aligned} &\max_{z} \sum_{j=1}^{|\mathcal{I}|N} [z]_j \\ &\text{subject to } \; 0 \le z \le \bigwedge_{\ell \in \mathcal{L}} \{A_\ell z + b_\ell\}\,,\; z \le \frac{1}{\lambda}\,, \end{aligned} \tag{4.10}$$

which takes the form of Problem (1.3).

4.2 Path planning for autonomous vehicles

Consider the following kinematic car-like model with rear-wheel drive and steering front wheels (see Figure 4.2):

$$\begin{cases} \dot{w} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{4.11}$$

with (w, y) representing the position of the center of the real wheel axle, θ the orientation angle and pair (v, ω) constituting the control input in which v and ω are, respectively, the linear and angular velocity. The relation between the angular velocity ω and the front wheel steering angle δ in given by relation $\omega = \frac{1}{l} v \tan\delta$, where l is the distance between the front and rear axle of the car-like model. The input control variables are constrained as follows

$$v_{\min} < v < v_{\max} \qquad \delta_{\min} < \delta < \delta_{\max},$$

with $v_{\min} < 0$ and $v_{\max} > 0$; Figure 4.3 shows the resulting set of admissible controls for v and ω assuming $v_{\min} = -v_{\max}$ and $\delta_{\min} = -\delta_{\max}$.

Among the many works devoted to solving the path-planning problem (see, for instance, [34, 41, 39, 23]); here, for computing a time-optimal parking maneuver for system (4.11), we follow the approach given by the HJB equation. However, if we take under consideration the fact that each change of direction from forward to reverse or viceversa is a time consuming operation for a road vehicle, it is desirable to maintain such changes limited. In order to generate a time-optimal path that takes into account these constraints, one can exploit the switching HJB equation presented in (4.2). In this scenario, the state space is given by a bounded and connected domain $\Omega \subseteq \mathbb{R}^2 \times [0, 2\pi)$ possibly partitioned into two spaces: Ω_{free}, representing the free space, and Ω_{obstacle}, representing the space occupied by obstacles.

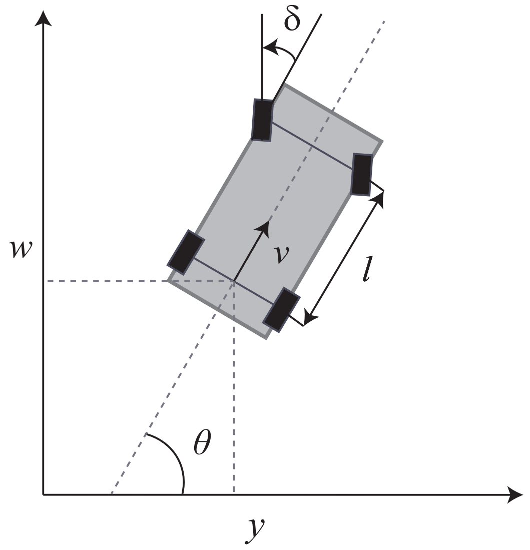


Figure 4.2: Car-like model.

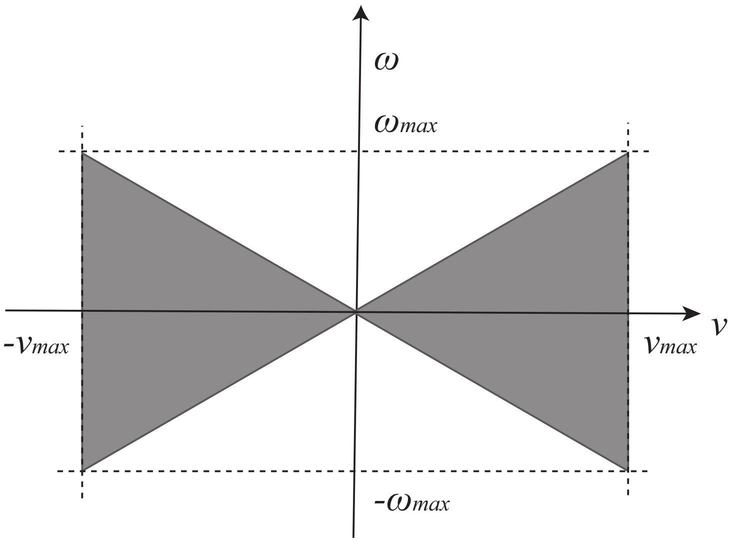


Figure 4.3: Set of admissible controls (v, ω).

These two spaces are such that $\Omega_{\text{free}} \cup \Omega_{\text{obstacle}} = \Omega$, and $\Omega_{\text{free}} \cap \Omega_{\text{obstacle}} = \varnothing$. Now, in order to use (4.2) we first need to define a finite state automaton and the switching control system associated to it. Here, we will present three possible finite state automata for addressing the path planning problem with limitations over direction changes.

4.2.1 Model 1

The first finite state automaton, depicted in Figure 4.4, is $\mathbb{A}_1 = (\mathcal{I}_1, \Sigma_1, \rho_1, \mathrm{F}, \mathcal{I}_1)$ in which $\mathcal{I}_1 = \{\mathrm{F}, \mathrm{R}\}$, with F and R representing, respectively, the control subsystem in which the car-like vehicle has a forward gear engaged and the one in which the backward gear is engaged; $\Sigma_1 = \{c\}$, with symbol c representing a gear switch, without loss of generality, initial state equal to F, and the set of final states coinciding with $\mathcal{I}_1$. The language accepted by $\mathbb{A}_1$ is given by $L(\mathbb{A}_1) = \Sigma_1^*$ with $\rho_1(\mathrm{F}, c) = \mathrm{R}$ and $\rho_1(\mathrm{R}, c) = \mathrm{F}$. Note that in this model, even though there is no given limit on the number of gear changes, we recall the presence of the switching cost function κ defined as $\kappa(\mathrm{F},\mathrm{R}) = \kappa(\mathrm{R},\mathrm{F}) = \kappa_0$, where $\kappa_0 > 0$ represents the time needed to the road vehicle for performing a direction change, which is always greater than zero. In this way, a longer trajectory with a lower number of direction changes can have a lower cost than a shorter one with a larger number of directions changes.

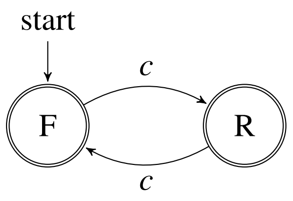

$$\begin{tikzcd} \arrow[d, "\text{start}"'] & \\ \mathrm{F} \arrow[r, bend left, "c"] & \mathrm{R} \arrow[l, bend left, "c"] \end{tikzcd}$$

Figure 4.4: Finite state automaton $\mathbb{A}_1$.

4.2.2 Model 2

A second model considers finite state automaton $\mathbb{A}_2 = (\mathcal{I}_2, \Sigma_2, \rho_2, P, \{0,1\})$, in which $\mathcal{I}_2 = \{1, \ldots, P\} \cup \{\mathrm{I}\}$, with $P \in \mathbb{N}$ representing the maximum number of gear switchings allowed, $i \in \{1, \ldots, P\}$ representing the residue number of gear switchings allowed and I representing an invalid state. Assuming the initial state is P, that is, the state in which the number of residue gear switchings is the largest possible, $\Sigma_2 = \{c\}$, with symbol c representing a gear switch, and the set of final states given by $\{0,1\}$. The language accepted by $\mathbb{A}_2$ is given by $L(\mathbb{A}_2) = \Sigma_2^P \cup \Sigma_2^{P-1}$, see (2.2), with $\rho_2(i,c) = i-1$, for $i \in \{1, \ldots, P\}$ and $\rho_2(0,c) = \mathrm{I}$. Figure 4.5 represents finite state automaton $\mathbb{A}_2$. As in the previous model, here we have only two control subsystems representing the car-like vehicle with a forward or reverse gear engaged; assuming without loss of generality that $f(x(t), P, u(t))$, see (4.1), is the dynamic associated to the forward gear engaged, $f(x(t), i, u(t))$, with $i \in \{0, \ldots, P\}$, will be the dynamic associated to the forward gear engaded if $i \mod 2 = P \mod 2$, otherwise it will be the one associated to the reverse gear engaged.

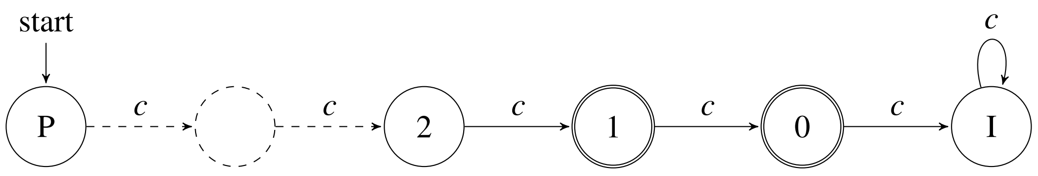


Figure 4.5: Finite state automaton $\mathbb{A}_2$.

4.2.3 Model 3

The third finite state automaton we present is the following one: $\mathbb{A}_3 = (\mathcal{I}_3, \Sigma_3, \rho_3, \text{B}, \mathcal{I}_3)$, in which $\mathcal{I}_3 = \{\text{Fl}, \text{Fc}, \text{Fr}, \text{Rl}, \text{Rc}, \text{Rr}\}$, with F and R representing, respectively, the vehicle with the forward and backward gear engaged, whilst l, c and r represent the front steering wheels of the vehicle, respectively, completely turned left, straight and right (i.e., with $\delta = \delta_{\min}$, $\delta = 0$ and $\delta = \delta_{\max}$). Any state of the automaton is a final one. The alphabet is given by $\Sigma_3 = \{\nwarrow, \uparrow, \nearrow, \swarrow, \downarrow, \searrow\}$. Assuming the initial state is Rc, that is, the state in which the vehicle has the reverse gear engaged and $\delta = 0$, Figure 4.6 represents the possible state transitions from state Rc; state transitions from other states are analogous to this one. The language accepted by $\mathbb{A}_3$ is given by $L(\mathbb{A}_3) = \Sigma_3^*$, (see (2.2)), and the transition function ρ_3 is defined as follows: $(\forall s \in \mathcal{I}_3)$

$$\begin{aligned} &\rho_3(s, \nwarrow) = \text{Fl}, \qquad &&\rho_3(s, \uparrow) = \text{Fc}, \qquad &&\rho_3(s, \nearrow) = \text{Fr}, \\ &\rho_3(s, \swarrow) = \text{Rl}, \qquad &&\rho_3(s, \downarrow) = \text{Rc}, \qquad &&\rho_3(s, \searrow) = \text{Rr}. \end{aligned}$$

In this model, we have as many control subsystems as the number of states of $\mathbb{A}_3$ representing the vehicle with a forward or reverse gear engaged and with front wheels completely turned left, straight or right. Also, notice that, as in the first model, even though there is no given limit over the number of gear or direction changes, again, the switching cost function κ provides an implicit bound over the number of state switchings.

4.2.4 Model 4

If one wishes to use the model represented by the finite state automaton $\mathbb{A}_3$ but with the additional constraint over the maximum number of allowed gear switchings of finite state automaton $\mathbb{A}_2$, one could combine them replacing each state of $\mathbb{A}_2$ with a copy of $\mathbb{A}_3$. More precisely, it is possible to define a finite state automaton $\mathbb{A}_4 = (\mathcal{I}_4, \Sigma_4, \rho_4, \text{Rc}_P, \{\mathcal{A}_1, \mathcal{A}_0\})$, in which $\mathcal{I}_4 = \bigcup_{i=0}^{P} \mathcal{A}_i \cup \{\text{I}\}$, with $P \in \mathbb{N}$ playing the same role as in finite state automaton $\mathbb{A}_2$, sets $\mathcal{A}_i = \{\text{Fl}_i, \text{Fc}_i, \text{Fr}_i, \text{Rl}_i, \text{Rc}_i, \text{Rr}_i\}$, for $i \in \{0, \ldots, P\}$, and I representing an invalid state. Alphabet Σ_4 is equal to Σ_3

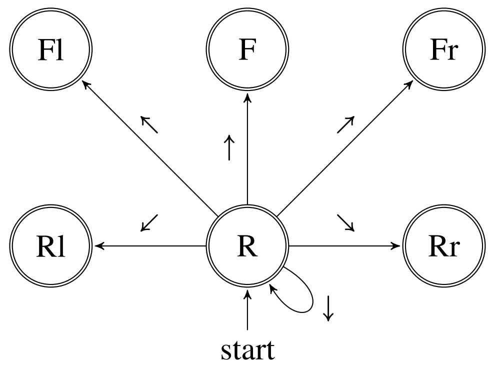


Figure 4.6: State transitions from state R for finite state automaton $\mathbb{A}_3$.

and transition function ρ_4 operates as follows: $(\forall i \in \{1,\ldots,P\})$ $(\forall s \in \mathcal{A}_i)$

$$\begin{array}{lll} \rho_4(s,\nwarrow) = \mathrm{Fl}_{i-1}, & \rho_4(s,\uparrow) = \mathrm{Fc}_{i-1}, & \rho_4(s,\nearrow) = \mathrm{Fr}_{i-1}, \\ \rho_4(s,\swarrow) = \mathrm{Rl}_{i-1}, & \rho_4(s,\downarrow) = \mathrm{Rc}_{i-1}, & \rho_4(s,\searrow) = \mathrm{Rr}_{i-1}, \end{array}$$

and $(\forall s \in \mathcal{A}_0)$ $(\forall \sigma \in \Sigma_4)$ $\rho_4(s,\sigma) = \mathrm{I}$. We set as initial state Rc_P, but it could be any state $s \in \mathcal{A}_P$. The set of final states is given by $\{\mathcal{A}_1, \mathcal{A}_0\}$, that is, any state in which at most one more gear switching is allowed.

4.3 Numerical experiments

As an example, let us considered three typical urban parking scenarios (two with a parallel parking and one with a perpendicular parking), and finite state automata $\mathbb{A}_2$ with $P = 10$. The numerical solution of the switching HJB equation was computed and several paths were generated starting from different initial states of the vehicle. In Figures 4.7, 4.8 and 4.9 we show some of the computed paths.

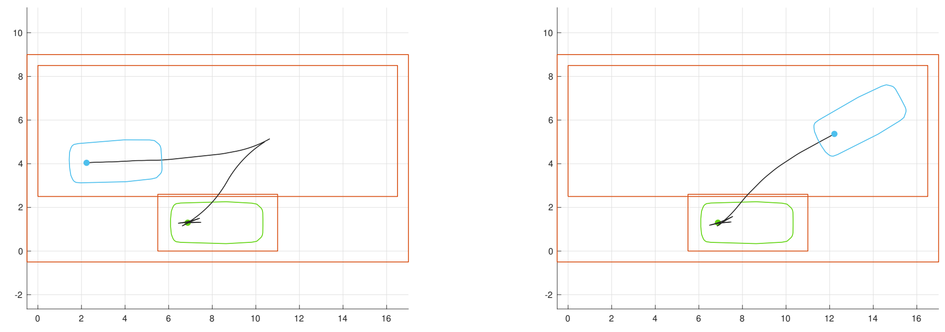


Figure 4.7: Paths for a parallel parking.

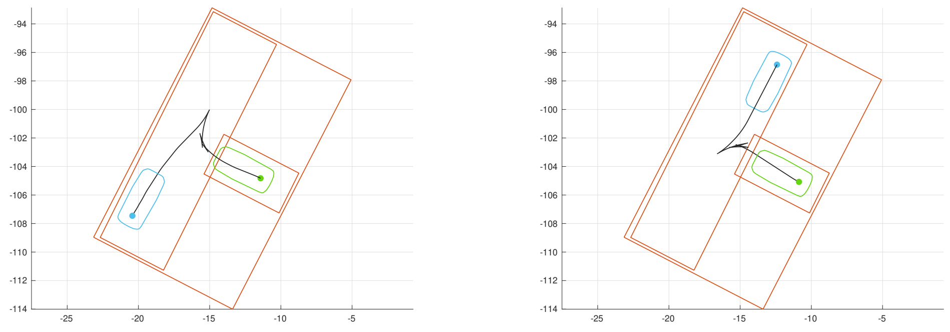


Figure 4.8: Paths for a perpendicular parking.

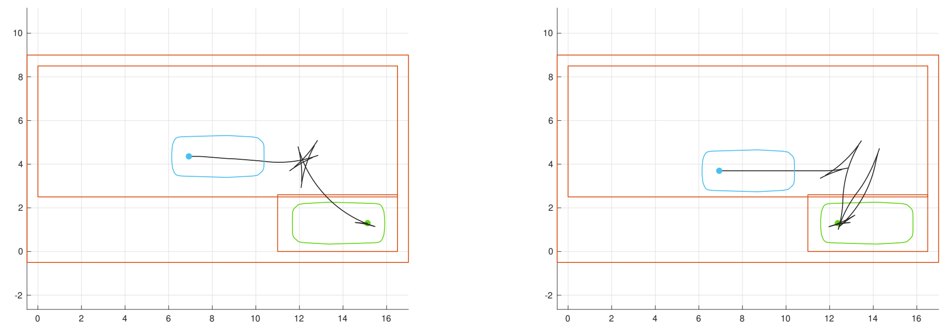


Figure 4.9: Paths for a parallel parking.

Chapter 5

An iterative algorithm for Problem (1.3)

In this chapter we analyze in further details Problems (5.4) and (5.4) and present iterative procedures for computing their solutions. Then, we examine the convergence speed of methods for solving Problem (5.4), in particular, we discuss how orderings for the newly introduced procedures can massively influence the speed of convergence. In conclusion, we test the introduced algorithms over different kind of randomly-generated networks.
The contents of this chapter are based on [12, 31] and improve the results of [32].

5.1 Characterization of Problem (1.1)

In this chapter we consider Problem (1.1) with the additional assumption $g(a) \geq a$ which guarantees that the feasible set of Problem (1.1)

$$\Sigma = \{x \in \mathbb{R}^n \mid a \leq x \leq g(x)\}$$

is such that $\Sigma \neq \emptyset$ (i.e., non-empty).

For any $\Gamma \subseteq \Sigma$ define $\bigvee_\Sigma \Gamma$ as the smallest $x \in \Sigma$, if it exists, such that $(\forall y \in \Gamma)\ x \geq y$. According to Definition 2.6, we call $\bigvee_\Sigma \Gamma$ the *least upper bound* of Γ (in Σ). Note that $\bigvee_\Sigma \emptyset = a$. The following proposition shows that $\bigvee_\Sigma \Gamma$ exists.

Proposition 5.1. *For any $\Gamma \subseteq \Sigma$, $\bigvee_\Sigma \Gamma$ exists.*

Proof. We first prove that, if $x, y \in \Sigma$, then $x \vee y \in \Sigma$ (recall that $\vee$ denotes the component-wise maximum). It is obvious that $x \vee y \geq a$. Thus, we only need to prove that, for each $j \in \{1, \dots, n\}$, $[y \vee x]_j \leq g_j(x \vee y)$. To see this, let us assume, w. l. o. g. , that $[x]_j \leq [y]_j$. Since $y \in \Sigma$, then $[y]_j \leq g_j(y)$. Moreover, $g_j(y) \leq g_j(y \vee x)$

since g_j is monotone non decreasing, so that $[y \vee x]_j \leq g_j(y \vee x)$ as we wanted to prove.

Let us consider a generic $\Gamma \subseteq \Sigma$ and call $\bigvee_{\mathcal{U}} \Gamma$ the least upper bound of Γ in $\mathcal{U} = \{x \in \mathbb{R}^n \mid a \leq x \leq U\}$. Since $\mathcal{U}$ is the cartesian product of complete lattices, it is a complete lattice itself, hence, the existence of $x^\star = \bigvee_{\mathcal{U}} \Gamma$ is ensured since $\Gamma \subseteq \mathcal{U}$. We now want to show that $x^\star \in \Sigma$. By definition of $\mathcal{U}$, $x^\star \geq a$, so we only need to prove that $x^\star \leq g(x^\star)$. By contradiction, assume that $x^\star \nleq g(x^\star)$, then, this means that there exists a component i of $x^\star$ such that $[x^\star]_i > [g(x^\star)]_i$, but, since $x^\star$ is the least upper bound of Γ in $\mathcal{U}$, there must be an element $x \in \Gamma$ such that $[x]_i > [g(x)]_i$, which goes against the fact that $\Gamma \subseteq \Sigma$. Hence, $x^\star \in \Sigma$. Now, by Lemma 2.28 of [17], if $\bigvee_{\mathcal{U}} \Gamma \in \Sigma$, then $\bigvee_{\Sigma} \Gamma$ exists and equals $\bigvee_{\mathcal{U}} \Gamma$. □

From now on we will omit Σ as subscript of $\bigvee_{\Sigma}$ and assume that for any $\Gamma \subseteq \Sigma$, $\bigvee \Gamma$ denotes the least upper bound of Γ in Σ. Similarly, define $\bigwedge \Gamma$ as the largest x, if it exists, such that $(\forall y \in \Gamma)\ x \leq y$, we call $\bigwedge \Gamma$ the *greatest lower bound* of Γ. For $x, y \in \Sigma$, note that $x \vee y = \bigvee\{x, y\}$, $x \wedge y = \bigwedge\{x, y\}$.

The following proposition characterizes set Σ with respect to operations $\vee$, $\wedge$. In particular, it shows that the component-wise minimum and maximum of each subset of Σ belongs to Σ.

Proposition 5.2. $\Sigma(\leq, \wedge, \vee)$ *is a complete lattice.*

Proof. It is a consequence of the Duality Principle (Theorem 2.5) and of Theorem 2.10. Indeed Σ has a bottom element (a) and $\bigvee \Gamma$ exists for any non-empty $\Gamma \subset \Sigma$ by Proposition 5.1. □

A consequence of the previous definition is that also $\bigwedge \Gamma$ exists.

The following proposition shows that the least upper bound x^+ of Σ is a fixed point of g and corresponds to an optimal solution of Problem (1.1).

Proposition 5.3. *Set*

$$x^+ = \bigvee \Sigma,$$

then i)

$$x^+ = g(x^+) \tag{5.1}$$

ii) x^+ is an optimal solution of problem (1.1).

Proof. i) It is a consequence of Knaster-Tarski Theorem (see Theorem 2.16), since $\Sigma(\leq, \wedge, \vee)$ is a complete lattice and g is an order-preserving map.

ii) By contradiction, assume that x^+ is not optimal, this implies that there exists $x \in \Sigma$ such that $f(x) > f(x^+)$. Being f monotonic increasing, this implies that there exists $i \in \{1, \ldots, n\}$ such that $[x]_i > [x^+]_i$, which implies that $x^+ \neq \bigvee \Sigma$. □

Remark 5.4. *The previous proposition shows that the actual form of function f is immaterial to the solution of Problem* (1.1)*, since the optimal solution is x^+ for any strictly monotonic increasing objective function f.*

The following defines a relaxed solution of Problem (1.1), obtained by allowing an error on fixed-point condition (5.1).

Definition 5.5. Let ε be a positive real constant, x is an ε-solution of (1.1) if

$$x \geq a, \text{ and } \|x - g(x)\|_\infty < \varepsilon .$$

The following proposition presents a sufficient condition that guarantees that a sequence of ε-solutions approaches x^+ as ε converges to 0.

Proposition 5.6. *If there exists $\delta > 0$ such that*

$$(\forall x, y \geq a)\ \frac{\|g(x) - g(y)\|_\infty}{\|x - y\|_\infty} \notin [1 - \delta, 1 + \delta] \tag{5.2}$$

then, there exists a constant M such that, for any $\varepsilon > 0$, if $x \in \mathbb{R}^n$ is an ε-solution of (1.1)*, then*

$$\|x - x^+\|_\infty \leq M\varepsilon .$$

Proof. Let x be an ε-solution. By Proposition 5.3 we have that

$$x - x^+ = g(x) - g(x^+) + \xi ,$$

where $\|\xi\|_\infty \leq \varepsilon$.

By assumption (5.2), either $\|g(x) - g(y)\|_\infty > (1+\delta)\|x-y\|_\infty$ or $\|g(x) - g(y)\|_\infty < (1-\delta)\|x-y\|_\infty$. In the first case,

$$\|x - x^+\|_\infty \geq -\|\xi\|_\infty + (1+\delta)\|x - x^+\|_\infty ,$$

in the second case,

$$\|x - x^+\|_\infty \leq \|\xi\|_\infty + (1-\delta)\|x - x^+\|_\infty .$$

In both cases it follows that

$$\|x - x^+\|_\infty \leq \delta^{-1}\|\xi\|_\infty \leq \delta^{-1}\varepsilon .$$

□

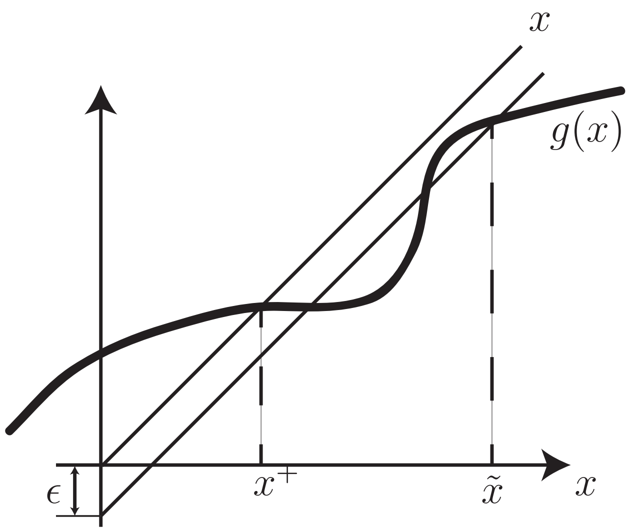


Figure 5.1: Representation of an instance of Problem (1.1) in which conditions (5.2) does not hold.

Remark 5.7. *If condition* (5.2) *is not satisfied, an ε-solution of* (1.1) *can be very distant from the optimal solution x^+. Figure 5.1 refers to a simple instance of Problem* (1.1) *with $x \in \mathbb{R}$, so that g is a scalar function. The optimal value x^+ corresponds to the maximum value of x such that $x \le g(x)$. The figure also shows $\tilde{x}$, which is an ε-solution, for the value of ε depicted in the figure. In this case there is a large separation between x^+ and $\tilde{x}$. Note that in this case function g does not satisfy* (5.2).

Remark 5.8. *If g is a contraction, namely, if there exists $\gamma \in [0,1)$, such that $(\forall x,y \in \mathbb{R}^n)\ \|g(x)-g(y)\|_\infty \le \gamma\|x-y\|_\infty$ (a subcase of* (5.2)*), then x^+ can be found with a standard fixed point iteration*

$$\begin{cases} x(k+1) = g(x) \\ x(0) = x_0, \end{cases} \tag{5.3}$$

and, given $\varepsilon > 0$, an ε-solution x of (1.1) *can be computed with Algorithm 1. This algorithm, given an input tolerance ε, function g and an initial solution $x_0 \in \mathbb{R}^n$, repeats the fixed point iteration $x = g(x)$ until x satisfies the definition of ε-solution, that is, until the infinity norm of error vector $\xi = x - g(x)$ is smaller than the assigned tolerance ε.*

The special structure of Problem (1.1) leads to a solution algorithm that is much more efficient than Algorithm 1 in terms of overall number of elementary operations. As a first step, we associate a graph to constraint g of Problem (1.1).

Algorithm 1 Fixed Point Iteration.

```
INPUT: initial vector $x_0$, tolerance $\varepsilon$, function $g$.
OUTPUT: vector $x$.

$x := x_0$

**repeat**
    $x_{\text{old}} := x$
    $x := g(x)$
    $\xi := x_{\text{old}} - x$
**until** $\|\xi\|_\infty \le \varepsilon$

**return** $x$
```

5.1.1 Selective update algorithm for Problem (1.1)

In Algorithm 1, each time line 8 is evaluated, the value of all components of x is updated according to the fixed point iteration $x = g(x)$, even though many of them may remain unchanged. We now present a more efficient procedure for computing an ε-solution of (1.1), in which we update only the value of those components of x that are known to undergo a variation. The algorithm is composed of two phases, an initialization and a main loop. In the *initialization*, x is set to an initial value x_0 that is known to satisfy $x_0 \ge x^+$. Then the fixed point error $\xi = x - g(x)$ is computed and all indexes $i \in \{1, \ldots, n\}$ for which $[\xi]_i > \varepsilon$ are inserted into a priority queue, ordered with respect to a policy that will be discussed later. In this way, at the end of the initialization, the priority queue contains all indexes i for which the corresponding fixed point error $[\xi]_i$ exceeds ε.

Then, the *main loop* is repeated until the priority queue is empty. First, we extract from the priority queue the index i with the highest priority. Then, we update its value by setting $[x]_i = g_i(x)$ and update the fixed point error ξ by setting $[\xi]_j = [x]_j - g_j(x)$ for all variables $j \in \mathcal{N}(i)$. This step is actually the key-point of the algorithm: we recompute the fixed point error *only* of those variables that correspond to components of g that we know to have been affected by the change in variable $[x]_i$. Finally, as in the initialization, all variables $j \in \mathcal{N}(i)$ such that the updated fixed-point error satisfies $[\xi]_j > \varepsilon$ are placed into the priority queue.

The order in which nodes are actually processed depends on the ordering of the priority queue. The choice of this ordering turns out to be critical in terms of computational cost for the algorithm, as can be seen in the numerical experiments in Section 5.3.3. Various orderings for the priority queue will be introduced in Section 5.3.3 and the ordering choice will be discussed in more detail. The procedure stops once the priority queue becomes empty, that is, once none of the updated nodes undergoes a significant variation. As we will show, the correctness of the algorithm is independent on the choice of the ordering of the priority queue.

We may think of graph $\mathbb{G}$ as a communication network in which each node transmits its updated value to its neighbours, whilst all other nodes maintain their value unchanged.

These considerations lead to Algorithm 2. This algorithm takes as input an initial vector $x_0 \in \mathbb{R}^n$, a tolerance ε, function g and the lower bound a. From lines 4 to 6 it initializes the solution vector x, the priority queue Q and the error vector ξ. From line 8 to 12 it adds into the priority queue those component nodes whose corresponding component of the error vector ξ is greater than tolerance ε. The priority with which a node is added to the queue will be discussed in Section 5.3.3, here symbol * denotes a generic choice of priority. Lines from 14 to 24 constitute the main loop. While the queue is not empty, the component node i with highest priority is extracted from the queue and its value is updated. Then, each component node j which is a neighbor of i is examined; the variation of node j is updated and, if it is greater than tolerance ε, neighbor j is added to the priority queue. After this, the component corresponding to node i in ξ is set to 0. Finally, once the queue becomes empty, the feasibility of solution x is checked and returned along with vector x. We remark that Algorithm 2 can be seen as a generalization of Algorithm 1 in [9], where a specific priority queue (namely, one based on the values of the nodes) was employed. Also note that Algorithm 2 can be seen as a bound-tightening technique (see, e. g., [6]) which, however, for this specific class of problem is able to return the optimal solution.

The following proposition characterizes Algorithm 2 and proves its correctness.

Proposition 5.9. *Assume that $x_0 \geq x^+$ and $g(x_0) \leq x_0$, then Algorithm 2 satisfies the following properties:*

i) At all times, $x \geq x^+$ and $x \geq g(x)$.

ii) After evaluation of line 6 and after every evaluation of line 23, $x = g(x) + \xi$ and $\xi \geq 0$.

iii) The algorithm terminates in a finite number of steps for any $\varepsilon > 0$.

iv) If Problem (1.1) *is feasible, output "feasible" is true.*

v) If output "feasible" is true, then x is an ε-feasible solution of Problem (1.1).

Proof. i) We prove both properties by induction. Note that x is updated only at line 16 and that line 16 is equivalent to $[x]_i = g_i(x)$. For $m \in \mathbb{N}$, let $x(m)$ be the value of x after the m-th evaluation of line 16. Note that $x(0) = x_0 \geq x^+$ and that x is changed only at step 16. Then $[x(m)]_i = g_i(x(m-1)) \geq g_i(x^+) = [x^+]_i$, where we have used the inductive hypothesis $x(m-1) \geq x^+$ and the fact that $g(x^+) = x^+$ (by Proposition 5.3).

Algorithm 2 Solution algorithm for Problem (1.1)

```
1:  INPUT: initial vector $x_0$, tolerance $\varepsilon$, function $g$, vector $a$.
2:  OUTPUT: vector $x$, bool $feasible$.
3:
4:  $x := x_0$
5:  $Q := \varnothing$
6:  $\xi := x - g(x)$
7:
8:  for $i \in \{1,\dots,n\}$ do
9:      if $[\xi]_i > \varepsilon$ then
10:         $Q :=$ Enqueue$(Q,(i,*))$
11:     end if
12: end for
13:
14: while $Q \neq \varnothing$ do
15:     $(Q,i) :=$ Dequeue$(Q)$
16:     $[x]_i := [x]_i - [\xi]_i$
17:     for all $j \in \mathcal{N}(i)$ do
18:         $[\xi]_j := [x]_j - g_j(x)$
19:         if $[\xi]_j > \varepsilon$ then
20:             $Q :=$ Enqueue$(Q,(j,*))$
21:         end if
22:     end for
23:     $[\xi]_i := 0$
24: end while
25:
26: $feasible := x \geq a$
27:
28: return $x, feasible$
```

Further, note that $g(x(0)) = g(x_0) \leq x_0$ by assumption. Moreover, $[x(m)]_i = g_i(x(m-1)) = g_i(x(m))$, since g_i does not depend on $[x]_i$ by assumption and variables $x(m)$, $x(m-1)$ differ only on the i-th component. By the induction hypothesis, $[x(m)]_i = g_i(x(m-1)) \leq [x(m-1)]_i$ which implies that $x(m) \leq x(m-1)$. Thus, in view of the monotonicity of g and of the inductive assumption, for $k \neq i$, $[g(x(m))]_k = g_k(x(m)) \leq g_k(x(m-1)) \leq [x(m-1)]_k = [x(m)]_k$.

ii) Condition $x = g(x) + \xi$ is satisfied after evaluating 6. Moreover, after evaluating line 23, $[x]_i = [g(x)]_i + [\xi]_i$ and all indices j for which potentially $[x]_j \neq [g(x)]_j + [\xi]_j$ belong to set $\mathcal{N}(i)$. For these indices, line 18 re-enforces $[x]_j = [g(x)]_j + [\xi]_j$. The fact that $\xi \geq 0$ is a consequence of point i).

iii) At each evaluation of line 16 the value of a component of x is decreased by at least ε. If the algorithm did not terminate, at some iteration we would have that $x \ngeq x^+$ which is not possible by i).

iv) If Problem (1.1) is feasible, then $x^+ \geq a$ is its optimal solution. By point 1), $x \geq x^+ \geq a$ and output "feasible" is true.

v) When the algorithm terminates, Q is empty, which implies than $\|x - g(x)\|_\infty \leq \varepsilon$, if "feasible" is true, it is also $x \geq a$ and x is an ε-solution. $\square$

5.2 Characterization of Problem (1.3)

In this section, we consider Problem (1.3) and we propose a solution method that exploits its linear structure and is more efficient than Algorithm 2. First of all, we show that Problem (1.3) belongs to class (1.1). To this end, set

$$P_\ell := I - D_\ell, \tag{5.4}$$

where $I \in \mathbb{R}^{n\times n}$ is the identity matrix and, for $\ell \in \mathcal{L}$, $D_\ell \in \mathbb{R}^{n\times n}$ is a diagonal matrix that contains the elements of A_ℓ on the diagonal. Note that here and in what follows we assume that all the diagonal entries of A_ℓ are lower than 1. Indeed, for values larger than or equal to 1 the corresponding constraints are redundant and can be eliminated. The proof of the following proposition is in the appendix.

Proposition 5.10. *Problem* (1.3) *can be reformulated as a problem of class* (1.1). *Namely, this is achieved by setting*

$$\hat{A}_\ell := P_\ell^{-1}(A_\ell - D_\ell), \quad \hat{b}_\ell := P_\ell^{-1} b_\ell \tag{5.5}$$

and $\hat{g}(x) = \bigwedge_{\ell\in\mathcal{L}} \{\hat{A}_\ell x + \hat{b}_\ell\} \wedge U$.

Proof. Given $A \in \mathbb{R}^{n\times n}$ let us define, for $i \in \{1, \dots, n\}$ the sum of the elements of row i

$$s_i(A) := \sum_{j=1}^{n} [A]_{ij}. \tag{5.6}$$

Note that, for any $\ell \in \mathcal{L}$, matrix P_ℓ defined in (5.4) is positive diagonal since, by assumption, all elements of D_ℓ are less than 1. One can rewrite the inequality of

Problem (1.3) as

$$\begin{aligned}
&\underline{0} \le \bigwedge_{\ell\in\mathcal{L}} \{A_\ell x - (D_\ell - D_\ell + I)x + b_\ell\}\\
\Leftrightarrow\ &\underline{0} \le \bigwedge_{\ell\in\mathcal{L}} \{(A_\ell - D_\ell)x - (I - D_\ell)x + b_\ell\}\\
\Leftrightarrow\ &\underline{0} \le \bigwedge_{\ell\in\mathcal{L}} \{(I - D_\ell)^{-1}(A_\ell - D_\ell)x - x + (I - D_\ell)^{-1} b_\ell\}\\
\Leftrightarrow\ &x \le \bigwedge_{\ell\in\mathcal{L}} \{(I - D_\ell)^{-1}(A_\ell - D_\ell)x + (I - D_\ell)^{-1} b_\ell\}\\
\Leftrightarrow\ &x \le \bigwedge_{\ell\in\mathcal{L}} \{P_\ell^{-1}(A_\ell - D_\ell)x + P_\ell^{-1} b_\ell\}
\end{aligned}$$

Then, set $\hat{A}_\ell := P_\ell^{-1}(A_\ell - D_\ell)$ and $\hat{b}_\ell := P_\ell^{-1} b_\ell$ and $\hat{g}(x) = \bigwedge_{\ell\in\mathcal{L}} \{\hat{g}_\ell(x)\} \wedge U$, where, for $\ell \in \mathcal{L}$,

$$\hat{g}_\ell(x) := \hat{A}_\ell x + \hat{b}_\ell. \tag{5.7}$$

Note that $\hat{g}$ is monotonic (since all entries of $\hat{A}_\ell$ are nonnegative) and for $i \in \{1,\ldots,n\}$, $[\hat{g}]_i$ is independent on x_i (since the diagonal entries of $\hat{A}_\ell$ are null). Note also that $\hat{b}_\ell$ is nonnegative. Hence, Problem (1.3) takes on the form of Problem (1.1). □

Then we apply the results for Problem (1.1) to Problem (1.3). The following proposition is a corollary of Proposition 5.3.

Proposition 5.11. *Problem* (1.3) *is feasible and its optimal solution* x^+ *satisfies the two equations*

$$x^+ = \bigwedge_{\ell\in\mathcal{L}} \left\{\hat{A}_\ell x^+ + \hat{b}_\ell\right\} \wedge U\,. \tag{5.8}$$

$$x^+ = \bigwedge_{\ell\in\mathcal{L}} \{A_\ell x^+ + b_\ell\} \wedge U\,. \tag{5.9}$$

Proof. Setting $g(x) = \bigwedge_{\ell\in\mathcal{L}} \left\{\hat{A}_\ell x + \hat{b}_\ell\right\} \wedge U$, note that $g(\underline{0}) = \bigwedge_{\ell\in\mathcal{L}} \left\{\hat{b}_\ell\right\} \wedge U \ge \underline{0}$, which implies that $\Sigma \neq \varnothing$ and that Problem (1.1) is feasible. Then, by Proposition 5.3, its solution x^+ satisfies $x^+ = g(x^+)$, which implies (5.8) and (5.9). □

The following result, needed below, can be found, e. g., in [25].

Lemma 5.12. *Let* $L \in \mathbb{R}_+$ *and* $\{g_i \mid i \in I\}$*, with* I *set of indices, be a family of functions* $g_i : \mathbb{R}^n \to \mathbb{R}^n$ *such that*

$$(\forall x, y \in \mathbb{R}^n)\ \|g_i(x) - g_i(y)\|_\infty \le L\|x - y\|_\infty.$$

Then, function $g(x) := \bigwedge_{i\in I} \{g_i(x)\}$ *also satisfies*

$$(\forall x, y \in \mathbb{R}^n)\ \|g(x) - g(y)\|_\infty \le L\|x - y\|_\infty.$$

The following proposition illustrates that if the infinity norm of all matrices A_ℓ is lower than 1, equation (5.9) is actually a contraction.

Proposition 5.13. *Assume that there exists a real constant* $\gamma \in [0,1)$ *such that*

$$(\forall \ell \in \mathcal{L}) \ \|A_\ell\|_\infty < \gamma, \tag{5.10}$$

then function

$$\bar{g}(x) = \bigwedge_{\ell \in \mathcal{L}} \{A_\ell x + b_\ell\} \wedge U\,. \tag{5.11}$$

is a contraction in infinity norm, in particular,

$$(\forall x, y \in \mathbb{R}^n) \ \|\bar{g}(x) - \bar{g}(y)\|_\infty \le \gamma \|x - y\|_\infty\,.$$

Proof. Note that, for any $\ell \in \mathcal{L}$, function $h(x) = A_\ell x + b_\ell$ is a contraction, in fact, for any $x, y \in \mathbb{R}^n$

$$\|h(x) - h(y)\|_\infty = \|A_\ell(x - y)\|_\infty \le \gamma \|x - y\|_\infty\,.$$

Then, the thesis is a consequence of Lemma 5.12. □

The following result proves that, under the same assumptions, also (5.8) is a contraction. The proof is in the appendix.

Proposition 5.14. *Assume that* (5.10) *holds and set*

$$\hat{A}_\ell = P_\ell^{-1}(A_\ell - D_\ell) \ \ \textit{and} \ \ \hat{b}_\ell = P_\ell^{-1} b_\ell,$$

with P_ℓ and D_ℓ defined as in (5.4). *Let*

$$\hat{g}(x) = \bigwedge_{\ell \in \mathcal{L}} \{\hat{A}_\ell x + \hat{b}_\ell\} \wedge U, \tag{5.12}$$

then $\hat{g}$ is a contraction in infinity norm, in particular,

$$(\forall x, y \in \mathbb{R}^n) \ \|\hat{g}(x) - \hat{g}(y)\|_\infty \le \hat{\gamma} \|x - y\|_\infty\,,$$

where

$$\hat{\gamma} := \max_{\substack{\ell \in \mathcal{L} \\ i \in \mathcal{V}}} \left\{ \frac{\gamma - [D_\ell]_{ii}}{1 - [D_\ell]_{ii}} \right\}. \tag{5.13}$$

Moreover, it holds that $\hat{\gamma} \le \gamma$.

Proof. Given P_ℓ as in (5.4) for $i \in \{1,\dots,n\}$ and $\ell \in \mathcal{L}$ we have that

$$s_i(\hat{A}_\ell) \le \frac{\gamma - [D_\ell]_{ii}}{[P_\ell]_{ii}},$$

where s_i is defined in (5.6) and $\hat{A}_\ell$ is defined as in (5.5). Let us note that

$$\max_{\substack{\ell \in \mathcal{L}\\ i \in \mathcal{V}}} \{s_i(\hat{A}_\ell)\} \le \max_{\substack{\ell \in \mathcal{L}\\ i \in \mathcal{V}}} \left\{ \frac{\gamma - [D_\ell]_{ii}}{[P_\ell]_{ii}} \right\} = \max_{\substack{\ell \in \mathcal{L}\\ i \in \mathcal{V}}} \left\{ \frac{\gamma - [D_\ell]_{ii}}{1 - [D_\ell]_{ii}} \right\} = \hat{\gamma},$$

where $\hat{\gamma}$ is defined as in (5.13). Note that the term on the left-hand side is the maximum of $s_i(A)$ for all possible $i \in \{1,\dots,n\}$ and for all possible matrices $A \in \mathbb{R}^{n\times n}$ which can be obtained by all possible combinations of the rows of matrices A_ℓ, with $\ell \in \mathcal{L}$. We prove that $\hat{\gamma} \le \gamma$, under the given assuptions. Indeed, it is immediate to see that function

$$S(d) := \frac{\gamma - d}{1 - d}$$

is monotone decreasing for any $d \in [0,\gamma]$. We remark that, for any $\ell \in \mathcal{L}$, $\|\hat{A}_\ell\|_\infty \le \hat{\gamma}$. Now, for any $x \in \mathbb{R}^n$, let us define $\hat{g}_U(x) := U$, while for any $\ell \in \mathcal{L}$, $\hat{g}_\ell(x)$ is defined as in (5.7). It is immediate to see that $(\forall x, y \in \mathbb{R}^n)$ $\|\hat{g}_i(x) - \hat{g}_i(y)\|_\infty \le \hat{\gamma}\|x-y\|_\infty$, for any $i \in \{1,\dots,n\} \cup \{U\}$. Then, by Lemma 5.12 we have that, for $\hat{g}(x) = \bigwedge_{k \in \mathcal{L} \cup \{U\}} \hat{g}_k(x)$, it holds that $(\forall x, y \in \mathbb{R}^n)$ $\|\hat{g}(x) - \hat{g}(y)\|_\infty \le \hat{\gamma}\|x-y\|_\infty$, that is, $\hat{g}$ is a contraction. □

Hence, in case (5.10) is satisfied, Problem (1.3) can be solved by Algorithm 1 using either $g = \bar{g}$ in (5.11) or $g = \hat{g}$ in (5.12). As we will show in Section 5.3, the convergence is faster in the second case.

Algorithm 2 can be applied to Problem (1.3), being a subclass of (1.1). Anyway, the linear structure of Problem (1.3) allows for a more efficient implementation, detailed in Algorithm 3. This algorithm takes as input an initial vector $x_0 \in \mathbb{R}^n$, a tolerance ε, matrices A_ℓ and vectors b_ℓ, for $\ell \in \mathcal{L}$, representing function g and the lower bound a. It operates like Algorithm 2 but it optimizes the operation performed in line 18 of Algorithm 2. Lines from 7 to 10 initialize the error vector ξ and they correspond to line 6 of Algorithm 2. Whilst, lines 22 from to 25 are the equivalent of line 18 of Algorithm 2 in which the special structure of Problem (1.3) is exploited in such a way that the updating of the j-th component of vector ξ only involves the evaluation of L scalar products and L scalar sums, with $L = |\mathcal{L}|$, as opposed to (up to) nL scalar products and nL scalar sums of Algorithm 2 applied to Problem (1.3).

Algorithm 3 Solution algorithm for Problem (1.3).

```
1:  INPUT: initial vector $x_0$, tolerance $\varepsilon$, matrices $A_\ell$, vectors $b_\ell$ for $\ell \in \mathcal{L}$, vector
    $a$.
2:  OUTPUT: vector $x$.
3:
4:  $x := x_0$
5:  $Q := \varnothing$
6:
7:  for all $\ell \in \mathcal{L}$ do
8:     $\eta_\ell := A_\ell x + b_\ell$
9:  end for
10: $\xi := x - \bigwedge_{\ell \in \mathcal{L}} \eta_\ell$
11:
12: for all $i \in \{1, \dots, n\}$ do
13:    if $([\xi]_i > \varepsilon)$ then
14:       $Q :=$ Enqueue$(Q, (i, *))$
15:    end if
16: end for
17:
18: while $Q \neq \varnothing$ do
19:    $(Q, i) :=$ Dequeue$(Q)$
20:    $[x]_i = [x]_i - [\xi]_i$
21:    for all $j \in \{1, \dots, n\}$ such that $i \in \mathcal{N}(j)$ do
22:       for all $\ell \in \mathcal{L}$ do
23:          $[\eta_\ell]_j := [\eta_\ell]_j - [A_\ell]_{ji} \cdot [\xi]_i$
24:       end for
25:       $[\xi]_j := [x]_j - \min_{\ell \in \mathcal{L}} [\eta_\ell]_j$
26:       if $[\xi]_j > \varepsilon$ then
27:          $Q :=$ Enqueue$(Q, (j, *))$
28:       end if
29:    end for
30:    $[\xi]_i = 0$
31: end while
32:
33: $feasible := x \geq a$
34:
35: return $x, feasible$
```

5.3 Convergence speed discussion

In this section, we will compare the convergence speed of various methods for solving Problem (1.3). First of all, note that Problem (1.3) can be reformulated as the linear problem (1.4). Hence, it can be solved with any general method for linear problems. As we will show, the performance of such methods is poor since they do not exploit the special stucture of Problem (1.4).

5.3.1 Fixed point iterations

In case hypothesis (5.10) is satisfied, as discussed in Section 5.2, Problem (1.3) can be solved by Algorithm 1 using either $g = \bar{g}$ in (5.11) or $g = \hat{g}$ in (5.12). In other words, x^+ can be computed with one of the following iterations:

$$\begin{cases} x(k+1) = \bar{g}(x(k)) = \bigwedge_{\ell \in \mathcal{L}} \{A_\ell x(k) + b_\ell\} \wedge U \\ x(0) = x_0, \end{cases} \tag{5.14}$$

$$\begin{cases} x(k+1) = \hat{g}(x(k)) = \bigwedge_{\ell \in \mathcal{L}} \{\hat{A}_\ell x(k) + \hat{b}_\ell\} \wedge U \\ x(0) = x_0, \end{cases} \tag{5.15}$$

where $x_0 \in \mathbb{R}^n$ is an arbitrary initial condition and $\hat{A}_\ell$ and $\hat{b}_\ell$ are defined as in (5.5).

We can compare the convergence rate of iterations (5.14) and (5.15). The speed of convergence of iteration (5.14) can be measured by the convergence rate:

$$\bar{\chi} := \max_{\substack{x \in \mathbb{R}^N \\ x \neq x^+}} \left\{ \frac{\|\bar{g}(x) - \bar{g}(x^+)\|_\infty}{\|x - x^+\|_\infty} \right\}.$$

Similarly, we call $\hat{\chi}$ the convergence rate of iteration (5.15). Note that, by Proposition 5.13, $\bar{\chi} \leq \gamma$ and, by Proposition 5.14, $\hat{\chi} \leq \max_{\substack{\ell \in \mathcal{L} \\ i \in \mathcal{V}}} \left\{ \frac{\gamma - [D_\ell]_{ii}}{1 - [D_\ell]_{ii}} \right\} \leq \gamma$. Hence, in general, we have a better upper bound of the convergence rate of iteration (5.15) than (5.14).

Now, let us assume that matrices $\{A_\ell\}_{\ell \in \mathcal{L}}$ are dominant diagonal, that is, there exists $\Delta \in \left[0, \frac{1}{2}\right)$ such that, $(\forall i \in \{1, \dots, n\})$ $(\forall \ell \in \mathcal{L})$

$$[A_\ell]_{ii} \geq (1 - \Delta)\gamma \quad \text{and} \quad \sum_{\substack{j=1 \\ j \neq i}}^{n} [A_\ell]_{ij} \leq \Delta\gamma. \tag{5.16}$$

Recall that in the dynamic programming framework discussed in Chapter 4 this is attained when h is small enough. In the following theorem, whose proof is proved in the Appendix, we state that, if Δ is small enough, iteration (5.15) has a faster convergence than iteration (5.3).

Proposition 5.15. *Assume that* (5.10) *holds and let* $\Delta \in \left[0, \frac{1}{2}\right)$ *be such that matrices* $\{A_\ell\}_{\ell \in \mathcal{L}}$ *satisfy* (5.16)*. Then, if the starting point* x_0 *is selected in such a way that* $x_0 \geq x^+$*, then the solutions of both* (5.14) *and* (5.15) *satisfy* $(\forall k \in \mathbb{N})\ x(k) \geq x^+$*. Moreover, if*

$$\Delta \in \left[0, \frac{\sqrt{1-\gamma}-(1-\gamma)}{\gamma}\right),$$

then for any $x \geq x^+$

$$\|\hat{g}(x) - x^+\|_\infty < \|\bar{g}(x) - x^+\|_\infty .$$

Proof. We first remark that $x_0 \geq x^+$ implies $x_k \geq x^+$ and $\bar{g}(x_k) \geq x^+$ for any k, where $\bar{g}$ is defined as in (5.11). Then, we provide a lower bound for $\|\bar{g}(x_k) - \bar{g}(x^+)\|_\infty$. Let $\bar{A} \in \mathbb{R}_+^{n\times n}$ and $\bar{b} \in \mathbb{R}_+^n$ be such that $\bar{A}x_k + \bar{b} = \bar{g}(x_k)$. Note that $\bar{A}$ is obtained by a combination of the rows of matrices A_ℓ, with $\ell \in \mathcal{L}$. In other words, for each $i \in \{1,\ldots,n\}$, $\left[\bar{A}\right]_{i*} = \left[A_{\ell_i}\right]_{i*}$ for some $\ell_i \in \mathcal{L}$. Then, in view of $x_k \geq x^+$, $x^+ \leq \bar{A}x^+ + \bar{b}$ and $\bar{A} \geq 0$,

$$\begin{aligned}\|\bar{g}(x_k) - \bar{g}(x^+)\|_\infty &= \left\|\bar{A}x_k + \bar{b} - x^+\right\|_\infty \geq \left\|\bar{A}x_k + \bar{b} - (\bar{A}x^+ + \bar{b})\right\|_\infty = \left\|\bar{A}(x_k - x^+)\right\|_\infty \geq \\ &\geq \left\|\operatorname{diag}(\bar{A})(x_k - x^+)\right\|_\infty \geq (1-\Delta)\gamma\|x_k - x^+\|_\infty ,\end{aligned}$$

where the last inequality follows from (5.16). Then, the result follows by observing that

$$\begin{aligned}&\frac{\Delta\gamma}{1-(1-\Delta)\gamma} < (1-\Delta)\gamma \Leftrightarrow \Delta\gamma < (1-\Delta)\gamma - (1-\Delta)^2\gamma^2 \Leftrightarrow \\ &\Leftrightarrow \gamma^2\Delta^2 + 2(1-\gamma)\gamma\Delta - (1-\gamma)\gamma < 0 \Leftrightarrow \Delta \in \left[0, \frac{\sqrt{1-\gamma}-(1-\gamma)}{\gamma}\right).\end{aligned}$$

□

5.3.2 Speed of Algorithm 3 and priority queue policy

As we will see in the numerical experiments section, Algorithm 3 solves Problem (1.3) more efficiently than iterations (5.14) and (5.15).

As we already mentioned in the previous section, the order in which we update the values of the nodes in the priority queue does not affect the convergence of the algorithm but impacts heavily on its convergence speed. We implemented four different queue policies, detailed in the following.

Node variation

The priority associated to an index i is given by the absolute value of the variation of $[x]_i$ in its last update. In this case, in lines 10, 20 of Algorithm 2 and lines 14, 27 of Algorithm 3, symbol $*$ is replaced by the corresponding component of ξ of the node added to the queue (see Table 5.1). This can be considered a "greedy" policy, in fact we update first the components of the solution $[x]_i$ associated to a larger variation $[\xi]_i$, in order to have a faster convergence of the current solution x to x^+.

Node values

The priority associated to an index i in the priority queue is given by $-[x]_i$. In this case, in lines 10, 20 of Algorithm 2 and lines 14, 27 of Algorithm 3, symbol $*$ is replaced by the opposite of the value of the node added to the queue (see Table 5.1). The rationale of this policy is the observation that, in Problem (1.4), components of x with lower values are more likely to appear in active constraints. This policy mimics Dijkstra's algorithm, in fact the indexes associated to the solution components with lower values are processed first.

FIFO e LIFO policies

The two remaining policies implement respectively the First In First Out (FIFO) policy, (i.e., a stack) and the Last In First Out (LIFO) policy (i.e., a queue). Namely, in case of FIFO, the nodes are updated in the order in which they are inserted in the queue. In case of LIFO, they are updated in reverse order.

In order to formally implement these two policies in a priority queue, we need to introduce a counter k initialized to 0 and incremented every time a node is added to the priority queue. In lines 10, 20 of Algorithm 2 and lines 14, 27 of Algorithm 3, symbol $*$ is replaced by k in case we want to implement a LIFO policy and by $-k$ for implementing a FIFO policy (see Table 5.1). These steps are required to formally represent these two policies in Algorithm 3. As said, these two policies can be more simply implemened with an unordered queue (for FIFO policy) or a stack (for LIFO policy). The rationale of this two policies is to avoid the overhead of managing a priority queue. In fact, inserting an entry into a priority queue of n elements has a time-cost of $O(\log n)$, while the same operation on an unordered queue or a stack has a cost of $O(1)$. Note that, with these policies, we increase the efficiency in the management of the set of the indexes that have to be updated at the expense of a possible less efficient update policy.

5.3.3 Numerical experiments

In this section, we test Algorithm 3 on randomly generated problems of class (1.3). We carried out two sets of tests. In the first one, we compared the solution time of Algorithm 3 with different priority queue policies with a commercial solver for linear problems (Gurobi). In the second class of tests, we compared the number

Policy	Alg.2 line 10, Alg.3 line 14	Alg.2 line 20, Alg.3 line 27
Variation	$Q := \text{Enqueue}(Q,(i,[\xi]_i))$	$Q := \text{Enqueue}\big(Q,\big(j,[\xi]_j\big)\big)$
Value	$Q := \text{Enqueue}(Q,(i,-[x]_i))$	$Q := \text{Enqueue}\big(Q,\big(j,-[x]_j\big)\big)$
FIFO	$Q := \text{Enqueue}(Q,(i,k))$ $k := k+1$	$Q := \text{Enqueue}(Q,(j,k))$ $k := k+1$
LIFO	$Q := \text{Enqueue}(Q,(i,-k))$; $k := k+1$	$Q := \text{Enqueue}(Q,(j,-k))$ $k := k+1$

Table 5.1: Possible priority queue policies.

of scalar multiplications executed by Algorithm 3 (with different priority queue policies) with the ones required by the fixed point iteration (5.14).

Random problems generation

The following procedure allows generating a random problem of class (1.3) with n variables. The procedure takes the following input parameters:

- $U \in \mathbb{R}^+$: an upper bound for the problem solution,
- $M_A \in \mathbb{R}^+$: maximum value for entries of $A_1,\dots,A_L$,
- $M_b \in \mathbb{R}^+$: maximum value for entries of $b_1,\dots,b_L$,
- $G_1,\dots,G_L$: graphs with n nodes.

A problem of class (1.3) is then obtained with the following operations, for $i \in \{1,\dots,L\}$:

- Set D_i as the adiacency matrix of graph G_i,
- define A_i as the matrix obtained from D_i by replacing each nonzero entry of D_i with a random number generated from a uniform distribution in interval $[0,M_A]$,
- define $b_i \in \mathbb{R}^n$ so that each entry is a random number generated from a uniform distribution in interval $[0,M_b]$.

Graphs $G_1,\dots,G_L$ are obtained from standard classes of random graphs, namely:

- the Barabási-Albert model [4], characterized by a scale-free degree distribution,

- the Newman-Watts-Strogatz model [38], that originates graphs with small-world properties,
- the Holm and Kim algorithm [26], that produces scale-free graphs with high clustering.

In our tests, we used the software NetworkX [22] to generate the random graphs.

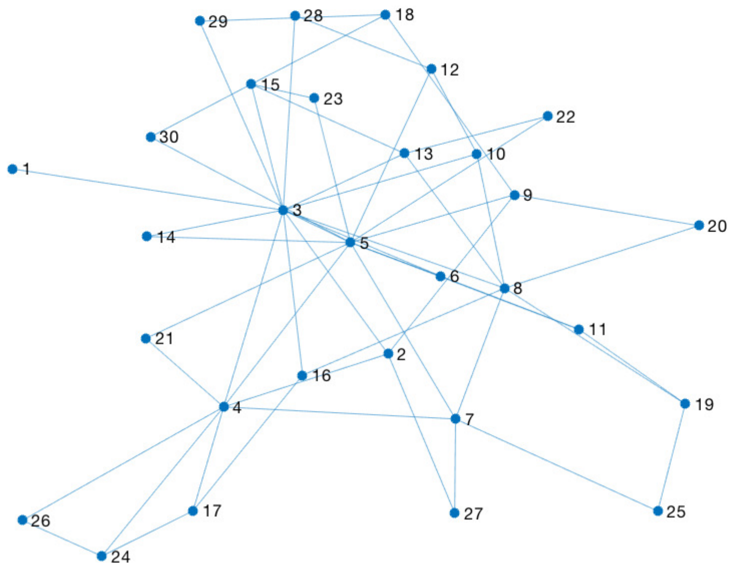


Figure 5.2: A graph with 30 nodes obtained with the Barabási-Albert model.

Test 1: solution time

We considered random instances of Problem (1.3) obtained with the following parameters: $U = 10^5$, $M_A = 0.5$, $M_b = 1$, $L = 4$, using random graphs with a varying number of nodes obtained with the following models.

- The Barabási-Albert model (see [4] for more details), in which each new node is connected to 5 existing nodes.
- The Watts-Strogatz model (see [38]), in which each node is connected to its 2 nearest neighbors and with shortcuts created with a probability of 3 divided by the number of nodes in the graph.
- The Holm and Kim algorithm (see [26]), in which 4 random edges are added for each new node and with a probability of 0.25 of adding an extra random edge generating a triangle.

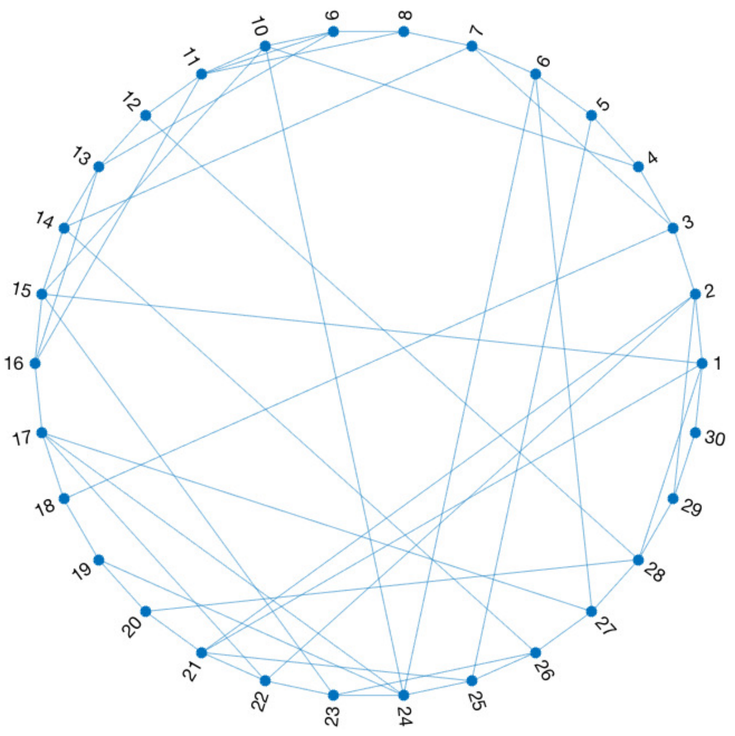


Figure 5.3: A graph with 30 nodes obtained with the Watts-Strogatz model.

Figures 5.5, 5.6 and 5.7 compare the solution times obtained with Algoritm 3 (using different queue policies) to those obtained with Gurobi. The figures refer to random graphs generated with Barabási-Albert model, Watts-Strogatz model and Holm and Kim algorithm, respectively. For each figure, the horizontal axis represents the number of variables (that are logarithmically spaced) and the vertical-axis represents the solution times (also logarithmically spaced), obtained as the average of 5 tests. For each graph type, the policies based on FIFO and node variation appear to be the best performing ones. In particular, for problems obtained from the Barabasi-Albert model (Figure 5.5) and Holm and Kim algorithm (Figure 5.7), the solution time obtained with these two policies are more than three orders of magnitude lower than Gurobi. Moreover, the solution time with FIFO policy is more than one order of magniture lower than Gurobi for problems obtained from Watts-Strogatz model (Figure 5.6). Note that, in every figure, Gurobi solution times are almost constant for small numbers of variables. A possible explanation could be that Gurobi performs some dimension-independent operations which, at small dimensions, are the most time-consuming ones. Note also that, in Figures 5.5 and 5.7, the solution times for node value and LIFO policies are missing starting from a certain number of variables. This is due to excessively high computational times, however, the first collected data points are enough for

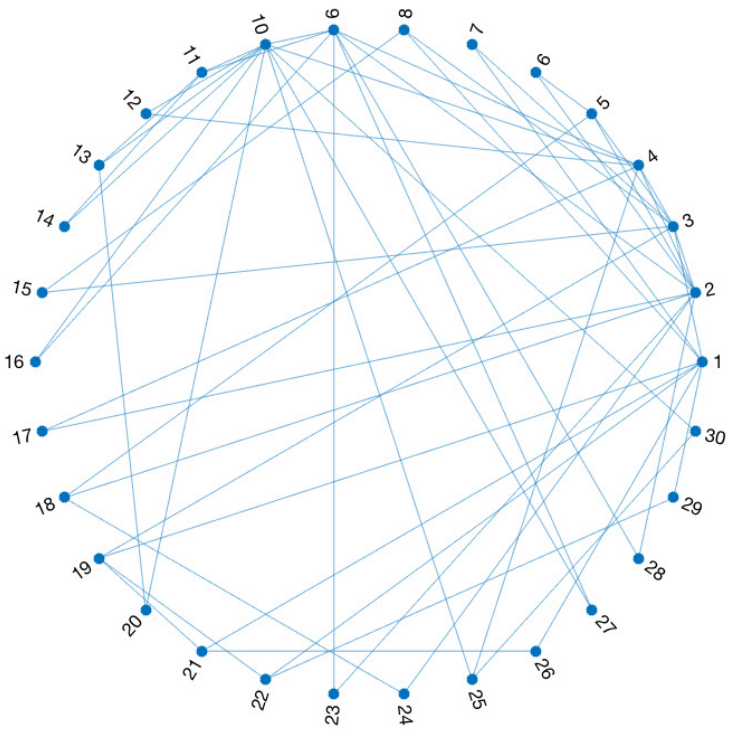


Figure 5.4: A graph with 30 nodes obtained with the Holm and Kim algorithm.

drawing conclusions on the performances of these policies which, as the number of variables grows, perform far worse than Gurobi.

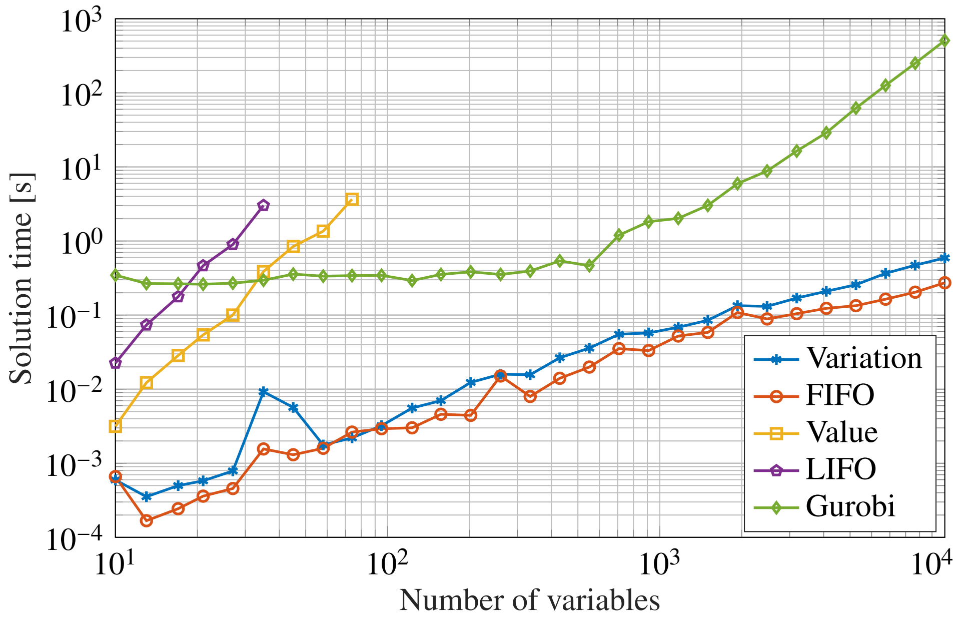


Figure 5.5: Solution times for graphs with growing number of nodes generated with Barabási-Albert model.

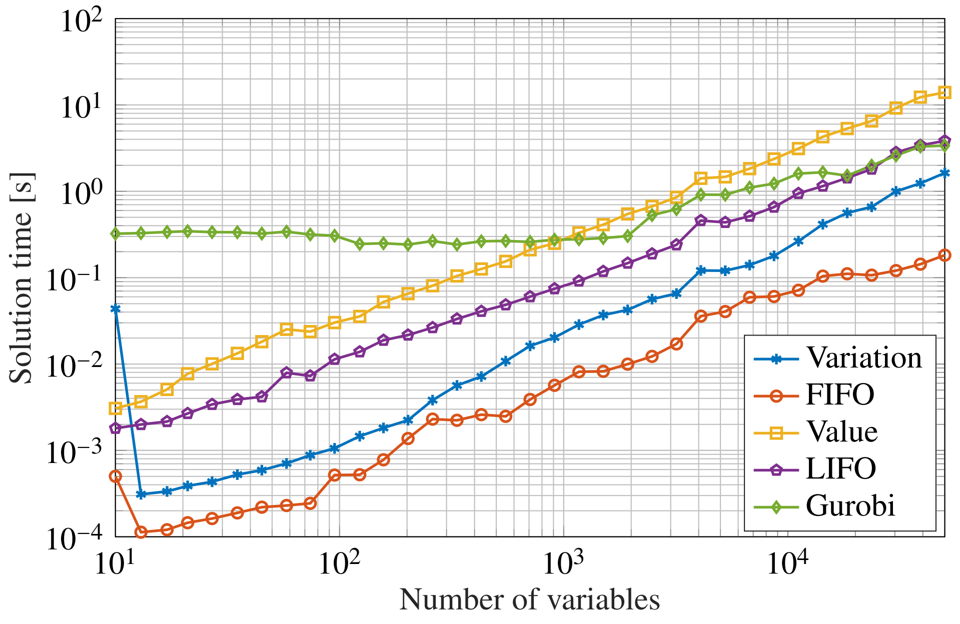


Figure 5.6: Solution times for graphs with growing number of nodes generated with Newman-Watts-Strogatz model.

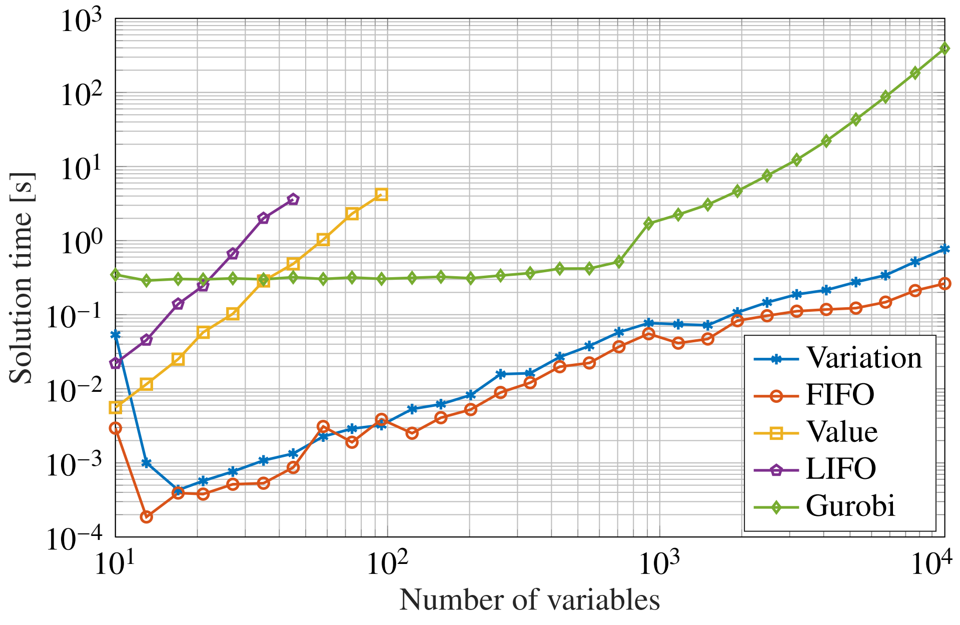


Figure 5.7: Solution times for graphs with growing number of nodes generated with Holm and Kim algorithm.

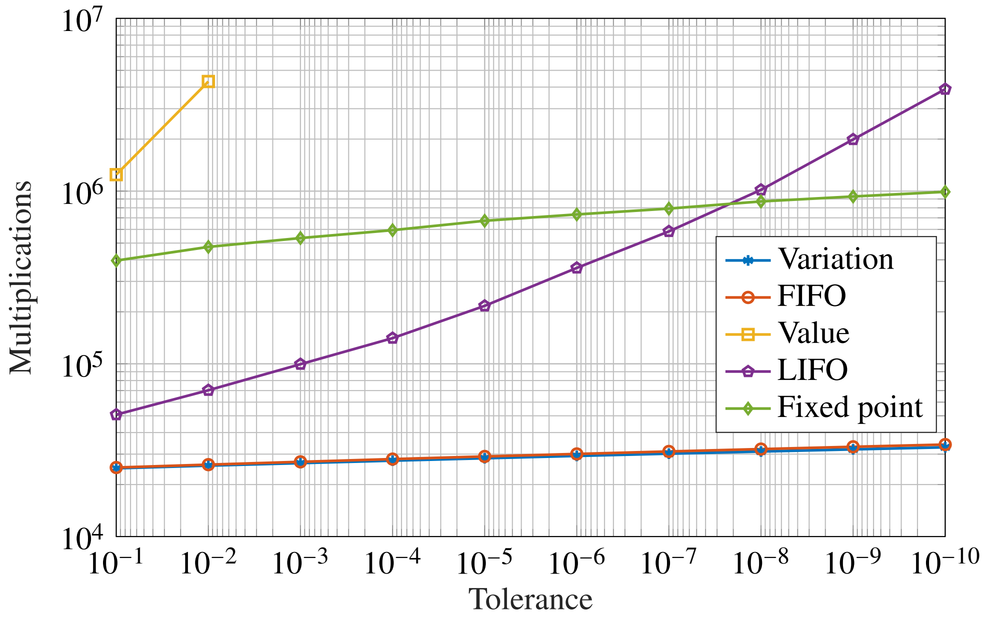


Figure 5.8: Scalar multiplications for different values of tolerance ε on a graph generated with Barabási-Albert model.

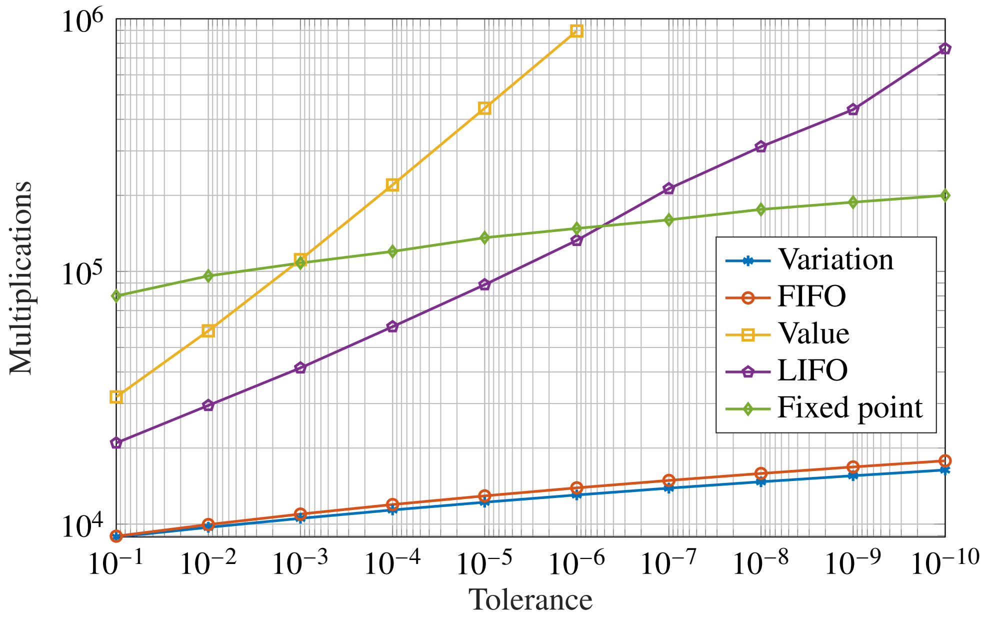


Figure 5.9: Scalar multiplications for different values of tolerance ε on a graph generated with Newman-Watts-Strogatz model.

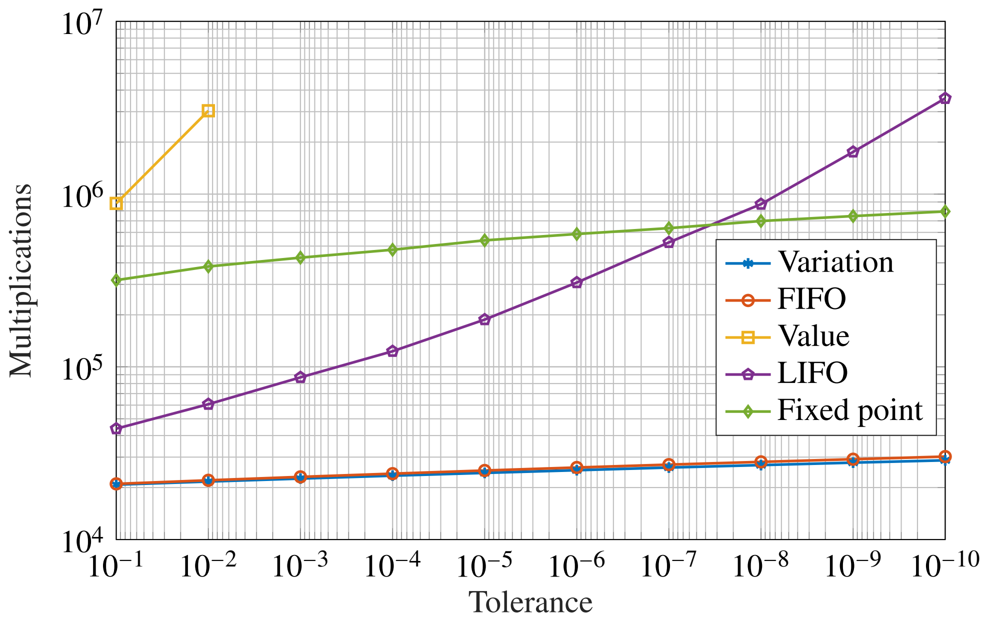


Figure 5.10: Scalar multiplications for different values of tolerance ε on a graph generated with Holm and Kim algorithm.

Test 2: number of operations

We considered three instances of Problem (1.3), obtained from the three classes of random graphs considered in the previous tests, with the same parameters and with 500 nodes. For each instance, we considered 10 logarithmically spaced values of tolerance ε between 10^{-1} and 10^{-10}. We solved each problem with the following methods:

- the preconditioned fixed point iteration (5.15),
- Algorithm 3 with FIFO, LIFO, node value and node variation policies.

The results are reported in Figures 5.8, 5.9 and 5.10. These figures show that the number of product operations required with node variation policy is much lower (of one order of magnitude) than those required by the fixed point iteration (5.14). The iteration based on FIFO, even though slightly less performing than the the node variation policy, also gives comparable results to it. Observe that, even though the iteration based on node variation requires (slightly) less scalar multiplications than the one based on FIFO, its solution times are worse than those obtained with the FIFO policy, since the management of the priority queue based on node variation is computationally more demanding than a First-In-First-Out data structure. The iteration based on nodes value provides poor performances even with high tolerances. Also, the iteration based on LIFO gives poor computational results, underperforming the fixed point iteration (5.14) for tolerances smaller than 10^{-7}, in Figures 5.8 and 5.10, and smaller than 10^{-6}, in Figure 5.9. Note that, in Figures 5.8, 5.9 and 5.10, below a certain value of the tolerance, the numbers of scalar multiplications for the priority queue based on node value are missing due to excessively high computational times. However, the first collected data points are enough for drawing conclusions on the performances of this policy.

As a concluding remark, we observe that all the experiments confirm our previous claim about the relevance of the ordering in the priority queue. While convergence is guaranteed for all the orderings we tested, speed of convergence and number of scalar multiplications turn out to be rather different between them. In what follows we give a tentative explanation of such different performances. The good performance of the node variation policy can be explained with the fact that such policy guarantees a quick reduction of the variables values. The LIFO and value orderings seem to update a small subset of variables before proceeding to update also the other variables. This is particularly evident in the case of the value policy, where only variables with small values are initially updated. The FIFO ordering guarantees a more uniform propagation of the updates, thus avoiding stagnation into small portions of the feasible region.
As a future development we could investigate the performances of probabilistic orderings, of those presented earlier with random perturbations or a combination of these in which the ordering policy changes from one to another cyclically.

Chapter 6

A parallel algorithm for Problem (1.3)

In this chapter, our aim is to provide a parallel procedure that allows to reduce the computational cost of iteration (5.3) in terms of scalar multiplications. When applying iteration (5.3), the value of all nodes is updated even though many nodes values may undergo a negligible variation, hence performing unnecessary multiplications. The main idea is to update only a specific subset of nodes: we will consider the nodes which have undergone a sufficiently large variation in their values in the previous iteration and update in the current one only the neighbors of these nodes.
In this chapter, which is based on [30], first, we introduce the updating procedure and then we present a convergence result. As a closing section, we provide some numerical experiments.

6.1 The consensus iteration

Given the multigraph $\mathbb{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \varphi, \psi)$ associated to Problem 1.3 (recall that, as we saw in Chapter 4, we can think about set of labels $\mathcal{L}$ as a set of controls), and given a tolerance $\varepsilon > 0$, let us define $C_\varepsilon : \mathbb{R}^n_+ \times \mathbb{R}^n_+ \to \mathbb{R}^n_+$ as follows:
$(\forall x, y \in \mathbb{R}^n_+)$ $(\forall i \in \mathcal{V})$

$$[C_\varepsilon(x,y)]_i := \begin{cases} [g(y)]_i, & \text{if } (\exists j \in \mathcal{V})\ i \in \mathcal{N}(j) \text{ and } \left|[y]_j - [x]_j\right| > \varepsilon \\ [y]_i, & \text{otherwise,} \end{cases} \tag{6.1}$$

where g is defined as in (5.11) and $\mathcal{N}(i)$ is defined as in (1.6), for each $i \in \mathcal{V}$. And consider the following sequence

$$\begin{cases} \tilde{x}_\varepsilon(k+2) = C_\varepsilon(\tilde{x}_\varepsilon(k), \tilde{x}_\varepsilon(k+1)) \\ \tilde{x}_\varepsilon(1) = g(x_0) \\ \tilde{x}_\varepsilon(0) = x_0, \end{cases} \tag{6.2}$$

where

$$(\forall i \in \mathcal{V})\ [x_0]_i = \frac{\max_{\ell \in \mathcal{L}} \|b_p\|_\infty}{1-\gamma}, \tag{6.3}$$

and $\gamma < 1$ is the Lipschitz constant of g. Note that this choice of x_0 is such that $(\forall i \in \mathcal{V})\ [x_0]_i \geq [x^\star]_i$, in fact,

$$\|x^\star\|_\infty = \left\| \bigwedge_{\ell \in \mathcal{L}} \{A_\ell x^\star + b_\ell\} \right\|_\infty \leq \left\| \bigvee_{\ell \in \mathcal{L}} A_\ell x^\star \right\|_\infty + \left\| \bigvee_{\ell \in \mathcal{L}} b_\ell \right\|_\infty \leq \gamma \|x^\star\|_\infty + \max_{\ell \in \mathcal{L}} \|b_\ell\|_\infty,$$

from which follows that

$$\|x^\star\|_\infty \leq \frac{\max_{\ell \in \mathcal{L}} \|b_\ell\|_\infty}{1-\gamma}.$$

Sequence (6.2) represents the procedure described earlier in which a node value is updated only if at least one of its direct predecessors value has changed more than the threshold ε. In other words, the procedure can be described as follows:

1. Set $\tilde{x}_\varepsilon(0) := x_0$, $\tilde{x}_\varepsilon(1) := g(x_0)$ and $k = 2$.
2. Set $I := \{i \in \mathcal{V} \mid |[\tilde{x}_\varepsilon(k)]_i - [\tilde{x}_\varepsilon(k-1)]_i| > \varepsilon\}$.
3. If $|I| > 0$, then for all $i \in I$ update $\mathcal{N}(i)$, increment k and go to (2). Otherwise, stop.

In the following, we will show that sequence (6.2) converges to the same limit as (5.3) but requiring a lower number of row-column product operations.
In this section we present the algorithms through which one can compute the approximated fixed point given a certain tolerance $\varepsilon > 0$. Algorithm 1 describes the classical fixed point iteration whilst Algorithm 4 illustrates the consensus iteration (6.1)–(6.2) for reducing the number of row-column product operations. Both algorithms are written in pseudocode.
Now, if we think about $\mathbb{G}$ as a communication network, through iteration (6.2), at each step, we choose a set of nodes which broadcast their value to their direct successors, whilst all other nodes maintain their value unchanged. We will show that the network converges to an equilibrium state that is the solution of (5.3). This method resembles the broadcast-based consensus algorithms presented in works such as [3] and [20]. This is why, in the following, we call (6.2) *consensus iteration*.

Algorithm 4 Consensus Iteration.

```
INPUT: initial vector $x_0$, tolerance $\varepsilon$, matrices $A_\ell$, vectors $b_\ell$ for $\ell \in \mathcal{L}$.
OUTPUT: vector $x$.

$x = x_0$

$x_{\text{old}} := x$
$x := \bigwedge_{\ell \in \mathcal{L}} \{A_\ell x + b_\ell\}$
$\xi := x_{\text{old}} - x$

**repeat**
    $I := \{i \in \mathcal{V} \mid [\xi]_i > \varepsilon\}$
    $\mathcal{N}(I) := \{j \in \mathcal{V} \mid (\exists i \in I)\ j \in \mathcal{N}(i)\}$
    **for** all $j \in \mathcal{N}(I)$ **do**
        $[x_{\text{old}}]_j := [x]_j$
        $[x]_j := \bigwedge_{\ell \in \mathcal{L}} \{[A_\ell]_{j*} x + [b_\ell]_j\}$
        $[\xi]_j := [x_{\text{old}}]_j - [x]_j$
    **end for**
**until** $\|\xi\|_\infty \le \varepsilon$

**return** $x$
```

6.2 Convergence discussion

In the following, we will show that for any $\varepsilon > 0$, iteration (6.2) is convergent to a certain $\tilde{x}^\star_\varepsilon \in \mathbb{R}^n$. Moreover the choice of a smaller ε leads to a value of $\tilde{x}^\star_\varepsilon$ that is closer to the fixed point $x^\star$ of (5.11). More formally, the main result of this chapter is the following one.

Theorem 6.1. *If* $(\forall \ell \in \mathcal{L})\ \|A_\ell\|_\infty \le \gamma$*, with* $\gamma < 1$*, then the sequence of* (6.1)–(6.2) *is such that for any* $\varepsilon > 0$*, the limit*

$$\lim_{k \to \infty} \tilde{x}_\varepsilon(k),$$

exists finite and, setting $\tilde{x}^\star_\varepsilon := \lim_{k \to \infty} \tilde{x}_\varepsilon(k)$*, it holds that*

$$\lim_{\varepsilon \to 0} \tilde{x}^\star_\varepsilon = x^\star. \tag{6.4}$$

Moreover, for all $\varepsilon > 0$*, the following inequality holds*

$$\|\tilde{x}^\star_\varepsilon - x^\star\|_\infty \le \frac{2\varepsilon}{1-\gamma}.$$

In other words, as $\varepsilon \to 0$, $\tilde{x}^\star_\varepsilon$ tends to $x^\star$ and the error norm is bounded from above by $\frac{2\varepsilon}{1-\gamma}$.

Consider iteration (6.1)–(6.2) with $\varepsilon = 0$, namely, $(\forall x, y \in \mathbb{R}^n)$ $(\forall i \in \mathcal{V})$

$$[C(x,y)]_i = \begin{cases} [T(y)]_i, & \text{if } (\exists j \in \mathcal{V}) \; i \in \mathcal{N}(j) \text{ and } \left|[y]_j - [x]_j\right| > 0 \\ [y]_i, & \text{otherwise.} \end{cases} \tag{6.5}$$

$$\begin{cases} \tilde{x}(k+2) = C(\tilde{x}(k), \tilde{x}(k+1)) \\ \tilde{x}(1) = g(x_0) \\ \tilde{x}(0) = x_0. \end{cases} \tag{6.6}$$

Lemma 6.2. *Sequence $\{\tilde{x}(k)\}_{k\in\mathbb{N}}$, defined in* (6.6)*, is bounded from below by $\{x(k)\}_{k\in\mathbb{N}}$, defined in* (5.3)*, that is, $(\forall k \in \mathbb{N})$ $\tilde{x}(k) \geq x(k)$.*

Proof. By the monotonicity of $\{x(k)\}_{k\in\mathbb{N}}$ and the definition of iteration (6.5)–(6.6), it follow immediately that $(\forall k \in \mathbb{N})$ $(\forall i \in \mathcal{V})$ $[\tilde{x}(k)]_i \geq [x(k)]_i$. □

Since $(\forall k \in \mathbb{N})$ $x(k) \geq x^\star$, by Lemma 6.2 we have that $(\forall k \in \mathbb{N})$ $\tilde{x}(k) \geq x^\star$.

Remark 6.3. $(\forall x \in \mathbb{R}^n)$ $x = C(x,x)$.

Since $\{\tilde{x}(k)\}_{k\in\mathbb{N}}$ is non-increasing and bounded from below by $x^\star$, it has to be convergent. Let $\tilde{x}^\star := \lim\limits_{k\to\infty} \tilde{x}(k)$, what we would like to ensure is that $\tilde{x}^\star = x^\star$. Actually, we are about to state a stronger fact.

Proposition 6.4. *Sequences $\{\tilde{x}(k)\}_{k\in\mathbb{N}}$ and $\{x(k)\}_{k\in\mathbb{N}}$ coincide, that is, $(\forall k \in \mathbb{N})$ $\tilde{x}(k) = x(k)$.*

Proof. Let us prove the result by induction.

- Base case: $\tilde{x}(0) = x(0)$ and $\tilde{x}(1) = x(1)$ by definition.
- Inductive step: let $p \in \mathbb{N}$ be such that

  $$(\forall k \in \mathbb{N}) \; k \leq p \Rightarrow \tilde{x}(k) = x(k),$$

  let us prove that $\tilde{x}(p+1) = x(p+1)$.
  By contradiction, let assume that

  $$\tilde{x}(p+1) \neq x(p+1), \tag{6.7}$$

  then, by Lemma 6.2,

  $$(\exists i \in \mathcal{V}) \; [\tilde{x}(p+1)]_i > [x(p+1)]_i.$$

This means that $[\tilde{x}(p+1)]_i = [\tilde{x}(p)]_i = [x(p)]_i$, that is,

$$(\forall j \in \mathcal{V})\ i \in \mathcal{N}(j) \Rightarrow [\tilde{x}(p)]_j = [\tilde{x}(p-1)]_j.$$

But this implies, by inductive hypothesis, that

$$(\forall j \in \mathcal{V})\ i \in \mathcal{N}(j) \Rightarrow [x(p)]_j = [x(p-1)]_j.$$

So

$$\begin{aligned}
[x(p+1)]_i = [g(x(p))]_i = \left[\bigwedge_{\ell\in\mathcal{L}} \{A_\ell x(p) + b_\ell\}\right]_i &= \\
= \min_{\ell\in\mathcal{L}} \left\{ \sum_{j\in\mathcal{N}^{-1}(i)} [A_\ell]_{ij}[x(p)]_j + [b_\ell]_i \right\} &= \\
= \min_{\ell\in\mathcal{L}} \left\{ \sum_{j\in\mathcal{N}^{-1}(i)} [A_\ell]_{ij}[x(p-1)]_j + [b_\ell]_i \right\} &= \\
= [g(x(p-1))]_i = [x(p)]_i.&
\end{aligned}$$

This reasoning can be applied to every $i \in \mathcal{V}$ such that $[\tilde{x}(p+1)]_i > [x(p+1)]_i$, and what we obtain is that $\tilde{x}(p+1) = x(p+1)$, which goes against hypothesis (6.7) $\lightning$. Since assumption (6.7) led to a contradiction, it must be $\tilde{x}(p+1) = x(p+1)$.

Given the arbitrariness of p the thesis holds true for all $p \in \mathbb{N}$. □

The result stated in Proposition 6.4 means that, for the initial condition x_0 as in (6.3), iteration (6.6) converges to the solution of problem (5.11). Let us now consider the family

$$\{C_\varepsilon : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n\}_{\varepsilon>0}$$

of functions defined as in (6.1) depending on parameter $\varepsilon > 0$.

Lemma 6.5. *Given sequence $\{\tilde{x}_\varepsilon(k)\}_{k\in\mathbb{N}}$ as defined in* (6.2)*, it holds that* $(\forall \varepsilon > 0)$ $(\forall k \in \mathbb{N})$ $\tilde{x}_\varepsilon(k) \geq x_\varepsilon(k)$*, where $\{x_\varepsilon(k)\}_{k\in\mathbb{N}}$ is defined as follows:*

$$\begin{cases} x_\varepsilon(k) = x(k), & \text{for } k = 1,2, \text{ or if } (k>2 \text{ and } \|x(k-1) - x(k-2)\| > \varepsilon) \\ x_\varepsilon(k) = x_\varepsilon(k-1), & \text{otherwise.} \end{cases}$$

Proof. The thesis of Lemma 6.5 follows by the same reasoning made for proving Lemma 6.2 applied to sequences $\{\tilde{x}_\varepsilon(k)\}_{k\in\mathbb{N}}$ and $\{x_\varepsilon(k)\}_{k\in\mathbb{N}}$. □

Since $(\exists k \in \mathbb{N})$ $(\exists x_\varepsilon^\star \in \mathbb{R}_+^n)$ $x_\varepsilon(k) = x_\varepsilon^\star$, sequence $\{\tilde{x}_\varepsilon(k)\}_{k\in\mathbb{N}}$ is non-increasing and bounded from below by $x_\varepsilon^\star$, so it has to be convergent.

Proof of Theorem 6.1. In order to show (6.4) we need to provide an upper bound over the distance between sequences $\{\tilde{x}_\varepsilon(k)\}_{k\in\mathbb{N}}$ and $\{x_\varepsilon(k)\}_{k\in\mathbb{N}}$. By definition, for $k \in \{0,1\}$, we know that $\tilde{x}_\varepsilon(k) = x_\varepsilon(k)$. Now let $\bar{n} \in \mathbb{N}$ be such that

$$\tilde{x}_\varepsilon(\bar{n}) \neq x_\varepsilon(\bar{n}) \text{ and } (\forall m \in \mathbb{N})\ m < \bar{n} \Rightarrow \tilde{x}_\varepsilon(m) = x_\varepsilon(m).$$

By Lemma 6.5,

$$\mathcal{C} := \{i \in \mathcal{V} \mid [\tilde{x}_\varepsilon(\bar{n})]_i > [x_\varepsilon(\bar{n})]_i\} \neq \varnothing.$$

This means that

$$(\forall i \in \mathcal{C})\ (\forall j \in \mathcal{V})\ i \in \mathcal{N}(j) \Rightarrow \left|[\tilde{x}_\varepsilon(\bar{n}-1)]_j - [\tilde{x}_\varepsilon(\bar{n}-2)]_j\right| < \varepsilon.$$

Now, since

$$(\forall \ell \in \mathcal{L})\ (\forall i \in \mathcal{V})\ \sum_{j=1}^{n} [A_\ell]_{ij} \leq \gamma,$$

it holds that $\|\tilde{x}_\varepsilon(\bar{n}) - x_\varepsilon(\bar{n})\| = \|C_\varepsilon(x_\varepsilon(\bar{n}-2), x_\varepsilon(\bar{n}-1)) - g(x_\varepsilon(\bar{n}-1))\| \leq \gamma\varepsilon =: E(0)$. This reasoning applies to any $\tilde{x}_\varepsilon(m)$, with $m \in \mathbb{N}$ such that $m \geq \bar{n}$, when estimating

$$\|C_\varepsilon(\tilde{x}_\varepsilon(m-1), \tilde{x}_\varepsilon(m)) - g(\tilde{x}_\varepsilon(m))\|.$$

Let us define

$$(\forall m \in \mathbb{N})\ E(m+1) := E(0) + \gamma E(m) = \sum_{p=0}^{m+1} \gamma^p E(0) = \sum_{p=1}^{m+2} \gamma^p \varepsilon.$$

If we now consider step $\bar{n}+1$, we have:

$$\begin{aligned}
\|\tilde{x}_\varepsilon(\bar{n}+1) - x_\varepsilon(\bar{n}+1)\| &= \|\tilde{x}_\varepsilon(\bar{n}+1) - g(\tilde{x}_\varepsilon(\bar{n})) + g(\tilde{x}_\varepsilon(\bar{n})) - x_\varepsilon(\bar{n}+1)\| \leq \\
&\leq \|\tilde{x}_\varepsilon(\bar{n}+1) - g(\tilde{x}_\varepsilon(\bar{n}))\| + \|g(\tilde{x}_\varepsilon(\bar{n})) - x_\varepsilon(\bar{n}+1)\| = \\
&= \|C_\varepsilon(\tilde{x}_\varepsilon(\bar{n}-1), \tilde{x}_\varepsilon(\bar{n})) - g(\tilde{x}_\varepsilon(\bar{n}))\| + \\
&\quad + \|g(\tilde{x}_\varepsilon(\bar{n})) - g(x_\varepsilon(\bar{n}))\| \leq \\
&\leq E(0) + \gamma E(0) = E(1).
\end{aligned}$$

And, for any $m \in \mathbb{N}$, it holds that

$$\begin{aligned}
\|\tilde{x}_\varepsilon(\bar{n}+m+1) - x_\varepsilon(\bar{n}+m+1)\| &\leq \|C_\varepsilon(\tilde{x}_\varepsilon(\bar{n}+m-1), \tilde{x}_\varepsilon(\bar{n}+m)) - g(\tilde{x}_\varepsilon(\bar{n}+m))\| + \\
&\quad + \|g(\tilde{x}_\varepsilon(\bar{n}+m)) - g(x_\varepsilon(\bar{n}+m))\| \leq \\
&\leq E(0) + \gamma E(m) = E(m+1).
\end{aligned}$$

However, we know that $\{\tilde{x}_\varepsilon(k)\}_{k\in\mathbb{N}}$ is stationary, in fact, if it were not like that, there would exists a node $i \in \mathcal{V}$ which updates its value an infinite number of times.

Now, letting $d := |[\tilde{x}_\varepsilon(\bar{n}-1)]_i - [x^\star_\varepsilon]_i|$, after at most $\lfloor d/\varepsilon \rfloor + 1$ updatings of i, that component would be smaller that the i-th component of $x^\star_\varepsilon$, which contradicts Lemma 6.5. So, since

$$(\forall \varepsilon > 0)\ (\exists \tilde{n} = \tilde{n}(\varepsilon) \in \mathbb{N})\ (\forall m \in \mathbb{N})\ m \geq \tilde{n} \Rightarrow (\tilde{x}_\varepsilon(m) = \tilde{x}^\star_\varepsilon \text{ and } x_\varepsilon(m) = x^\star_\varepsilon),$$

if we let $\tilde{m} := \tilde{n} - \bar{n}$, it follows that

$$\|\tilde{x}_\varepsilon(\tilde{n}) - x_\varepsilon(\tilde{n})\| \leq E(\tilde{m}) = \sum_{p=1}^{\tilde{m}+1} \gamma^p \varepsilon \leq \sum_{p=0}^{\infty} \gamma^p \varepsilon = \frac{\varepsilon}{1-\gamma}.$$

This means that

$$(\forall \varepsilon > 0)\ \|\tilde{x}^\star_\varepsilon - x^\star_\varepsilon\| \leq c\varepsilon,$$

with $c := \frac{1}{1-\gamma}$. Moreover, $\lim_{\varepsilon \to 0} x^\star_\varepsilon = x^\star$ and for any $\varepsilon > 0$ it holds that $\|x^\star_\varepsilon - x^\star\| \leq c\varepsilon$, so we can conclude that

$$\lim_{\varepsilon \to 0} \tilde{x}^\star_\varepsilon = x^\star$$

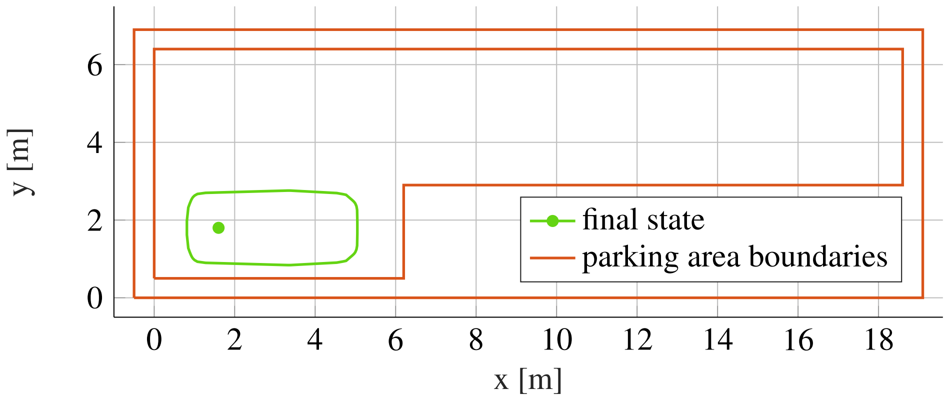
and

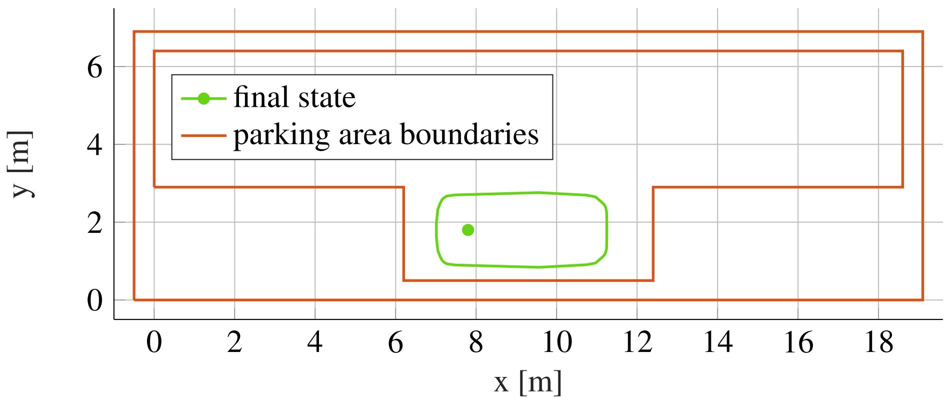
$$\|\tilde{x}^\star_\varepsilon - x^\star\| \leq \|\tilde{x}^\star_\varepsilon - x^\star_\varepsilon\| + \|x^\star_\varepsilon - x^\star\| \leq 2c\varepsilon. \tag{6.8}$$

□

6.3 Numerical experiments

We considered three instances of Problem (4.10) for the car-like model introduced in 4.2, using the finite state automaton presented in Section 4.2.2, obtained from three parking scenarios represented in Figure 6.1, in which a maximum of 10 maneuvers is allowed. The state space of the car-like vehicle is given by $\Omega = [-0.5, 19.5] \times [0, 7] \times [0, 2\pi)$ and the average number of variables used for discretizing Ω associated to these problems is of the order of $7 \cdot 10^4$.





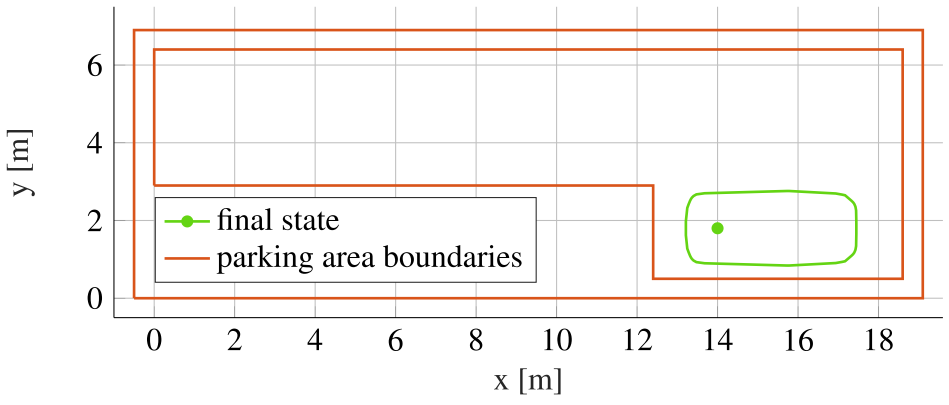


Figure 6.1: Parking scenarios.

For each instance, 10 logarithmically spaced values of tolerance ε between 10^{-1} and 10^{-10} were considered. Each instance has been solved with the following methods:

- the preconditioned fixed point iteration (5.15),
- Algorithm 4 (the consensus iteration (6.2)) after preconditioning.

The results are given in Figures 6.2, 6.3 and 6.4. Figure 6.2 shows that the mean number of product operations required by the consensus iteration (6.2) with preconditioning is on average $30,8\%$ smaller than the one required by the preconditioned fixed point iteration (5.15). Figure 6.3 shows the mean infinity norm of the error given by the iterations used for solving the problem, as expected, the infinity norm associated to the consensus iteration is on average one order of magnitude greater that the one associated to the preconditioned fixed point iteration. Note that the error is computed considering the solution obtained from the fixed point iteration with a precision of 10^{-16} as the exact one. Figure 6.4 compares the theoretical upper bound obtained in (6.8) with the infinity norm of the consensus iteration error for different values of tolerance ε confirming the theoretical result. Note that, out of the three instances of Problem (4.10) considered, the results shown in Figure 6.4 refer to only one of these, in particular the first one, since (6.8) involves the Lipschitz constant of g which is obviously problem dependent.

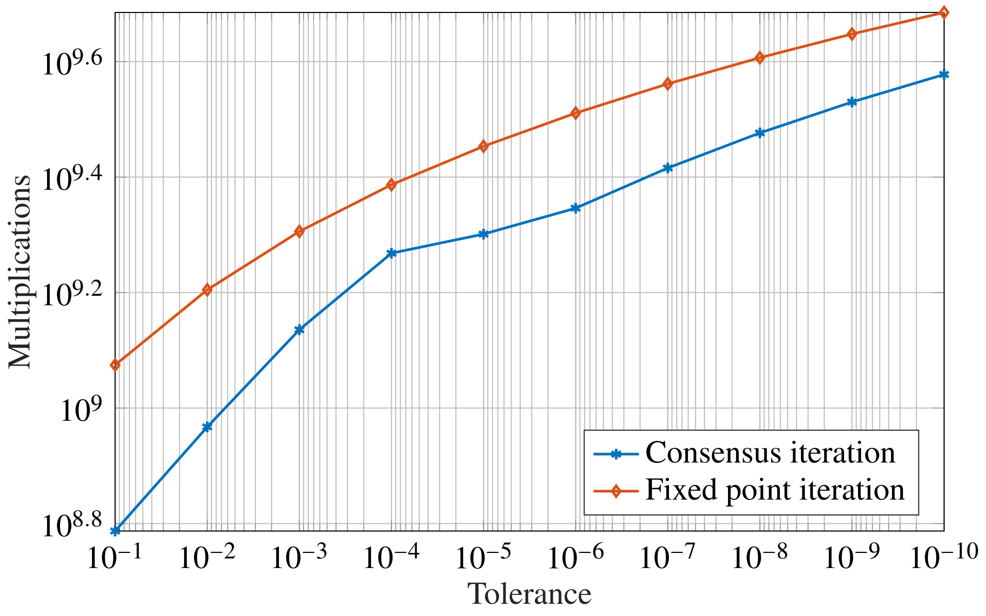


Figure 6.2: Scalar multiplications for different values of tolerance ε.

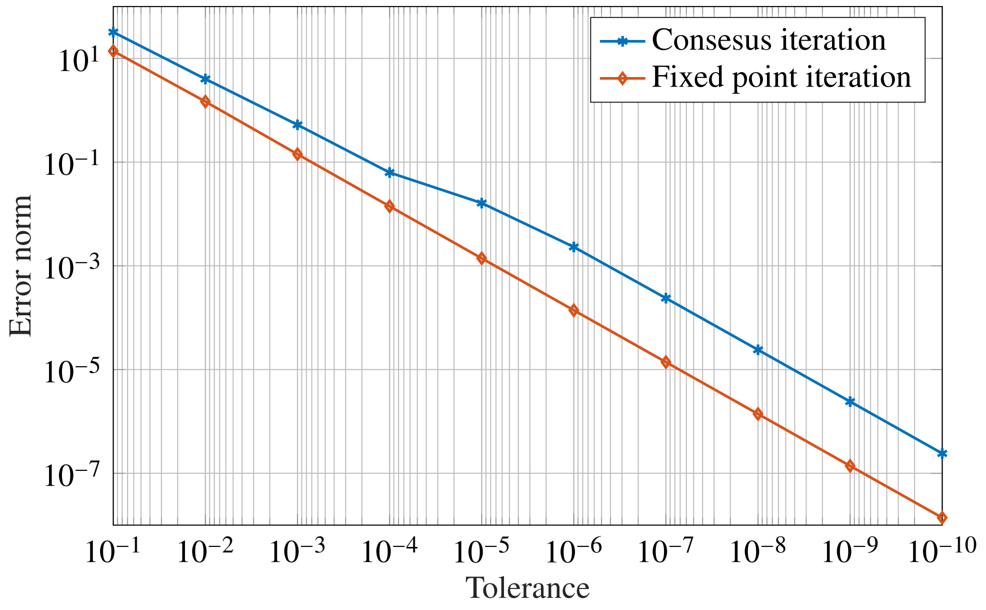


Figure 6.3: Errors in infinity norm for different values of tolerance ε.

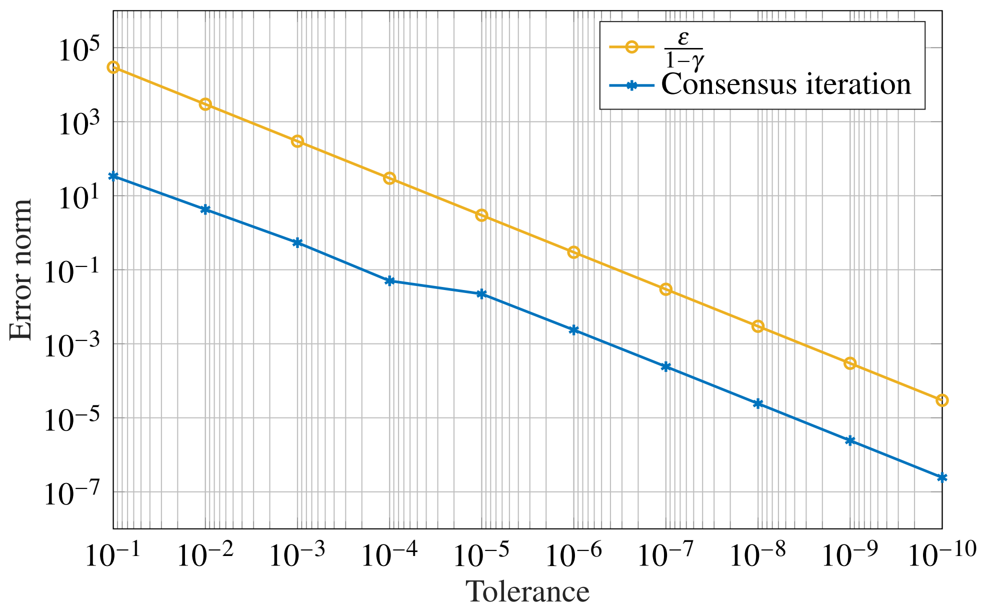


Figure 6.4: Consensus error in infinity norm for different values of tolerance ε compared to the the theoretical upper bound.

As a final observation, we note that the experiments we carried out highlight the benefit of iteration (5.4) at reducing the computational cost of iteration (6.2) by avoiding unnecessary updates.

Chapter 7

Speed planning for autonomous vehicles in continuous-time domain

In this chapter we address a speed planning problem similar to the one introduced in Section 3.1, but we will solve it in the continuous time domain. In particular, after having introduced the problem in Section 7.1 and discussed the contribution presented in this chapter, a feasibility condition and an operator for computing the optimal solution (Theorem 7.2) are presented in Section 7.3, along with some numerical examples given in Section 7.4. In conclusion, Section 7.5 provides the proof of the main result.
The results presented in this chapter are based on [13] and they are a development of those given in [15, 16].

7.1 Minimum-time velocity planning

Let us consider the following problem

$$\min_{v \in W^{1,\infty}([0,s_f])} \int_0^{s_f} v^{-1}(s) ds \tag{7.1a}$$

$$v^-(s) \le v(s) \le v^+(s), \qquad s \in [0, s_f], \tag{7.1b}$$

$$\alpha^-(s) \le 2v'(s)v(s) \le \alpha^+(s), \qquad s \in [0, s_f], \tag{7.1c}$$

$$|k(s)| v(s)^2 \le \beta(s), \qquad s \in [0, s_f], \tag{7.1d}$$

where $W^{1,\infty}\left(\left[0, s_f\right]\right)$ is the Sobolev space on $\left[0, s_f\right]$ of functions with first weak derivative in $L^\infty\left(\left[0, s_f\right]\right)$ (see Definition2.28). Here, v^-, v^+, α^-, α^+ are assigned functions, with v^-, v^+ non-negative. The objective function (7.1a) is the total maneuver time and constraints (7.1b), (7.1c), (7.1d) limit velocity and the tangential and normal components of acceleration.

Problem (7.1) belongs to a class of problems that includes optimal velocity planning for manipulators and has the form

$$\begin{gathered}\min_{v \in W^{1,\infty}([0,s_f])} \int_0^{s_f} v^{-1}(s)ds \\ a_i(s)\dot{v}(s)v(s) + b_i(s)v(s)^2 + c_i(s) \leq 0, \quad i \in \{1,\dots,m\},\end{gathered} \tag{7.2}$$

where m is the number of constraints and a_i, b_i, c_i are assigned functions. It is clear that Problem (7.1) is a special case of Problem (7.2), obtained for a specific choice of a_i, b_i, c_i.

References [15, 16] present an algorithm, with linear-time computational complexity with respect to the number of variables, that provides an optimal solution of (7.1) after spatial discretization. Namely, the path is divided into n intervals of equal length and Problem (7.1) is approximated with a finite dimensional one in which the derivative of v is substituted with a finite difference approximation.

In this chapter, we compute directly the exact continuous-time solution of Problem (7.1) without performing a finite-dimensional reduction. The main result of the chapter is presented in Theorem 7.2. It gives a sufficient and necessary condition for the feasibility of Problem (7.1) and presents its optimal solution, which is computed as the pointwise minimum of the solutions of two ODEs.

The proposed method presents some resemblances with the method of "numerical integration", introduced for problems of class (7.2). For instance, [43] proposes a method, based on the identification of "characteristic switching points" in which the maximum velocity is attained. This simplifies the calculation of the optimal velocity profile. A related algorithm is presented in [29]. Recent paper [40] presents various properties of numerical integration methods. Anyway, the method we propose is simpler and more efficient since it leverages the special structure of Problem (7.1) with respect to the more general problem (7.2).

With respect to existing literature, the new contributions provided in what follows are the undermentioned ones:

- A necessary and sufficient condition for the feasibility of Problem (7.1) is presented (see part *i.* of Theorem 7.2).
- A simple operator, based on the solution of two ordinary differential equations, that computes the optimal solution is proposed (see part *ii.* of Theorem 7.2).

Note that these results correspond to the generalization to the continuous-time case of the results presented in [16] for the spatially-discretized version of Problem (7.1). In fact, the work presented in this chapter shares some of its fundamental ideas with [16]. Namely:

- The feasible set of Problem (7.1) has the algebraic structure of a lattice, if equipped with the operations of pointwise minimum and maximum.
- The optimal solution of Problem (7.1) corresponds to the supremal element of this lattice.
- The optimal solution of Problem (7.1) is obtained with a projection operation and its optimality is proven by the Knaster-Tarski Fixpoint Theorem.

However, solving the problem in a function space requires various nontrivial technical extensions to the proofs presented in [16].

7.2 Problem formulation

Let $\gamma : [0, s_f] \to \mathbb{R}^2$ be a C^2 function such that $(\forall \lambda \in [0, s_f])\ \|\gamma'(\lambda)\| = 1$. The image set $\gamma([0, s_f])$ represents the path followed by a vehicle, $\gamma(0)$ the initial configuration and $\gamma(s_f)$ the final one. We want to compute the speed-law that minimizes the overall transfer time while satisfying some kinematic and dynamic requirements. To this end, let $\lambda : [0, t_f] \to [0, s_f]$ be a differentiable monotone increasing function that represents the vehicle position as a function of time and let $v : [0, s_f] \to [0, +\infty]$ be such that, $(\forall t \in [0, t_f])\ \dot{\lambda}(t) = v(\lambda(t))$. In this way, $v(s)$ is the vehicle velocity at position s. The position of the vehicle as a function of time is given by $x : [0, t_f] \to \mathbb{R}^2$, $x(t) = \gamma(\lambda(t))$, the velocity and acceleration are given by

$$\begin{aligned} \dot{x}(t) &= \gamma'(\lambda(t))v(\lambda(t)), \\ \ddot{x}(t) &= a_L(t)\gamma'(\lambda(t)) + a_N(t)\gamma'^{\perp}(\lambda(t)), \end{aligned}$$

where $a_L(t) = v'(\lambda(t))v(\lambda(t))$ and $a_N(t)(t) = k(\lambda(t))v(\lambda(t))^2$ are, respectively, the longitudinal and normal components of acceleration. Here $k : [0, s_f] \to \mathbb{R}$ is the scalar curvature, defined as $k(s) = \langle \gamma''(s), \gamma'(s)^{\perp} \rangle$.

We require to travel the distance s_f in minimum-time while satisfying constraints on the vehicle velocity and on its longitudinal and normal acceleration. The minimum-time problem can be approached by searching a velocity profile v which is the solution of Problem (7.1).

Let $f, g : I \to \mathbb{R}$, define $f \wedge g$, $f \vee g$, as, respectively, the pointwise minimum and maximum operations; moreover let us define the partial order $\leq$ as follows

$$f \leq g \iff (\forall x \in I)\ f(x) \leq g(x).$$

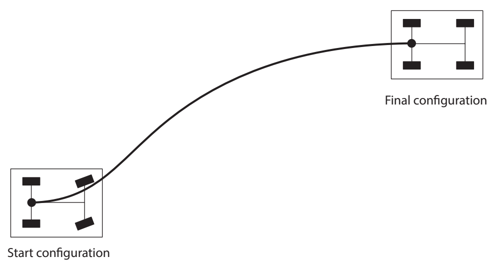


Figure 7.1: A path to follow for an autonomous car-like vehicle.

It is convenient to make the change of variables $w = v^2$ (see also [47]), so that Problem (7.1) takes on the form

$$\min_{w\in W^{1,\infty}([0,s_f])} \int_0^{s_f} w(s)^{-\frac{1}{2}} ds \tag{7.3a}$$

$$\begin{aligned} &\mu^-(s) < w(s) \le \mu^+(s), && s\in[0,s_f],\\ &\alpha^-(s) \le w'(s) \le \alpha^+(s), && s\in[0,s_f], \end{aligned}$$

where

$$\mu^+(s) = v^+(s)^2 \wedge \frac{\beta(s)}{k(s)}, \qquad \mu^-(s) = v^-(s)^2 \tag{7.4}$$

represent the upper bound of w, (depending on the velocity bound v^+ and the curvature k) and the lower bound of w, respectively.
In what follows, we actually address the following problem, which is slightly more general than (7.3),

$$\min_{w\in W^{1,\infty}([0,s_f])} \quad \Psi(w)$$

$$\mu^-(s) < w(s) \le \mu^+(s), \qquad s\in[0,s_f], \tag{7.5a}$$

$$\alpha^-(s) \le w'(s) \le \alpha^+(s), \qquad s\in[0,s_f], \tag{7.5b}$$

where $\Psi : W^{1,\infty}([0,s_f]) \to \mathbb{R}$ is order reversing (i.e., $\left(\forall v,w \in W^{1,\infty}([0,s_f])\right)$ $v \ge w \Rightarrow \Psi(v) \le \Psi(w)$) and μ^-, μ^+, α^-, $\alpha^+ \in L^\infty([0,s_f])$ are assigned functions with $\mu^-, \alpha^+ \ge 0$, $\alpha^- \le 0$. Note that the objective function (7.3a) is order reversing, so that Problem (7.3) has the form (7.5). Consider the following:

Definition 7.1. Let $\mathcal{Q}$ be the subset of $W^{1,\infty}([0,s_f,])$ such that $\mu \in \mathcal{Q}$ if $\operatorname{sgn}(\mu' - \alpha^+)$ and $\operatorname{sgn}(\mu' - \alpha^-)$ are Riemann integrable (i.e., in view of the boundedness of the sgn function, a. e. continuous), where $\operatorname{sgn} : \mathbb{R} \to \{-1,0,1\}$ is defined as

$$\operatorname{sgn}(x) = \begin{cases} 1, & \text{if } x > 0\\ 0, & \text{if } x = 0\\ -1, & \text{if } x < 0. \end{cases}$$

7.3 A feasibility condition and the optimal solution

Define the *forward operator* $F : \mathcal{Q} \to W^{1,\infty}([0,s_f])$ such that $F(\mu) = \phi$, where ϕ is the solution of the following differential equation

$$\begin{cases} \phi'(x) = f(x,\phi) = \begin{cases} \alpha^+(x) \wedge \mu'(x), & \text{if } \phi(x) \geq \mu(x) \\ \alpha^+(x), & \text{if } \phi(x) < \mu(x) \end{cases} \\ \phi(0) = \mu(0). \end{cases} \tag{7.6}$$

Note that the solution of (7.6) exists and is unique by Theorem 2.32, since function f is bounded on $[0,s_f]$, the subset of $[0,s_f] \times \mathbb{R}$ in which f is discontinuous has measure zero and,

$$(\forall x \in [0,s_f])\ (\forall u,y \in \mathbb{R})\ (u-y)(f(x,u) - f(x,y)) \leq 0.$$

Conversely, define the *backward operator* $B : \mathcal{Q} \to W^{1,\infty}([0,s_f])$, such that $B(\mu) = \phi$, where ϕ is the solution of

$$\begin{cases} \phi'(x) = \begin{cases} \alpha^-(x) \vee \mu'(x), & \text{if } \phi(x) \geq \mu(x) \\ \alpha^-(x), & \text{if } \phi(x) < \mu(x) \end{cases} \\ \phi(s_f) = \mu(s_f), \end{cases} \tag{7.7}$$

whose existence and uniqueness hold for the same reasons as (7.6).
Finally, define the *meet operator* $M : \mathcal{Q} \to W^{1,\infty}([0,s_f])$ as

$$M(\mu) = F(\mu) \wedge B(\mu). \tag{7.8}$$

We claim that the *meet operator* M allows checking the feasibility of Problem (7.5) and that, in case Problem (7.5) is feasible, function $v^* = M(\mu^+)$ represents its optimal solution.
These statements are formalized in the following theorem which constitutes the main result of this chapter.

Theorem 7.2. *Let $\mu^+ \in \mathcal{Q}$, then the following statements hold:*

i. Problem (7.5) *is feasible if and only if function $w^* = M(\mu^+)$ satisfies*

$$w^* \geq \mu^-.$$

ii. If Problem (7.5) *is feasible, then function $w^* = M(\mu^+)$ is its optimal solution.*

Proofs of the results. Part *i.* follows from Proposition 7.11, part *ii.* follows from Proposition 7.12 (see Section 7.5).

7.4 Numerical examples

7.4.1 Example 1

As a first example consider the path shown in Figure 7.3, whose curvature is defined as

$$k(s) = \begin{cases} 0, & \text{if } s \in [0, l_1) \\ k_\tau(s), & \text{if } s \in [l_1, l_2] \\ 1/R, & \text{if } s \in (l_2, l_3) \\ k_\tau(s), & \text{if } s \in [l_3, l_4] \\ 0, & \text{if } s \in (l_4, s_f] \end{cases} \tag{7.9}$$

where $k_\tau(s)$ is the 6-th degree Hermite polynomial used to guarantee the following interpolation conditions:

$$\begin{aligned} & k(l_1) = k(l_4) = 0, \\ & k(l_2) = k(l_3) = 1/R, \\ & k'(l_i) = k''(l_i) = 0, \qquad i \in \{1, \dots, 4\}. \end{aligned}$$

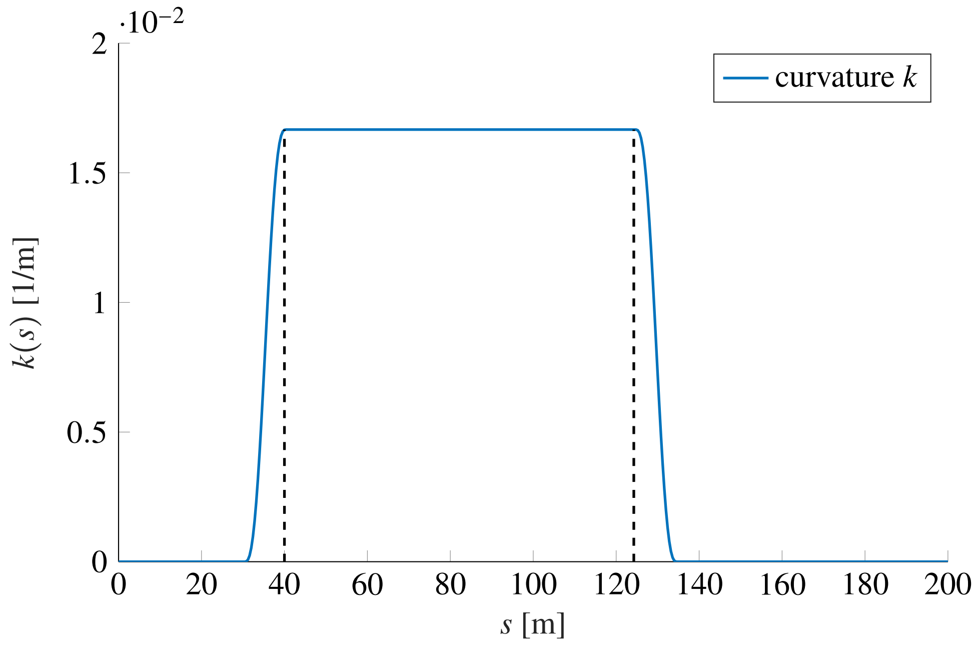


Figure 7.2: The curvature function k in (7.9) of the curve discussed in the first example.

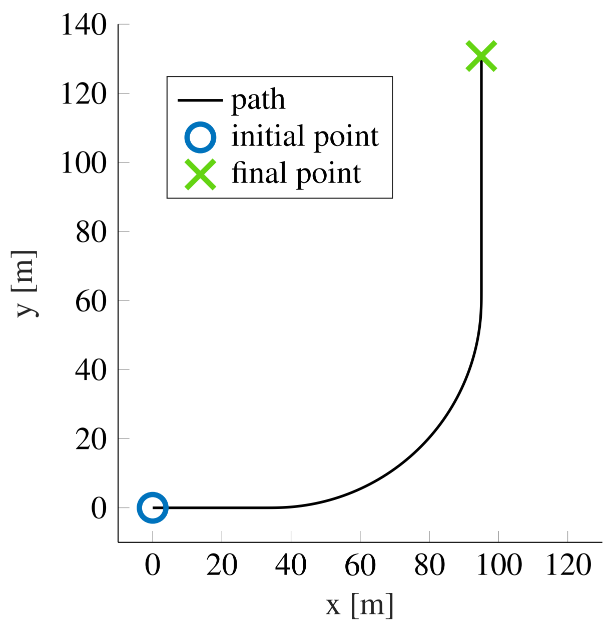


Figure 7.3: The black line represents the path, the red circle and the black cross represent the starting point and, respectively, the end point.

In this example, the total length is $s_f = 200$ and the minimum-time velocity planning problem is addressed with $[l_1, l_2, l_3, l_4] = [30,\ 40,\ 124.2478,\ 134.2478]$, $R = 60$. The velocity bounds v^+ and v^- are set as follows: $v^+(0) = v^-(0) = 0$, $v^+(s_f) = v^-(s_f) = 22$, while, for each $s \in (0, s_f)$, $v^-(s) = 0$ and $v^+(s) = 36.1$. The longitudinal acceleration limits are $\alpha^- = -10.5$ and $\alpha^+ = 4$, and the maximal normal acceleration is $\beta = 7$.
The following results are obtained by numerically solving equations (7.6), (7.7) with a standart Runge-Kutta 45 integration scheme. Figure 7.4 shows the upper-bound function μ^+ obtained by (7.4) and the corresponding functions $F(u)$ and $B(u)$ computed as the solution of equations (7.6) and (7.7), respectively. Figure 7.5 shows the optimal solution w^* obtained with (7.8). In this example, the vehicle starts with zero velocity and accelerates to the upper bound. Then, it follows the velocity bound in order to respect the maximum velocity constraint due to the lateral acceleration on the curve. After that, at the end of the constant bound, the vehicle accelerates and reaches a second local maximum velocity after which it decelerates quickly in order to reach the final velocity $v^+(s_f)$.

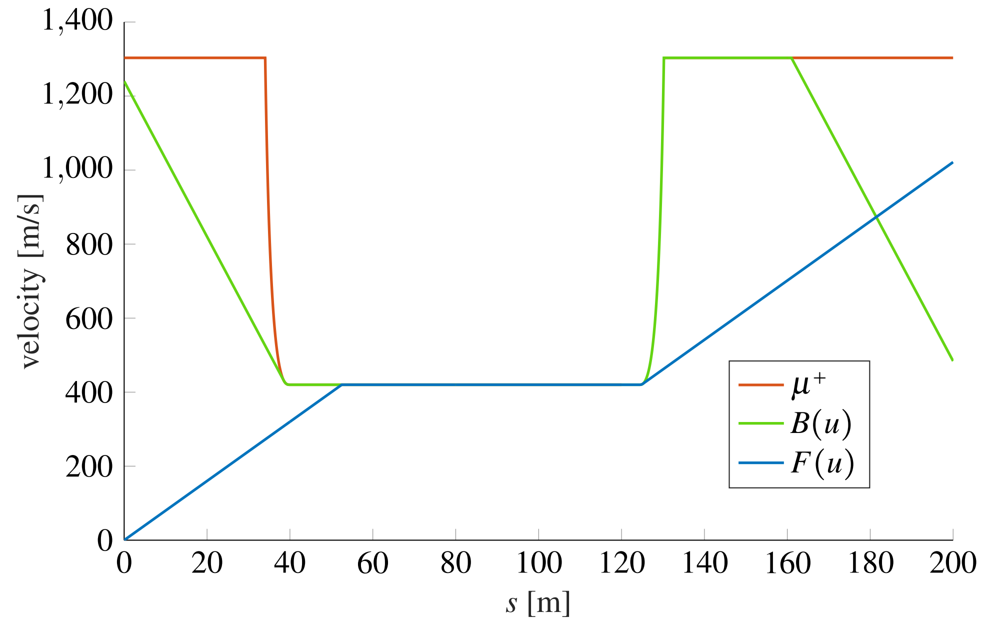


Figure 7.4: Example 1: The red line represents function μ^+ defined in (7.4), the blue line represents $F(u)$ and the green line represents $B(u)$.

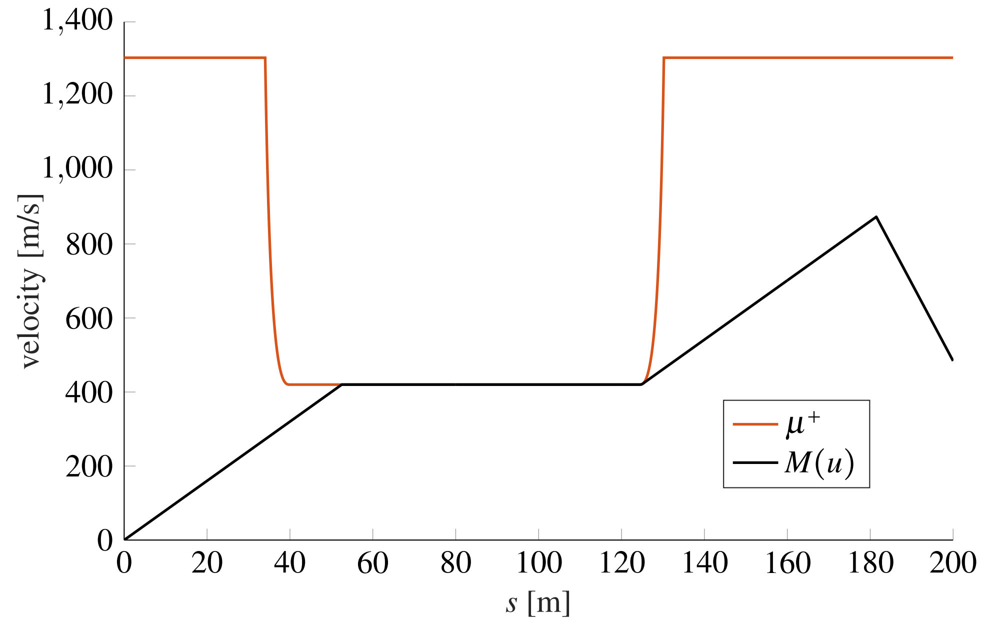


Figure 7.5: Example 1: The red line represents μ^+ defined in (7.4), the black line represents the optimal solution $w^* = M(u)$.

7.4.2 Example 2

As a second example, consider the same path and constraints as in the first example, with different initial and final conditions: $v^-(0) = v^+(0) = 0$, $v^-(s_f) = v^+(s_f) = 35$. Figure 7.6 shows function w^* obtained by (7.8). In this case, Problem (7.5) is unfeasible by Theorem 7.2, being $w^*(s_f) < v^-(s_f)^2$. In fact, the allowed maximum longitudinal acceleration is not sufficient to reach the final condition on velocity.

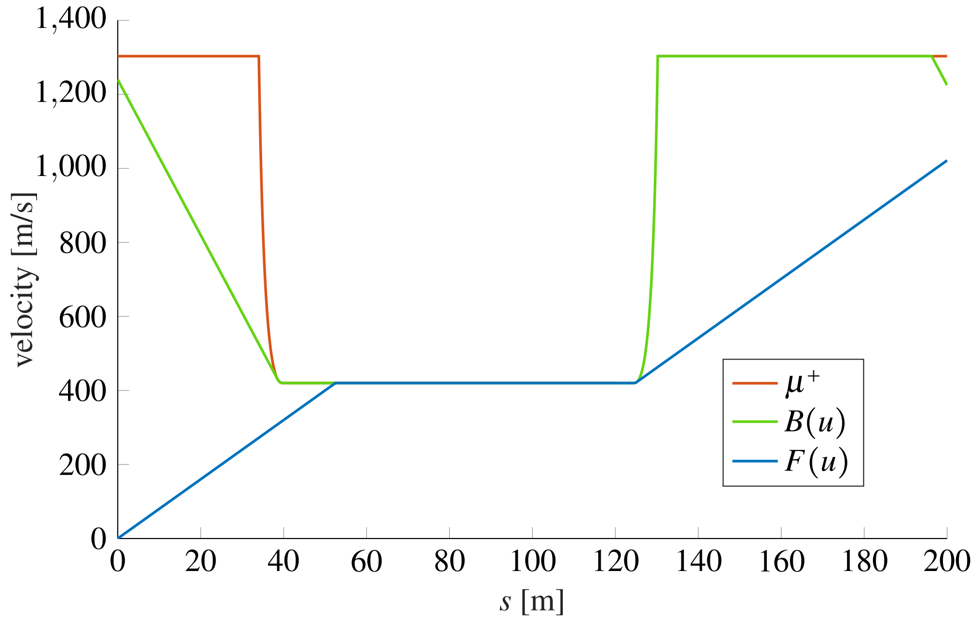


Figure 7.6: Example 2. The green line represents the velocity function $B(u)(s)$ while the black one depicts function $w^* = M(u)$. The final velocity condition is not satisfied: $w^*(s_f) \neq v_f^2$.

As a third example, consider a curve obtained by a quintic polynomial curve which interpolates coordinates $x = [0,\ 2,\ 2.60,\ 1.75,\ 3]$, $y = [0,\ -0.5,\ 0,\ 2,\ 3]$ (see Figure 7.7). The velocity planning is addressed with $v^+(0) = v^-(0) = 0$, $v^+(s_f) = v^-(s_f) = 0$ and with $v^-(s) = 0$ and $v^+(s) = 1.3$ for each $s \in \left(0, s_f\right)$. The longitudinal acceleration limits are $\alpha^- = -0.1$, $\alpha^+ = 0.1$, and the maximal normal acceleration $\beta = 0.05$. The resulting optimal velocity profile is plotted in Figure 7.8.

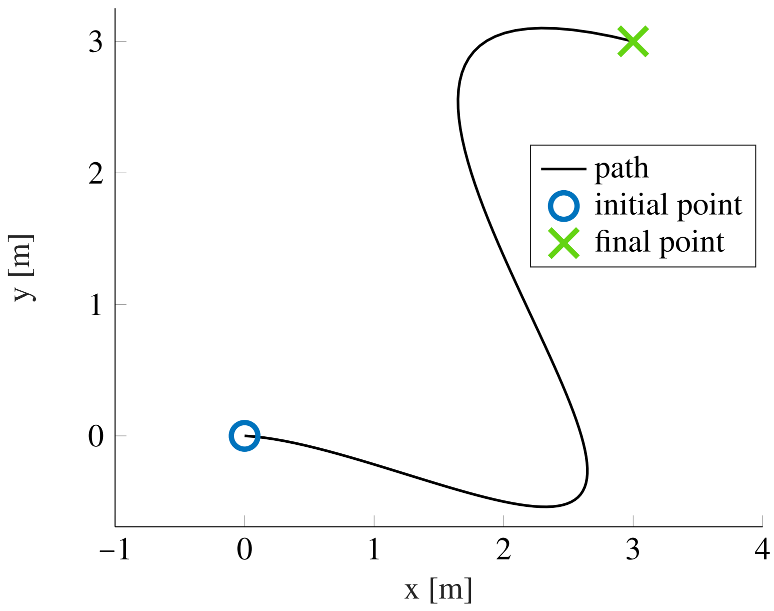


Figure 7.7: A path obtained by quintic-splines interpolation. The black line represents the path while the circle and the cross represent the start and the end point, respectively.

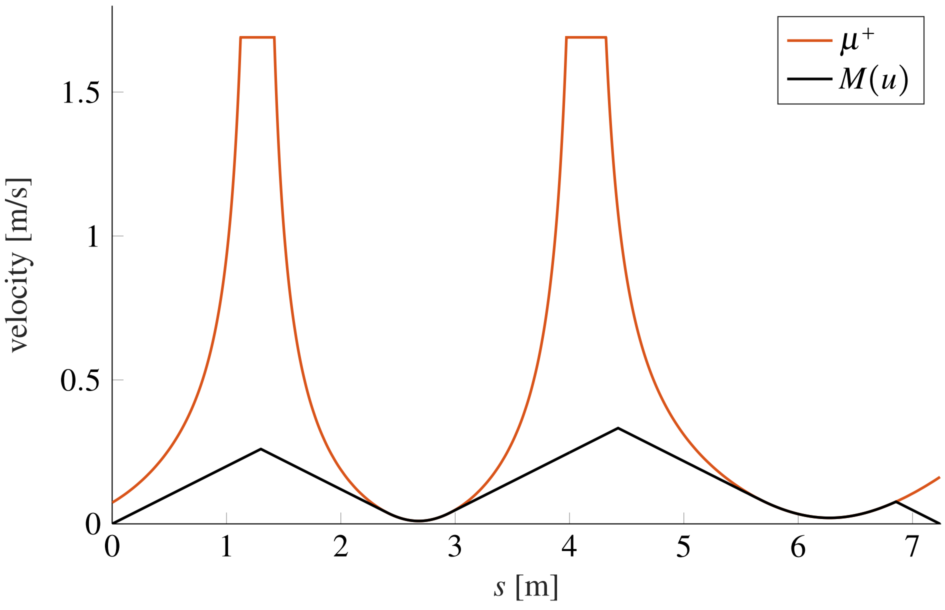


Figure 7.8: The red line represents function μ^+ defined in (7.4), the black line is the optimal velocity profile $w^* = M(u)$.

7.5 Proofs

Given the interval $I = \left[0, s_f\right]$, let $P = \{u \in L^\infty(I) \mid 0 \le u(s) \le \|\mu^+\|_\infty$ for almost every $s \in I\}$. Note that $P(\le, \wedge, \vee)$, with $\le$ defined as in (2.1) on $L^\infty(I)$, is a complete lattice. Hence, for each subset $S \subseteq P$, there exists a unique least upper bound $u \in P$, such that,

$$(\forall v \in P)\ (\forall w \in S)\ w \le v \iff u \le v.$$

Given function $\chi : \mathbb{R} \to \{0,1\}$ defined as follows

$$\chi(s) = \begin{cases} 1, & \text{if } s \ge 0 \\ 0, & \text{otherwise,} \end{cases}$$

let us define for all $x, y \in I$, function $A : I \times I \to \mathbb{R}$ as

$$A(x,y) = \int_x^y \{\alpha^+(\xi)\chi(y-x) + \alpha^-(\xi)\chi(x-y)\}\, d\xi. \tag{7.10}$$

Define, also, operators $\bar{F}, \bar{B}, \bar{M} : P \to P$, such that, for $\mu \in P$, $\bar{F}(\mu)$ and $\bar{B}(\mu)$ are given as follows

$$\begin{cases} \bar{F}(\mu)(x) = \bigwedge\limits_{y \le x} \{\mu(y) + A(y,x)\} \\ \bar{F}(\mu)(0) = \mu(0), \end{cases}$$

$$\begin{cases} \bar{B}(\mu)(x) = \bigwedge\limits_{y \ge x} \{\mu(y) + A(y,x)\} \\ \bar{B}(\mu)(s_f) = \mu(s_f). \end{cases}$$

and $\bar{M}(\mu) = \bar{F}(\mu) \wedge \bar{B}(\mu)$. Observe that $(\forall x \in I)$

$$\bar{M}(\mu)(x) = \bigwedge_{y \le x} \{\mu(y) + A(y,x)\} \wedge \bigwedge_{y \ge x} \{\mu(y) + A(y,x)\} = \bigwedge_{y \in I} \{\mu(y) + A(y,x)\}. \tag{7.11}$$

As we will show in Proposition 7.9, the operators we just introduced are extensions of those defined, respectively, in (7.6), (7.7) and (7.8).
Recalling Definitions 2.12 and 2.13; even though we will only use the fact that operators $\bar{F}, \bar{B}$ and $\bar{M}$ are order-preserving, for completeness we state the following Proposition.

Proposition 7.3. *Operators $\bar{F}$, $\bar{B}$ and $\bar{M}$ are meet preserving and order-preserving.*

Proof. Let $u,v \in P$, $w = u \wedge v$. We want to show that $\bar{F}(w) = \bar{F}(u) \wedge \bar{F}(v)$. By definition of $\bar{F}$ we have that $\bar{F}(w)(0) = \bar{F}(u)(0) \wedge \bar{F}(v)(0)$ and $(\forall x \in I)$

$$\begin{aligned}\bar{F}(w)(x) &= \bigwedge_{y \le x} \{w(y) + A(y,x)\} = \bigwedge_{y \le x} \{(u \wedge v)(y) + A(y,x)\} = \\ &= \bigwedge_{y \le x} \{u(y) \wedge v(y) + A(y,x)\} = \bigwedge_{y \le x} \{u(y) + A(y,x)\} \wedge \bigwedge_{y \le x} \{v(y) + A(y,x)\} = \\ &= \bar{F}(u)(x) \wedge \bar{F}(v)(x).\end{aligned}$$

The proof that $\bar{B}(u \wedge v) = \bar{B}(u) \wedge \bar{B}(v)$ is analogous. Finally, $\bar{M}(u \wedge v) = \bar{F}(u \wedge v) \wedge \bar{B}(u \wedge v) = \bar{F}(u) \wedge \bar{F}(v) \wedge \bar{B}(u) \wedge \bar{B}(v) = \bar{F}(u) \wedge \bar{B}(u) \wedge \bar{F}(v) \wedge \bar{B}(v) = \bar{M}(u) \wedge \bar{M}(v)$. Since maps $\bar{F}, \bar{B}, \bar{M}$ are meet preserving, they are also order-preserving (see Proposition 2.14). □

Proposition 7.4. *Function $A : I \times I \to \mathbb{R}$ defined as in* (7.10) *is a hemi-metric, that is, it satisfies the following properties:*

i. $(\forall x,y \in I)\ A(x,y) \ge 0$,

ii. $(\forall x \in I)\ A(x,x) = 0$,

iii. $(\forall x,y,z \in I)\ A(x,z) \le A(x,y) + A(y,z)$ *(i.e., the triangular inequality holds). Moreover, equality holds if $x \ge y \ge z$ or $x \le y \le z$.*

Proof. *i.* It holds, since α^+ is non-negative and α^- is non-positive over I.

ii. It holds trivially by definition of A.

iii. For $y \ge z \ge x$:

$$\begin{aligned}A(x,z) &= \int_x^z \{\alpha^+(\xi)\chi(z-x) + \alpha^-(\xi)\chi(x-z)\}d\xi = \int_x^z \alpha^+(\xi)d\xi \le \\ &\le \int_x^y \alpha^+(\xi)d\xi \le \int_x^y \alpha^+(\xi)d\xi - \int_z^y \alpha^-(\eta)d\eta = \\ &= \int_x^y \alpha^+(\xi)d\xi + \int_y^z \alpha^-(\eta)d\eta = \\ &= \int_x^y \{\alpha^+(\xi)\chi(y-x) + \alpha^-(\xi)\chi(x-y)\}d\xi + \\ &+ \int_y^z \{\alpha^+(\eta)\chi(z-y) + \alpha^-(\eta)\chi(y-z)\}d\eta = \\ &= A(x,y) + A(y,z).\end{aligned}$$

The same reasoning applies also to the case when $z \geq x \geq y$, $x \geq z \geq y$ or $y \geq x \geq z$. Next, let us show that equality holds for any $x \leq y \leq z$:

$$\begin{aligned} A(x,z) &= \int_x^z \{\alpha^+(\xi)\chi(z-x) + \alpha^-(\xi)\chi(x-z)\}\,d\xi = \int_x^z \alpha^+(\xi)\,d\xi = \\ &= \int_x^y \alpha^+(\xi)\,d\xi + \int_y^z \alpha^+(\eta)\,d\eta = \\ &= \int_x^y \{\alpha^+(\xi)\chi(y-x) + \alpha^-(\xi)\chi(x-y)\}\,d\xi + \\ &\quad + \int_y^z \{\alpha^+(\eta)\chi(z-y) + \alpha^-(\eta)\chi(y-z)\}\,d\eta = \\ &= A(x,y) + A(y,z). \end{aligned}$$

The proof that the equality holds also for any $x \geq y \geq z$ is analogous.

□

Proposition 7.5. *Function $\bar{M}$ satisfies the following properties, $\forall \mu \in P$,*

i. $\bar{M}(\mu) \leq \mu$,

ii. $\bar{M}^2(\mu) = \bar{M}(\mu)$, *where* $\bar{M}^2(\mu)$ *stands for* $\bar{M}(\bar{M}(\mu))$.

Proof. *i.* It is a consequence of the definition of $\bar{M}$.

ii. Let us now show that $\bar{F}(\bar{M}(\mu)) = \bar{M}(\mu)$: the fact that $\bar{F}(\bar{M}(\mu)) \leq \bar{M}(\mu)$ follows by the definition of $\bar{F}$ whilst, to prove the opposite inequality, note that, by Proposition 7.4 and (7.11),

$$\begin{aligned} \bar{F}(\bar{M}(\mu))(x) &= \bigwedge_{y \leq x} \{\bar{M}(\mu)(y) + A(y,x)\} = \bigwedge_{y \leq x} \left\{ \bigwedge_{z \in I} \{\mu(z) + A(z,y)\} + A(y,x) \right\} \geq \\ &\geq \bigwedge_{z \in I} \{\mu(z) + A(z,x)\} = \bar{M}(\mu)(x). \end{aligned}$$

In the same way it can be proved that $\bar{B}(\bar{M}(\mu)) = \bar{M}(\mu)$, from which it follows that $\bar{M}(\bar{M}(\mu)) = \bar{M}(\mu)$.

□

Proposition 7.6.

$$\bar{M}(\mu^+) = \bigvee \{u \in P \mid u \leq \bar{M}(u), u \leq \mu^+\}$$

Proof. Set $U = \{u \in P \mid u \leq \mu^+\}$. Note that $U(\leq,\wedge,\vee)$ is a sublattice of $P(\leq,\wedge,\vee)$, moreover, by *i.* of Proposition 7.5, if $u \in U$, then $\bar{M}(u) \in U$. Since $\bar{M}$ is order-preserving by Proposition 7.3, by the Knaster-Tarski Fixpoint Theorem (see Theorem 2.16)

$$u^* = \bigvee \{u \in P \mid u \leq \bar{M}(u), u \leq \mu^+\}$$

is such that u^* is the greatest fixed point of $\bar{M}$ such that $u^* \in U$. Let $u = \bar{M}(\mu^+)$, by part *ii.* of Proposition 7.5, we know that u is also a fixed point of $\bar{M}$, thus, by definition of u^*, $u^* \geq u$. To prove that $u^* = u$, that is, to prove that u is also the greatest fixed point, it remains to show that $u^* \leq u$. To this end, assume, by contradiction, that $u^* \nleq u$. Since $u^* = \bar{M}(u^*)$, $u = \bar{M}(\mu^+)$ and the fact that $\bar{M}$ is order-preserving, it follows that $u^* \nleq \mu^+$, which contradicts the definition of u^* $\lightning$. □

Remark 7.7. *Given $u, v \in P$, if $u \nleq v$, this does not imply that $u \geq v$ and $u \neq v$, as u and v may not be comparable with respect to partial order $\leq$.*

Proposition 7.8. *The following two statements are equivalent:*

i. *Set $\{u \in P \mid u = \bar{M}(u),\ \mu^- \leq u \leq \mu^+\}$ is not empty.*

ii. $\bar{M}(\mu^+) \geq \mu^-$.

Proof.

ii. $\Rightarrow$ *i.*) It follows from the fact that $u^* = \bar{M}(\mu^+)$ is such that $\bar{M}(u^*) = u^*$ by part *ii.* of Proposition 7.5.

i. $\Rightarrow$ *ii.*) By contradiction, assume that $\bar{M}(\mu^+) \ngeq \mu^-$. Choose any $w \in P$ such that $w = \bar{M}(w)$ and $w \leq \mu^+$. By Proposition 7.6, $\bar{M}(w) \leq \bar{M}(\mu^+)$, hence $w \leq \bar{M}(\mu^+)$, but $\bar{M}(\mu^+) \ngeq \mu^-$, so $w \ngeq \mu^-$. Thus, being w any fixed point of $\bar{M}$ such that $w \leq \mu^+$, set $\{u \in P \mid u = \bar{M}(u),\ \mu^- \leq u \leq \mu^+\}$ is empty $\lightning$.

□

Proposition 7.9. *If $\mu \in \mathcal{Q}$, then $\bar{F}(\mu), \bar{B}(\mu) \in W^{1,\infty}(I)$ satisfy a. e.*

$$\begin{cases} \bar{F}(\mu)'(x) = \begin{cases} \alpha^+(x) \wedge \mu'(x), & \text{if } \bar{F}(\mu)(x) \geq \mu(x) \\ \alpha^+(x), & \text{if } \bar{F}(\mu)(x) < \mu(x) \end{cases} \\ \bar{F}(\mu)(0) = \mu(0), \end{cases} \tag{7.12}$$

and

$$\begin{cases} \bar{B}(\mu)'(x) = \begin{cases} \alpha^-(x) \vee \mu'(x), & \text{if } \bar{B}(\mu)(x) \geq \mu(x) \\ \alpha^-(x), & \text{if } \bar{B}(\mu)(x) < \mu(x) \end{cases} \\ \bar{B}(\mu)(s_f) = \mu(s_f). \end{cases} \tag{7.13}$$

Proof. Let $J = \{x \in I \mid \operatorname{sgn}(\mu' - \alpha^+) \text{ is continuous at } x\}$. Note that, since $\mu \in \mathcal{Q}$, J contains almost all elements of I. Let $x \in J$, then

$$\lim_{h\to 0^+} \frac{\bar{F}(\mu)(x+h) - \bar{F}(\mu)(x)}{h} = \lim_{h\to 0^+} h^{-1}\left[\bigwedge_{y\le x+h}\{\mu(y) + A(y,x+h)\} - \bar{F}(\mu)(x)\right] =$$

$$= \lim_{h\to 0^+} h^{-1}\left[\left(\bigwedge_{y\le x}\{\mu(y) + A(y,x+h)\} - \bar{F}(\mu)(x)\right)\wedge\right.$$
$$\left.\wedge\left(\bigwedge_{x<y\le x+h}\{\mu(y) + A(y,x+h)\} - \bar{F}(\mu)(x)\right)\right] \tag{7.14}$$

Since $A(y,x+h) = A(y,x) + A(x,x+h)$ by Proposition 7.4, the first parenthesis of (7.14) reduces to

$$\bigwedge_{y\le x}\{\mu(y) + A(y,x+h)\} - \bar{F}(\mu)(x) = \bar{F}(\mu)(x) + A(x,x+h) - \bar{F}(\mu)(x) = A(x,x+h).$$

Being $\operatorname{sgn}(\mu' - \alpha^+)$ continuous at x, it is possible to choose $h > 0$ sufficiently small such that $\operatorname{sgn}(\mu' - \alpha^+)$ is constant on interval $[x,x+h]$. Set $J^+ = \{x \in J : \mu'(x) - \alpha^+(x) > 0\}$ and $J^- = J \setminus J^+$. Then, the second parenthesis of (7.14) can be rewritten as

$$\bigwedge_{x<y\le x+h}\{\mu(y) + A(y,x+h)\} - \bar{F}(\mu)(x) = \begin{cases} \mu(x+h) - \bar{F}(\mu)(x), & \text{if } x \in J^- \\ \mu(x) + A(x,x+h) - \bar{F}(\mu)(x), & \text{if } x \in J^+, \end{cases}$$

since in the former case the minimum of $\mu(y) + A(y,x+h)$ over $[x,x+h]$ is attained at $x+h$, whilst in the latter is attained at x.
Hence, we have that

$$\lim_{h\to 0^+} \frac{\bar{F}(\mu)(x+h) - \bar{F}(\mu)(x)}{h} =$$
$$= \begin{cases} \lim_{h\to 0^+} h^{-1}\left[A(x,x+h) \wedge (\mu(x+h) - \bar{F}(\mu)(x))\right], & \text{if } x \in J^- \\ \lim_{h\to 0^+} h^{-1}\left[A(x,x+h) \wedge (\mu(x) + A(x,x+h) - \bar{F}(\mu)(x))\right], & \text{if } x \in J^+ \end{cases}$$
$$= \begin{cases} \alpha^+(x) \wedge \lim_{h\to 0^+} h^{-1}(\mu(x+h) - \bar{F}(\mu)(x)), & \text{if } x \in J^- \\ \alpha^+(x) \wedge \lim_{h\to 0^+} h^{-1}(\mu(x) + A(x,x+h) - \bar{F}(\mu)(x)), & \text{if } x \in J^+ \end{cases}$$

$$= \begin{cases} \alpha^+(x) \wedge \mu'(x), & \text{if } x \in J^- \text{ and } \bar{F}(\mu)(x) \geq \mu(x) \\ \alpha^+(x) \wedge +\infty = \alpha^+(x), & \text{if } x \in J^- \text{ and } \bar{F}(\mu)(x) < \mu(x) \\ \alpha^+(x) = \alpha^+(x) \wedge \mu'(x), & \text{if } x \in J^+ \text{ and } \bar{F}(\mu)(x) \geq \mu(x) \\ \alpha^+(x) \wedge +\infty = \alpha^+(x), & \text{if } x \in J^+ \text{ and } \bar{F}(\mu)(x) < \mu(x) \end{cases}$$

$$= \begin{cases} \alpha^+(x) \wedge \mu'(x), & \text{if } \bar{F}(\mu)(x) \geq \mu(x) \\ \alpha^+(x), & \text{if } \bar{F}(\mu)(x) < \mu(x). \end{cases}$$

Note that, by definition of $\bar{F}$, $\bar{F}(\mu)(x) \leq \mu(x)$ must hold. In conclusion, we proved that $\bar{F}(u) \in W^{1,\infty}(I)$ and satisfies (7.12).
Applying the same reasoning it can be proved that $\bar{B}(u) \in W^{1,\infty}(I)$ and satisfies (7.13). □

Proposition 7.10. *Assume that $\mu^+ \in Q$ and let $w \in P$, then w is feasible for Problem* (7.5) *(i.e., it satisfies constraints* (7.5a) *and* (7.5b)*), if and only if $\mu^- \leq w \leq \mu^+$ and $\bar{M}(w) = w$.*

Proof. $\Rightarrow$) Assume that w is feasible for Problem (7.5), then w satisfies a. e. $w' \leq \alpha^+$. Thus, $\phi = w$ is the solution of (7.6) for $\mu = w$, which implies that $\bar{F}(w) = w$. Analogously $\bar{B}(w) = w$, so that $\bar{M}(w) = w$. Moreover, since w satisfies the bounds of Problem (7.5), it follows that $\mu^- \leq w \leq \mu^+$.

$\Leftarrow$) Condition (7.5a) holds by hypothesis. Since $\bar{M}(w) = w$, then it must be $\bar{F}(w) = w$. In fact, if by contradiction $\bar{F}(w) < w$, $\bar{M}(w) \leq \bar{F}(w) < w$, which contradicts the assumption $\lightning$. Then $\bar{F}(w) = w$, implies that, a. e., $\bar{F}(w)' = w'$ which by definition of $\bar{F}(w)'$ in (7.6), implies that, a. e., $w' \leq \alpha^+$. Analogously, it must be $\bar{B}(w) = w$ which implies that, a. e. $w' \geq \alpha^-$ and condition (7.5b) holds.

□

Proposition 7.11. *Assume that $\mu^+ \in Q$. Then, Problem* (7.5) *is feasible if and only if $\bar{M}(\mu^+) \geq \mu^-$.*

Proof. $\Rightarrow$) Let w be a feasible solution of Problem (7.5). Then, by Proposition 7.10, $\bar{M}(w) = w \leq \mu^+$. Hence, being $\bar{M}$ order-preserving, $\bar{M}(\mu^+) \geq \bar{M}(w) \geq \mu^-$.

$\Leftarrow$) $w = \bar{M}(\mu^+)$ satisfies $\bar{M}(w) = w$ (by part *ii.* of Proposition 7.5) and $\mu^+ \leq w \leq \mu^-$ (by hypothesis). Hence, by Proposition 7.10, w is a feasible solution of Problem (7.5).

□

Proposition 7.12. *If $\mu^+ \in Q$ and Problem* (7.5) *is feasible, then $w^* = \bar{M}(\mu^+)$ is its optimal solution.*

Proof. By contradiction, assume that there exists a feasible $\tilde{w}$ such that $\Psi(\tilde{w}) < \Psi(w^*)$. Since $\tilde{w}$ is feasible, Proposition 7.10 implies that $\bar{M}(\tilde{w}) = \tilde{w}$. Moreover, since Ψ is order reversing, $\tilde{w} \nleq w^*$. This is not possible since $w^* = \bigvee\{w \in P \mid w \leq \bar{M}(w), w \leq \mu^+\} \geq \tilde{w}$, by Proposition 7.6 $\lightning$. $\square$

Bibliography

[1] R. Adams and J. Fournier. *Sobolev Spaces*, volume 140 of *Pure and Applied Mathematics*. Elsevier, 2nd edition, June 2003.

[2] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf. Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):943–949, Aug 2008.

[3] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, 57(7):2748–2761, July 2009.

[4] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[5] M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008.

[6] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex minlp. *Optimization Methods and Software*, 24(4-5):597–634, 2009.

[7] W. D. Blizard. Multiset theory. *Notre Dame Journal of Formal Logic*, 30(1):36–66, 1989.

[8] J. Bobrow, S. Dubowsky, and J. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985.

[9] F. Cabassi, L. Consolini, and M. Locatelli. Time-optimal velocity planning by a bound-tightening technique. *Computational Optimization and Applications*, 70(1):61–90, May 2018.

[10] I. Capuzzo Dolcetta and L. C. Evans. Optimal switching for ordinary differential equations. *SIAM Journal on Control and Optimization*, 22(1):143–161, Jan 1984.

[11] C. Chen, Y. He, C. Bu, J. Han, and X. Zhang. Quartic Bézier curve based trajectory generation for autonomous vehicles with curvature and velocity constraints. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6108–6113, May 2014.

[12] L. Consolini, M. Laurini, and M. Locatelli. Graph-based algorithms for the efficient solution of a class of optimization problems. *arXiv:1809.01970 [math.OC]*, 2018.

[13] L. Consolini, M. Laurini, M. Locatelli, and A. Minari. A solution of the minimum-time velocity planning problem based on lattice theory. *arXiv:1809.01959 [math.OC]*, 2018.

[14] L. Consolini, M. Locatelli, A. Minari, A. Nagy, and I. Vajk. Optimal time-complexity speed planning for robot manipulators. *CoRR*, abs/1802.03294, 2018.

[15] L. Consolini, M. Locatelli, A. Minari, and A. Piazzi. A linear-time algorithm for minimum-time velocity planning of autonomous vehicles. In *Proceedings of the 24th Mediterranean Conference on Control and Automation (MED), IEEE*, 2016.

[16] L. Consolini, M. Locatelli, A. Minari, and A. Piazzi. An optimal complexity algorithm for minimum-time velocity planning. *Systems and Control Letters*, 103:50–57, 2017.

[17] B. Davey and H. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[18] A. Filippov. Differential equations with discontinuous right-hand sides. *Matematicheskii Sbornik*, 51:99–128 (In Russian), 1960.

[19] A. Filippov. *Differential Equations with Discontinuous Righthand Sides: Control Systems*, volume 18 of *Mathematics and its Applications*. Springer Netherlands, 1988.

[20] M. Franceschelli, A. Giua, and C. Seatzu. A gossip-based algorithm for discrete consensus over heterogeneous networks. *IEEE Transactions on Automatic Control*, 55(5):1244–1249, May 2010.

[21] M. Frego, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli. Semi-analytical minimum time solutions for a vehicle following clothoid-based trajectory subject to velocity constraints. In *2016 European Control Conference (ECC)*, pages 2221–2227, June 2016.

[22] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Aug 2008.

[23] L. Han, Q. H. Do, and S. Mita. Unified path planner for parking an autonomous vehicle based on rrt. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5622–5627, 2011.

[24] S. V. Hanly. An algorithm for combined cell-site selection and power control to maimize cellular spread spectrum capacity. *IEEE Journal on Selected Areas in Communications*, 13(7):1332–1340, September 1995.

[25] J. Heinonen. *Lectures on Lipschitz Analysis*. Bericht (Jyväskylän yliopisto. Matematiikan ja tilastotieteen laitos). University of Jyväskylä, 2005.

[26] P. Holme and B. J. Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65(026107):1–4, Jan 2002.

[27] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Pearson Education, 2nd edition, 2001.

[28] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, 1986.

[29] T. Kunz and M. Stilman. Time-optimal trajectory generation for path following with bounded acceleration and velocity. *Robotics: Science and Systems VIII*, 2012.

[30] M. Laurini, L. Consolini, and M. Locatelli. A consensus approach to dynamic programming. In D. Dochain, D. Henrion, and D. Peaucelle, editors, *Proceedings of the 20th IFAC World Congress*, volume 50, pages 8435–8440, July 2017.

[31] M. Laurini, L. Consolini, and M. Locatelli. A multigraph-based selective update method for the efficient solution of dynamic programming. In *2018 IEEE 57th Conference on Decision and Control (CDC)*, Dec 2018. to appear.

[32] M. Laurini, P. Micelli, L. Consolini, and M. Locatelli. A jacobi-like acceleration for dynamic programming. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 7371–7376, Dec 2016.

[33] X. Li, Z. Sun, A. Kurt, and Q. Zhu. A sampling-based local trajectory planner for autonomous driving along a reference path. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 376–381, June 2014.

[34] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.

[35] D. Liu and Q. Wei. Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems. *IEEE Transactions on Cybernetics*, 43(2):779–789, April 2013.

[36] V. Muñoz, A. Ollero, M. Prado, and A. Simón. Mobile robot trajectory planning with dynamic and kinematic constraints. In *Proc. of the 1994 IEEE Int. Conf. on Robotics and Automation*, volume 4, pages 2802–2807, San Diego, CA, May 1994.

[37] Á. Nagy and I. Vajk. Lp-based velocity profile generation for robotic manipulators. *International Journal of Control*, pages 1–11, 2017.

[38] M. E. J. Newman and D. J. Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, 1999.

[39] J. Reeds and L. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990.

[40] P. Shen, X. Zhang, and Y. Fang. Essential properties of numerical integration for time-optimal path-constrained trajectory planning. *IEEE Robotics and Automation Letters*, 2(2):888–895, April 2017.

[41] Z. Shiller and Y.-R. Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7(2):241–249, April 1991.

[42] M. Sipser. *Introduction to the Theory of Computation*. CENGAGE Learning, 3rd edition, 2012.

[43] J. J. E. Slotine and H. S. Yang. Improving the efficiency of time-optimal path-following algorithms. *IEEE Transactions on Robotics and Automation*, 5(1):118–124, Feb 1989.

[44] R. Solea and U. Nunes. Trajectory planning with velocity planner for fully-automated passenger vehicles. In *IEEE Intelligent Transportation Systems Conference, ITSC '06*, pages 474 –480, September 2006.

[45] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[46] E. Velenis and P. Tsiotras. Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation. *Journal of Optimization Theory and Applications*, 138(2):275–296, 2008.

[47] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl. Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10), Oct 2009.

[48] J. Villagra, V. Milanés, J. Pérez, and J. Godoy. Smooth path and speed planning for an automated public transport vehicle. *Robotics and Autonomous Systems*, 60:252–265, 2012.

[49] S. Wang, F. Gao, and K. L. Teo. An upwind finite-difference method for the approximation of viscosity solutions to hamilton-jacobi-bellman equations. *IMA Journal of Mathematical Control and Information*, 17(2):167–178, 2000.

[50] R. Yates and C. Y. Huang. Integrated power control and base staion assignment. *IEEE Transactions on Vehicular Technology*, 44(3), August 1995.

[51] R. D. Yates. A framework for uplink power control in cellular radio systems. *IEEE Journal on Selected Areas in Communications*, 13(7):1341–1348, September 1995.

[52] H. Zhang and M. R. James. On computation of optimal switching hjb equation. In *Proceedings of the 45th IEEE Conference on Decision & Control*, pages 2704–2709, 2006.