

The Bounded Acceleration Shortest Path problem: complexity and solution algorithms

Stefano Ardizzoni, Luca Consolini, Mattia Laurini and Marco Locatelli*

Abstract

The purpose of this work is to introduce and characterize the Bounded Acceleration Shortest Path (BASP) problem, a generalization of the Shortest Path (SP) problem. This problem is associated to a graph: the nodes represent positions of a mobile vehicle and the arcs are associated to pre-assigned geometric paths that connect these positions. BASP consists in finding the minimum-time path between two nodes. Differently from SP, we require that the vehicle satisfy bounds on maximum and minimum acceleration and speed, that depend on the vehicle position on the currently traveled arc. We prove that BASP is NP-hard and define solution algorithm that achieves polynomial time-complexity under some additional hypotheses on problem data.

1 Introduction

The purpose of this work is to introduce and characterize the Bounded Acceleration Shortest Path (BASP) problem, a generalization of the Shortest Path (SP) problem. We consider a graph associated to a path and speed planning problem for a mobile vehicle. The graph nodes represent vehicle positions and the arcs are associated to pre-assigned geometric paths that connect these positions. BASP consists in finding the minimum-time path between two nodes. Differently from SP, BASP requires that the vehicle satisfy bounds on maximum and minimum acceleration and speed, that depend on the vehicle position on the currently traveled arc. Figure 1 presents a simple scenario that allows to illustrate BASP and its difference with SP. This figure shows some fixed paths connecting positions A, B, C, D . The vehicle starts from A with zero speed and must reach D with zero speed. The solution of the SP problem corresponds to path $ABCD$, which is the one of shortest length. BASP consists in finding the shortest-time path under acceleration and speed constraints. In this case, we assume that the vehicle acceleration and deceleration are bounded by a common constant and that its speed is bounded only on arc BC . For instance, this may be due to the fact that BC is an arc of a circle of small radius

*All authors are with the Dipartimento di Ingegneria e Architettura, Università degli Studi di Parma, Parco Area delle Scienze 181/A, 43124 Parma, Italy. E-mails: {stefano.ardizzoni, luca.consolini, mattia.laurini, marco.locatelli}@unipr.it

and the vehicle speed on BC has to be limited in order to avoid excessive lateral acceleration, which may cause slideslip. If the bound on acceleration and deceleration is sufficiently high, the solution of BASP corresponds to path AD . Indeed, even if this second path is longer, it can be travelled with a greater mean speed due to the absence of speed bounds. To clarify this fact, see Figures 2 and 3. Figure 2 represents the fastest speed profile on $ABCD$. The x -axis corresponds to the arc-length position on path $ABCD$ and the y -axis represents the squared speed. In this representation, arc-length intervals of constant acceleration or deceleration correspond to straight lines. Note that speed has to be reduced before entering into arc BC in order to respect the speed bound on BC . Figure 3 represents the fastest speed profile on AD . Due to the absence of speed bounds, the vehicle accelerates till the midpoint of the path and then decelerates to the end node D . Even if path AD is longer than $ABCD$, it can be travelled with a shorter time. In Section 3.1, we will justify the structure of the optimal speed profiles reported in Figures 2 and 3.

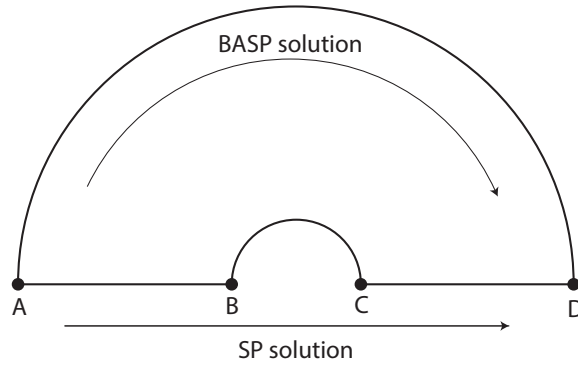


Figure 1: Comparison of BASP and SP solutions.

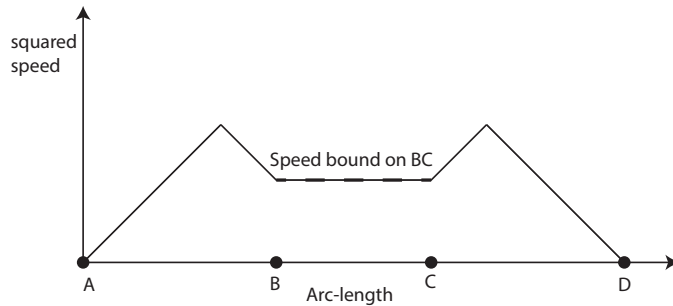


Figure 2: Optimal speed profile on $ABCD$.

BASP is a generalization of SP. Indeed, if we remove the maximum and minimum acceleration bounds, BASP reduces to SP, in which the cost of each arc is the time needed to travel the path associated to the arc at maximum speed. In the

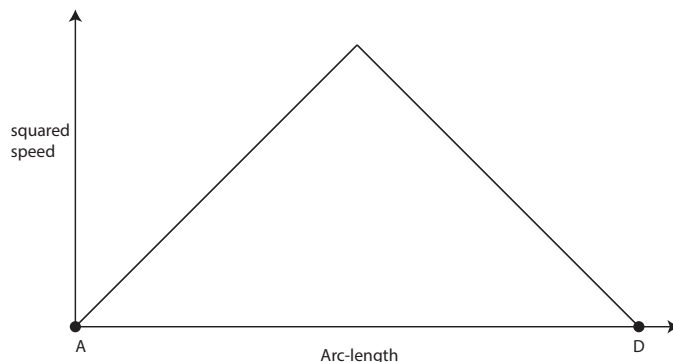


Figure 3: Optimal speed profile on AD .

general case, BASP is more complex than SP. Indeed, in Proposition 4.2, we will show that BASP is NP-hard. However, if we make some additional assumptions on problem parameters, we obtain a subclass of BASP, that we call k -BASP, that can be solved with polynomial time-complexity. Roughly speaking, a BASP instance belongs to k -BASP if the problem data are such that no more than $k - 2$ arcs can be travelled with a speed profile starting from zero speed and of maximum acceleration, then followed by one of maximum deceleration and ending with zero speed, without violating the maximum speed constraint. In Section 5, we will define k -BASP more precisely and we will present a simple upper bound on constant k . In Proposition 5.5, we will show that k -BASP can be solved by Dijkstra's algorithm with polynomial time complexity with respect to the graph size (its number of nodes and edges), provided that k is fixed. We will also present Algorithm 6.5, which is able to adaptively find constant k .

Statement of contribution. To our knowledge, BASP has not been explicitly considered in literature, so that the main results presented here are new. In particular, we believe that the most relevant contributions are:

- Proposition 4.2, that shows that BASP is NP-hard.
- Proposition 5.5, that shows that k -BASP can be solved with a polynomial time-complexity, provided that k is fixed.
- The adaptive Algorithm 6.5, that is able to efficiently solve a large set of BASP instances.

1.1 Problem motivation

One relevant application of this work is the optimization of automated guided vehicles (AGVs) motions in automated warehouses. Automated warehouses are rapidly

spreading in manufacturing and logistics because of their speed, flexibility, and reliability. In order to ensure the smooth functioning and to increase the overall efficiency of the system, such fleets of AGVs need be coordinated at different levels of control: task allocation, localization, path planning, motion planning and vehicle management (see, for instance, [11], for a more in depth discussion).

In automated warehouses, AGVs are commonly moved between fixed operating points. These points may be associated to shelves locations, where packages are stored or retrieved, to the end of production lines, where the AGV picks up a final product, and to additional intermediate locations, used for routing. Between these operating points, the vehicle follows preassigned connecting paths (see Figure 4). The vehicle motion must satisfy constraints on maximum speed and max-

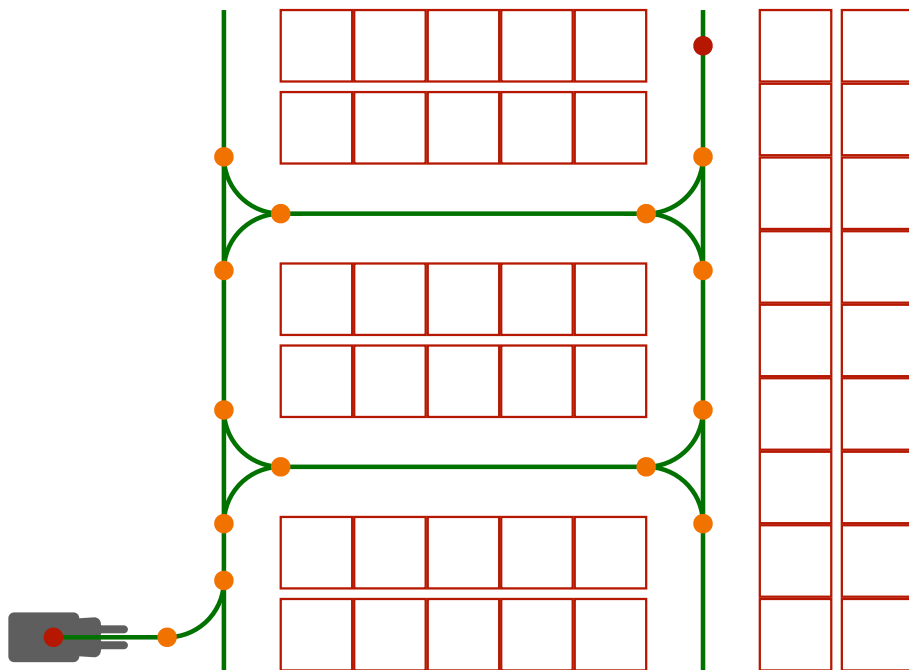


Figure 4: An example scenario.

imum tangential and transversal accelerations, that depend on the vehicle position on the path.

The algorithms developed in this paper allow to find the time-optimal path for a single AGV that travels between operating points, taking into account path-dependent bounds on maximum acceleration, deceleration and speed.

1.2 Related works

As said, to our knowledge, BASP has not been explicitly addressed in literature. However, various works address related path planning problems for AGVs. In those scenarios in which AGVs are allowed to move freely within their environ-

ment and no predetermined circuits are available, one need to employ environmental representations such as cell decomposition methods ([1]) or trajectory maps ([12]). In particular, among cell decomposition methods, [4] presents an algorithm based on a modification of Dijkstra’s algorithm in which edge weights depend on previously visited edges. Note that our work shares some similarities with [4] in regards of the idea of the history-dependent edges weight and in the way the extended graph associated to the addressed problem is defined. However, [4] focuses on a different problem. In fact, it introduces a cell decomposition method whose goal is to obtain a feasible path taking into account the vehicle maximum curvature radius. Instead, our work focuses on selecting the optimal path among a set of already feasible paths while obtaining the optimal speed profile as well. Moreover, the algorithm introduced in [4] operates introducing a set of labels which can potentially be very expensive in terms of memory usage and the history parameter is given in input and is not adaptively computed, losing the guarantee of optimality.

In many industrial scenarios, AGVs move along predetermined circuits. The representation of such paths is usually graph-based. The problem of finding the optimal path connecting two positions within a facility turns, then, into the problem of finding the shortest path connecting a pair of nodes in a graph. There are various graph searching algorithms that are used to this end such as A*, Lifelong Planning A* ([9]), D* ([6]) and D* Lite algorithms. Among these, the most widely used ([8]) are A* and D* Lite algorithms.

A* algorithm ([10]) is a heuristic method that allows to compute the optimal path (if it exists) ([7]) by exploring the graph beginning from the starting node along the most promising directions according to a heuristic function that estimates the cost from the current position to the target node.

2 Notation

A directed graph is a pair $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ where \mathbb{V} is a set of nodes and $\mathbb{E} \subset \{(x, y) \in \mathbb{V}^2 \mid x \neq y\}$ is a set of directed arcs. A path p on \mathbb{G} is a sequence of adjacent vertices of \mathbb{E} . That is, $p = \sigma_1 \cdots \sigma_m$, where, for $i \in \{1, \dots, m-1\}$, $(\sigma_i, \sigma_{i+1}) \in \mathbb{E}$. We denote by $P(\mathbb{G})$ the set of all paths of \mathbb{G} . An alphabet $\Sigma = \{\sigma_1, \dots, \sigma_n\}$ is a set whose elements are called symbols. A word is any finite sequence of symbols. We denote the set of all words over Σ by Σ^* , that also contains the empty word ε , while Σ_i represents the set of all words of length up to $i \in \mathbb{N}$, that is, words composed of up to i symbols, including the empty word ε . Given a word $w \in \Sigma^*$, we denote its length by $|w|$. Given a directed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, we can think of \mathbb{V} as an alphabet. In this way, any path $p \in P(\mathbb{G})$ is a word in \mathbb{V}^* . Given $s, t \in \Sigma^*$, the word obtained by writing t after s is called the concatenation of s and t and is denoted by $st \in \Sigma^*$. We also say that t is a suffix of st and that s is a prefix of st . For $r \in \mathbb{V}^*$, we denote by \vec{r} the rightmost symbol of r . In the following, it will be convenient to represent paths of \mathbb{G} as strings composed of symbols in \mathbb{V} . This will allow us to use the concatenation operation on paths and to use prefixes and suffixes to represent

portions of paths.

For $x \in \mathbb{R}$, we denote the ceiling of x by $\lceil x \rceil = \min\{i \in \mathbb{Z} \mid i \geq x\}$. For $a, b \in \mathbb{R}$, we set $a \wedge b = \min\{a, b\}$ and $a \vee b = \max\{a, b\}$, as the minimum and maximum operations, respectively. Further, \mathbb{R}^+ denotes the set of nonnegative real numbers.

Finally, given an interval $I \subseteq \mathbb{R}$, let us recall that $W^{1,\infty}(I)$ is the Sobolev space of functions in $L^\infty(I)$ with weak derivative of order one with finite L^∞ -norm. For $f, g \in W^{1,\infty}(I)$, we denote with $f \wedge g$ and $f \vee g$ the point-wise minimum and maximum of f and g , respectively.

3 Problem formulation

We first present the speed planning problem on an assigned path, following our previous work [3]. Then, we introduce the BASP problem, that considers both speed planning and path selection.

3.1 Speed planning along an assigned path

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

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

where $a_L(t) = v'(\xi(t))v(\xi(t))$ and $a_N(t) = \kappa(\xi(t))v(\xi(t))^2$ are the longitudinal and normal components of acceleration, respectively. Here, $\kappa: [0, \lambda_f] \rightarrow \mathbb{R}$ is the scalar curvature, defined as $\kappa(\lambda) = \langle \gamma''(\lambda), \gamma'(\lambda)^\perp \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the scalar product.

We require to travel distance λ_f in minimum-time while satisfying, for every $t \in [0, \xi^{-1}(\lambda_f)]$, $0 \leq v^-(\xi(t)) \leq v(\xi(t)) \leq v^+(\xi(t))$, $|a_N(\xi(t))| \leq \beta(\xi(t))$, $\alpha^-(\xi(t)) \leq a_L(\xi(t)) \leq \alpha^+(\xi(t))$. Here, functions $v^-, v^+, \alpha^-, \alpha^+, \beta$ are arc-length dependent bounds on the vehicle speed and on its longitudinal and normal acceleration. It is convenient to make the change of variables $w = v^2$ (see [13]), so that

our problem takes on the following form

$$\min_{w \in W^{1,\infty}([0, \lambda_f])} \int_0^{\lambda_f} w(\lambda)^{-\frac{1}{2}} d\lambda \quad (1a)$$

$$\mu^-(\lambda) < w(\lambda) \leq \mu^+(\lambda), \quad \lambda \in [0, \lambda_f], \quad (1b)$$

$$\alpha^-(\lambda) \leq w'(\lambda) \leq \alpha^+(\lambda), \quad \lambda \in [0, \lambda_f], \quad (1c)$$

where

$$\mu^+(\lambda) = v^+(\lambda)^2 \wedge \frac{\beta(\lambda)}{\kappa(\lambda)}, \quad \mu^-(\lambda) = v^-(\lambda)^2 \quad (2)$$

represent the upper bound on w (depending on speed bound v^+ and curvature κ) and the lower bound on w , respectively.

We actually address the following problem, which is slightly more general than (1),

$$\min_{w \in W^{1,\infty}([0, \lambda_f])} \Psi(w) \quad (3a)$$

$$\mu^-(\lambda) \leq w(\lambda) \leq \mu^+(\lambda), \quad \lambda \in [0, \lambda_f], \quad (3b)$$

$$\alpha^-(\lambda) \leq w'(\lambda) \leq \alpha^+(\lambda), \quad \lambda \in [0, \lambda_f], \quad (3c)$$

where $\Psi : W^{1,\infty}([0, \lambda_f]) \rightarrow \mathbb{R}$ is order reversing (i.e., $(\forall x, y \in [0, \lambda_f]) x \geq y \Rightarrow \Psi(x) \leq \Psi(y)$) and $\mu^-, \mu^+, \alpha^-, \alpha^+ \in L^\infty([0, \lambda_f])$ are assigned functions with $\mu^-, \alpha^+ \geq 0$ and $\alpha^- \leq 0$. Initial and final conditions on speed can be included in the definition of functions μ^- and μ^+ . For instance, to set initial condition $w(0) = w_0$, it is sufficient to define $\mu^+(0) = \mu^-(0) = w_0$.

Note that the objective function (1a) is order reversing, so that Problem (1) has form (3).

3.2 Solution of Problem (3)

We summarize the method presented in [3] and begin with introducing a subset of $W^{1,\infty}([0, \lambda_f])$ as a technical requirement.

Definition 3.1. Let Q_{α^-, α^+} be the subset of $W^{1,\infty}([0, \lambda_f])$ such that $\mu \in Q$ if $\text{sign}(\mu' - \alpha^+)$ and $\text{sign}(\mu' - \alpha^-)$ are Riemann integrable (i.e., in view of the boundedness of the sign function, almost everywhere continuous), where $\text{sign} : \mathbb{R} \rightarrow \{-1, 0, 1\}$ is defined as

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

Note that $\mu \in Q_{\alpha^-, \alpha^+}$ if functions $\mu' - \alpha^+$ and $\mu' - \alpha^-$ change sign a finite number of times in interval $[0, \lambda_f]$. In the following, we assume that $\mu^+ \in Q_{\alpha^-, \alpha^+}$.

To solve Problem (3), we define operators $F, B, M : Q \rightarrow W^{1,\infty}([0, \lambda_f])$ where Q is defined as in Definition 3.1 and, for $\mu \in Q$, $F(\mu)$ and $B(\mu)$ are given as follows

$$\begin{cases} F(\mu)'(\lambda) = \begin{cases} \alpha^+(\lambda) \wedge \mu'(\lambda) & \text{if } F(\mu)(\lambda) \geq \mu(\lambda) \\ \alpha^+(\lambda) & \text{if } F(\mu)(\lambda) < \mu(\lambda) \end{cases} \\ F(\mu)(0) = \mu(0), \end{cases} \quad (4)$$

$$\begin{cases} B(\mu)'(\lambda) = \begin{cases} \alpha^-(\lambda) \wedge \mu'(\lambda) & \text{if } B(\mu)(\lambda) \geq \mu(\lambda) \\ \alpha^-(\lambda) & \text{if } B(\mu)(\lambda) < \mu(\lambda) \end{cases} \\ B(\mu)(\lambda_f) = \mu(\lambda_f). \end{cases} \quad (5)$$

Finally, for $\mu \in Q$, operator M is defined as

$$M(\mu) = F(\mu) \wedge B(\mu). \quad (6)$$

The solution of Problem (3) is given by $w = M(\mu^+)$ (see [3]). We call F, B, M the forward operator, the backward operator and the meet operator, respectively. Roughly speaking, given a maximum squared speed profile $\mu^+ \in Q$, starting from $\mu^+(0)$ and up to $\mu^+(\lambda_f)$, $F(\mu^+)$ grows with the maximum allowed acceleration α^+ while staying below μ^+ , and, if it touches μ^+ , coincides with it until μ^+ grows with an acceleration higher than α^+ , in which case $F(\mu^+)$ behaves again as previously explained. Analogously, operator B acts in the same way as F but backwards and with $-\alpha^-$ as maximum acceleration. Finally, meet operator M is the point-wise minimum between forward operator F and backward operator B . Moreover, Problem (3) is feasible if and only if $\mu^- \leq w$.

In order to further clarify the meaning of these operators, we will consider a simple example. Let us examine the path shown in Figure 5, which represents a path whose total length is 200 m. The speed bounds v^+ and v^- in (2) are set as follows: $v^+(0) = v^-(0) = 0 \text{ ms}^{-1}$, $v^+(200) = v^-(200) = 22 \text{ ms}^{-1}$, whilst, for each $\lambda \in (0, 200)$, $v^-(\lambda) = 0 \text{ ms}^{-1}$ and $v^+(\lambda) = 36.1 \text{ ms}^{-1}$. The longitudinal acceleration limits are $\alpha^- = -2.78 \text{ ms}^{-2}$ and $\alpha^+ = 2.78 \text{ ms}^{-2}$, and the maximal normal acceleration is $\beta = 4.9 \text{ ms}^{-2}$. Figure 6 shows the upper-bound function μ^+ obtained by (2), with $\mu^+(0) = 0$, to impose zero initial speed, and the corresponding functions $F(\mu^+)$ and $B(\mu^+)$ computed as the solution of (4) and (5), respectively. Figure 6 shows $F(\mu^+)$ and $B(\mu^+)$, while Figure 7 shows the optimal solution $w = M(\mu^+)$ obtained by (6). In this example, the initial speed is zero, then the profile grows to the upper bound μ^+ ; next, it follows it in order to respect the maximum speed constraint due to the lateral acceleration on the curve. After that, at the end of the path part of higher curvature, it grows again and reaches a second local maximum speed after which it decreases in order to meet the final speed requirement $v^+(200)$. Note that we can compute an approximated solution of $w = M(\mu^+)$ by using a finite difference approximation of equations (4) and (5). As shown in [2], this can be done with an algorithm that has linear time-complexity with respect to the number of discretization points. Further, note that if functions

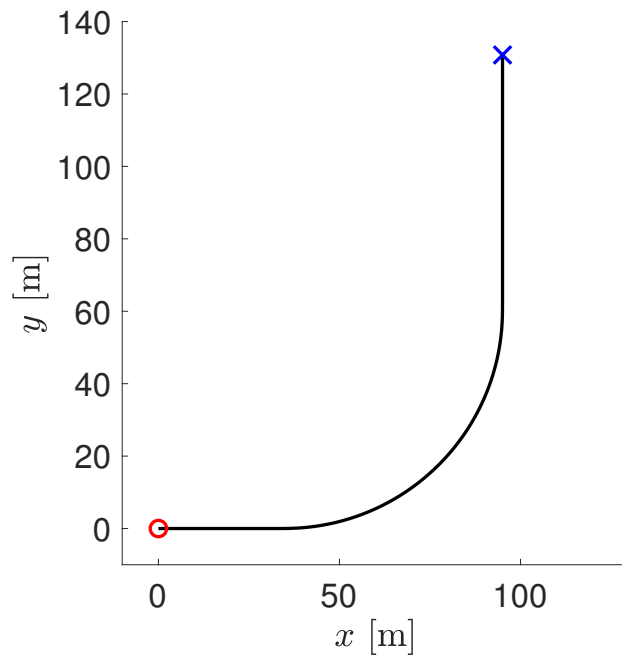


Figure 5: The black line represents the path, while the red circle and the black cross represent the starting point and the end point, respectively.

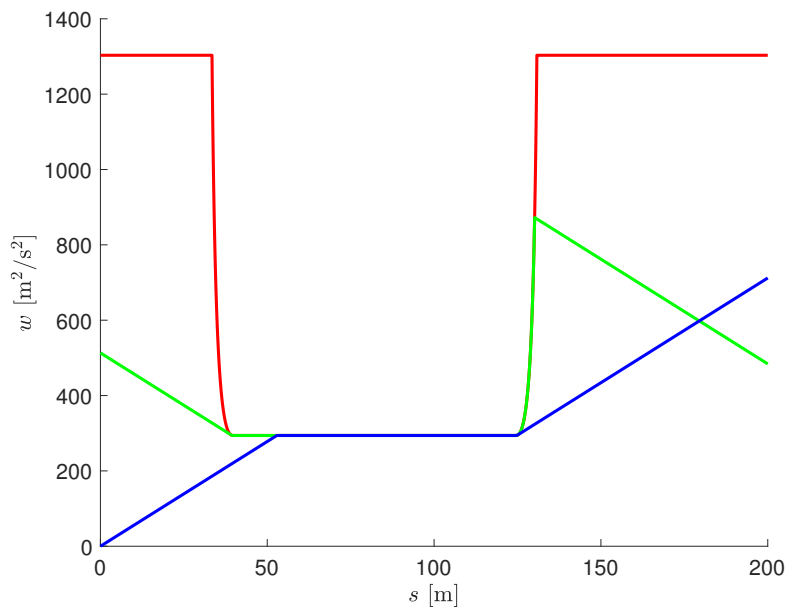


Figure 6: The red line represents μ^+ defined in (2), the blue one represents $F(\mu^+)$, whilst the green one represents $B(\mu^+)$.

μ^+ , α^- , α^+ are piecewise-constant, then w is piecewise linear (as in the simple

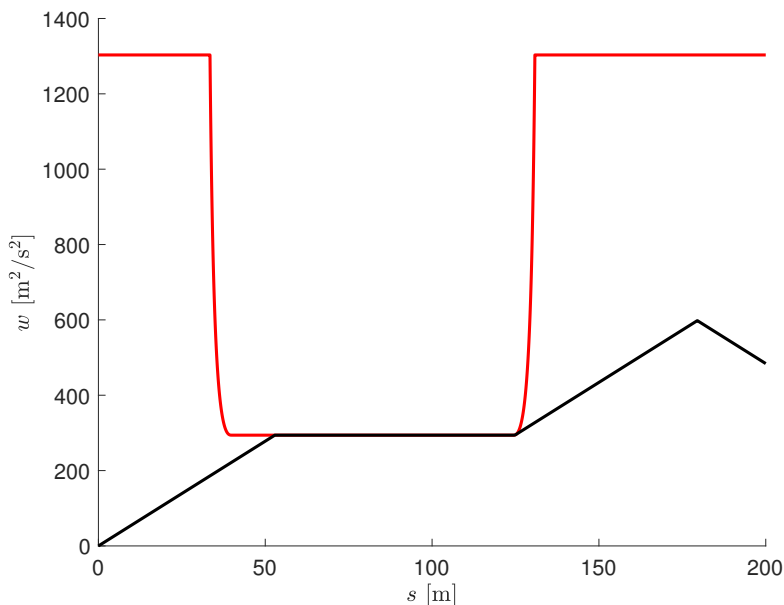


Figure 7: Example 1: The red line represents μ^+ defined in (2), whilst the black one represents optimal solution $w = M(\mu^+)$.

examples of Figures 2 and 3) and can be directly computed without actually integrating differential equations (4) and (5).

3.3 Bounded Acceleration Shortest Path Problem

Before defining BASP in a formal way, we present an example. Consider the setting represented in Figure 8. Here, the circles represent the positions of 7 AGV configurations, while the arrows represent the associated orientation angles. For instance, each configuration can be an operating point useful to the management of an automated warehouse. It may be a position along the racks, to insert or retrieve packages from the shelves, a position at the end of the production lines, to pickup finished products, or some intermediate location, used for routing. These configurations are connected by 10 fixed directed paths. We can associate a directed graph to this setting, reported in Figure 9. Namely, each configuration corresponds to a vertex and each path to a directed arc. We associate to each path bounds on maximum and minimum velocity and acceleration, that may depend on the arc-length position along the path, following the procedure presented in Section 3.1. Roughly speaking, solving BASP consists in finding the time-optimal motion from a source to a destination configuration. This requires finding both the geometrical path (i.e., the optimal sequence of directed arcs) and the time-optimal speed law along this path that satisfied the constraints associated to each travelled arc. Note that, once the path is known, this last task can be done with the method presented in Section 3.1.

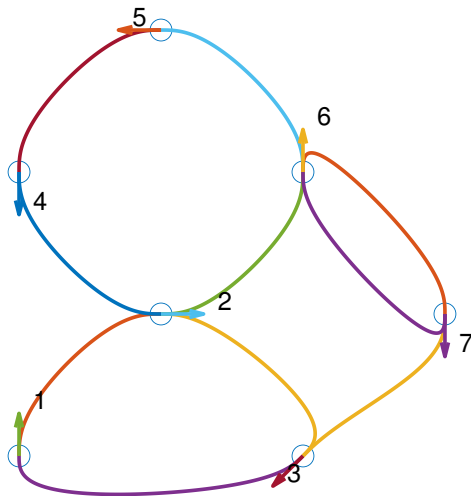


Figure 8: Layout with 7 positions.

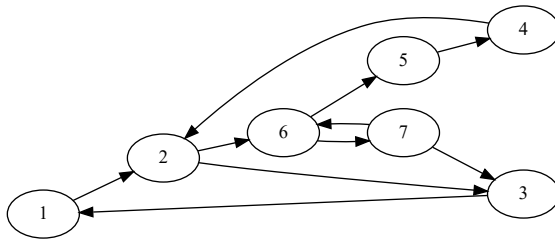


Figure 9: Directed graph associated to the setting in Figure 8.

We now present BASP problem in more general terms. Let us consider a directed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, with $\mathbb{V} = \{\sigma_1, \dots, \sigma_N\}$. Each node σ_i , $i \in \{1, \dots, N\}$, represents an operating point $Q_i \in \mathbb{R}^2$.

Each arc $\theta = (\sigma_i, \sigma_j) \in \mathbb{E}$ represents a fixed directed path between two operating points and is associated to an arc-length parameterized path γ_θ of length $\ell(\theta)$, such that $\gamma_\theta(0) = Q_i$ and $\gamma_\theta(\ell(\theta)) = Q_j$.

In the following, we denote the set of all possible paths on \mathbb{G} simply by P . Similarly, for $s, f \in \mathbb{V}$, we denote by P_s the subset of P consisting in all paths starting from s and by $P_{s,f}$ the subset of P consisting in all paths starting from s and ending in f . We extend this definition to subsets of \mathbb{V} , that is, if $S, F \subset \mathbb{V}$, we denote by $P_{S,F}$ the set of all paths starting from nodes in S and ending in nodes in F .

Given a path $p = \sigma_1 \cdots \sigma_m$, its length $\ell(p)$ is defined as the sum of the lengths

of its individual arcs, that is,

$$\ell(p) = \sum_{i=1}^{m-1} \ell(\sigma_i, \sigma_{i+1}).$$

To setup our problem, we need to associate four real-valued functions to each edge $\theta \in \mathbb{E}$: the maximum and minimum allowed acceleration and squared speed. The domain of each function is the arc-length coordinate on path γ_θ . Then, given a specific path p , we are able to define a speed optimization problem of form (3) by considering the constraints associated to the edges that compose p . We define the set of edge functions as

$$\mathcal{E} = \{\varphi : \mathbb{E} \times \mathbb{R}^+ \rightarrow \mathbb{R}\}.$$

If $\varphi \in \mathcal{E}$, $\theta \in \mathbb{E}$, $\lambda \in \mathbb{R}^+$, $\varphi(\theta, \lambda)$ denotes the value of φ on edge θ at position λ . Note that $\varphi(\theta, \lambda)$ will be relevant only for $\lambda \in [0, \ell(\theta)]$. Given a path $p = \sigma_1 \cdots \sigma_m$, we associate to $\varphi \in \mathcal{E}$ a function $\varphi_p : [0, \ell(p)] \rightarrow \mathbb{R}$ in the following way. Define functions $\Theta : [0, \ell(p)] \rightarrow \mathbb{N}$, $\Lambda : [0, \ell(p)] \rightarrow \mathbb{R}$ such that $\Theta(\lambda) = \max\{i \in \mathbb{N} \mid \ell(\sigma_1 \cdots \sigma_i) \leq \lambda\}$ and $\Lambda(\lambda) = \ell(\sigma_1 \cdots \sigma_{\Theta(\lambda)})$. In this way, $\Theta(\lambda)$ is such that $\theta(\lambda) = (\sigma_{\Theta(\lambda)}, \sigma_{\Theta(\lambda)+1})$ is the edge that contains the position at arc-length λ along p and $\Lambda(\lambda)$ is the sum of the lengths of all arcs up to node $\sigma_{\Theta(\lambda)}$ in p . Then, we define $\varphi_p(\lambda) = \varphi(\theta(\lambda), \lambda - \Lambda(\lambda))$.

Given $\hat{\mu}^+, \hat{\mu}^-, \hat{\alpha}^+, \hat{\alpha}^- \in \mathcal{E}$ and path $p \in P$, let $\mathbb{B} = (\hat{\mu}^-, \hat{\mu}^+, \hat{\alpha}^-, \hat{\alpha}^+)$. Assume $(\forall \theta \in \mathbb{E}) \hat{\mu}^+(\theta, \cdot) \in \mathcal{Q}_{\hat{\alpha}^-(\theta, \cdot), \hat{\alpha}^+(\theta, \cdot)}$ and define

$$T_{\mathbb{B}}(p) = \min_{w \in W^{1,\infty}([0, \ell(p)])} \Psi(w),$$

as the solution of Problem (3) along path p with $\mu^- = \hat{\mu}_p^-$, $\mu^+ = \hat{\mu}_p^+$, $\alpha^- = \hat{\alpha}_p^-$, $\alpha^+ = \hat{\alpha}_p^+$. In this way, $T_{\mathbb{B}}(p)$ is the minimum-time required to traverse path p , respecting the speed and acceleration constraints defined in \mathbb{B} . We set $T_{\mathbb{B}}(p) = +\infty$ if Problem (3) is not feasible.

The following is the main problem discussed in this paper.

Problem 3.2 (Bounded Acceleration Shortest Path Problem (BASP)). *Given a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, $\mu^+, \mu^-, \alpha^-, \alpha^+ \in \mathcal{E}$, $\mathbb{B} = (\mu^+, \mu^-, \alpha^-, \alpha^+)$, $s \in \mathbb{V}$, and $F \subset \mathbb{V}$, find*

$$\min_{p \in P_{s,F}} T_{\mathbb{B}}(p).$$

In other words, we want to find the path p that minimizes the transfer time between source node s and a destination node in F , taking into account bounds on speed and accelerations on each traversed arc (represented by arc functions $\mu^+, \mu^-, \alpha^-, \alpha^+$). The following properties are a direct consequence of the definition of $T_{\mathbb{B}}(p)$.

Proposition 3.3. *The following properties hold:*

i) If $p_1, p_2 \in P$ are such that $p_1 p_2 \in P$, then

$$T_{\mathbb{B}}(p_1 p_2) \geq T_{\mathbb{B}}(p_1) + T_{\mathbb{B}}(p_2).$$

ii) If $\mathbb{B} = (\mu^+, \mu^-, \alpha^-, \alpha^+)$, $\hat{\mathbb{B}} = (\hat{\mu}^+, \hat{\mu}^-, \hat{\alpha}^-, \hat{\alpha}^+)$ are such that $(\forall \theta \in \mathbb{E})$
 $(\forall \lambda \in [0, \ell(\theta)]) [\mu^-(\theta, \lambda), \mu^+(\theta, \lambda)] \subset [\hat{\mu}^-(\theta, \lambda), \hat{\mu}^+(\theta, \lambda)]$ and $[\alpha^-(\theta, \lambda),$
 $\alpha^+(\theta, \lambda)] \subset [\hat{\alpha}^-(\theta, \lambda), \hat{\alpha}^+(\theta, \lambda)]$, then $(\forall p \in P)$

$$T_{\mathbb{B}}(p) \geq T_{\hat{\mathbb{B}}}(p).$$

In particular, the first property states that the minimum time for travelling the composite path $p_1 p_2$ is greater or equal to sum of the times needed for travelling p_1 and p_2 separately. In fact, in the first case, the speed must be continuous when passing from p_1 to p_2 (due to the acceleration bounds), but this constraint does not need to be satisfied when the speed profiles for p_1 and p_2 are computed separately.

4 Complexity

We discuss the complexity of a simplified version of Problem 3.2, in which maximum and minimum acceleration and speed are constant on each arc.

Problem 4.1 (Bounded Acceleration Shortest Path Problem with constant bounds (BASP-C)). *Solve Problem 3.2 with the additional assumption that there exist functions $\alpha^-, \alpha^+, \mu^-, \mu^+ : \mathbb{E} \rightarrow \mathbb{R}$ such that, $(\forall \theta \in \mathbb{E}) (\forall \lambda \in \mathbb{R}^+) \alpha^-(\theta, \lambda) = \alpha^-(\theta)$, $\alpha^+(\theta, \lambda) = \alpha^+(\theta)$, $\mu^-(\theta, \lambda) = \mu^-(\theta)$, $\mu^+(\theta, \lambda) = \mu^+(\theta)$.*

We will show that BASP-C is NP-hard, which implies that the more general BASP is also NP-hard.

A special case of BASP-C is the classical Shortest Path (SP) problem, where a distance/time $d(\theta)$ is associated to each edge and a minimum distance/time path from source node s to destination node f is searched for. This is the special case when $\alpha^+(\theta) = +\infty$ and $\alpha^-(\theta) = -\infty$ for all edges $\theta \in \mathbb{E}$. In this case, speed can be changed instantaneously, so that we can run along each edge at the maximum allowed speed along that edge, so that $d(\theta) = \frac{\ell(\theta)}{\mu^+(\theta)}$. The classical SP problem is known to be solvable in polynomial time, e.g., by Dijkstra's algorithm. BASP-C can be viewed as a generalization of the SP problem, but, differently from SP, we prove that BASP-C is NP-hard. The following proposition characterizes the complexity of Problem 4.1.

Proposition 4.2. *Problem BASP-C is NP-hard.*

Proof. See Appendix 8.2. □

As said, this implies that also BASP is NP-hard. However, we also prove that, under additional assumptions, BASP admits a pseudo-polynomial algorithm, i.e., an algorithm running in polynomial time with respect to the values of the input but not with respect to the number of bits required to represent them.

Proposition 4.3. *Let us assume that the maximum and minimum acceleration along each arc are fixed values. W.l.o.g., we assume that $\alpha^+(\theta) = +1$ and $\alpha^-(\theta) = -1$ for each $\theta \in \mathbb{E}$. Moreover, we also assume that all lengths $\ell(\theta)$ are positive integer values. Then, BASP admits a pseudo-polynomial time algorithm.*

Proof. See Appendix 8.3 □

5 The k -BASP problem

As stated in Proposition 4.2, BASP is NP-hard. In the previous section we commented that SP can be viewed as a special case of BASP. In fact, also BASP can be viewed as an SP problem but defined on a different graph with respect to the original one. More precisely, here we introduce some restrictions on parameters \mathbb{B} that allow reducing BASP to a standard SP problem on an extended graph, that can be solved by Dijkstra's algorithm. Let $p \in P$, define

$$\ell^+(p) = \min \left\{ \left\{ \lambda \in [0, \ell(p)] \mid \int_0^\lambda \alpha_p^+(q) dq = \mu_p^+(\lambda) \right\}, +\infty \right\},$$

$$\ell^-(p) = \max \left\{ \left\{ \lambda \in [0, \ell(p)] \mid -\int_\lambda^{\ell(p)} \alpha_p^-(q) dq = \mu_p^+(\lambda) \right\}, -\infty \right\}.$$

In this way, $\ell^+(p)$ is the smallest value of $\lambda \in [0, \ell(p)]$ for which the solution of F in (4), with $\alpha^+ = \alpha_p^+$, starting from initial condition $F(0) = 0$, reaches the squared speed upper bound $\mu^+(\lambda)$. Note that $\ell^+(p) = \infty$ if no such value of λ exists. Similarly, $\ell^-(p)$ is the largest value of $\lambda \in [0, \ell(p)]$ for which the solution of B in (5), with $\alpha^- = \alpha_p^-$, starting from initial condition $B(\ell(p)) = 0$, reaches $\mu^+(\lambda)$. Again, $\ell^-(p) = -\infty$ if no such value of λ exists. Note that if $p, t, pt \in P$, $\ell^+(pt) \leq \ell^+(p)$ and $\ell^-(pt) \geq \ell^-(p)$ (actually, equalities hold if the values are all finite). Finally, we define

$$K(\mathbb{B}) = \min\{k \in \mathbb{N} \mid (\forall p \in P_s) |p| \geq k \Rightarrow \ell^+(p) \leq \ell^-(p)\}. \quad (7)$$

We call k -BASP any instance of Problem 3.2 that satisfies $K(\mathbb{B}) \leq k$. For instance, consider the simple graph depicted in Figure 10. Here, $\mathbb{V} = \{s, 1, 2, f\}$, $\mathbb{E} = \{(s, 1), (1, 2), (2, f)\}$, $(\forall \theta \in \mathbb{E}) \alpha^-(\theta) = -1$, $\alpha^+(\theta) = 1$, $\mu^-(\theta) = 0$, $\ell(\theta) = 1$, moreover $\mu^+((s, 1)) = 1$, $\mu^+((1, 2)) = \frac{2}{3}$, $\mu^+((2, f)) = 1$. In this case, $P_s = \{s, s1, s12, s12f\}$. Moreover, $K(\mathbb{B}) > 2$, since $\ell^+(s1) = 1 > 0 = \ell^-(s1)$ as reported in Figure 11. Further, $\ell^+(s12) < \ell^-(s12)$ and $\ell^+(12f) < \ell^-(12f)$ and $s12, 12f$ are the only paths of length 3. Figure 12 shows the computation of $\ell^+(s12)$ and $\ell^-(s12)$, the computation of $\ell^+(12f)$ and $\ell^-(12f)$ is analogous. Hence, in this example, $K(\mathbb{B}) = 3$.

Note that $K(\mathbb{B}) - 1$ represents the maximum number of vertices of a path that can be traveled with a speed profile of maximum acceleration, followed by one of maximum deceleration, starting and ending with zero speed, without violating



Figure 10: Simple graph with source node s and final node f .

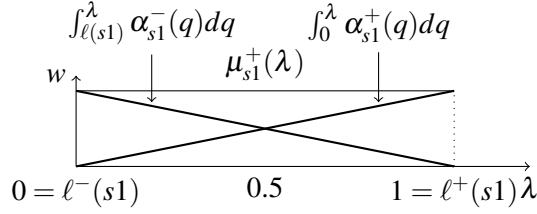


Figure 11: Computation of $\ell^+(s1) = 1$, $\ell^-(s1) = 0$.

the maximum speed constraint. The following expression provides a simple upper bound on $K(\mathbb{B})$

$$K(\mathbb{B}) \leq 1 + \left[2 \max_{\theta \in \mathbb{E}} \frac{\max_{\lambda \in [0, \ell(\theta)]} \mu^+(\theta, \lambda)}{\min_{\lambda \in [0, \ell(\eta)]} (|\alpha^+(\theta, \lambda) \wedge \alpha^-(\theta, \lambda)|) \ell(\theta)} \right]. \quad (8)$$

Note that $K(\mathbb{B}) = 1$ only if $\alpha_- = -\infty$ and $\alpha^+ = +\infty$, that is, if we do not consider acceleration bounds. We will present an algorithm that solves k -BASP in polynomial time complexity with respect to $|\mathbb{V}|$ and $|\mathbb{E}|$. However, note that the complexity is exponential with respect to k , so that a correct estimation of $K(\mathbb{B})$ is critical. In general, bound (8) overestimates $K(\mathbb{B})$. In section 6.2 we will present a simple method for correctly estimating $K(\mathbb{B})$.

Define $\text{Suff}_k : P \rightarrow \mathbb{V}_k$ such that, if $|p| \leq k$, $\text{Suff}_k(p) = p$ and if $|p| > k$, $\text{Suff}_k(p)$ is the suffix of p of length k . Function Suff_k allows to introduce a partially defined transition function $\Gamma : \mathbb{V}_k \times \mathbb{V} \rightarrow \mathbb{V}_k$ by setting

$$\Gamma(r, \sigma) = \begin{cases} \text{Suff}_k(r\sigma), & \text{if } r\sigma \in P \\ \text{is not defined,} & \text{if } r\sigma \notin P. \end{cases}$$

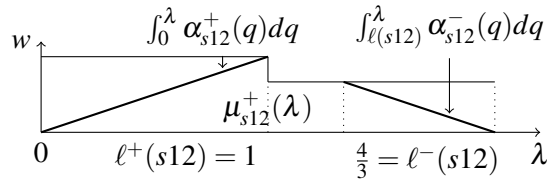


Figure 12: Computation of $\ell^+(s12) = 1$, $\ell^-(s12) = \frac{4}{3}$.

We recall that \mathbb{V}_k represents the subset of language \mathbb{V}^* composed of strings with maximum length k , including the empty string ε .

Define the incremental cost function $\eta : P_s \times \mathbb{V} \rightarrow \mathbb{R}^+$ such that, for $p \in P_s$ and $\sigma \in \mathbb{V}$,

$$\eta(p, \sigma) = \begin{cases} T_{\mathbb{B}}(p\sigma) - T_{\mathbb{B}}(p), & \text{if } p\sigma \in P_s \\ +\infty, & \text{otherwise.} \end{cases}$$

In other words, $\eta(p, \sigma)$ is the difference between the minimum-time required for traversing $p\sigma$ and the minimum-time required for traversing p . For simplicity of notation, from now on we will denote $T_{\mathbb{B}}$ simply as T . The following proposition shows that the incremental cost is always strictly positive.

Proposition 5.1. $\eta(p, \sigma) \geq T(\sigma)$.

Proof. By i) of Proposition 3.3, $T(p\sigma) \geq T(p) + T(\sigma)$. □

The following property, whose proof is presented in the Appendix, plays a key role in the solution algorithm.

Proposition 5.2. Let $p_1, p_2, t \in P$ be such that $p_1t, p_2t \in P$ and $\ell^+(t) \leq \ell^-(t)$, then $(\forall \sigma \in \mathbb{V})$

$$T(p_1t\sigma) - T(p_1t) = T(p_2t\sigma) - T(p_2t).$$

The following is a direct consequence of Proposition 5.2. It states that, given $p \in P$ and $\sigma \in \mathbb{V}$, the incremental cost $\eta(p, \sigma)$ does not depend on the complete path p , but only on $\text{Suff}_k(p)$ (its last k symbols).

Proposition 5.3. If $K(\mathbb{B}) \leq k$ and $p, p' \in P$ are such that $\text{Suff}_k(p) = \text{Suff}_k(p')$, then $(\forall \sigma \in \mathbb{V})$

$$\eta(p, \sigma) = \eta(p', \sigma).$$

Define function $\hat{\eta} : \mathbb{V}_k \times \mathbb{V} \rightarrow \mathbb{R}^+$, such that $\hat{\eta}(r, \sigma) = \eta(p, \sigma)$ where $p \in P$ is any path such that $r = \text{Suff}_k(p)$. We set $\hat{\eta}(r, \sigma) = +\infty$ if such path does not exist. Note that function $\hat{\eta}$ is well-defined by Proposition 5.3, being $\eta(p, \sigma)$ identical among all paths p such that $r = \text{Suff}_k(p)$. In particular, Proposition 5.3 holds for $p' = \text{Suff}_k(p) = r$, so that we can compute $\hat{\eta}$ as

$$\hat{\eta}(r, \sigma) = \eta(r, \sigma).$$

In the following, since $\hat{\eta}$ is the restriction of η on $\mathbb{V}_k \times \mathbb{V}$, we will denote $\hat{\eta}$ simply by η .

The value k can be viewed as the amount of memory required to solve the problem: once a node is reached, the optimal path from such node to the target one depends on the last k visited nodes. If $k = 1$, it only depends on the current node itself (i.e., no memory is required). This is the situation with the classical SP problem. More generally, $k > 1$, so that the optimal way to complete the path

does not only depend on the current node, but also on the sequence of $k - 1$ nodes visited before reaching it.

Define function $V : \mathbb{V}_k \rightarrow \mathbb{R}$ as

$$V(r) = \min_{p \in P_s | \text{Suff}_k p = r} T_{\mathbb{B}}(p). \quad (9)$$

Note that the solution of BASP corresponds to $\min_{r \in \mathbb{V}_k | \vec{r} \in F} V(r)$ (we recall that \vec{r} is the last vertex of r). For $r \in \mathbb{V}_k$, define the set of predecessors of r as $\text{Prec}(r) = \{\vec{r} \in \mathbb{V}_k \mid r = \Gamma(\vec{r}, \vec{r})\}$. The following proposition presents an expression for $V(r)$ that holds if condition $\ell^+(r') \leq \ell^-(r')$ is satisfied for all predecessors r' of r .

Proposition 5.4. *Let $r \in \mathbb{V}_k$, if $(\forall r' \in \text{Prec}(r)) \ell^+(r') \leq \ell^-(r')$, then*

$$V(r) = \min_{r' \in \text{Prec}(r)} \{V(r') + \eta(r', \vec{r})\}. \quad (10)$$

Proof.

$$\begin{aligned} V(r) &= \min_{p \in P_s | \text{Suff}_k p = r} T(p) = \\ &= \min_{q \in P_s | \text{Suff}_k q\vec{r} = r} \{T(q\vec{r}) - T(q) + T(q)\} = \\ &= \min_{q \in P_s | \text{Suff}_k q\vec{r} = r} \{T(q) + T((\text{Suff}_k q)\vec{r}) - T(\text{Suff}_k q)\} = \\ &= \min_{q \in P_s | \text{Suff}_k q\vec{r} = r} \{T(q) + \eta(\text{Suff}_k q, \vec{r})\} = \\ &= \min_{q \in P_s, r' \in \text{Prec}(r) | \text{Suff}_k q = r'} \{T(q) + \eta(r', \vec{r})\} = \\ &= \min_{r' \in \text{Prec}(r)} \{V(r') + \eta(r', \vec{r})\}, \end{aligned}$$

where we used the facts that $T(q\sigma) - T(q) = T(\text{Suff}_k q\sigma) - T(\text{Suff}_k q)$, by Proposition 5.2, and that $q \in P_s$ is such that $\text{Suff}_k q\vec{r} = r$ if and only if $\text{Suff}_k q \in \text{Prec}(r)$. \square

As a consequence of Proposition 5.4, if $(\forall r \in \mathbb{V}_k) \ell^+(r) \leq \ell^-(r)$, $V(r)$ corresponds to the length of the shortest path from s to r on the extended directed graph $\tilde{\mathbb{G}} = (\tilde{\mathbb{V}}, \tilde{\mathbb{E}})$, where $\tilde{\mathbb{V}} = \mathbb{V}_k$ and $(r_1, r_2) \in \tilde{\mathbb{E}}$ if $r_2 = \Gamma(r_1, \vec{r}_2)$ is defined, in this case its length is $\eta(r_1, \vec{r}_2)$. The upper part of Figure 13 shows a graph consisting of 3 nodes. Node $s = 1$ is the source (indicated by the entering arrow) and the double border shows the final node $F = \{3\}$. The lower part of Figure 13 represents the corresponding extended graph, obtained for $k = 2$, consisting of 13 nodes (the cardinality of \mathbb{V}_2). Note that some of the nodes are unreachable from the initial state, these are represented with dotted edges.

Solving k -BASP corresponds to finding a minimum-length path on $\tilde{\mathbb{G}}$ that connects node $s \in \mathbb{V}_k$ to $\hat{F} = \{r \in \mathbb{V}_k \mid \vec{r} \in F\}$. Note that the set of final states for the extended graph \hat{F} contains all paths $p \in \mathbb{V}_k$ that end in an element of F . In the extended graph reported in Figure 13, this corresponds to finding a minimum-length path from starting node 1 to one of the final nodes 3, 13, 23, 33. Note that the unreachable nodes play no role in this procedure. We can find a minimum-length path by Dijkstra's algorithm applied on $\tilde{\mathbb{G}}$, leading to the following complexity result.

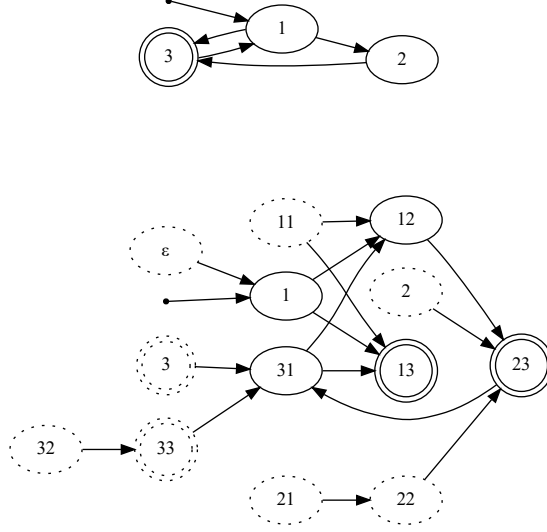


Figure 13: A graph and the corresponding extended graph for $k = 2$.

Proposition 5.5. k -BASP can be solved with complexity $O(|\mathbb{V}|^{k-1}|\mathbb{E}| + (|\mathbb{V}|^k \log |\mathbb{V}|^k))$.

Proof. Dijkstra's algorithm has time complexity $O(|E| + |V| \log |V|)$, where $|E|$ and $|V|$ are the cardinalities of the edge and vertex sets. In our case, $|V| = |\tilde{\mathbb{V}}| = |\mathbb{V}_k| = \sum_{i=0}^k |\mathbb{V}|^i = O(|\mathbb{V}|^k)$, $|E| = |\tilde{\mathbb{E}}| \leq |\mathbb{V}_{k-1}\mathbb{E}| = O(|\mathbb{V}|^{k-1}|\mathbb{E}|)$, which imply the thesis. \square

The following remark establishes again that SP can be viewed as a special case of BASP when no bound on the acceleration is imposed.

Remark 5.6. If $(\forall \sigma \in \mathbb{V}) (\forall \lambda) \alpha^-(\sigma, \lambda) = -\infty, \alpha^+(\sigma, \lambda) = +\infty$, then $K(\mathbb{B}) = 1$. The resulting 1-BASP reduces to a standard SP problem on graph \mathbb{G} and can be solved with time complexity $O(|\mathbb{E}| + |\mathbb{V}| \log |\mathbb{V}|)$.

6 Adaptive A* algorithm for k -BASP

The computation method based on Dijkstra's algorithm on the extended graph $\tilde{\mathbb{G}}$, presented in the previous section, has two main disadvantages. First, the extended graph has $\sum_{j=1}^k |\mathbb{V}|^j$ nodes, so that the time required by Dijkstra's algorithm grows exponentially with k . We will show that it is possible to mitigate this problem and reduce the number of visited nodes by using A* algorithm with a suitable heuristic. Second, the estimation of $k = K(\mathbb{B})$ from its definition is not an easy task. We will show that it is quite easy to adaptively find the correct value of k by starting from $k = 2$ and increasing k if needed.

6.1 Upper bounds on $T_{\mathbb{B}}(p)$

To implement the A^* algorithm, we need to define a heuristic function $h : \mathbb{V}_k \rightarrow \mathbb{R}$, such that, for $r \in \mathbb{V}_k$, $h(r)$ is a lower bound on $\min_{p \in P_{\vec{r}, \hat{F}}} T(p)$, that is, the minimum time needed for traveling from \vec{r} to a final state in \hat{F} . In general, we can compute lower bounds for BASP by relaxing the acceleration constraints α^- , α^+ . Namely, let $\hat{\mathbb{B}}$ be a parameter set obtained by relaxing acceleration constraints in \mathbb{B} . Then, if $K(\hat{\mathbb{B}}) < K(\mathbb{B})$, by Proposition 5.5, the solution of BASP for parameters $\hat{\mathbb{B}}$ can be computed with a lower computational time than the solution with parameters \mathbb{B} . In particular, we obtain a very simple lower bound by removing acceleration bounds altogether, that is, by setting $\alpha^- = -\infty$ and $\alpha^+ = +\infty$. In this way, the vehicle is allowed to travel at maximum speed everywhere along the path and the incremental cost function $\eta(p, \sigma)$ is given by the time needed to travel γ_σ at maximum speed, that is:

$$\eta(p, \sigma) = \int_0^{\ell(\vec{p}\sigma)} \frac{1}{\sqrt{\mu^+(\vec{p}, \sigma, \lambda)}} d\lambda.$$

Define the heuristic $h : \mathbb{V}_k \rightarrow \mathbb{R}^+$ as

$$h(r) = \min_{p \in P_{\vec{r}, \hat{F}}} T_{\hat{\mathbb{B}}}(p). \quad (11)$$

Note that, if $\alpha^- = -\infty$ and $\alpha^+ = +\infty$, h corresponds to the solution of 1-BASP and all values of h can be efficiently precomputed by Dijkstra's algorithm (see Remark 5.6).

The following proposition shows that h is admissible and consistent, so that A^* algorithm, with heuristic h , provides the optimal solution of k -BASP and its time-complexity is no worse than Dijkstra's algorithm (see for instance Theorems 2.9 and 2.10 of [5]).

Proposition 6.1. *Heuristic h satisfies the following two properties:*

- i) $(\forall r \in \mathbb{V}_k) h(r) \leq \min_{q \in P_{\vec{r}, \hat{F}}} T_{\hat{\mathbb{B}}}(q)$ (admissibility).
- ii) $(\forall r \in \mathbb{V}_k) (\forall \sigma \in \mathbb{V}) h(r) \leq \eta(r, \sigma) + h(\Gamma(r, \sigma))$ (consistency).

Proof. i) $h(r) = \min_{p \in P_{\vec{r}, \hat{F}}} T_{\hat{\mathbb{B}}}(p) \leq \min_{q \in P_{\vec{r}, \hat{F}}} T_{\mathbb{B}}(q)$, since $\hat{\mathbb{B}}$ is a relaxation of \mathbb{B} .

ii) $h(r) = \min_{p \in P_{\vec{r}, \hat{F}}} T_{\hat{\mathbb{B}}}(p) \leq T_{\hat{\mathbb{B}}}(\sigma) + \min_{p \in P_{\sigma, \hat{F}}} T_{\hat{\mathbb{B}}}(p) \leq T_{\mathbb{B}}(\sigma) + \min_{p \in P_{\sigma, \hat{F}}} T_{\mathbb{B}}(p) \leq \eta(r, \sigma) + \min_{p \in P_{\sigma, \hat{F}}} T_{\mathbb{B}}(p) = \eta(r, \sigma) + h(\Gamma(r, \sigma))$, where $T_{\hat{\mathbb{B}}}(\sigma) \leq T_{\mathbb{B}}(\sigma)$ by ii) of Proposition 3.3 and $T_{\mathbb{B}}(\sigma) \leq \eta(r, \sigma)$ by Proposition 5.1. \square

Since heuristic h is admissible and consistent, A^* is equivalent to Dijkstra's algorithm, with the only difference that the incremental cost function $\eta(r, \sigma)$ is substituted with modified cost

$$\tilde{\eta}(r, \sigma) = \eta(r, \sigma) + h(\Gamma(r, \sigma)) - h(r) \quad (12)$$

(see Lemma 2.3 of [5] for a complete discussion). A description of A^* algorithm can be found in literature (for instance, see Algorithm 2.13 of [5]). For the sake of

completeness, we report a possible implementation. We define a priority queue \mathcal{Q} that contains open nodes, that is, nodes that have already been generated but have not yet been visited. Namely, \mathcal{Q} is an ordered set of pairs $(r, t) \in \mathbb{V}_k \times \mathbb{R}^+$, in which $r \in \mathbb{V}_k$ and t is a lower bound for the time associated to the best completion of r to a path arriving at a final state. We need to perform the following operations on \mathcal{Q} : find its element with the minimal t -value, insert a pair, and update the queue if a node improves its t -value due to the discovery of a shorter path. Accordingly, we define the following operations on \mathcal{Q} . Operation $\text{INSERT}(\mathcal{Q}, (r, t))$ inserts couple (r, t) into \mathcal{Q} , operation $(r, t) = \text{DELETEMIN}(\mathcal{Q})$ returns the first couple of \mathcal{Q} , that is, the couple (r, t) with the minimum time t , and removes this couple from \mathcal{Q} . Finally, operation $\text{DECREASEKEY}(\mathcal{Q}, (r, t))$ assumes that \mathcal{Q} already contains a couple (r, t') with $t' > t$ and substitutes this couple with (r, t) . Further, we consider three partially defined maps $\text{VALUE} : \mathbb{V}_k \rightarrow \mathbb{R}$, $\text{PARENT} : \mathbb{V}_k \rightarrow \mathbb{V}_k$, $\text{CLOSED} : \mathbb{V}_k \rightarrow \{0, 1\}$, such that, for $r \in \mathbb{V}_k$, $\text{VALUE}(r)$ is the current best upper estimate of $V(r)$, $\text{PARENT}(r)$ is the parent node of r and $\text{CLOSED}(r) = 1$ if node r has already been visited. Maps VALUE , PARENT , and CLOSED can be implemented as hashtables. For a complete discussion on A* algorithm and the data structures involved, we refer again the reader to [5].

Algorithm 6.2 (A* algorithm for k-BASP).

- 1) [initialization] Set $\mathcal{Q} = \{(s, h(s))\}$, $\text{VALUE}(s) = 0$.
- 2) [expansion] Set $(r, t) = \text{DELETEMIN}(\mathcal{Q})$ and set $\text{CLOSED}(r) = 1$. If $\vec{r} \in \hat{F}$, then t is the optimal solution and the algorithm terminates, returning maps VALUE , PARENT . Otherwise, for each $\sigma \in \mathbb{V}$ for which $\Gamma(r, \sigma)$ is defined, set $r' = \Gamma(r, \sigma)$, $t' = t + \tilde{\eta}(r, \sigma)$. If $\text{CLOSED}(r') = 1$, go to 3). Else, if $\text{VALUE}(r')$ is undefined $\text{INSERT}(\mathcal{Q}, (r', t'))$. Otherwise, if $t' < \text{VALUE}(r')$, set $\text{VALUE}(r') = t'$, $\text{PARENT}(r') = r$ and do $\text{DECREASEKEY}(\mathcal{Q}, (r', t'))$.
- 3) [loop] If \mathcal{Q} is not empty go back to 2), otherwise no solution exists.

Proposition 6.3. *Algorithm 6.2 terminates and returns the optimal solution (if it exists), with a time-complexity not higher than Dijkstra's algorithm.*

Proof. It is a consequence of the fact that heuristic h is admissible and consistent (see, for instance, Theorems 2.9 and 2.10 of [5]). \square

Note that, at the end of Algorithm 6.2, $\text{VALUE}(f)$ is the optimal value of k-BASP and the optimal path from s to set F can be reconstructed from map PARENT .

6.2 Adaptive search for k

One possible limitation of Algorithm 6.2 is that estimating $K(\mathbb{B})$ from its definition can be difficult. A correct estimation of $K(\mathbb{B})$ is critical for the efficiency of the algorithm. Indeed, if $K(\mathbb{B})$ is overestimated, the time-complexity of the algorithm is higher than it would be with a correct estimate. On the other hand, if $K(\mathbb{B})$ is

underestimated, Algorithm 6.2 is not correct since Proposition 5.4 does not hold. Here we propose an algorithm that adaptively find a suitable value for k in Algorithm 6.2, that may be lower or equal to the true value of $K(\mathbb{B})$, but, in any case, allows to find the optimal solution of BASP. First, we define the modified cost function $W : \mathbb{V}_k \rightarrow \mathbb{R}$ as

$$W(r) = V(r) + h(r),$$

where V is given by (9) and h is the heuristic given by (11).

If $(\forall r \in \mathbb{V}_k) \ell^+(r) \leq \ell^-(r)$, then W is the solution of

$$\begin{cases} W(r) = \min_{r' \in \text{Prec}(r)} \{W(r') + \tilde{\eta}(r', \vec{r})\} \\ W(s) = h(s) \end{cases} \quad (13)$$

Indeed, following the same steps of the proof of Proposition 5.4

$$\begin{aligned} W(r) &= V(r) + h(r) = \\ &\min_{r' \in \text{Prec}(r)} \{V(r') + \eta(r', \vec{r}) + h(r) + h(r') - h(r')\} \\ &= \min_{r' \in \text{Prec}(r)} \{W(r') + \tilde{\eta}(r', \vec{r})\}. \end{aligned}$$

Hence, $W(r)$ corresponds to the length of the shortest path from s to r on $\tilde{\mathbb{G}}$, with arc-length given according to $\tilde{\eta}$. If condition $\ell^+(r) \leq \ell^-(r)$ is not satisfied for all $r \in \mathbb{V}_k$, equation (13) does not hold for all $r \in \mathbb{V}_k$ and W does not represent the solution of a shortest path problem. However, the following proposition shows that we can still find a lower bound \hat{W} of W that does correspond to the solution of a shortest path problem.

Proposition 6.4. *Let $\hat{W} : \mathbb{V}_k \rightarrow \mathbb{R}$ be the solution of*

$$\begin{cases} \hat{W}(r) = \min_{r' \in \text{Prec}(r)} \{\hat{W}(r') + \hat{\eta}(r', \vec{r})\} \\ \hat{W}(s) = 0 \end{cases} \quad (14)$$

where

$$\hat{\eta}(r', \vec{r}) = \begin{cases} \tilde{\eta}(r', \vec{r}), & \text{if } \ell^+(r') \leq \ell^-(r') \text{ or } |r'| < k \\ h(r) - h(r'), & \text{otherwise.} \end{cases}$$

Then, $(\forall r \in \mathbb{V}_k)$

i) $\hat{W}(r) \leq W(r)$.

ii) if $(\forall \vec{r} \in \mathbb{V}_k \mid \hat{W}(\vec{r}) \leq \hat{W}(r)) \ell^+(\vec{r}) \leq \ell^-(\vec{r})$, then $\hat{W}(r) = W(r)$.

Proof. i) For $r \in \mathbb{V}_k$, let $p \in P_s$ be such that $\text{Suff}_k p \in \text{Prec}(r)$. If $\ell^+(\text{Suff}_k p) \leq \ell^-(\text{Suff}_k p)$, in view of Proposition 5.2, $T(p\vec{r}) = T(p) + \eta(\text{Suff}_k p, \vec{r})$, otherwise, obviously, $T(p\vec{r}) \geq T(p)$. Hence, in both cases, by the definition of $\tilde{\eta}$ in (12), $T(p\vec{r}) + h(r) \geq T(p) + h(\text{Suff}_k p) + \tilde{\eta}(\text{Suff}_k p, \vec{r})$. By contradiction, assume that

there exists a non-empty subset $A \subset \mathbb{V}_k$ such that $(\forall r \in A) \hat{W}(r) > W(r)$. Let $\bar{r} = \operatorname{argmin}_{\hat{r} \in A} W(\hat{r})$, then,

$$\begin{aligned} W(\bar{r}) &= V(\bar{r}) + h(\bar{r}) = \min_{p \in P_s | \operatorname{Suff}_k p = \bar{r}} T(p) + h(\bar{r}) = \\ &\min_{q \in P_s | \operatorname{Suff}_k q \in \operatorname{Prec}(\bar{r})} T(q\vec{r}) + h(\bar{r}) \geq \\ &\min_{q \in P_s | \operatorname{Suff}_k q \in \operatorname{Prec}(\bar{r})} \{T(q) + h(\operatorname{Suff}_k(q)) + \hat{\eta}(\operatorname{Suff}_k q, \vec{r})\} = \\ &\min_{r' \in \operatorname{Prec}(\bar{r})} \{\hat{W}(r') + \hat{\eta}(r', \vec{r})\} = \hat{W}(\bar{r}), \end{aligned}$$

where we used the fact that $W(r') = \hat{W}(r')$, that follows from the definition of \bar{r} , since the value of r' that attains the minimum is such that $W(r') < W(\bar{r})$. Then, the obtained inequality contradicts the fact that $\hat{W}(\bar{r}) > W(\bar{r})$.

ii) Let $A \subset \mathbb{V}$ be the set of values of $r \in \mathbb{V}$ for which ii) does not hold and, by contradiction, assume that A is not empty and let $\hat{r} = \operatorname{argmin}_{r \in A} \hat{W}(r)$. Then, by the definition of \hat{r} , it satisfies the following two properties. First, $(\forall \bar{r} \in \mathbb{V}_k | \hat{W}(\bar{r}) \leq \hat{W}(\hat{r})) \ell^+(\bar{r}) \leq \ell^-(\bar{r})$, moreover $\hat{W}(\hat{r}) \neq W(\hat{r})$.

Note that, from the definitions of \hat{W} , $W(s) = \hat{W}(s)$. Then,

$$\begin{aligned} W(\hat{r}) &= \min_{p \in P_s | \operatorname{Suff}_k p = \hat{r}} T(p) + h(\hat{r}) = \\ &\min_{q \in P_s | \operatorname{Suff}_k q \in \operatorname{Prec}(\hat{r})} \{T(q\vec{r}) + h(\operatorname{Suff}_k q) - h(\operatorname{Suff}_k q) + h(\hat{r})\} \\ &= \min_{r' \in \operatorname{Prec}(\hat{r})} \{\hat{W}(r') + \hat{\eta}(r', \vec{r})\} = \hat{W}(\hat{r}), \end{aligned}$$

which contradicts the definition of \hat{r} . Here, we used equation (12) and the fact that, since $\hat{W}(r') < \hat{W}(\hat{r})$ and by the definition of \hat{r} , $\hat{W}(r') = W(r')$. \square

Proposition 6.4 implies that $\hat{W}(r)$ is a lower bound of $W(r)$ and that it corresponds to the length of the shortest path from s to r on the extended directed graph \tilde{G} , with arc-length given in accordance to (14), namely by the value of function $\hat{\eta}$. Hence, $\hat{W}(f)$ can be computed by Dijkstra's algorithm (which is equivalent to compute V with A* algorithm, with heuristic h). The algorithm that we are going to present is based on the following basic observation. If A* algorithm computes $f^* = \operatorname{argmin}_{f \in \hat{F}} \hat{W}(f)$ by visiting only nodes for which $\ell^+(r) \leq \ell^-(r)$, then ii) of Proposition 6.4 is satisfied for $r = f^*$ and $\hat{W}(f^*) = W(f^*)$ is the optimal value of k -BASP. If this is not the case, we increase k by 1 and re-run the A* algorithm. Note that the algorithm starts with $k = 2$, since, according to its definition, $K(\mathbb{B}) = 1$ only if no acceleration bounds are present and, in this case, BASP is equivalent to a standard SP and can be solved by Dijkstra's algorithm.

Algorithm 6.5 (Adaptive A* algorithm for k -BASP).

1) Set $k = 2$.

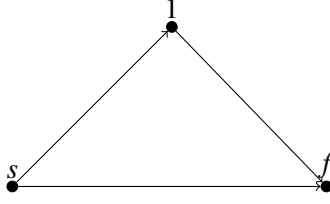


Figure 14: Simple graph considered in Example 6.6.

- 2) Execute A^* algorithm and, at every visit of a new node r , if none of the two conditions $\ell^+(r) \leq \ell^-(r)$ and $|r| < k$ hold, set $k = k + 1$ and repeat step 2).

Note that the algorithm does not compute the exact value $K(\mathbb{B})$. Rather, it underestimates it. More precisely, it stops with the smallest k value needed to solve BASP problem between the given source and destination nodes. This is illustrated by the following example.

Example 6.6. *Let*

$$\mathbb{G} = (\mathbb{V}, \mathbb{E}), \quad \mathbb{V} = \{s, 1, f\}, \quad \mathbb{E} = \{(s, 1), (1, f), (s, f)\},$$

be the graph represented in Figure 14, with the following set of bounds \mathbb{B} :

$$\begin{aligned} (s, 1) &\rightarrow \alpha^- = -1, \alpha^+ = 1, \mu^- = 0, \mu^+ = 4 \\ (1, f) &\rightarrow \alpha^- = -1, \alpha^+ = 1, \mu^- = 0, \mu^+ = 4 \\ (s, f) &\rightarrow \alpha^- = -2, \alpha^+ = 2, \mu^- = 0, \mu^+ = 3, \end{aligned}$$

and edge lengths

$$\ell((s, 1)) = 2, \quad \ell((1, f)) = 2, \quad \ell((s, f)) = 3.$$

The speed is further bounded to be equal to 0 both in s and in f . In this case it is easily seen that $K(\mathbb{B}) = 3$, since along path $s1f$ the maximum speed is never reached under the given bounds on the acceleration and the graph does not contain paths with more vertices. However, the A^* algorithm is first run with $k = 2$. With such value, the heuristic has the following value for the different paths of length less or equal than $k = 2$

$$\begin{aligned} h(s) &= 1, \quad h(1) = 0.5, \quad h(f) = 0, \\ h(s1) &= 0.5, \quad h(1f) = 0, \quad h(sf) = 0. \end{aligned}$$

These are easily computed by solving an SP problem with edge lengths equal to

$$d_{s1} = 0.5, \quad d_{1f} = 0.5, \quad d_{sf} = 1,$$

obtained by the formula $d_e = \frac{\ell(e)}{\mu^+(e)}$ for each edge e . The queue \mathcal{Q} is then initialized with $\{(s, 1)\}$ with $\text{VALUE}(s) = 0$. Next, we remove $(s, 1)$ from the queue and set $\text{CLOSED}(s) = 1$, and we insert in the queue $(1, h(s) + T_{\mathbb{B}}(s1) - T_{\mathbb{B}}(s) + h(1) - h(s)) = (1, 2.5)$ and $(1, h(s) + T_{\mathbb{B}}(sf) - T_{\mathbb{B}}(s) + h(f) - h(s)) = (1, \sqrt{6})$, and we set

$$\text{PARENT}(1) = \text{PARENT}(f) = s.$$

Thus,

$$\mathcal{Q} = \left\{ \left(f, \sqrt{6} \right), (1, 2.5) \right\}.$$

Since the minimum is attained by $(f, \sqrt{6})$, we remove it from the queue, we check whether $\ell^+(sf) \leq \ell^-(sf)$, which is the case since $\ell^+(sf) = \ell^-(sf) = 1.5$, and we stop since we reached the target node f . The minimum path is recovered from PARENT (in this case it is simply path sf) and the minimum time to travel from s to f is $\sqrt{6}$.

Proposition 6.7. Algorithm 6.2 terminates with $k \leq K(\mathbb{B})$ and returns an optimal solution.

Proof. By Definition of K , if $k = K(\mathbb{B})$ condition $\ell^+(r) \leq \ell^-(r)$ is satisfied for all r . Hence, there exists $k \leq K(\mathbb{B})$ for which the algorithm terminates. Let $r \in \mathbb{V}_k$, with $\vec{r} \in F$ be the last-visited node before the termination of the algorithm. By ii) of Proposition 6.4, we have that $\hat{W}(r) = W(r) = V(r)$ (since $h(r) = 0$), but, by definition, $V(r)$ is the shortest time for reaching a node in F . \square

7 Numerical experiments

7.1 Nodes associated to different orientations

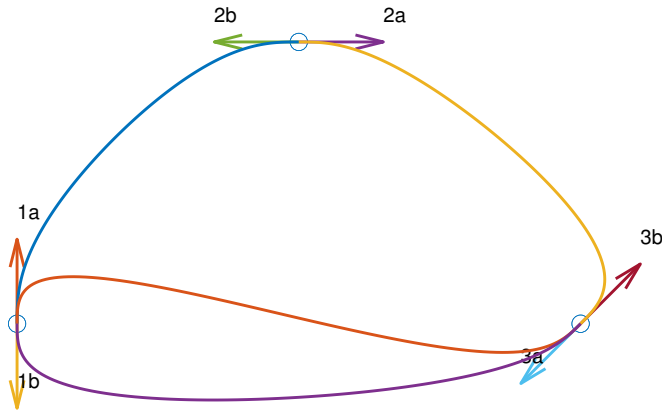


Figure 15: Graph with replicated nodes for the two possible directions.

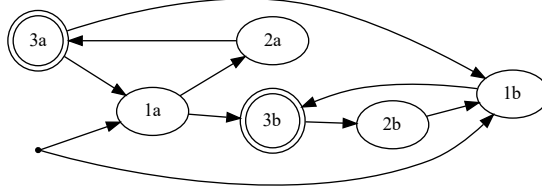


Figure 16: Directed graph associated to the setting in Figure 15.

Consider the setting represented in Figure 15. There are 3 positions connected by 4 paths. The paths are given by spline curves and are chosen in order to have a nonzero continuous first derivative at connection points. In this way, the path obtained by combining two adjacent arcs has piecewise-continuous curvature. In order to associate a graph to the setting of Figure 15, we actually need to assign two nodes to each position, associated to opposite curve directions. For instance, there is a direct arc from node $1a$ to node $2a$, but not from $1a$ to $2b$, since node $2b$ is associated to a direction which is opposite to the one that we would obtain by following the path from the first to the second position. In this way, the setting of Figure 15 is associated to the graph reported in Figure 16. Here, position 1 is the initial one and is associated to the two initial nodes $1a$ and $1b$. This is due to the fact that we assume that the vehicle is initially at rest, so that it can start along both directions associated to $1a$ and $1b$. Similarly, final position 3 is associated to the two states $3a$, $3b$, due to the fact that we accept both orientations for reaching the final position. Handling two initial states is not problematic, since it is sufficient to solve the problem twice, starting from both initial states $1a$ and $1b$, and then choosing the best solution.

7.2 A 16-vertex graph

We run all simulations on an Intel core i5 (7200u) with 8GB of RAM. As a simple example, we consider the 16-configuration setting represented in Figure 17. Each configuration $i \in \{1, \dots, 16\}$ is associated to a direction $\theta_i \in [0, 2\pi]$ and to a position $Q_i \in \mathbb{R}^2$. According to the method presented in Section 7.1, we associate the setting of Figure 17 to a graph with 32 nodes. In order to satisfy the maximum acceleration constraint, for each edge we set the constant squared speed bound $\mu^*((i, j)) = a_N r_{ij}$, where r_{ij} is the minimum curvature radius of the path that connects Q_i to Q_j . The normal acceleration $a_N = 2 \text{ m/s}^2$ and the maximum tangential acceleration and deceleration $\alpha^+ = -\alpha^- = 0.5 \text{ m/s}^2$ are constant and equal for all arcs.

As an example, we chose as source configuration $s = 16$ and, as final one, $f = 6$, and we computed three different solutions:

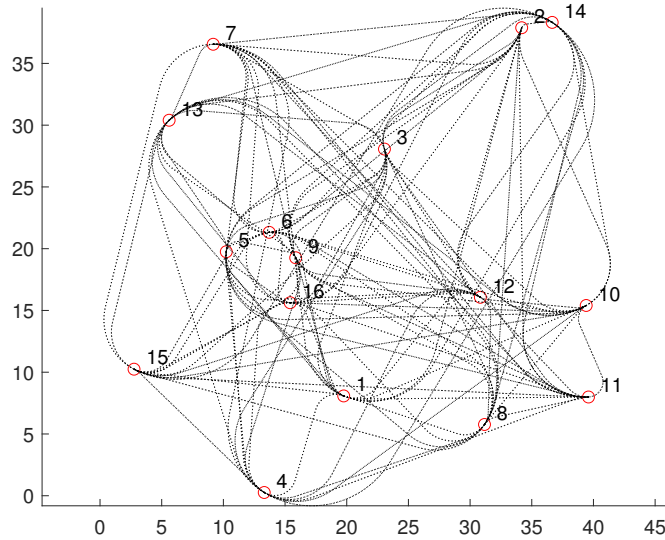


Figure 17: Graph with 16 nodes.

- the solution of BASP;
- the solution of BASP with infinite acceleration and deceleration (1-BASP);
- the shortest path (SP).

Note that the solutions of SP and 1-BASP can be computed by Dijkstra's algorithm. To compute the solution of BASP, we used Algorithm 6.5. Figure 18 represents the solutions of the three problems. Note that, in this case, they are all different. In particular, in Figure 19, we show the speed profile of the solution of BASP, while, in Figure 20, we show the speed profile of the solution of 1-BASP, which is the solution of BASP with infinite acceleration. Observe that the path obtained as the solution of 1-BASP, being 49 m long, is longer than the path obtained as the solution of BASP, which is 42 m long. However, if we allow infinite acceleration, this longer path is the minimum-time one.

The path corresponding to the solution of BASP changes according to the chosen acceleration bounds. In particular, if we choose a small enough acceleration bound, for example $\alpha^+ = -\alpha^- = 0.1 \text{ m/s}^2$, then the path corresponding to the solution of BASP coincides with the shortest one. Instead, if the acceleration bounds are large enough, for example $\alpha^+ = -\alpha^- = 1 \text{ m/s}^2$, the path corresponding to the solution of BASP coincides with the one obtained from the solution of 1-BASP (i.e., the infinite acceleration fastest path).

7.3 Randomly generated problems

We performed various tests on randomly generated problems of different sizes, obtained with the following procedure. First, we generated a random graph with n

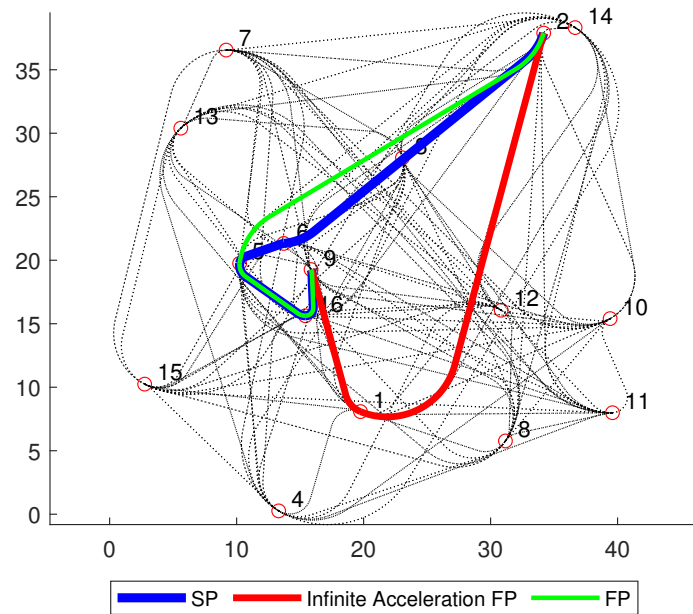


Figure 18: The three different solutions of BASP, 1-BASP and SP.

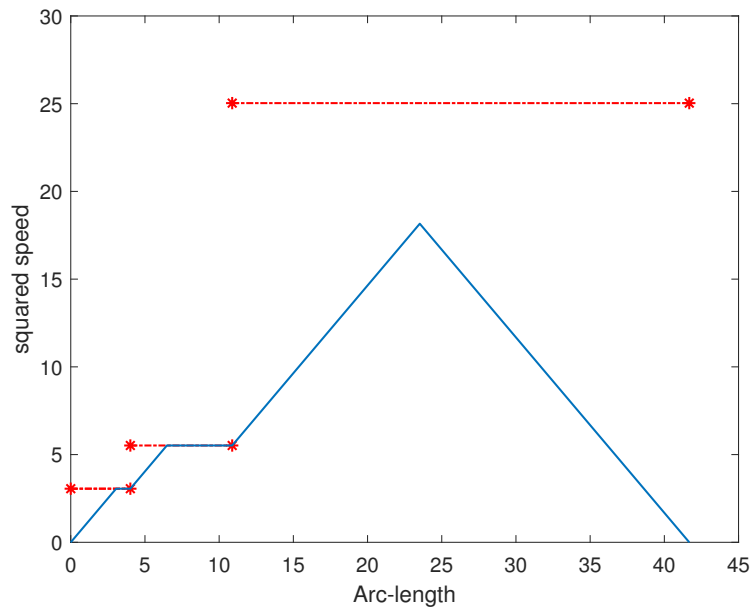


Figure 19: Speed profile of the solution of BASP with $\alpha^+ = -\alpha^- = 0.5 \text{ m/s}^2$.

nodes with Python package NetworkX (networkx.org), using function `geographical_threshold_graph`. Essentially, each node is associated to a position, obtained by choosing a random element of set $[0, 1] \times [0, 1]$. The edges

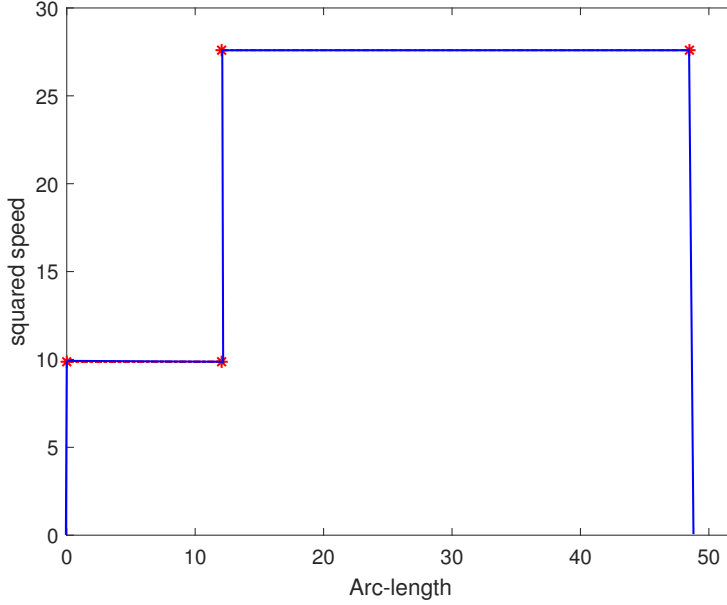


Figure 20: Speed profile of the solution of 1-BASP with infinite acceleration.

are randomly determined in such a way that closer nodes have a higher connection probability. We multiplied the obtained position by factor $10\sqrt{n}$, in order to obtain the same average nodes density independently on n . For a more detailed description of `geographical_threshold_graph`, we refer the reader to `NetworkX` documentation. Then, we associated a random angle θ_i to each node, obtained from a uniform distribution in $[0, 2\pi]$. In this way, each node of the random graph is associated to a vehicle configuration, consisting of a position and an angle. Set $\tau(\theta_i) = [\cos \theta_i, \sin \theta_i]^T$. Each edge (i, j) is associated to a *Dubins path*, which is defined as the shortest curve of bounded curvature that connects the configurations associated to nodes i and j , with initial tangent parallel to $\tau(\theta_i)$ and final tangent parallel to $\tau(\theta_j)$. We chose the minimum turning radius for the path associated to edge (i, j) as $r_{ij} = \min \{ \ell((i, j)) / (d(\theta_i, \theta_j)), 4 \}$ where $d(x, y)$ is the angular distance between angles x and y .

We defined the problem graph \mathbb{G} as described in Section 7.1. In particular, we associated two nodes to each configuration, representing opposite directions. In this way, we obtain a problem graph with $2n$ nodes. We set the acceleration and deceleration bounds constant for all paths and equal to 0.1 m/s^2 . The upper squared speed bound is constant for each arc and given by $2r$, where r is the minimum curvature radius of the path associated to the arc. In our tests we used the adaptive A* algorithm (Algorithm 6.5). First, we ran simulations for 10 values of n , logarithmically spaced between 100 and 1000. For each value of n , we generated 20 different graphs and, for each one of them, we ran 10 simulations, randomly choosing the source and the target node. Figure 21 shows the mean values and

the distributions of the computational time. We considered as solved only those instances for which the algorithm took less than 100 seconds to find the solution: for $n = 1000$, 5% of the instances have not been solved, while all the instances have been solved for the other values of n . Table 1 shows, for each value of n , the

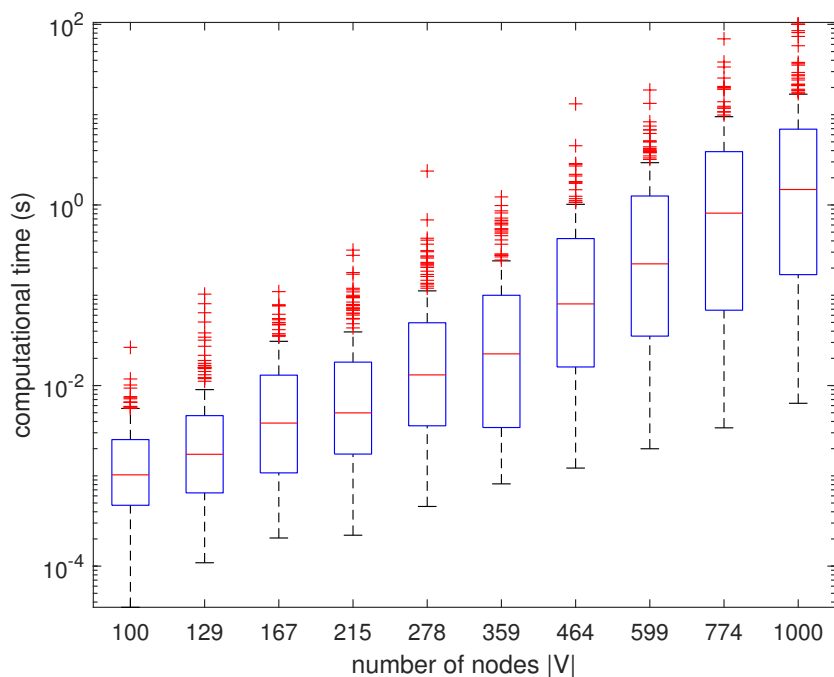


Figure 21: Box-and-whisker plot of the computational time to solve BASP on the different instances.

percentages of the tests in which Algorithm 6.5 terminates with a given value of k .

number of nodes	$k = 3$	$k = 4$	$k = 5$	$k = 6$
100	90 %	10%	-	-
129	81.5 %	17.5%	-	-
167	70.5%	28%	1%	0.5%
215	73.5%	25%	1.5 %	-
278	64.5%	30.5%	5 %	-
359	67.5 %	31 %	1.5%	-
464	49 %	44.5%	6.5 %	-
599	40 %	55%	5 %	-
744	37.5%	53.5%	9 %	-
1000	37.5%	57.9%	4.1%	0.5%

Table 1: Percentages of the values of k for each dimension of the graph.

In the previous section, we showed that, for a given problem instance, the path

p^* corresponding to the solution of BASP is in general different from the path \hat{p} obtained as the solution of BASP with infinite acceleration bounds. We ran some numerical experiments to compare the travel times $T_{\mathbb{B}}(p^*)$ and $T_{\mathbb{B}}(\hat{p})$. Namely, we generated 50 different random graphs with $n = 100$ with the procedure presented above. For each instance, we considered 10 problems obtained by randomly choosing the source node and the target node. Then, we solved BASP with different acceleration bounds. Namely, for each problem instance, we considered equal and constant maximum acceleration and deceleration bounds, chosen in the range $[0.1, 5.6]$ m/s^2 .

In Figure 22, we compare the optimal travel times along the two paths. Namely, for each value of the acceleration and deceleration bounds, we report the percentage difference $100 \frac{T_{\mathbb{B}}(\hat{p}) - T_{\mathbb{B}}(p^*)}{T_{\mathbb{B}}(p^*)}$ obtained for each test.

We observe that for low acceleration and deceleration bounds the difference is significant, while as the acceleration and deceleration bounds increase, the travel time difference between the two paths tends to be smaller. This is due to the fact that, if the acceleration/deceleration bounds are sufficiently high, paths p^* and \hat{p} are the same.

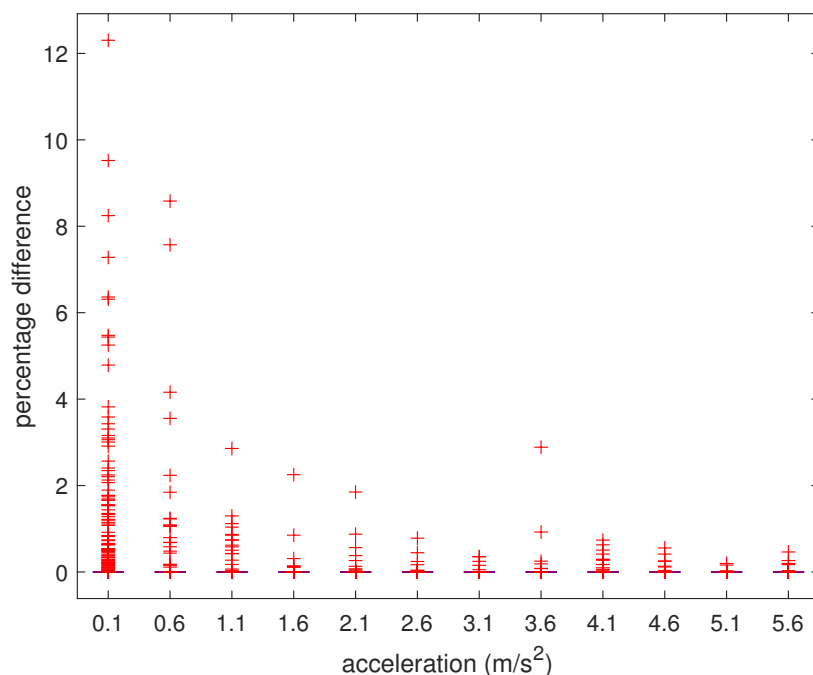


Figure 22: Percentage difference between the travel time of the infinite acceleration FP and the travel time of the FP.

7.4 Real industrial applications

Here we present two problems taken from real industrial applications, representing two automated warehouses. The problem data have been provided by packaging company Ocme S.r.l., based in Parma, Italy. The first problem is described by a graph of 399 nodes. The acceleration and deceleration bounds are constant, equal for all paths, and given by $\alpha^+ = 0.28 \text{ m/s}^2$ and $\alpha^- = -0.19 \text{ m/s}^2$. The speed bounds are constant for each arc but vary among different arcs, according to the associated paths curvatures, and they take values in the interval $[0.136, 1.7] \text{ ms}^{-1}$. The arc-lengths take values between 0.628 m and 10.87 m and have an average value of 2.86 m.

We ran 1000 simulations by randomly choosing the source node and the target node. The average value and the standard deviation of the computational time are 0.0036 s and 0.0062 s, respectively. In Figure 23, we report the distribution of the computational time of the considered instances. In the same figure we also show a box-and-whisker plot that reports the final value of k obtained by Algorithm 6.5 for solving each instance.

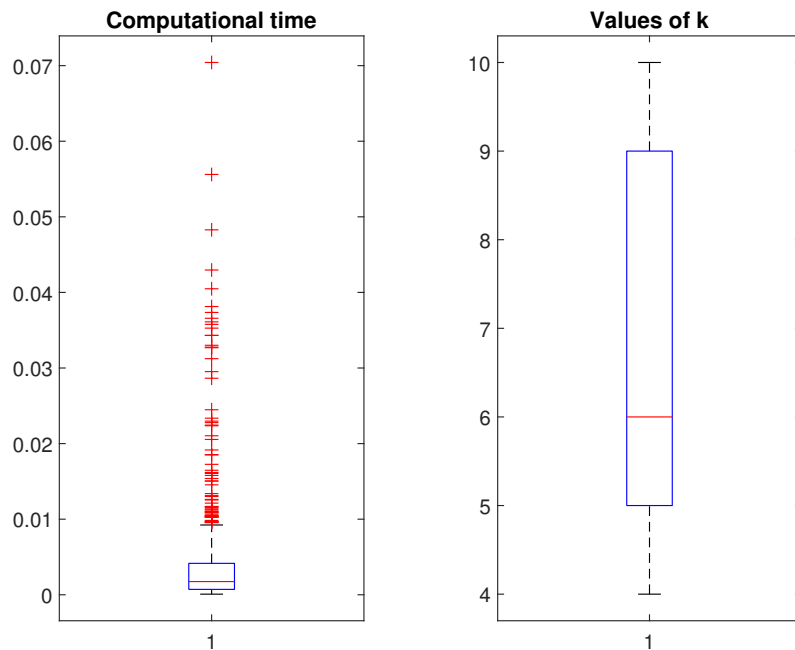


Figure 23: Box-and-whisker plot of the 1000 simulations on the 399-vertex graph.

We also considered a second problem, representing a larger automated warehouse, described by a graph of 3399 nodes. The acceleration and deceleration ramps are the same as in the previous example, while the maximum speed bounds belongs to the interval $[0.086, 1.7] \text{ ms}^{-1}$. The arc-lengths take values between 0.2 m and 16.352 m and have an average value of 3.569 m.

As in the first example, we ran 1000 simulations by randomly choosing the

source and the target nodes and we found that the average value and the standard deviation of the computational time are 0.0128 s and 0.0058 s, respectively.

In Figure 24, we report the box-and-whisker plots of the computational time and of the final value for k in Algorithm 6.5, for each instance. Note that both the mean computational time and the final value of k of the second example are larger than those of the first one. This is due to the fact that the second problem has a larger number of nodes. We can also note that the mean computational times of these two real-life examples are much lower than those of the random tests of comparable size presented in Section 7.3. This is probably due to the fact that the graphs associated to the two industrial problems have a lower connectivity than the randomly generated ones. Indeed, most nodes in the two industrial problems represent positions in corridors and are connected only to two other nodes: the preceding and the following one along the corridor. Note that this is common in problems associated to automated warehouses, since these facilities often have many long corridors.

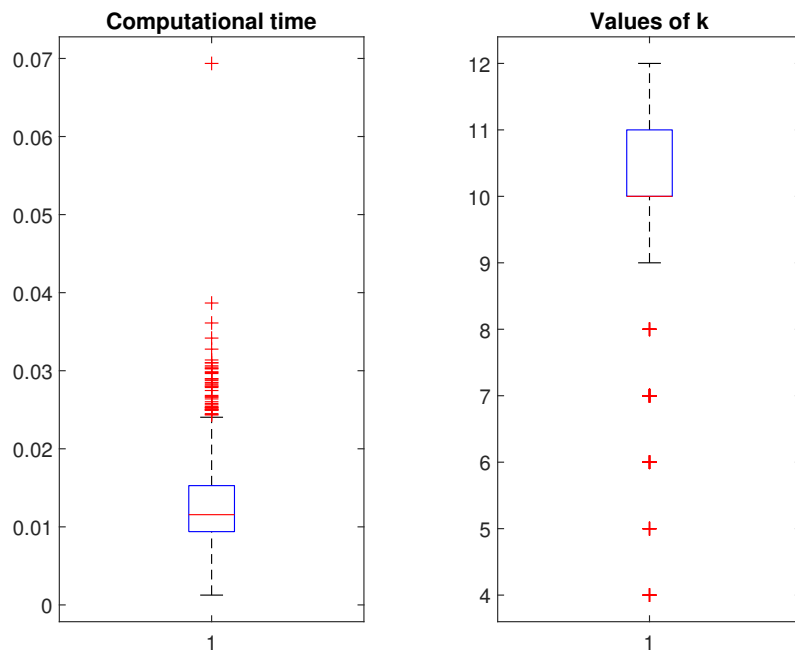


Figure 24: Box-and-whisker plot of the 1000 simulations on the 3399-vertex graph.

7.5 Example with non constant speed bounds

In all previous simulations, we considered problem instances in which acceleration and speed bounds are constant along each arc. However, the setting of BASP, as defined in 3.2, allows for arc-length dependent bounds on each arc. Here, we considered a problem instance of this more general form, illustrated by Figure 25. We considered 9 configurations, each one associated to a position on the plane and

to a direction angle. We defined the connecting paths by an order 5 interpolating spline, with initial and final conditions that guarantee the continuity of the tangent vector on connection points.

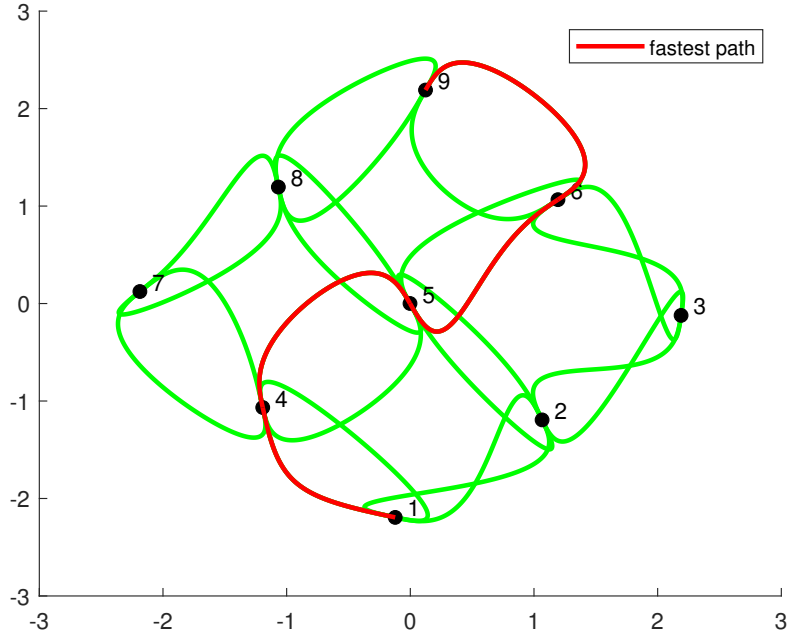


Figure 25: Graph with 9 nodes.

We choose a maximum speed bound $v_{\max} = 1.7 \text{ m/s}$ and a maximum normal acceleration $a_N = 0.56 \text{ m/s}^2$. The speed bound is a continuous function defined at each point λ of a path as $v(\lambda) = \min \left\{ v_{\max}, \sqrt{a_N / |\kappa(\lambda)|} \right\}$, where κ is the scalar curvature of the path, which is a function whose absolute value is the inverse of the radius of the circle that locally approximates the geometric path.

Figure 25 also shows the solution of BASP, with source node $s = 1$, while Figure 26 shows the corresponding speed profile.

8 Conclusions

The main contributions of this work are the definition of BASP, the proof of its NP-hardness, and the definition of a solution algorithm that achieves polynomial time-complexity under some hypotheses on problem data.

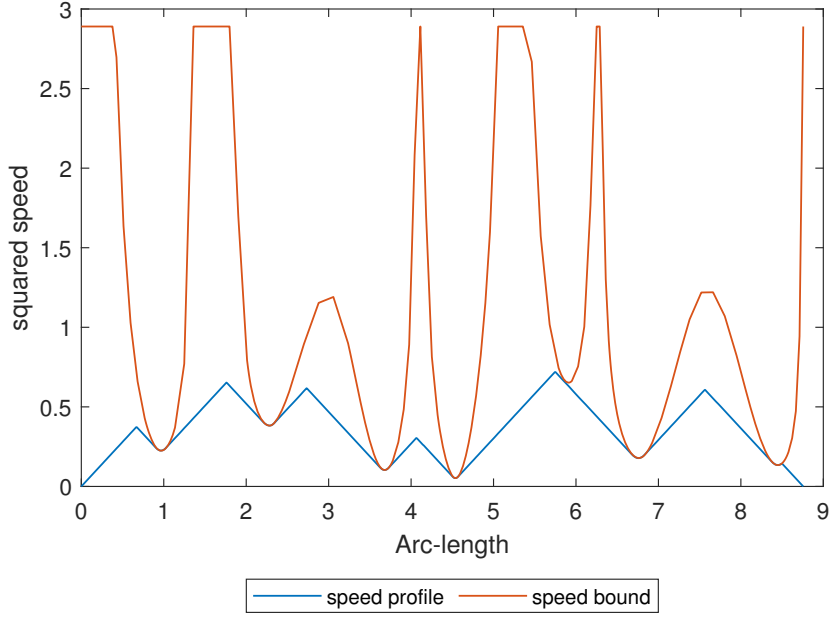


Figure 26: Speed profile of the fastest path.

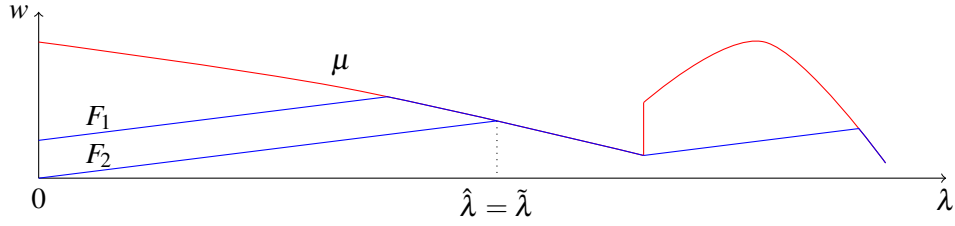


Figure 27: Illustration of the proof of Proposition 8.1.

Appendix

Proposition 8.1. Let $\mu, \alpha : [0, +\infty) \rightarrow \mathbb{R}^+$, let F_1, F_2 be the solutions of the following equations,

$$\begin{cases} F_i'(\lambda) = \begin{cases} \alpha(\lambda) \wedge \mu'(\lambda) & \text{if } F_i(\lambda) \geq \mu(\lambda) \\ \alpha(\lambda) & \text{if } F_i(\lambda) < \mu(\lambda) \end{cases} \\ F_i(0) = w_{0,i}, \end{cases} \quad (15)$$

with $0 \leq w_{0,i} \leq \mu(0)$, for $i \in \{1, 2\}$, and let $\bar{\lambda}$ be such that $\mu(\bar{\lambda}) = \int_0^{\bar{\lambda}} \alpha(\lambda) d\lambda$. Then $(\forall \lambda \geq \bar{\lambda}) F_1(\lambda) = F_2(\lambda)$.

Proof. Figure 27 illustrates the following proof. W.l.o.g., assume that $w_{0,1} \geq w_{0,2}$. This implies that $(\forall \lambda \geq 0) F_1(\lambda) \geq F_2(\lambda)$. Indeed, assume by contradiction that

there exists $\bar{\lambda}$ such that $F_1(\bar{\lambda}) < F_2(\bar{\lambda})$, then, by continuity of F_1 and F_2 , this implies that there exists $\hat{\lambda} \leq \bar{\lambda}$ such that $F_1(\hat{\lambda}) = F_2(\hat{\lambda})$, thus $(\forall \lambda \geq \hat{\lambda}) F_1(\lambda) = F_2(\lambda)$, since, for $\lambda \geq \hat{\lambda}$, $F_1(\lambda)$ and $F_2(\lambda)$ solve the same differential equation with the same initial condition at $\lambda = \hat{\lambda}$, contradicting the assumption.

Further, note that $(\exists \tilde{\lambda} \in (0, \bar{\lambda}]) F_2(\tilde{\lambda}) = \mu(\tilde{\lambda})$. Indeed, if by contradiction

$$(\forall \lambda \in (0, \bar{\lambda}]) F_2(\lambda) < \mu(\lambda),$$

then

$$(\forall \lambda \in (0, \bar{\lambda}]) F_2'(\lambda) = \alpha(\lambda),$$

so that

$$F_2(\bar{\lambda}) - F_2(0) = \int_0^{\bar{\lambda}} \alpha(\lambda) d\lambda = \mu(\bar{\lambda}),$$

which contradicts the assumption.

Hence, $(\exists \hat{\lambda} \in \mathbb{R}^+) F_2(\hat{\lambda}) = F_1(\hat{\lambda}) = \mu(\hat{\lambda})$ and, consequently,

$$(\forall \lambda \geq \hat{\lambda}) F_1(\lambda) = F_2(\lambda),$$

which implies the thesis, being $\bar{\lambda} \geq \hat{\lambda}$. \square

For $p \in P$, $\lambda \in [0, \ell(p)]$, we set $\mathscr{W}_p(\lambda) = w$, where w is the solution of Problem (3) for path p . In other words $\mathscr{W}_p(\lambda)$ is the square of the optimal speed profile for traversing path p , evaluated at arc-length λ , with respect to p .

Proposition 8.2. 1) Let $p_1, p_2, q \in P$, be such that $p_1q, p_2q \in P$, then

$$(\forall \lambda \geq \ell^+(q)) \mathscr{W}_{p_1q}(\ell(p_1) + \lambda) = \mathscr{W}_{p_2q}(\ell(p_2) + \lambda).$$

2) Let $p, q_2, q_1 \in P$, be such that $pq_1, pq_2 \in P$, then

$$(\forall \lambda \leq \ell^-(p)) \mathscr{W}_{pq_1}(\lambda) = \mathscr{W}_{pq_2}(\lambda).$$

Proof. We only prove 1), the proof of 2) is analogous. Note that, for $\lambda \geq 0$, $\mathscr{W}_{p_1q}(\lambda + \ell(p_1)) = \min\{F_1(\lambda), B(\lambda)\}$, $\mathscr{W}_{p_2q}(\lambda + \ell(p_2)) = \min\{F_2(\lambda), B(\lambda)\}$, where F_1, F_2 are the solution of (15) with $\mu = \mu^+$ and initial conditions $w_{0,1} = \mathscr{W}_{p_1}(\ell(p_1))$ and $w_{0,2} = \mathscr{W}_{p_2}(\ell(p_2))$, respectively, and B is the solution of (5) with $\mu = \mu^+$. By Proposition 8.1, for $\lambda \geq \ell^+(q)$, $F_1(\lambda) = F_2(\lambda)$. Consequently, $(\forall \lambda \geq \ell^+(q)) \mathscr{W}_{p_1q}(\ell(p_1) + \lambda) = \mathscr{W}_{p_2q}(\ell(p_2) + \lambda)$. \square

8.1 Proof of Proposition 5.2

Let Ψ be defined as in (3a), then

$$\begin{aligned} T(p_1t\sigma) - T(p_1t) &= \\ &= \int_0^{\ell(p_1t\sigma)} \Psi(\mathscr{W}_{p_1t\sigma}(\lambda)) d\lambda - \int_0^{\ell(p_1t)} \Psi(\mathscr{W}_{p_1t}(\lambda)) d\lambda = \\ &= \int_{\ell(p_1) + \ell^-(t)}^{\ell(p_1t\sigma)} \Psi(\mathscr{W}_{p_1t\sigma}(\lambda)) d\lambda - \int_{\ell(p_1) + \ell^-(t)}^{\ell(p_1t)} \Psi(\mathscr{W}_{p_1t}(\lambda)) d\lambda, \end{aligned}$$

where we used the fact that, by ii) of Proposition 8.2, $(\forall \lambda \leq \ell(p_1) + \ell^-(t)) \Psi(\mathcal{W}_{p_1 t \sigma}(\lambda)) = \Psi(\mathcal{W}_{p_1 t}(\lambda))$. Similarly, we have that $T(p_2 t \sigma) - T(p_2 t) = \int_{\ell(p_2) + \ell^-(t)}^{\ell(p_2 t \sigma)} \Psi(\mathcal{W}_{p_2 t \sigma}(\lambda)) d\lambda - \int_{\ell(p_2) + \ell^-(t)}^{\ell(p_2 t)} \Psi(\mathcal{W}_{p_2 t}(\lambda)) d\lambda$. Moreover, by i) of Proposition 8.2, we have that $(\forall \lambda \geq \ell^+(t \sigma)) \mathcal{W}_{p_1 t \sigma}(\ell(p_1) + \lambda) d\lambda = \mathcal{W}_{p_2 t \sigma}(\ell(p_2) + \lambda) d\lambda$ and $(\forall \lambda \geq \ell^+(t)) \mathcal{W}_{p_1 t}(\ell(p_1) + \lambda) d\lambda = \mathcal{W}_{p_2 t}(\ell(p_2) + \lambda) d\lambda$ which imply that $T(p_1 t \sigma) - T(p_1 t) = T(p_2 t \sigma) - T(p_2 t)$, since $\ell^+(t) \leq \ell^-(t)$ and, as noticed in Section 5, $\ell^+(t \sigma) \leq \ell^+(t)$. \square

8.2 Proof of Proposition 4.2

Let $s \in \mathbb{V}$ be the departure node and $f \in \mathbb{V}$ be the arrival node. Let v_s be the initial speed at node s and v_f be the final speed at node f . We would like to select a path in \mathbb{G} from s to f , such that the time needed to run along the path by fulfilling the maximum speed, the maximum and minimum acceleration constraints along the edges, and the boundary conditions v_s and v_f , is minimized. We show that this problem is NP-hard by a polynomial reduction of the NP-complete *Partition* problem to *BASP-C*. In the *Partition* problem, given a set $N = \{1, \dots, n\}$ of positive integer values w_1, \dots, w_n , we would like to establish whether N can be partitioned into two subsets N_1 and N_2 such that $\sum_{i \in N_1} w_i = \sum_{i \in N_2} w_i = \frac{W}{2}$. Given an instance of the *Partition* problem we polynomially reduce it to an instance of *BASP-C* as follows. Let $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ be such that:

$$\mathbb{V} = N \cup \{0, n+1\}, \quad \mathbb{E} = \{(i, j) \in \mathbb{V}^2 \mid i < j\}.$$

We set the following lengths for the arcs:

$$\ell(i, j) = \begin{cases} 0, & i = 0 \\ w_i, & \text{otherwise.} \end{cases}$$

For what concerns the maximum speed values, we set $(\forall e \in \mathbb{E}) \mu^+(e) = +\infty$ (unbounded maximum speed), while we set the maximum acceleration $\alpha^+ = 1$ and the minimum acceleration $\alpha^- = -1$ for all arcs. The starting node s is node 0, with $v_s = 0$, while the final node f is $n+1$ with $v_f = \sqrt{W}$. Each path p from node 0 to node $n+1$ has the following structure

$$0 \ i_1 \ i_2 \ \dots \ i_r \ n+1,$$

with $0 < i_1 < i_2 < \dots < i_r$. Let us denote by $N_p = \{i_1, i_2, \dots, i_r\}$ the set of intermediate nodes in p . The length of path p is $\ell(p) = \sum_{i \in N_p} w_i$. Let us first assume that $\ell(p) < \frac{W}{2}$. In this case the maximum speed which can be reached at the end of the path is $v_u = \alpha^+ t_p$, where t_p fulfills $\ell(p) = \frac{1}{2} \alpha^+ t^2$, (i.e., $t_p = \sqrt{2\ell(p)}$). Thus, $v_u = \sqrt{2\ell(p)} < v_f$, (i.e., no path p with $\ell(p) < \frac{W}{2}$ is able to meet the boundary condition v_f). Thus, we restrict our attention to paths p such that $\ell(p) \geq \frac{W}{2}$. A lower bound for the time needed to run along the path is given again by the solution of the following simple equation $\ell(p) = \frac{1}{2} \alpha^+ t^2$, (i.e., $t_p = \sqrt{2\ell(p)}$). Note that this is a

lower bound since, with the maximum acceleration, after this time we reach speed $v_u = \alpha^+ t_p = \sqrt{2\ell(p)} \geq \sqrt{W}$, so that we might need to decelerate in order to meet the boundary condition v_f . Since $\ell(p) \geq \frac{W}{2}$, we have that the lower bound can be further bounded from below by \sqrt{W} . Finally, we observe that such lower bound can be attained if and only if the Partition problem admits a solution. In such case we can set $N_1 = N_p$ and $N_2 = N \setminus N_p$. Thus, we have established that an instance of the Partition problem admits a solution if and only if the corresponding instance of the BASP has optimal value equal to \sqrt{W} .

8.3 Proof of Proposition 4.3

We modify the original problem as follows. First, we split each arc θ into $\ell(\theta)$ arcs of length 1 by introducing along the original arc $\ell(\theta) - 1$ intermediate nodes (recall that $\ell(\theta)$ is assumed to be integer). In this way, we have a new graph with node set \mathbb{V}' such that $|\mathbb{V}'| = |\mathbb{V}| + \sum_{\theta \in \mathbb{E}} \ell(\theta) - |\mathbb{E}|$ and arc set \mathbb{E}' where each arc $\theta \in \mathbb{E}$ is replaced by $\ell(\theta)$ arcs and all arcs have length equal to 1. The new arcs inherit the speed and acceleration bounds of the original ones. Next, we observe that at optimal solutions there is a finite number of speeds which can be reached at each node. These include all squared speeds $\mu^+(\theta)$ for $\theta \in \mathbb{E}$ but also all speeds which can be reached starting from one squared speed $\mu^+(\theta)$ and then moving with a maximum (or minimum) acceleration along a path p of length $\ell(p)$, provided that we never reach the maximum speed along an arc of the path and that the speed never falls below 0. In order to meet the last two requirements, the value $\ell(p)$ is bounded from above by $\frac{1}{2}(\max_{\theta \in \mathbb{E}} \mu^+(\theta))^2$. Indeed, the time t_p required for a path p of length $\ell(p)$, assuming that the initial speed is 0, is given by the solution of

$$\ell(p) = \frac{1}{2} \alpha^+ t^2,$$

so that the corresponding variation of the speed is $\alpha^+ t_p$ which needs to be lower than $\max_{\theta \in \mathbb{E}} \mu^+(\theta)$. Recalling that $(\forall \theta \in \mathbb{E}) \alpha^+ = 1$, we must have that $\sqrt{2\ell(p)} \leq \max_{\theta \in \mathbb{E}} \mu^+(\theta)$ or, equivalently, $\ell(p) \leq \frac{1}{2}(\max_{\theta \in \mathbb{E}} \mu^+(\theta))^2$. Now, let us denote by \mathcal{V} the set of different possible speeds. In view of the previous observations, we have that $|\mathcal{V}| \leq |\mathbb{E}|(1 + \frac{1}{2}(\max_{\theta \in \mathbb{E}} \mu^+(\theta))^2)$. Now we create a new graph with node set $\mathbb{V}' \times \mathcal{V}$, (i.e., each node is a pair made up by a node in \mathbb{V}' and one of the possible speeds in \mathcal{V}). Thus, the number of nodes is

$$|\mathbb{V}'| |\mathcal{V}| \leq (|\mathbb{V}| + \sum_{\theta \in \mathbb{E}} \ell(\theta) - |\mathbb{E}|)(|\mathbb{E}|(1 + \frac{1}{2}(\max_{\theta \in \mathbb{E}} \mu^+(\theta))^2)).$$

For what concerns the arc set, in this graph an arc between node (i, w_i) and node (j, w_j) exists if there exists an arc $(i, j) \in \mathbb{E}'$. The distance associated to this arc is the minimum time for a path from i to j with the boundary conditions w_i and w_j , which can be easily computed by the forward-backward algorithm. Then we can solve our problem by applying, e.g., Dijkstra's algorithm to this graph. Dijkstra's complexity is bounded from above by the square of the number of nodes and

is, thus, polynomial with respect to the size and the values of data of the original problem, which proves pseudo-polynomiality.

References

- [1] Ahmad, Abadi, & Vaclav, Prenosil. 2015 (May). Safe path planning using cell decomposition approximation. *Pages 8–14 of: Hrubý, Miroslav (ed), International Conference Distance learning simulation and communication 'DLSC 2015'*.
- [2] Consolini, Luca, Locatelli, Marco, Minari, Andrea, & Piazzzi, Aurelio. 2017. An optimal complexity algorithm for minimum-time velocity planning. *Systems & Control Letters*, **103**, 50–57.
- [3] Consolini, Luca, Laurini, Mattia, Locatelli, Marco, & Minari, Andrea. 2020. A solution of the minimum-time speed planning problem based on lattice theory. *Journal of the Franklin Institute*, **357**, 7617–7637.
- [4] Cowlagi, Raghvendra V., & Tsiotras, Panagiotis. 2009. Shortest Distance Problems in Graphs Using History-Dependent Transition Costs with Application to Kinodynamic Path Planning. *Pages 414–419 of: 2009 American Control Conference*.
- [5] Edelkamp, S., & Schroedl, S. 2011. *Heuristic Search: Theory and Applications*. Elsevier Science.
- [6] Ferguson, D., & Stentz, A. 2007. Field D*: an interpolation-based path planner and replanner. *Springer Tracts in Advanced Robotics*, **28**, 239–253.
- [7] Gelperin, D. 1977. On the optimality of A*. *Artificial Intelligence*, **8**(1), 69–76.
- [8] Kim, Dae Hwan, Hai, Nguyen Trong, & Joe, Woong Yeol. 2018. A Guide to Selecting Path Planning Algorithm for Automated Guided Vehicle (AGV). *Pages 587–596 of: Duy, Vo Hoang, Dao, Tran Trong, Zelinka, Ivan, Kim, Sang Bong, & Phuong, Tran Thanh (eds), AETA 2017 - Recent Advances in Electrical Engineering and Related Sciences: Theory and Application*. Cham: Springer International Publishing.
- [9] Koenig, S., Likhachev, M., & Furcy, D. 2004. Lifelong planning A*. *Artificial Intelligence*, **155**(1–2), 93–146.
- [10] Nilsson, N. J. 1971. *Problem-Solving Methods in Artificial Intelligence*. New York: McGraw-Hill.
- [11] Ryck, M. De, Versteyhe, M., & Debrouwere, F. 2020. Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *Journal of Manufacturing Systems*, **54**, 152–173.

- [12] Seif, Roudabe, & Oskoei, Mohammadreza Asghari. 2015. Mobile Robot Path Planning by RRT* in Dynamic Environments. *International Journal of Intelligent Systems and Applications (IJISA)*, **7**(5), 24–30.
- [13] Verschuer, D., Demeulenaere, B., Swevers, J., Schutter, J. De, & Diehl, M. 2009. Time-Optimal Path Tracking for Robots: A Convex Optimization Approach. *IEEE Transactions on Automatic Control*, **54**(10), 2318–2327.