AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your article. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

Carefully check the page proofs (and coordinate with all authors); additional changes or updates WILL NOT be accepted after the article is published online/print in its final form. Please check author names and affiliations, funding, as well as the overall article for any errors prior to sending in your author proof corrections.

AQ:1 = Please confirm or add details for any funding or financial support for the research of this article.
AQ:2 = Please confirm the postal code for Università di Parma.
AQ:3 = Please specify the section numbers for the phrase "next sections."

A Sequential Algorithm for Jerk Limited Speed Planning

Luca Consolini, *Member, IEEE*, Marco Locatelli, and Andrea Minari

***Abstract*—In this article, we discuss a sequential algorithm for the computation of a minimum-time speed profile over a given path, under velocity, acceleration, and jerk constraints. Such a problem arises in industrial contexts, such as automated warehouses, where LGVs need to perform assigned tasks as fast as possible in order to increase productivity. It can be reformulated as an optimization problem with a convex objective function, linear velocity and acceleration constraints, and nonconvex jerk constraints, which, thus, represents the main source of the difficulty. While existing nonlinear programming (NLP) solvers can be employed for the solution of this problem, it turns out that the performance and robustness of such solvers can be enhanced by the sequential line-search algorithm proposed in this article. At each iteration, a feasible direction, with respect to the current feasible solution, is computed, and a step along such direction is taken in order to compute the next iterate. The computation of the feasible direction is based on the solution of a linearized version of the problem, and the solution of the linearized problem, through an approach that strongly exploits its special structure, represents the main contribution of this work. The efficiency of the proposed approach with respect to existing NLP solvers is proven through different computational experiments.**

***Note to Practitioners*—This article was motivated by the needs of LGV manufacturers. In particular, it presents an algorithm for computing the minimum-time speed law for an LGV along a preassigned path, respecting assigned velocity, acceleration, and jerk constraints. The solution algorithm should be: 1) *fast*, since speed planning is made continuously throughout the workday, not only when an LGV receives a new task but also during the execution of the task itself, since conditions may change, e.g., if the LGV has to be halted for security reasons and 2) *reliable*, i.e., it should return solutions of high quality, because a better speed profile allows to save time and even small percentage improvements, say a 5% improvement, has a considerable impact on the productivity of the warehouse, and, thus, determines a significant economic gain. The algorithm that we propose meets these two requirements, and we believe that it can be a useful tool for LGV manufacturers and users. It is obvious that the proposed method also applies to the speed planning problem for vehicles other than LGVs, e.g., road vehicles.**

AQ:1 Manuscript received June 29, 2021; accepted August 24, 2021. This article was recommended for publication by Associate Editor M. Robba and Editor C. Seatzu upon evaluation of the reviewers' comments. This work was supported in part by the Programme "FIL-Quota Incentivante" of the University of Parma and in part by the Fondazione Cariparma. *(Corresponding author: Marco Locatelli.)*

AQ:2 The authors are with the Dipartimento di Ingegneria e Architettura, Università di Parma, 43121 Parma, Italy (e-mail: luca.consolini@unipr.it; marco.locatelli@unipr.it; andrea.minari2@studenti.unipr.it).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TASE.2021.3111758.

Digital Object Identifier 10.1109/TASE.2021.3111758

***Index Terms*—Optimization, sequential line-search method, speed planning.**

I. Introduction

An IMPORTANT problem in motion planning is the computation of the minimum-time motion of a car-like vehicle from a start configuration to a target one while avoiding collisions (obstacle avoidance) and satisfying kinematic, dynamic, and mechanical constraints (for instance, on velocities, accelerations, and maximal steering angle). This problem can be approached in two ways.

1) As a minimum-time trajectory planning, where both the path to be followed by the vehicle and the timing law on this path (i.e., the vehicle's velocity) are simultaneously designed. For instance, one could use the RRT* algorithm (see [1]).
2) As a (geometric) path planning followed by a minimum-time speed planning on the planned path (see [2]).

In this article, following the second paradigm, 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. The problem can be reformulated as an optimization problem, and it is quite relevant from the practical point of view. In particular, in automated warehouses, the speed of LGVs needs to be planned under acceleration and jerk constraints. As previously mentioned, the solution algorithm should be both *fast* and *reliable*. In our previous work [3], we proposed an optimal time-complexity algorithm for finding the time-optimal speed law that satisfies constraints on maximum velocity and tangential and normal acceleration. In the subsequent work [4], we included a bound on the derivative of the acceleration with respect to the arc length. In this article, we consider the presence of jerk constraints (constraints on the time derivative of the acceleration). The resulting optimization problem is nonconvex and, for this reason, is significantly more complex than the ones that we discussed in [3] and [4]. The main contribution of this work is the development of a line-search algorithm for this problem based on the sequential solution of convex problems. The proposed algorithm meets both requirements of being fast and reliable. The former is met by heavily exploiting the special structure of the optimization problem, the latter by the theoretical guarantee that the returned solution is a first-order stationary point (in practice, a local minimizer) of the optimization problem.

A. Problem Statement

Here, we introduce the problem at hand more formally. Let $\boldsymbol{\gamma}:[0, s_f] \to \mathbb{R}^2$ be a smooth function. The image set $\boldsymbol{\gamma}([0, s_f])$ is the path to be followed, $\boldsymbol{\gamma}(0)$ the initial configuration, and $\boldsymbol{\gamma}(s_f)$ the final one. Function $\boldsymbol{\gamma}$ has arc-length parameterization, such that $(\forall \lambda \in [0, s_f])$, $\|\boldsymbol{\gamma}'(\lambda)\| = 1$. In this way, s_f is the path length. We want to compute the speed-law that minimizes the overall transfer time (i.e., the time needed to go from $\boldsymbol{\gamma}(0)$ to $\boldsymbol{\gamma}(s_f)$). To this end, let $\lambda:[0, t_f] \to [0, s_f]$ be a differentiable monotone increasing function that represents the vehicle's arc-length position along the curve 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 derivative of the vehicle arc-length position, which corresponds to the norm of its velocity vector at position s. The position of the vehicle as a function of time is given by $\mathbf{x}:[0, t_f] \to \mathbb{R}^2$, $\mathbf{x}(t) = \boldsymbol{\gamma}(\lambda(t))$. The velocity and acceleration are given, respectively, by

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

where $a_T(t) = v'(\lambda(t))v(\lambda(t))$ and $a_N(t) = k(\lambda(t))v(\lambda(t))^2$ are, respectively, the tangential and normal components of the acceleration (i.e., the projections of the acceleration vector $\ddot{\mathbf{x}}$ on the tangent and the normal to the curve). Moreover $\boldsymbol{\gamma}'^{\perp}(\lambda)$ is the normal to vector $\boldsymbol{\gamma}'(\lambda)$ and the tangent of $\boldsymbol{\gamma}'$ at λ. Here, $k:[0, s_f] \to \mathbb{R}$ is the scalar curvature, defined as $k(s) = <\boldsymbol{\gamma}''(s), \boldsymbol{\gamma}'(s)^{\perp}>$. Note that $|k(s)| = \|\boldsymbol{\gamma}''(s)\|$. In the following, we assume that $k(s) \in \mathcal{C}^1([0, s_f], \mathbb{R})$. The total maneuver time, for a given velocity profile $v \in C^1([0, s_f], \mathbb{R})$, is returned by the functional

$$\mathcal{F} : C^1\big(\big[0, s_f\big], \mathbb{R}\big) \to \mathbb{R}, \quad \mathcal{F}(v) = \int_0^{s_f} v^{-1}(s)ds. \tag{1}$$

In our previous work [3], we considered the problem

$$\min_{v \in \mathcal{V}} \mathcal{F}(v) \tag{2}$$

where the feasible region $\mathcal{V} \subset C^1([0, s_f], \mathbb{R})$ is defined by the following set of constraints:

$$v(0) = 0, \quad v\big(s_f\big) = 0 \tag{3a}$$
$$0 \le v(s) \le v_{\max}, \quad s \in \big[0, s_f\big] \tag{3b}$$
$$|v'(s)v(s)| \le A, \quad s \in \big[0, s_f\big] \tag{3c}$$
$$|k(s)|v(s)^2 \le A_N, \quad s \in \big[0, s_f\big] \tag{3d}$$

where $v_{\max}$, A, and A_N are upper bounds for the velocity, the tangential acceleration, and the normal acceleration, respectively. Constraints (3a) are the initial and final interpolation conditions, while constraints (3b)–(3d) limit velocity and the tangential and normal components of acceleration. In [3], we presented an algorithm, with linear-time computational complexity with respect to the number of variables, which provides an optimal solution of (2) after spatial discretization. One limitation of this algorithm is that the obtained velocity profile is Lipschitz[1] but not differentiable so that the vehicle's acceleration is discontinuous. With the aim of obtaining a smoother velocity profile, in the subsequent work [4], we required that the velocity be differentiable, and we imposed a Lipschitz condition (with constant J) on its derivative. In this way, after setting $w = v^2$, the feasible region of the problem $\mathcal{W} \subset C^1([0, s_f], \mathbb{R})$ is defined by the set of functions $w \in C^1([0, s_f], \mathbb{R})$ that satisfy the following set of constraints:

$$w(0) = 0, \quad w\big(s_f\big) = 0 \tag{4a}$$
$$0 \le w(s) \le v_{\max}^2, \quad s \in \big[0, s_f\big] \tag{4b}$$
$$\frac{1}{2}|w'(s)| \le A, \quad s \in \big[0, s_f\big] \tag{4c}$$
$$|k(s)|w(s) \le A_N, \quad s \in \big[0, s_f\big] \tag{4d}$$
$$|w'(s_1) - w'(s_2)| \le J|s_1 - s_2|, \quad s_1, s_2 \in \big[0, s_f\big]. \tag{4e}$$

Then, we end up with the problem

$$\min_{w \in \mathcal{W}} G(w) \tag{5}$$

where the objective function is

$$G : C^1\big(\big[0, s_f\big], \mathbb{R}\big) \to \mathbb{R}, \quad G(w) = \int_0^{s_f} w^{-1/2}(s)ds. \tag{6}$$

The objective function (6) and constraints (4a)–(4d) correspond to the ones in problem (2) after the substitution $w = v^2$. Note that this change of variable is well known in the literature. It has been first proposed in [5], while, in [6], it is observed that Problem (2) becomes convex after this change of variable. The added set of constraints (4e) is a Lipschitz condition on the derivative of the squared velocity w. It is used to enforce a smoother velocity profile by bounding the second derivative of the squared velocity with respect to arc length. Note that constraints (4) are linear, and the objective function (6) is convex. In [4], we proposed an algorithm for solving a finite-dimensional approximation of Problem (4). The algorithm exploited the particular structure of the resulting convex finite-dimensional problem. This article extends the results of [4]. It considers a nonconvex variation of Problem (4), in which constraint (4e) is substituted with a constraint on the time derivative of the acceleration $|\dot{a}(t)| \le J$, where $a(t) = (d/dt)v(\lambda(t)) = v'(\lambda(t))v(\lambda(t)) = (1/2)w'(\lambda(t))$. Then, we set

$$j_L(t) = \dot{a}(t) = \frac{1}{2}w''(s(t))\sqrt{(w(s(t)))}.$$

This quantity is commonly called "jerk." Bounding the absolute value of jerk allows to avoid sudden changes of acceleration and obtain a smoother motion. Then, we end up with the following minimum-time problem.

Problem 1 (Smooth Minimum-Time Velocity Planning Problem: Continuous Version):

$$\min_{w \in C^2} \int_0^{s_f} w(s)^{-1/2}\, ds$$
$$w(0) = 0, \quad w\big(s_f\big) = 0$$
$$0 \le w(s) \le \mu^+(s), \quad s \in \big[0, s_f\big]$$
$$\frac{1}{2}|w'(s)| \le A, \quad s \in \big[0, s_f\big] \tag{7}$$
$$\frac{1}{2}|w''(s)\sqrt{w(s)}| \le J \quad s \in \big[0, s_f\big] \tag{8}$$

[1] A function $f:\mathbb{R} \to \mathbb{R}$ is *Lipschitz* if there exists a real positive constant L such that $(\forall x, y \in \mathbb{R})$ $|f(x) - f(y)| \le L|x - y|$.

where μ^+ is the square velocity upper bound depending on the shape of the path, i.e.,

$$\mu^+(s) = \min\left\{v_{\max}^2, \frac{A_N}{|k(s)|}\right\}$$

where $v_{\max}$, A_N, and k are the maximum vehicle velocity, the maximum normal acceleration, and the path curvature, respectively. Parameters A and J are the bounds representing the limitations on the (tangential) acceleration and the jerk, respectively. For the sake of simplicity, we consider constraints (7) and (8) symmetric and constant. However, the following development could be easily extended to the nonsymmetric and nonconstant case. Note that the jerk constraint (8) is nonconvex. The continuous problem is discretized as follows. We subdivide the path into $n-1$ intervals of equal length, i.e., we evaluate function w at points $s_i = ((i-1)s_f)/(n-1)$, $i = 1, \ldots, n$, so that we have the following n-dimensional vector of variables:

$$\mathbf{w} = (w_1, w_2, \ldots, w_n) = (w(s_1), w(s_2), \ldots, w(s_n)).$$

Then, the finite dimensional version of the problem is given as follows.

Problem 2 (Smooth Minimum-Time Velocity Planning Problem: Discretized Version):

$$\min_{\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1}} + \sqrt{w_i}} \tag{9}$$

$$0 \le \mathbf{w} \le \mathbf{u} \tag{10}$$

$$w_{i+1} - w_i \le 2hA, \quad i = 1, \ldots, n-1 \tag{11}$$

$$w_i - w_{i+1} \le 2hA, \quad i = 1, \ldots, n-1 \tag{12}$$

$$(w_{i-1} - 2w_i + w_{i+1})\sqrt{\frac{\ell_i(\mathbf{w})}{4}} \le 2h^2 J, \quad i = 2, \ldots, n-1 \tag{13}$$

$$-(w_{i-1} - 2w_i + w_{i+1})\sqrt{\frac{\ell_i(\mathbf{w})}{4}} \le 2h^2 J, \quad i = 2, \ldots, n-1 \tag{14}$$

where

$$\ell_i(\mathbf{w}) = w_{i+1} + w_{i-1} + 2w_i \tag{15}$$

while $u_i = \mu^+(s_i)$, for $i = 1, \ldots, n$, and, in particular, $u_1 = 0$ and $u_n = 0$ since we are assuming that the initial and final velocities are equal to 0. The objective function (9) is an approximation of (6) given by the Riemann sum of the intervals obtained by dividing each interval $[s_i, s_{i+1}]$, for $i = 1, \ldots, n-1$, in two subintervals of the same size. Constraints (11) and (12) are obtained by a finite difference approximation of w'. Constraints (13) and (14) are obtained by using a second-order central finite difference to approximate w'', while w is approximated by a weighted arithmetic mean of three consecutive samples. Due to jerk constraints (13) and (14), Problem 2 is nonconvex and cannot be solved with the algorithm presented in [4].

B. Main Result

The main contribution of this article is the development of a new solution algorithm for finding a local minimum of the nonconvex Problem 2. As detailed in next sections, we propose to solve Problem 2 by a line-search algorithm based on the sequential solution of convex problems. The algorithm is an iterative one where the following operations are performed at each iteration. AQ:3

1) Constraint Linearization: We first define a convex problem by linearizing constraints (13) and (14) through a first-order Taylor approximation around the current point $\mathbf{w}^{(k)}$. Different from other sequential algorithms for nonlinear programming (NLP) problems, we keep the original convex objective function. The linearized problem is introduced in Section II.

2) Computation of a Feasible Descent Direction: The convex problem (actually, a relaxation of such problem) is solved in order to compute a feasible descent direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$. The main contribution of this article lies in this part. The computation requires the minimization of a suitably defined objective function through a further iterative algorithm. At each iteration of this algorithm, the following operations are performed:

C. Objective Function Evaluation

Such evaluation requires the solution of a problem with the same objective function but subject to a subset of the constraints. The special structure of the resulting subproblem is heavily exploited in order to solve it efficiently. This is the topic of Section III.

D. Computation of a Descent Step

Some Lagrange multipliers of the subproblem define a subgradient for the objective function. This can be employed to define a linear programming (LP) problem that returns a descent step for the objective function. This is the topic of Section IV.

Line Search: Finally, a standard line search along the half-line $\mathbf{w}^{(k)} + \alpha\boldsymbol{\delta}\mathbf{w}^{(k)}$, $\alpha \ge 0$, is performed.

Sections II–IV detail all what we discussed above. Next, in Section V, we present different computational experiments.

E. Comparison With Existing Literature

Although many works consider the problem of minimum-time speed planning with acceleration constraints (see [7]–[9]), relatively few consider jerk constraints. Perhaps, this is also due to the fact that the jerk constraint is nonconvex so that its presence significantly increases the complexity of the optimization task. One can use a general-purpose NLP solver (such as SNOPT or IPOPT) for finding a local solution of Problem 2, but the required time is, in general, too large for the speed planning application. As outlined in Section I-D, in this work, we tackle this problem through an approach based on the solution of a sequence of convex subproblems. There are different approaches in the literature based on the sequential solution of convex subproblems. In [10], it is first observed that the problem with acceleration constraints but no jerk constraints for robotic manipulators can be reformulated as a convex one with linear constraints, and it is solved by a sequence of LP problems obtained by linearizing the objective function at the current point, i.e., the objective function is replaced by its supporting hyperplane at the current point, and by introducing a trust region centered at the current point. In [11] and [12], it is further observed that this problem can be solved very efficiently through the solution of a sequence of 2-D LP problems. In [13], an interior point barrier method is used to solve the same problem based on Newton's method. Each Newton step requires the solution of a KKT system, and an efficient way to solve such systems is proposed in that work. Moving to approaches also dealing with jerk constraints, we mention [14]. In this work, it is observed that jerk constraints are nonconvex but can be written as the difference between two convex functions. Based on this observation, the authors solve the problem by a sequence of convex subproblems obtained by linearizing at the current point the concave part of the jerk constraints and by adding a proximal term in the objective function that plays the same role as a trust region, preventing from taking too large steps. In [15] a slightly different objective function is considered. Rather than minimizing the traveling time along the given path, the integral of the squared difference between the maximum velocity profile and the computed velocity profile is minimized. After representing time-varying control inputs as products of parametric exponential and polynomial functions, the authors reformulate the problem in such a way that its objective function is convex quadratic, while nonconvexity lies in difference-of-convex functions. The resulting problem is tackled through the solution of a sequence of convex subproblems obtained by linearizing the concave part of the nonconvex constraints. In [16], the problem of speed planning for robotic manipulators with jerk constraints is reformulated in such a way that nonconvexity lies in simple bilinear terms. Such bilinear terms are replaced by the corresponding convex and concave envelopes, obtaining the so-called McCormick relaxation, which is the tightest possible convex relaxation of the nonconvex problem. Other approaches dealing with jerk constraints do not rely on the solution of convex subproblems. For instance, in [17], a concatenation of fifth-order polynomials is employed to provide smooth trajectories, which results in quadratic jerk profiles, while, in [18], cubic polynomials are employed, resulting in piecewise constant jerk profiles. The decision process involves the choice of the phase durations, i.e., of the intervals over which a given polynomial applies. A very recent and interesting approach to the problem with jerk constraints is [19]. In this work, an approach based on numerical integration is discussed. Numerical integration has been first applied under acceleration constraints in [20] and [21]. In [19], jerk constraints are taken into account. The algorithm detects a position s along the trajectory where the jerk constraint is singular, that is, the jerk term disappears from one of the constraints. Then, it computes the speed profile up to s by computing two maximum jerk profiles and then connecting them by a minimum jerk profile, found by a shooting method. In general, the overall solution is composed of a sequence of various maximum and minimum jerk profiles. This approach does not guarantee reaching a local minimum of the traversal time. Moreover, since Problem 4 has velocity and acceleration constraints, the jerk constraint is singular for all values of s so that the algorithm presented in [19] cannot be directly applied to Problem 4.

Some algorithms use heuristics to quickly find sub-optimal solutions of acceptable quality. For instance, Villagra *et al.* [22] propose an algorithm that applies to curves composed of clothoids, circles, and straight lines. The algorithm does not guarantee the local optimality of the solution. Raineri and Guarino Lo Bianco [23] present an efficient heuristic algorithm. Also, this method does not guarantee global nor local optimality. Various works in the literature consider jerk bounds in the speed optimization problem for robotic manipulators instead of mobile vehicles. This is a slightly different problem but mathematically equivalent to Problem (1). In particular, paper [24] presents a method based on the solution of a large number of nonlinear and nonconvex subproblems. The resulting algorithm is slow due to a large number of subproblems; moreover, the authors do not prove its convergence. Zhang *et al.* [25] propose a similar method that gives a continuous-time solution. Again, the method is computationally slow since it is based on the numerical solution of a large number of differential equations; moreover, this article does not contain proof of convergence or local optimality. Some other works replace the jerk constraint with *pseudo-jerk*, that is, the derivative of the acceleration with respect to arc length, obtaining a constraint analogous to (4e) and ending up with a convex optimization problem. For instance, Zhang *et al.* [26] add to the objective function a pseudo-jerk penalizing term. This approach is computationally convenient, but substituting (8) with (4e) may be overly restrictive at low speeds.

F. Statement of Contribution

The method presented in this article is a sequential convex one that aims at finding a local optimizer of Problem 2. To be more precise, as usual with nonconvex problems, only convergence to a stationary point can usually be proved. However, the fact that the sequence of generated feasible points is decreasing with respect to the objective function values usually guarantees that the stationary point is a local minimizer, except in rather pathological cases (see [27, p. 19]). Moreover, in our experiments, even after running a local solver from different starting points, we have never been able to detect local minimizers better than the one returned by the method we propose. Thus, a possible, nontrivial, topic for future research could be that of proving the global optimality of the solution. To the best of our knowledge and as detailed in the following, this algorithm is more efficient than the ones existing in the literature since it leverages the special structure of the subproblems obtained as local approximations of Problem 2. We discussed this class of problems in our previous work [28]. This structure allows computing very efficiently a feasible descent direction for the main line-search algorithm; it is one of the key elements that allow us to outperform generic NLP solvers. In summary, the main contributions of this work are: 1) on the theoretical side, the development of an approach for which a rigorous mathematical analysis has been

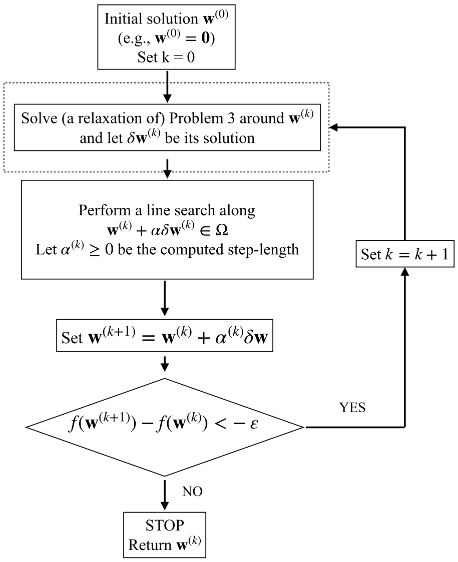

Fig. 1. Flowchart of algorithm SCA. The dashed block corresponds to a call of the procedure `ComputeUpdate`, proposed to solve Problem 3, which represents the main contribution of this article.

performed, proving a convergence result to a stationary point (see Section II) and 2) on the computational side, to exploit heavily the structure of the problem in order to implement the approach in a fairly efficient way (see Sections III and IV) so that its computing times outperform those of nonlinear solvers and are competitive with heuristic approaches that are only able to return suboptimal solutions of lower quality (see Section V).

II. Sequential Algorithm Based on Constraint Linearization

To account for the nonconvexity of Problem 2, we propose a line-search method based on the solution of a sequence of special structured convex problems. Throughout this article, we call this algorithm Sequential Convex Algorithm (SCA), and its flowchart is shown in Fig. 1. It belongs to the class of Sequential Convex Programming algorithms, where, at each iteration, a convex subproblem is solved. In what follows, we denote by $\mathbf{\Omega}$ the feasible region of Problem 2. At each iteration k, we replace the current point $\mathbf{w}^{(k)} \in \mathbf{\Omega}$ with a new point $\mathbf{w}^{(k)} + \alpha^{(k)}\boldsymbol{\delta}\mathbf{w}^{(k)} \in \mathbf{\Omega}$, where the step size $\alpha^{(k)} \in [0, 1]$ is obtained by a *line search* along the descent direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$, which, in turn, is obtained through the solution of a convex problem. The constraints of the convex problem are linear approximations of (10)–(14) around $\mathbf{w}^{(k)}$, while the objective function is the original one. Then, the problem that we consider to compute the direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ is given in the following (superscript k of $\mathbf{w}^{(k)}$ is omitted):

Problem 3:

$$\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1}+\delta w_{i+1}}+\sqrt{w_i+\delta w_i}} \tag{16}$$

$$\mathbf{l_B} \leq \boldsymbol{\delta}\mathbf{w} \leq \mathbf{u_B} \tag{17}$$

$$\delta w_{i+1} - \delta w_i \leq b_{A_i}, \quad i = 1, \dots, n-1 \tag{18}$$

$$\delta w_i - \delta w_{i+1} \leq b_{Di}, \quad i = 1, \dots, n-1 \tag{19}$$

$$\delta w_i - \eta_i \delta w_{i-1} - \eta_i \delta w_{i+1} \leq b_{N_i}, \quad i = 2, \dots, n-1 \tag{20}$$

$$\eta_i \delta w_{i-1} + \eta_i \delta w_{i+1} - \delta w_i \leq b_{P_i}, \quad i = 2, \dots, n-1 \tag{21}$$

where $\mathbf{l_B} = -\mathbf{w}$ and $\mathbf{u_B} = \mathbf{u} - \mathbf{w}$ (recall that $\mathbf{u}$ has been introduced in (10), and its components have been defined immediately in Problem 2), while parameters $\boldsymbol{\eta}$, $\mathbf{b_A}$, $\mathbf{b_D}$, $\mathbf{b_N}$, and $\mathbf{b_P}$ depend on the point $\mathbf{w}$ around which the constraints (10)–(14) are linearized. More precisely, we have

$$\begin{aligned}
b_{A_i} &= 2hA - w_{i+1} + w_i \\
b_{Di} &= 2hA - w_i + w_{i+1} \\
\eta_i &= \frac{3w_{i+1} + 3w_{i-1} + 2w_i}{2(w_{i+1} + w_{i-1} + 6w_i)} \\
b_{P_i} &= \sqrt{\ell_i(\mathbf{w})}\frac{8h^2 J + (w_{i-1} - 2w_i + w_{i+1})\sqrt{\ell_i(\mathbf{w})}}{2(w_{i+1} + w_{i-1} + 6w_i)} \\
b_{N_i} &= \sqrt{\ell_i(\mathbf{w})}\frac{8h^2 J - (w_{i-1} - 2w_i + w_{i+1})\sqrt{\ell_i(\mathbf{w})}}{2(w_{i+1} + w_{i-1} + 6w_i)}
\end{aligned} \tag{22}$$

where ℓ_i is defined in (15). The following proposition is an immediate consequence of the feasibility of $\mathbf{w}$.

Proposition 1: All parameters $\boldsymbol{\eta}$, $\mathbf{b_A}$, $\mathbf{b_D}$, $\mathbf{b_N}$, and $\mathbf{b_P}$ are nonnegative.

The proposed approach follows some standard ideas of sequential quadratic approaches employed in the literature about nonlinearly constrained problems. However, a quite relevant difference is that the true objective function (9) is employed in the problem to compute the direction, rather than a quadratic approximation of such function. This choice comes from the fact that the objective function (9) has some features (in particular, convexity and being decreasing), which, combined with the structure of the linearized constraints, allows for an efficient solution of Problem 3. Problem 3 is a convex problem with a nonempty feasible region ($\boldsymbol{\delta}\mathbf{w} = \mathbf{0}$ is always a feasible solution) and, consequently, can be solved by existing NLP solvers. However, such solvers tend to increase computing times since they need to be called many times within the iterative algorithm SCA. The main contribution of this article lies in the routine `computeUpdate` (see dashed block in Fig. 1), which is able to solve Problem 3 and efficiently returns a descent direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$. To be more precise, we solve a *relaxation* of Problem 3. Such relaxation, as well as the routine to solve it, is detailed in Sections III and IV. In Section III, we present efficient approaches to solve some subproblems, including proper subsets of the constraints. Then, in Section IV, we address the solution of the relaxation of Problem 3.

Remark 1: It is possible to see that, if one of the constraints (13) and (14) is active at $\mathbf{w}^{(k)}$, then, along the direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ computed through the solution of the linearized Problem 3, it holds that $\mathbf{w}^{(k)}+\alpha\boldsymbol{\delta}\mathbf{w}^{(k)} \in \mathbf{\Omega}$ for any sufficiently small $\alpha > 0$. In other words, small perturbations of the current solution $\mathbf{w}^{(k)}$ along direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ do not lead outside the feasible region $\mathbf{\Omega}$. This fact is illustrated in Fig. 2. Let us

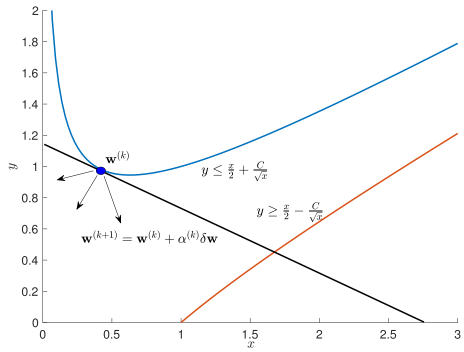

Fig. 2. Constraints (13) and (14) and their linearization ($C = 4h^2 J$).

rewrite constraints (13) and (14) as follows:

$$\left|(x-2y)\sqrt{x}\right| \leq C \tag{23}$$

where $x = \ell_i(\mathbf{w})$, $y = 2w_i$, and $C = 4h^2 J$ is a constant. The feasible region associated with constraint (23) is reported in Fig. 2. In particular, it is the region between the blue and red curves. Suppose that constraint $y \leq (x/2)+(C/2\sqrt{x})$ is active at $\mathbf{w}^{(k)}$ (the case when $y \geq (x/2) - (C/2\sqrt{x})$ is active can be dealt with in a completely analogous way). If we linearize such constraint around $\mathbf{w}^{(k)}$, then we obtain a linear constraint (black line in Fig. 2), which defines a region completely contained into the one defined by the nonlinear constraint $y \leq (x/2)+(C/2\sqrt{x})$. Hence, for each direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ feasible with respect to the linearized constraint, we are always able to perform sufficiently small steps, without violating the original nonlinear constraints, i.e., for $\alpha > 0$ small enough, it holds that $\mathbf{w}^{(k)} + \alpha\boldsymbol{\delta}\mathbf{w}^{(k)} \in \boldsymbol{\Omega}$.

Constraints (13) and (14) can also be rewritten as follows:

$$w_{i-1} + w_{i+1} - 2w_i - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}} \leq 0 \tag{24}$$
$$2w_i - w_{i-1} - w_{i+1} - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}} \leq 0. \tag{25}$$

Note that the functions on the left-hand side of these constraints are concave. Now, we can define a variant of Problem 3 where constraints (20) and (21) are replaced by the following linearizations of constraints (24) and (25):

$$-\beta_i \delta w_{i-1} - \beta_i \delta w_{i+1} + \delta w_i \leq b'_{N_i} \tag{26}$$
$$\theta_i \delta w_{i-1} + \theta_i \delta w_{i+1} - \delta w_i \leq b'_{P_i} \tag{27}$$

where

$$\theta_i = \frac{1 + 2h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}{2 - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}$$
$$\beta_i = \frac{1 - 2h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}{2 + 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}$$
$$b'_{N_i} = \frac{6h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}}}{2 + 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}$$
$$b'_{P_i} = \frac{6h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}}}{2 - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}. \tag{28}$$

The following proposition states that constraints (26) and (27) are tighter than constraints (20) and (21).

Proposition 2: For all $i = 2, \ldots, n-1$, it holds that $\beta_i \leq \eta_i \leq \theta_i$. Equality $\eta_i = \theta_i$ holds if the corresponding nonlinear constraint (24) is active at the current point $\mathbf{w}$. Similarly, $\eta_i = \beta_i$ holds if the corresponding nonlinear constraint (25) is active at the current point $\mathbf{w}$.

Proof: We only prove the results about θ_i and η_i. Those about β_i and η_i are proved in a completely analogous way. By definition of η_i and θ_i, we need to prove that

$$\frac{3w_{i+1} + 3w_{i-1} + 2w_i}{w_{i+1} + 6w_i + w_{i-1}} \leq \frac{1 + 2h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}{2 - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}.$$

After few simple computations, this inequality can be rewritten as

$$4h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}} \geq (w_{i-1} - 2w_i + w_{i+1})$$

which holds in view of feasibility of $\mathbf{w}$ and, moreover, holds as an equality if constraint (24) is active at the current point $\mathbf{w}$, as we wanted to prove. □

In view of this result, by replacing constraints (20) and (21) with (26) and (27), we reduce the search space of the new displacement $\boldsymbol{\delta}\mathbf{w}$. On the other hand, the following proposition states that, with constraints (26) and (27), no line search is needed along the direction $\boldsymbol{\delta}\mathbf{w}$, i.e., we can always choose the step length $\alpha = 1$.

Proposition 3: If constraints (26) and (27) are employed as a replacement of constraints (20) and (21) in the definition of Problem 3, then, for each feasible solution $\boldsymbol{\delta}\mathbf{w}$ of this problem, it holds that $\mathbf{w} + \boldsymbol{\delta}\mathbf{w} \in \Omega$.

Proof: For the sake of convenience, let us rewrite Problem 2 in the following more compact form:

$$\min \ f(\mathbf{w} + \boldsymbol{\delta}\mathbf{w})$$
$$\mathbf{c}(\mathbf{w} + \boldsymbol{\delta}\mathbf{w}) \leq 0 \tag{29}$$

where the vector function $\mathbf{c}$ contains all constraints of Problem 2 and the nonlinear ones are given as in (24) and (25) (recall that, in that case, vector $\mathbf{c}$ is a vector of concave functions). Then, Problem 3 can be written as follows:

$$\min \ f(\mathbf{w} + \delta\mathbf{w}) \quad \mathbf{c}(\mathbf{w}) + \nabla\mathbf{c}(\mathbf{w})\delta\mathbf{w} \leq 0. \tag{30}$$

Now, it is enough to observe that, by concavity,

$$\mathbf{c}(\mathbf{w} + \boldsymbol{\delta}\mathbf{w}) \leq \mathbf{c}(\mathbf{w}) + \nabla\mathbf{c}(\mathbf{w})\boldsymbol{\delta}\mathbf{w}$$

so that each feasible solution of (30) is also feasible for (29). □

The above proposition states that the feasible region of Problem 3, when constraints (26) and (27) are employed in its definition, is a subset of the feasible region $\boldsymbol{\Omega}$ of the original Problem 2. As a final result of this section, we state the following theorem, which establishes convergence of algorithm SCA to a stationary (KKT) point of Problem 2.

Theorem 1: If algorithm SCA is run for an infinite number of iterations and there exists some positive integer value K such that for all iterations $k \geq K$, constraints (26) and (27) are always employed in the definition of Problem 3, and then, the sequence of points $\{\mathbf{w}^{(k)}\}$ generated by the algorithm converges to a KKT point of Problem 2.

In order to prove the theorem, we first need to prove some lemmas.

Lemma 1: The sequence $\{f(\mathbf{w}^{(k)})\}$ of the function values at points generated by algorithm SCA converges to a finite value.

Proof: The sequence is nonincreasing and bounded from below, e.g., by the value $f(\mathbf{u}_B)$, in view of the fact that the objective function f is monotonic decreasing. Thus, it converges to a finite value. □

Next, we need the following result based on strict convexity of the objective function f.

Lemma 2: For each $\delta > 0$ sufficiently small, it holds that

$$\min\left\{\max\{f(\mathbf{x}), f(\mathbf{y})\} - f\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) \right.$$
$$\left. : \mathbf{x}, \mathbf{y} \in \boldsymbol{\Omega},\ \|\mathbf{x}-\mathbf{y}\| \geq \delta\right\} \geq \varepsilon_\delta > 0. \quad (31)$$

Proof: Due to strict convexity, it holds that, $\forall \mathbf{x} \neq \mathbf{y}$,

$$\max\{f(\mathbf{x}), f(\mathbf{y})\} - f\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) > 0.$$

Moreover, the function is a continuous one. Next, we observe that the region

$$\{\mathbf{x}, \mathbf{y} \in \boldsymbol{\Omega} :\ \|\mathbf{x}-\mathbf{y}\| \geq \delta\}$$

is a compact set. Thus, by the Weierstrass theorem, the minimum in (31) is attained, and it must be strictly positive, as we wanted to prove. □

Finally, we prove that also the sequence of points generated by algorithm SCA converges to some point, feasible for Problem 2.

Lemma 3: It holds that

$$\|\delta\mathbf{w}^{(k)}\| \to 0.$$

Proof: Let us assume, by contradiction, that, over some infinite subsequence with index set $\mathcal{K}$, it holds that $\|\delta\mathbf{w}^{(k)}\| \geq 2\rho > 0$ for all $k \in \mathcal{K}$, i.e.,

$$\|\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}\| \geq 2\rho > 0 \quad (32)$$

where $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \delta\mathbf{w}^{(k)}$. Over this subsequence, it holds, by strict convexity, that

$$f\left(\mathbf{w}^{(k+1)}\right) \leq f\left(\mathbf{w}^{(k)}\right) - \xi \quad \forall k \in \mathcal{K} \quad (33)$$

for some $\xi > 0$. Indeed, it follows by optimality of $\mathbf{w}^{(k)} + \delta\mathbf{w}^{(k)}$ for Problem 3 and convexity of f that

$$f\left(\mathbf{w}^{(k+1)}\right) \leq f\left(\frac{\mathbf{w}^{(k+1)} + \mathbf{w}^{(k)}}{2}\right) \leq f\left(\mathbf{w}^{(k)}\right)$$

so that

$$\max\left\{f\left(\mathbf{w}^{(k)}\right), f\left(\mathbf{w}^{(k+1)}\right)\right\} = f\left(\mathbf{w}^{(k)}\right).$$

Then, it follows from (32) and Lemma 2 that we can choose $\xi = \varepsilon_\rho > 0$. Thus, since (33) holds infinitely often, we should have $f(\mathbf{w}^{(k)}) \to -\infty$, which, however, is not possible in view of Lemma 1. □

Now, we are ready to prove Theorem 1.

Proof: As a consequence of Lemma 3, it also holds that

$$\mathbf{w}^{(k)} \to \bar{\mathbf{w}} \in \boldsymbol{\Omega}. \quad (34)$$

Indeed, all points $\mathbf{w}^{(k)}$ belong to the compact feasible region $\boldsymbol{\Omega}$ so that the sequence $\{\mathbf{w}^{(k)}\}$ admits accumulation points. However, due to Lemma 3, the sequence cannot have distinct accumulation points.

Now, let us consider the compact reformulation (29) of Problem 2 and the related linearization (30), equivalent to Problem 3 with the linearized constraints (26) and (27). Since the latter is a convex problem with linear constraints, its local minimizer $\delta\mathbf{w}^{(k)}$ (unique in view of strict convexity of the objective function) fulfills the following KKT conditions:

$$\begin{aligned}
\nabla f\left(\mathbf{w}^{(k)} + \delta\mathbf{w}^{(k)}\right) + \boldsymbol{\mu}_k^\top \nabla\mathbf{c}\left(\mathbf{w}^{(k)}\right) &= \mathbf{0}\\
\mathbf{c}\left(\mathbf{w}^{(k)}\right) + \nabla\mathbf{c}\left(\mathbf{w}^{(k)}\right)\delta\mathbf{w}^{(k)} &\leq 0\\
\boldsymbol{\mu}_k^\top\left(\mathbf{c}\left(\mathbf{w}^{(k)}\right) + \nabla\mathbf{c}\left(\mathbf{w}^{(k)}\right)\delta\mathbf{w}^{(k)}\right) &= 0\\
\boldsymbol{\mu}_k &\geq \mathbf{0} \qquad (35)
\end{aligned}$$

where $\boldsymbol{\mu}_k$ is the vector of Lagrange multipliers. Now, by taking the limit of system (35), possibly over a subsequence, in order to guarantee convergence of the multiplier vectors $\boldsymbol{\mu}_k$ to a vector $\bar{\boldsymbol{\mu}}$, in view of Lemma 3 and (34), we have that

$$\begin{aligned}
\nabla f(\bar{\mathbf{w}}) + \bar{\boldsymbol{\mu}}^\top \nabla\mathbf{c}(\bar{\mathbf{w}}) &= \mathbf{0}\\
\mathbf{c}(\bar{\mathbf{w}}) &\leq 0\\
\bar{\boldsymbol{\mu}}^\top \mathbf{c}(\bar{\mathbf{w}}) &= 0\\
\bar{\boldsymbol{\mu}} &\geq \mathbf{0}
\end{aligned}$$

or, equivalently, the limit point $\bar{\mathbf{w}}$ is a KKT point of Problem 2, as we wanted to prove. □

Remark 2: In algorithm SCA at each iteration, we solve to optimality Problem 3. This is indeed necessary for the final iterations to prove the convergence result stated in Theorem 1. However, during the first iterations, it is not necessary to solve the problem to optimality: finding a feasible descent direction is enough. This does not alter the theoretical properties of the algorithm and allows to reduce the computing times.

In the rest of this article, we refer to constraints (18) and (19) as acceleration constraints, while constraints (20) and (21) [or (26) and (27)] are called (linearized) negative acceleration rate (NAR) and positive acceleration rate (PAR) constraints, respectively. Also, note that, in the different subproblems discussed in the following, we always refer to the linearization with constraints (20) and (21) and, thus, with parameters η_i, but the same results also hold for the linearization with constraints (26) and (27) and, thus, with parameters θ_i and β_i.

III. Subproblem With Acceleration and NAR Constraints

In this section, we propose an efficient method to solve Problem 3 when PAR constraints are removed. The solution of this subproblem becomes part of an approach to solve a suitable relaxation of Problem 3 and, in fact, under very mild assumptions, to solve Problem 3 itself. This is clarified in Section IV. We discuss: 1) the subproblem including only (17) and the acceleration constraints (18) and (19); 2) the subproblem including only (17) and the NAR constraints (20);

and 2) the subproblem including all constraints (17)–(20). Throughout the section, we need the results stated in the following two propositions. Let us consider problems with the following form, where $N = \{1, \ldots, n\}$ and $M_j = \{1, \ldots, m_j\}$, $j \in N$:

$$\begin{aligned} &\min\ g(x_1, \ldots, x_n) \\ &\qquad x_j \le a_{i,j} x_{j-1} + b_{i,j} x_{j+1} + c_{i,j}, \quad i \in M_j,\ j \in N \\ &\qquad \ell_j \le x_j \le u_j, \quad j \in N \end{aligned} \tag{36}$$

where g is a monotonic decreasing function; $a_{i,j}, b_{i,j}, c_{i,j} \ge 0$, for $i \in M_j$ and $j \in N$; $a_{i,1} = 0$ for $i \in M_1$; and $b_{i,n} = 0$ for $i \in M_n$. The following result is proven in [28]. Here, we report the proof in order to make this article self-contained. We denote by P the feasible polytope of problem (36). Moreover, we denote by $\mathbf{z}$ the componentwise maximum of all feasible solutions in P, i.e., for each $j \in N$, $z_j = \max_{\mathbf{x} \in P} x_j$ (note that the above maximum value is attained since P is a polytope).

Proposition 4: The unique optimal solution of (36) is the componentwise maximum $\mathbf{z}$ of all its feasible solutions.

Proof: If we are able to prove that the componentwise maximum $\mathbf{z}$ of all feasible solutions is itself a feasible solution, by monotonicity of g, it must also be the unique optimal solution. In order to prove that $\mathbf{z}$ is feasible, we proceed as follows. For $j \in N$, let $\mathbf{x}^{*j}$ be the optimal solution of $\max_{\mathbf{x} \in P} x_j$ so that $z_j = x_j^{*j}$. Since $\mathbf{x}^{*j} \in P$, then it must hold that $\ell_j \le z_j \le u_j$. Moreover, let us consider the generic constraint

$$x_j \le a_{i,j} x_{j-1} + b_{i,j} x_{j+1} + c_{i,j}$$

for $i \in M_j$. It holds that

$$\begin{aligned} z_j = x_j^{*j} &\le a_{i,j} x_{j-1}^{*j} + b_{i,j} x_{j+1}^{*j} + c_{i,j} \\ &\le a_{i,j} z_{j-1} + b_{i,j} z_{j+1} + c_{i,j} \end{aligned}$$

where the first inequality follows from feasibility of $\mathbf{x}^{*j}$, while the second follows from nonnegativity of a_{ij} and b_{ij} and the definition of $\mathbf{z}$. Since this holds for all $j \in N$, the result is proven. □

Now, consider the problem obtained from (36) by removing some constraints, i.e., by taking $M'_j \subseteq M_j$ for each $j \in N$

$$\begin{aligned} &\min\ g(x_1, \ldots, x_n) \\ &\qquad x_j \le a_{i,j} x_{j-1} + b_{i,j} x_{j+1} + c_{i,j}, \quad i \in M'_j,\ j \in N \\ &\qquad \ell_j \le x_j \le u_j, \quad j \in N. \end{aligned} \tag{37}$$

Later, we also need the result stated in the following proposition.

Proposition 5: The optimal solution $\bar{\mathbf{x}}^\star$ of problem (37) is an upper bound for the optimal solution $\mathbf{x}^\star$ of problem (36), i.e., $\bar{\mathbf{x}}^\star \ge \mathbf{x}^\star$.

Proof: It holds that $\mathbf{x}^\star$ is a feasible solution of problem (37) so that, in view of Proposition 4, $\bar{\mathbf{x}}^\star \ge \mathbf{x}^\star$ holds. □

A. Acceleration Constraints

The simplest case is the one where we only consider the acceleration constraints (18) and (19), besides constraints (17) with a generic upper bound vector $\mathbf{y} \ge \mathbf{0}$. The problem to be solved is

Problem 4:

$$\begin{aligned} &\min_{\boldsymbol{\delta}\mathbf{w} \in \mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1} + \delta w_{i+1}} + \sqrt{w_i + \delta w_i}} \\ &\qquad \mathbf{l_B} \le \boldsymbol{\delta}\mathbf{w} \le \mathbf{y} \\ &\qquad \delta w_{i+1} - \delta w_i \le b_{A_i}, \quad i = 1, \ldots, n-1 \\ &\qquad \delta w_i - \delta w_{i+1} \le b_{D_i}, \quad i = 1, \ldots, n-1. \end{aligned}$$

It can be seen that such a problem belongs to the class of problems (36). Therefore, in view of Proposition 4, the optimal solution of Problem 4 is the componentwise maximum of its feasible region. Moreover, in [3], it has been proven that Algorithm 1, based on a forward and a backward iteration and with $O(n)$ computational complexity, returns an optimal solution of Problem 4.

Algorithm 1 Routine `SolveAcc` for the Solution of the Problem With Acceleration Constraints

input : Upper bound $\mathbf{y}$
output: $\boldsymbol{\delta}\mathbf{w}$
$\delta w_1 = 0$, $\delta w_n = 0$;
for $i = 1$ **to** $n-1$ **do**
  $\delta w_{i+1} = \min\{\delta w_i + b_{A_i}, y_{i+1}\}$
for $i = n-1$ **to** 1 **do**
  $\delta w_i = \min\{\delta w_{i+1} + b_{A_i}, y_i\}$
return $\boldsymbol{\delta}\mathbf{w}$

B. NAR Constraints

Now, we consider the problem only including NAR constraints (20) and constraints (17) with upper bound vector $\mathbf{y}$

Problem 5:

$$\begin{aligned} &\min_{\boldsymbol{\delta}\mathbf{w} \in \mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1} + \delta w_{i+1}} + \sqrt{w_i + \delta w_i}} \\ &\qquad \mathbf{0} \le \boldsymbol{\delta}\mathbf{w} \le \mathbf{y} \qquad (38) \\ &\qquad \delta w_i \le \eta_i(\delta w_{i-1} + \delta w_{i+1}) + b_{N_i}, \quad i = 2, \ldots, n-1 \qquad (39) \end{aligned}$$

where $y_1 = y_n = 0$ because of the boundary conditions. Also, this problem belongs to the class of problems (36) so that Proposition 4 states that its optimal solution is the componentwise maximum of its feasible region. Problem 5 can be solved by using the graph-based approach presented in [4] and [28]. However, Cabassi *et al.* [4] show that, by exploiting the structure of a simpler version of the NAR constraints, it is possible to develop an algorithm more efficient than the graph-based one. Our purpose is to extend the results presented in [4] to a case with different and more challenging NAR constraints in order to develop an efficient algorithm outperforming the graph-based one.

Now, let us consider the restriction of Problem 5 between two generic indexes s and t such that $1 \le s < t \le n$, obtained by fixing $\delta w_s = y_s$ and $\delta w_t = y_t$ and by considering only the

NAR and upper bound constraints at $s+1,\dots,t-1$. Let $\boldsymbol{\delta}\mathbf{w}^*$ be the optimal solution of the restriction. We first prove the following lemma.

Lemma 4: The optimal solution $\boldsymbol{\delta}\mathbf{w}^*$ of the restriction of Problem 5 between two indexes s and t, $1 \le s < t \le n$, is such that, for each $j \in \{s+1,\dots,t-1\}$, either $\delta w_j^* \le y_j$ or $\delta w_j^* \le \eta_j(\delta w_{j+1}^* + \delta w_{j-1}^*) + b_{N_j}$ holds as an equality.

Proof: It is enough to observe that, in case both inequalities were strict for some j, then, in view of the monotonicity of the objective function, we could decrease the objective function value by increasing the value of δw_j^*, thus contradicting optimality of $\boldsymbol{\delta}\mathbf{w}^*$. □

Note that the above result also applies to the full Problem 5, which corresponds to the special case $s = 1$, $t = n$ with $y_1 = y_n = 0$. In view of Lemma 4, we have that there exists an index j, with $s < j \le t$, such that: 1) $\delta w_j^* = y_j$; 2) the upper bound constraint is not active at $s+1,\dots,j-1$; and 3) all NAR constraints $s+1,\dots,j-1$ are active. Then, j is the lowest index in $\{s+1,\dots,t-1\}$ where the upper bound constraint is active If index j were known, then the following observation allows returning the components of the optimal solution between s and j. Let us first introduce the following definitions of matrix $\mathbf{A}$ and vector $\mathbf{q}$:

$$\mathbf{A} = \begin{bmatrix} 1 & -\eta_{s+1} & 0 & \cdots & 0 \\ -\eta_{s+2} & 1 & -\eta_{s+2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -\eta_{j-1} & 1 \end{bmatrix}$$

$$\mathbf{q} = \begin{bmatrix} b_{N\,s+1} + \eta_{s+1} y_s \\ b_{N\,s+2} \\ \vdots \\ b_{N\,j-2} \\ b_{N\,j-1} + \eta_{j-1} y_j \end{bmatrix}. \tag{40}$$

Note that $\mathbf{A}$ is the square submatrix of the NAR constraints restricted to rows $s+1$ up to $j-1$ and the related columns.

Observation 1: Let $\boldsymbol{\delta}\mathbf{w}^*$ be the optimal solution of the restriction of Problem 5 between s and t and let $s < j$. If constraints $\delta w_s^* \le y_s$, $\delta w_j^* \le y_j$, and $\delta w_i^* \le \eta_i(\delta w_{i+1}^* + \delta w_{i-1}^*) + b_{N_i}$, for $i = s+1,\dots,j-1$, are all active, then $\delta w_{s+1}^*,\dots,\delta w_{j-1}^*$ are obtained by the solution of the following tridiagonal system:

$$\begin{aligned} &\delta w_s = y_s \\ &\qquad \delta w_r - \eta_r \delta w_{r+1} - \eta_r \delta w_{r-1} = b_{N\,r}, \quad r = s+1,\dots,j-1 \\ &\qquad \delta w_j = y_j \end{aligned}$$

or, equivalently, as

$$\begin{aligned} &\delta w_{s+1} - \eta_{s+1}\bar{x}_{s+2} \\ &= b_{N\,s+1} + \eta_{s+1} y_s \\ &\quad \delta w_r - \eta_r \delta w_{r+1} - \eta_r \delta w_{r-1} = b_{N\,r}, \quad r = s+2,\dots,j-2 \\ &\quad \delta w_{s+1} - \eta_{s+1}\bar{x}_{s+2} = b_{N\,s+1} + \eta_{s+1} y_s. \end{aligned} \tag{41}$$

In the matrix form, the above tridiagonal linear system can be written as

$$\mathbf{A}\boldsymbol{\delta}\mathbf{w}_{s+1,j-1}^* = \mathbf{q} \tag{42}$$

where matrix $\mathbf{A}$ and vector $\mathbf{q}$ are defined in (40) and $\boldsymbol{\delta}\mathbf{w}_{s+1,j-1}^*$ is the restriction of vector $\boldsymbol{\delta}\mathbf{w}$ to its components between $s+1$ and $j-1$.

Tridiagonal systems

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \quad i = 1,\dots,m$$

with $a_1 = c_m = 0$ can be solved through so-called Thomas algorithm [29] with $O(m)$ operations. In order to detect the lowest index $j \in \{s+1,\dots,t-1\}$ such that the upper bound constraint is active at j, we propose Algorithm 2, also called `SolveNAR` and described in what follows. We initially set $j = t$. Then, at each iteration, we solve the linear system (42). Let $\bar{\mathbf{x}} = (\bar{x}_{s+1},\dots,\bar{x}_{j-1})$ be its solution. We check whether it is feasible and optimal or not. Namely, if there exists $k \in \{s+1,\dots,j-1\}$ such that either $\bar{x}_k < 0$ or $\bar{x}_k > y_k$, then $\bar{\mathbf{x}}$ is unfeasible, and consequently, we need to reduce j by 1. If $\bar{x}_k = y_k$ for some $k \in \{s+1,\dots,j-1\}$, then we also reduce j by 1 since j is not in any case the lowest index of the optimal solution where the upper bound constraint is active. Finally, if $0 \le \bar{x}_k < y_k$, for $k = s+1,\dots,j-1$, then we need to verify if $\bar{\mathbf{x}}$ is the best possible solution over the interval $\{s+1,\dots,j-1\}$. We are able to check that after proving the following result.

Proposition 6: Let matrix $\mathbf{A}$ and vector $\mathbf{q}$ be defined as in (40). The optimal solution $\boldsymbol{\delta}\mathbf{w}^*$ of the restriction of Problem 5 between s and t satisfies

$$\delta w_s^* = y_s, \quad \delta w_r^* = \bar{x}_r, \; r = s+1,\dots,j-1, \; \delta w_j^* = y_j \tag{43}$$

if and only if the optimal value of the LP problem

$$\begin{aligned} \max_{\boldsymbol{\epsilon}} \quad & \mathbf{1}^T\boldsymbol{\epsilon} \\ & \mathbf{A}\boldsymbol{\epsilon} \le \mathbf{0} \\ & \boldsymbol{\epsilon} \le \bar{\mathbf{y}} - \bar{\mathbf{x}} \end{aligned} \tag{44}$$

is strictly positive or, equivalently, if the following system admits no solution:

$$\mathbf{A}^T\boldsymbol{\lambda} = \mathbf{1}, \quad \boldsymbol{\lambda} \ge \mathbf{0}. \tag{45}$$

Proof: Let us first assume that $\boldsymbol{\delta}\mathbf{w}^*$ does not fulfill (43). Then, in view of Lemma 4, j is not the lowest index such that the upper bound is active at the optimal solution, and consequently, $\delta w_k^* = y_k > \bar{x}_k$ for some $k \in \{s+1,\dots,j-1\}$. Such optimal solution must be feasible, and in particular, it must satisfy all NAR constraints between $s+1$ and $j-1$ and the upper bound constraints between $s+1$ and j, i.e.,

$$\begin{aligned} &\delta w_{s+1}^* - \eta_{s+1}\delta w_{s+2}^* \\ &\le b_{N\,s+1} + \eta_{s+1} y_s \\ &\quad \delta w_r^* - \eta_r \delta w_{r+1}^* - \eta_r \delta w_{r-1}^* \le b_{N\,r}, \quad r = s+2,\dots,j-2 \\ &\quad \delta w_{j-1}^* - \eta_{j-1}\delta w_{j-2}^* - \eta_{j-1}\delta w_j^* \le b_{N\,j-1} \\ &\quad \delta w_r^* \le y_r, \quad r = s+1,\dots,j. \end{aligned}$$

In view of $\delta w_j^* \le y_j$ and $\eta_{j-1} \ge 0$, $\boldsymbol{\delta}\mathbf{w}^*$ also satisfies the following system of inequalities:

$$\begin{aligned} &\delta w_{s+1}^* - \eta_{s+1}\delta w_{s+2}^* \\ &\le b_{N\,s+1} + \eta_{s+1} y_s \\ &\quad \delta w_r^* - \eta_r \delta w_{r+1}^* - \eta_r \delta w_{r-1}^* \le b_{N\,r}, \quad r = s+2,\dots,j-2 \end{aligned}$$

$$\delta w^*_{j-1} - \eta_{j-1}\delta w^*_{j-2} \le b_{N\,j-1} + \eta_{j-1} y_j$$
$$\delta w^*_r \le y_r, \quad r = s+1,\dots,j-1.$$

After making the change of variables $\delta w^*_r = \bar{x}_r + \epsilon_r$ for $r = s+1,\dots,j-1$, and recalling that $\bar{\mathbf{x}}$ solves system (41), the system of inequalities can be further rewritten as

$$\begin{aligned}
&\epsilon_{s+1} - \eta_{s+1}\epsilon_{s+2} \le 0\\
&\qquad \epsilon_r - \eta_r\epsilon_{r+1} - \eta_r\epsilon_{r-1} \le 0, \quad r = s+2,\dots,j-2\\
&\qquad \epsilon_{j-1} - \eta_{j-1}\epsilon_{j-2} \le 0\\
&\qquad \epsilon_r \le y_r - \bar{x}_r, \quad r = s+1,\dots,j-1.
\end{aligned}$$

Finally, recalling the definition of matrix $\mathbf{A}$ and vector $\mathbf{q}$ given in (40), this can also be written in a more compact form as

$$\begin{aligned}
\mathbf{A}\boldsymbol{\epsilon} &\le \mathbf{0}\\
\boldsymbol{\epsilon} &\le \bar{\mathbf{y}} - \bar{\mathbf{x}}.
\end{aligned}$$

If $\delta w^*_k = y_k > \bar{x}_k$ for some $k \in \{s+1,\dots,j-1\}$, then the system must admit a solution with $\epsilon_k > 0$. This is equivalent to prove that problem (44) has an optimal solution with at least one strictly positive component, and the optimal value is strictly positive. Indeed, in view of the definition of matrix $\mathbf{A}$, problem (44) has the structure of the problems discussed in Proposition 4. More precisely, to see that, we need to remark that maximizing $\mathbf{1}^T\boldsymbol{\epsilon}$ is equivalent to minimizing the decreasing function $-\mathbf{1}^T\boldsymbol{\epsilon}$. Then, observing that $\boldsymbol{\epsilon} = \mathbf{0}$ is a feasible solution of problem (44), by Proposition 4, the optimal solution $\boldsymbol{\epsilon}^*$ must be a nonnegative vector, and since at least one component, namely, component k, is strictly positive, then the optimal value must also be strictly positive.

Conversely, let us assume that the optimal value is strictly positive, and $\boldsymbol{\epsilon}^*$ is an optimal solution with at least one strictly positive component. Then, there are two possible alternatives. Either the optimal solution $\boldsymbol{\delta}\mathbf{w}^*$ of the restriction of Problem 5 between s and t is such that $\delta w^*_j < y_j$, in which case (43) obviously does not hold, or $\delta w^*_j = y_j$. In the latter case, let us assume by contradiction that (43) holds. We observe that the solution that is defined as follows:

$$\begin{aligned}
x'_s &= y_s\\
x'_r &= \bar{x}_r + \epsilon^*_r = \delta w^*_r + \epsilon^*_r, \quad r = s+1,\dots,j-1\\
x'_j &= y_j = \delta w^*_j\\
x'_r &= \delta w^*_r, \quad r = j+1,\dots,t
\end{aligned}$$

is feasible for the restriction of Problem 5 between s and t. Indeed, by feasibility of $\boldsymbol{\epsilon}^*$ in problem (44), all upper bound and NAR constraints between s and $j-1$ are fulfilled. Those between, $j+1$ and t, are also fulfilled by the feasibility of $\boldsymbol{\delta}\mathbf{w}^*$. Then, we only need to prove that the NAR constraint at j is satisfied. By feasibility of $\boldsymbol{\delta}\mathbf{w}^*$ and in view of $\epsilon^*_{j-1}, \eta_j \ge 0$, we have that

$$\begin{aligned}
x'_j = \delta w^*_j &\le \eta_j \delta w^*_{j-1} + \eta_j \delta w^*_{j+1} + b_{N\,j}\\
&\le \eta_j\left(\delta w^*_{j-1} + \epsilon_{j-1}\right) + \eta_j \delta w^*_{j+1} + b_{N\,j}\\
&= \eta_j x'_{j-1} + \eta_j x'_{j+1} + b_{N\,j}.
\end{aligned}$$

Thus, $\mathbf{x}'$ is feasible such that $\mathbf{x}' \ge \boldsymbol{\delta}\mathbf{w}^*$ with at least one strict inequality (recall that at least one component of $\boldsymbol{\epsilon}^*$ is strictly positive), which contradicts the optimality of $\boldsymbol{\delta}\mathbf{w}^*$ (recall that the optimal solution must be the componentwise maximum of all feasible solutions).

In order to prove the last part, i.e., problem (44) has a positive optimal value if and only if (45) admits no solution, and we notice that the optimal value is positive if and only if the feasible point $\boldsymbol{\epsilon} = \mathbf{0}$ is not an optimal solution, or equivalently, the null vector is not a KKT point. Since, at $\boldsymbol{\epsilon} = \mathbf{0}$, constraints $\boldsymbol{\epsilon} \le \bar{\mathbf{y}} - \bar{\mathbf{x}}$ cannot be active, then the KKT conditions for problem (44) at this point are exactly those established in (45), where vector $\boldsymbol{\lambda}$ is the vector of Lagrange mutlpliers for constraints $\mathbf{A}\boldsymbol{\epsilon} \le \mathbf{0}$. This concludes the proof. □

Then, if (45) admits no solution, (43) does not hold, and again, we need to reduce j by 1. Otherwise, we can fix the optimal solution between s and j according to (43). After that, we recursively call the routine `SolveNAR` on the remaining subinterval $\{j,\dots,t\}$ in order to obtain the solution over the full interval.

Remark 3: In Algorithm 2, routine `isFeasible` is the routine used to verify if, for $k = s+1,\dots,j-1$, $0 \le \bar{x}_k < y_k$, while `isOptimal` is the procedure to check optimality of $\bar{\mathbf{x}}$ over the interval $\{s+1,\dots,j-1\}$, i.e., (43) holds.

Now, we are ready to prove that Algorithm 2 solves Problem 5.

Proposition 7: The call `solveNAR(`$\mathbf{y}$`, 1, `n`)` of Algorithm 2 returns the optimal solution of Problem 5.

Proof: After the call `solveNAR(`$\mathbf{y}$`, 1, `n`)`, we are able to identify the portion of the optimal solution between 1 and some index j_1, $1 < j_1 \le n$. If $j_1 = n$, then we are done. Otherwise, we make the recursive call `solveNAR(`$\mathbf{y}$`, `j_1`, `n`)`, which enables to identify also the portion of the optimal solution between j_1 and some index j_2, $j_1 < j_2 \le n$. If $j_2 = n$, then we are done. Otherwise, we make the recursive call `solveNAR(`$\mathbf{y}$`, `j_2`, `n`)` and so on. After at most n recursive calls, we are able to return the full optimal solution. □

Algorithm 2 `SolveNAR`($\mathbf{y}, s, t$)

input : Upper bound $\mathbf{y}$ and two indices s and t with $1 \le s < t \le n$
output: $\boldsymbol{\delta}\mathbf{w}^*$

```
Set j = t;
δw* = y;
while j ≥ s + 1 do
    Compute the solution x̄ of the linear system (42);
    if isFeasible(x̄) and isOptimal(x̄) then
        Break;
    else
        Set j = j − 1;
for i = s + 1, ..., j − 1 do
    Set δw*_i = x̄_i;
return δw* = min{δw*, SolveNAR(δw*, j, t)};
```

Remark 4: Note that Algorithm 2 involves solving a significant amount of linear systems, both to compute $\bar{\mathbf{x}}$ and verify its optimality [see (42) and (45)]. Some tricks can be employed to reduce the number of operations. Some of these are discussed in [30].

The following proposition states the worst case complexity of `solveNAR(`$\mathbf{y}$`,1,`n`)`.

Proposition 8: Problem 5 can be solved with $O(n^3)$ operations by running the procedure `SolveNAR`$(\mathbf{y}, 1, n)$ and by using the Thomas algorithm for the solution of each linear system.

Proof: In the worst case, at the first call, we have $j_1 = 2$ since we need to go all the way from $j = n$ down to $j = 2$. Since, for each j, we need to solve a tridiagonal system, which requires at most $O(n)$ operations, the first call of `SolveNAR` requires $O(n^2)$ operations. This is similar for all successive calls, and since the number of recursive calls is at most $O(n)$, the overall effort is at most of $O(n^3)$ operations. □

In fact, what we observed is that the practical complexity of the algorithm is much better, namely, $\Theta(n^2)$.

C. Acceleration and NAR Constraints

Now, we discuss the problem with acceleration and NAR constraints, with upper bound vector $\mathbf{y}$, i.e.,

Problem 6:

$$\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1}+\delta w_{i+1}}+\sqrt{w_i+\delta w_i}}$$

$$\mathbf{l_B} \leq \boldsymbol{\delta}\mathbf{w} \leq \mathbf{y}$$

$$\delta w_{i+1} - \delta w_i \leq b_{A_i}, \quad i = 1, \ldots, n-1$$

$$\delta w_i - \delta w_{i+1} \leq b_{Di}, \quad i = 1, \ldots, n-1$$

$$\delta w_i - \eta_i \delta w_{i-1} - \eta_i \delta w_{i+1} \leq b_{N_i}, \quad i = 2, \ldots, n-1.$$

We first remark that Problem 6 has the structure of problem (36) so that, by Proposition 4, its unique optimal solution is the componentwise maximum of its feasible region. As for Problem 5, we can solve Problem 6 by using the graph-based approach proposed in [28]. However, Cabassi *et al.* [4] show that, if we adopt a very efficient procedure to solve Problems 4 and 5, then it is worth splitting the full problem into two separated ones and use an iterative approach (see Algorithm 3). Indeed, Problems 4–6 share the common property that their optimal solution is also the componentwise maximum of the corresponding feasible region. Moreover, according to Proposition 5, the optimal solutions of Problems 4 and 5 are valid upper bounds for the optimal solution (actually, also for any feasible solution) of the full Problem 6. In Algorithm 3, we first call the procedure `SolveACC` with input the upper bound vector $\mathbf{y}$. Then, the output of this procedure, which, according to what we have just stated, is an upper bound for the solution of the full Problem 6, satisfies $\boldsymbol{\delta}\mathbf{w}_{\text{Acc}} \leq \mathbf{y}$, and becomes the input for a call of the procedure `SolveNAR`. The output $\boldsymbol{\delta}\mathbf{w}_{\text{NAR}}$ of this call is again an upper bound for the solution of the full Problem 6, and it satisfies $\boldsymbol{\delta}\mathbf{w}_{\text{NAR}} \leq \boldsymbol{\delta}\mathbf{w}_{\text{Acc}}$. This output becomes the input of a further call to the procedure `SolveACC`, and we proceed in this way until the distance between two consecutive output vectors falls below a prescribed tolerance value ε. The following proposition states that the sequence of output vectors generated by the alternate calls to the procedures `SolveACC` and `SolveNAR` converges to the optimal solution of the full Problem 6.

Proposition 9: Algorithm 3 converges to the the optimal solution of Problem 6 when $\varepsilon = 0$ and stops after a finite number of iterations if $\varepsilon > 0$.

Proof: We have observed that the sequence of alternate solutions of Problems 4 and 5, here denoted by $\{\mathbf{y}_t\}$, is: 1) a sequence of valid upper bounds for the optimal solution of Problem 6; 2) componentwise monotonic nonincreasing; and 3) componentwise bounded from below by the null vector. Thus, if $\varepsilon = 0$, an infinite sequence is generated, which converges to some point $\bar{\mathbf{y}}$, which is also an upper bound for the optimal solution of Problem 6 but, more precisely, by continuity, is also a feasible point of the problem and, is thus, also the optimal solution of the problem. If $\varepsilon > 0$, due to the convergence to some point $\bar{\mathbf{y}}$, at some finite iteration, the exit condition of the while loop must be satisfied. □

Algorithm 3 Algorithm `SolveACCNAR` for the Solution of Problem 6

input : The upper bound $\mathbf{y}$ and the tolerance ε
output: The optimal solution $\boldsymbol{\delta}\mathbf{w}^*$ and the optimal value f^*

$\boldsymbol{\delta}\mathbf{w}_{\text{Acc}} =$ `SolveACC`$(\mathbf{y})$;
$\boldsymbol{\delta}\mathbf{w}_{\text{NAR}} =$ `SolveNAR`$(\boldsymbol{\delta}\mathbf{w}_{\text{Acc}}, 1, n)$;
while $\|\boldsymbol{\delta}\mathbf{w}_{NAR} - \boldsymbol{\delta}\mathbf{w}_{Acc}\| > \varepsilon$ **do**
    $\boldsymbol{\delta}\mathbf{w}_{\text{Acc}} =$ `SolveACC`$(\boldsymbol{\delta}\mathbf{w}^*)$;
    $\boldsymbol{\delta}\mathbf{w}_{\text{NAR}} =$ `SolveNAR`$(\boldsymbol{\delta}\mathbf{w}_{\text{Acc}}, 1, n)$;
$\boldsymbol{\delta}\mathbf{w}^* = \boldsymbol{\delta}\mathbf{w}_{\text{NAR}}$;
return $\boldsymbol{\delta}\mathbf{w}^*$, `evaluateObj`$(\boldsymbol{\delta}\mathbf{w}^*)$

IV. Descent Method for the Case of Acceleration, PAR, and NAR Constraints

Unfortunately, PAR constraints (21) do not satisfy the assumptions requested in Proposition 4 in order to guarantee that the componentwise maximum of the feasible region is the optimal solution of Problem 3. However, in Section III, we have shown that Problem 6, i.e., Problem 3 without the PAR constraints, can be efficiently solved by Algorithm 3. Our purpose then is to separate the acceleration and NAR constraints from the PAR constraints.

Definition 1: Let $f:\mathbb{R}^n \to \mathbb{R}$ be the objective function of Problem 3, and let $\mathcal{D}$ be the region defined by the acceleration and NAR constraints (the feasible region of Problem 6). We define the function $F:\mathbb{R}^n \to \mathbb{R}$ as follows:

$$F(\mathbf{y}) = \min\{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{D}, \mathbf{x} \leq \mathbf{y}\}.$$

Namely, F is the optimal value function of Problem 6 when the upper bound vector is $\mathbf{y}$.

Proposition 10: Function F is a convex function.

Proof: Since Problem 6 is convex, then the optimal value function F is convex (see [31, Sec. 5.6.1]). □

Now, let us introduce the following problem:

Problem 7:

$$\min_{\mathbf{y}\in\mathbb{R}^n} \quad F(\mathbf{y}) \tag{46}$$

$$\eta_i(y_{i-1}+y_{i+1}) - y_i \le b_{P_i}, \quad i=2,\ldots,n-1 \tag{47}$$

$$\mathbf{l_B} \le \mathbf{y} \le \mathbf{u_B}. \tag{48}$$

Such a problem is a relaxation of Problem 3. Indeed, each feasible solution of Problem 3 is also feasible for Problem 7, and the value of F at such solution is equal to the value of the objective function of Problem 3 at the same solution. We solve Problem 7 rather than Problem 3 to compute the new displacement $\boldsymbol{\delta}\mathbf{w}$. More precisely, if $\mathbf{y}^*$ is the optimal solution of Problem 7, then we set

$$\boldsymbol{\delta}\mathbf{w} = \arg\min_{\mathbf{x}\in\mathcal{D},\mathbf{x}\le\mathbf{y}^*} f(\mathbf{x}). \tag{49}$$

In the following proposition, we prove that, under a very mild condition, the optimal solution of Problem 7 computed in (49) is feasible and, thus, optimal for Problem 3 so that, although we solve a relaxation of the latter problem, we return an optimal solution for it.

Proposition 11: Let $\mathbf{w}^{(k)}$ be the current point. If

$$\ell_j(\boldsymbol{\delta}\mathbf{w}) \le \ell_j\left(\mathbf{w}^{(k)}\right)\left(3+\min\left\{0,\xi\left(\mathbf{w}^{(k)}\right)\right\}\right), \quad j=2,\ldots,n-1 \tag{50}$$

where $\boldsymbol{\delta}\mathbf{w}$ is computed through (49) and

$$\xi\left(\mathbf{w}^{(k)}\right) = \frac{\sqrt{\ell_j\left(\mathbf{w}^{(k)}\right)}\left(w_{j-1}^{(k)} + w_{j+1}^{(k)} - 2w_j^{(k)}\right)}{2h^2 J} \ge -2$$

(the inequality follows from feasibility of $\mathbf{w}^{(k)}$), then $\boldsymbol{\delta}\mathbf{w}$ is feasible for Problem 3, both if the nonlinear constraints are linearized as in (20) and (21), and if they are linearized as in (26) and (27).

Proof: First, we notice that, if we prove the result for the tighter constraints (26) and (27), then it must also hold for constraints (20) and (21). Thus, we prove the result only for the former. By definition (49), $\boldsymbol{\delta}\mathbf{w}$ satisfies the acceleration and NAR constraints so that

$$\begin{aligned}
\delta w_j &\le \delta w_{j+1} + b_{D_j}\\
\delta w_j &\le \delta w_{j-1} + b_{A_{j-1}}\\
\delta w_j &\le \beta_j\left(\delta w_{j+1} + \delta w_{j-1}\right) + b'_{N_j}\\
\delta w_j &\le y_j^*.
\end{aligned}$$

At least one of these constraints must be active; otherwise, δw_j could be increased, thus contradicting optimality. If the active constraint is $\delta w_j \le \beta_j(\delta w_{j+1} + \delta w_{j-1}) + b'_{N_j}$, then constraint (27) can be rewritten as follows:

$$\begin{aligned}
4h^2 J\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{3}{2}}&\left(\delta w_{j+1} + 2\delta w_j + \delta w_{j-1}\right)\\
&\le 12h^2 J\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{1}{2}}
\end{aligned}$$

or, equivalently,

$$\ell_j(\boldsymbol{\delta}\mathbf{w}) \le 3\ell_j\left(\mathbf{w}^{(k)}\right)$$

implied by (50), and thus, the constraint is satisfied under the given assumption. If $\delta w_j = y_j^*$, then

$$\theta_j\left(\delta w_{j-1} + \delta w_{j+1}\right) \le \theta_j\left(y_{j-1}^* + y_{j+1}^*\right) \le y_j^* + b'_{P_j} = \delta w_j + b'_{P_j}$$

where the second inequality follows from the fact that $\mathbf{y}^*$ satisfies the PAR constraints. Now, let $\delta w_j = \delta w_{j+1} + b_{D_j}$ (the case when $\delta w_j \le \delta w_{j-1} + b_{A_{j-1}}$ is active can be dealt with in a completely analogous way). First, we observe that $\delta w_j \ge \delta w_{j-1} - b_{D_{j-1}}$. Then,

$$2\delta w_j \ge \delta w_{j+1} + \delta w_{j-1} + b_{D_j} - b_{D_{j-1}}.$$

In view of the definitions of b_{D_j} and $b_{D_{j-1}}$, this can also be written as

$$2\delta w_j \ge \delta w_{j+1} + \delta w_{j-1} + w_{j+1}^{(k)} - 2w_j^{(k)} + w_{j-1}^{(k)}. \tag{51}$$

Now, after recalling the definitions of θ_j and b'_{P_j} given in (28), and setting $\Delta = h^2 J$, (27) can be rewritten as

$$\begin{aligned}
2\delta w_j \ge \delta w_{j+1} + \delta w_{j-1} &+ 2\Delta\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{3}{2}}\ell_j(\boldsymbol{\delta}\mathbf{w})\\
&- 6\Delta\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{1}{2}}.
\end{aligned}$$

Taking into account (51), such inequality certainly holds if

$$\begin{aligned}
w_{j+1}^{(k)} - 2w_j^{(k)} + w_{j-1}^{(k)} &\ge 2\Delta\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{3}{2}}\ell_j(\boldsymbol{\delta}\mathbf{w})\\
&\quad - 6\Delta\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{1}{2}}
\end{aligned}$$

which is equivalent to

$$\ell_j(\boldsymbol{\delta}\mathbf{w}) \le \ell_j\left(\mathbf{w}^{(k)}\right)\left(3 + \xi\left(\mathbf{w}^{(k)}\right)\right).$$

This is also implied by (50). □

Assumption (50) is mild. In order to fulfill it, one can impose restrictions on $\delta w_{j-1}, \delta w_j$ and δw_{j+1}. In fact, in the computational experiments, we did not impose such restrictions unless a positive step-length along the computed direction $\boldsymbol{\delta}\mathbf{w}$ could not be taken (which, however, never occurred in our experiments).

Now, let us turn our attention toward the solution of Problem 7. In order to solve it, we propose a descent method. We can exploit the information provided by the dual optimal solution $\boldsymbol{\nu} \in \mathbb{R}_+^n$ associated with the upper bound constraints of Problem 6. Indeed, from the sensitivity theory, we know that the dual solution is related to the gradient of the optimal value function F (see Definition 1) and provides information about how it changes its value for small perturbations of the upper bound values (for further details, see [31, Secs. 5.6.2 and 5.6.5]). Let $\mathbf{y}^{(t)}$ be a feasible solution of Problem 7 and $\boldsymbol{\nu} \in \mathbb{R}_+^n$ be the Lagrange multipliers of the upper bound constraints of Problem 6 when the upper bound is $\mathbf{y}^{(t)}$. Let

$$\varphi_i = b_{P_i} - \eta_i\left(y_{i-1}^{(t)} + y_{i+1}^{(t)}\right) + y_i^{(t)}, \quad i=2,\ldots,n-1.$$

Then, a *feasible descent direction* $\mathbf{d}^{(t)}$ can be obtained by solving the following LP problem:

Problem 8:

$$\min_{\mathbf{d}\in\mathbb{R}^n} -\boldsymbol{\nu}^T\mathbf{d} \tag{52}$$

$$\eta_i(d_{i-1}+d_{i+1}) - d_i \le \varphi_i, \quad i=2,\ldots,n-1 \tag{53}$$

$$\mathbf{l_B} \le \mathbf{y}^{(t)} + \mathbf{d} \le \mathbf{u_B} \tag{54}$$

where the objective function (52) imposes that $\mathbf{d}^{(t)}$ is a descent direction, while constraints (53) and (54) guarantee feasibility with respect to Problem 7. Problem 8 is an LP problem, and consequently, it can easily be solved through a standard LP solver. In particular, we employed GUROBI [32]. Unfortunately, since the information provided by the dual optimal solution $\boldsymbol{\nu}$ is local and related to small perturbations of the upper bounds, it might happen that $F(\mathbf{y}^{(t)}+\mathbf{d}^{(t)}) \geq F(\mathbf{y}^{(t)})$. To overcome this issue, we introduce a trust-region constraint in Problem 8. Thus, let $\sigma^{(t)} \in \mathbb{R}_+$ be the radius of the trust region at iteration t; then, we have

Problem 9:

$$\min_{\mathbf{d}\in\mathbb{R}^n} -\boldsymbol{\nu}^T \mathbf{d} \tag{55}$$

$$\eta_i(d_{i-1}+d_{i+1}) - d_i \leq \varphi_i, \quad i = 2, \ldots, n-1 \tag{56}$$

$$\bar{\mathbf{l}}_{\mathbf{B}} \leq \mathbf{d} \leq \bar{\mathbf{u}}_{\mathbf{B}} \tag{57}$$

where $\bar{l}_{B_i} = \max\{l_{B_i} - y_i^{(t)}, -\sigma^{(t)}\}$ and $\bar{u}_{B_i} = \min\{u_{B_i} - y_i^{(t)}, \sigma^{(t)}\}$ for $i = 1, \ldots, n$. After each iteration of the descent algorithm, we change the radius $\sigma^{(t)}$ according to the following rules.

1) If $F(\mathbf{y}^{(t)}+\mathbf{d}^{(t)}) \geq F(\mathbf{y}^{(t)})$, then we set $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)}$, and we tight the trust region by decreasing $\sigma^{(t)}$ by a factor $\tau \in (0, 1)$.
2) If $F(\mathbf{y}^{(t)}+\mathbf{d}^{(t)}) < F(\mathbf{y}^{(t)})$, then we set $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)}+\mathbf{d}^{(t)}$ and enlarge the radius $\sigma^{(t)}$ by a factor $\rho > 1$.

The proposed descent algorithm is sketched in Fig. 3, which reports the flowchart of the procedure `ComputeUpdate` used in algorithm SCA. We initially set $\mathbf{y}^{(0)} = \mathbf{0}$. At each iteration t, we evaluate the objective function $F(\mathbf{y}^t)$ by solving Problem 6 with upper bound vector $\mathbf{y}^{(t)}$ through a call of the routine `solveACCNAR` (see Algorithm 3). Then, we compute the Lagrange multipliers $\boldsymbol{\nu}^{(t)}$ associated with the upper bound constraints. After that, we compute a candidate descent direction $\mathbf{d}^{(t)}$ by solving Problem 9. If $\mathbf{d}^{(t)}$ is a descent step, then we set $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} + \mathbf{d}^{(t)}$ and enlarge the radius of the trust region; otherwise, we do not move to a new point, and we tight the trust region and solve again Problem 9. The descent algorithm stops as soon as the radius of the trust region becomes smaller than a fixed tolerance ε_1.

Remark 5: Note that we initially set $\mathbf{y}^{(0)} = \mathbf{0}$. However, any feasible solution of Problem 9 does the job, and actually, starting with a good initial solution may enhance the performance of the algorithm.

Remark 6: Problem 9 is an LP and can be solved by any existing LP solver. However, a suboptimal solution to Problem 9, obtained by a heuristic approach, is also acceptable. Indeed, we observe that: 1) an *optimal* descent direction is not strictly required and 2) a heuristic approach allows to reduce the time needed to get a descent direction. In this article, we employed a possible heuristic, whose description can be found in [30], but the development of further heuristic approaches is a possible topic for future research.

V. Computational Experiments

In this section, we present various computational experiments performed in order to evaluate the approaches proposed in Sections III and IV.

In particular, we compared solutions of Problem 2 computed by algorithm SCA to solutions obtained with commercial NLP solvers. Note that, with a single exception, we did not carry out a direct comparison with other methods specifically tailored to Problem 2 for the following reasons.

1) Some algorithms (such as [22] and [23]) use heuristics to quickly find suboptimal solutions of acceptable quality but do not achieve local optimality. Hence, comparing their solution times with SCA would not be fair. However, in one of our experiments (see Experiment 4), we made a comparison between the most recent heuristic proposed in [23] and algorithm SCA, both in terms of computing times and in terms of the quality of the returned solution.
2) The method presented in [26] does not consider the (nonconvex) jerk constraint but solves a convex problem whose objective function has a penalization term that includes pseudojerk. Due to this difference, a direct comparison with SCA is not possible.
3) The method presented in [24] is based on the numerical solution of a large number of nonlinear and nonconvex subproblems and is, therefore, structurally slower than SCA, whose main iteration is based on the efficient solution of the convex Problem 3.

In the first two experiments, we compare the computational time of IPOPT, a general-purpose NLP solver [33], with that of algorithm SCA over some randomly generated instances of Problem 2. In particular, we tested two different versions of the algorithm SCA. The first version, called SCA-H in what follows, employs the heuristic mentioned in Remark 6. Since the heuristic procedure may fail in some cases, in such cases, we also need an LP solver. In particular, in our experiments, we used GUROBI whenever the heuristic did not produce either a feasible solution to Problem 9 or a descent direction. In the second version, called SCA-G in what follows, we always employed GUROBI to solve Problem 9. For what concerns the choice of the NLP solver IPOPT, we remark that we chose it after a comparison with two further general-purpose NLP solvers, SNOPT and MINOS, which, however, turned out to perform worse than IPOPT on this class of problems.

In the third experiment, we compare the performance of the two implemented versions of algorithm SCA applied to two specific paths and see their behavior as the number n of discretized points increases.

In the fourth experiment, we compare the solutions returned by algorithm SCA with those returned by the heuristic recently proposed in [23].

Finally, in the fifth experiment, we present a real-life speed planning task for an LGV operating in an industrial setting, using real problem bounds and paths layouts, provided by an automation company based in Parma, Italy.

We remark that, according to our experiments, the special purpose routine `solveACCNAR` (Algorithm 3) strongly outperforms general-purpose approaches, such as the graph-based approach proposed in [28], and GUROBI, when solving Problem 6 (which can be converted into an LP as discussed in [28]).

Finally, we remark that we also tried to solve the convex Problem 3 arising at each iteration of the proposed

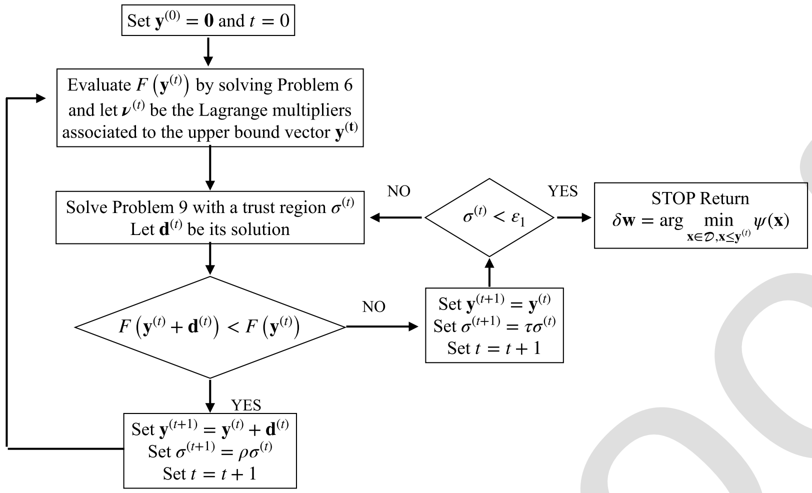

Fig. 3. Flowchart of the routine `ComputeUpdate`.

method with an NLP solver in place of the procedure `ComputeUpdate`, presented in this article. However, the experiments revealed that, in doing this, the computing times become much larger even with respect to the single call to the NLP solver for solving the nonconvex Problem 2.

All tests have been performed on an IntelCore i7-8550U CPU at 1.8 GHz. Both for IPOPT and algorithm SCA, the null vector was chosen as a starting point. The parameters used within algorithm SCA were $\varepsilon = 1e^{-8}$, $\varepsilon_1 = 1e^{-6}$ (tolerance parameters), $\rho = 4$, and $\tau = 0.25$ (trust-region update parameters). The initial trust region radius $\sigma^{(0)}$ was initialized to 1 in the first iteration $k = 0$ but adaptively set equal to the size of the last update $\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\|_\infty$ in all subsequent iterations (this adaptive choice allowed to reduce computing times by more than a half). We remark that algorithm SCA has been implemented in MATLAB, so we expect better performance after a C/C++ implementation.

A. Experiments 1 and 2

In Experiment 1, we generated a set of 50 different paths, each of which was discretized setting $n = 100$, $n = 500$, and $n = 1000$ sample points. The instances were generated by assuming that the traversed path was divided into seven intervals over which the curvature of the path was assumed to be constant. Thus, the n-dimensional upper bound vector $\mathbf{u}$ was generated as follows. First, we fixed $u_1 = u_n = 0$, i.e., the initial and final speeds must be equal to 0. Next, we partitioned the set $\{2, \ldots, n-1\}$ into seven subintervals I_j, $j \in \{1, \ldots, 7\}$, which corresponds to intervals with constant curvature. Then, for each subinterval, we randomly generated a value $u_j \in (0, \tilde{u}]$, where $\tilde{u}$ is the maximum upper bound (which was set equal to 100 $\mathrm{m^2s^{-2}}$). Finally, for each $j \in \{1, \ldots, 7\}$, we set $u_k = \tilde{u}_j \ \forall k \in I_j$. The maximum acceleration parameter A is set equal to 2.78 $\mathrm{ms^{-2}}$ and the maximum jerk J to 0.5 $\mathrm{ms^{-3}}$, while the path length is $s_f = 60$ m. The values for A and J allow a comfortable motion for a ground transportation vehicle (see [34]).

In Experiment 2, we generated a further set of 50 different paths, each of which was discretized using $n = 100$, $n = 500$, and $n = 1000$ variables. These new instances were randomly generated such that the traversed path was divided into up to five intervals over which the curvature could be zero, linear with respect to the arc length or constant. We chose this kind of path since they are able to represent the curvature of a road trip (see [35]). The maximum squared speed along the path was fixed equal to 192.93 $\mathrm{m^2s^{-2}}$ (corresponding to a maximum speed of 50 $\mathrm{kmh^{-1}}$, a typical value for an urban driving scenario). The total length of the paths was fixed to $s_f = 1000$ m, while parameter A was set equal to 0.25 $\mathrm{ms^{-2}}$, J to 0.025 $\mathrm{ms^{-3}}$, and A_N to 4.9 $\mathrm{ms^{-2}}$.

The results are reported in Table I, in which we show the average (minimum and maximum) computational times for SCA-H, SCA-G, and IPOPT. They show that algorithm SCA-H is the fastest one, while SCA-G is slightly faster than IPOPT at $n = 100$ but clearly faster for a larger number of sample points n. In general, we observe that both SCA-H and SCA-G tend to outperform IPOPT as n increases. Moreover, while the computing times for IPOPT at $n = 100$ are not much worse than those of SCA-H and SCA-G, we should point out that, at this dimension, IPOPT is sometimes unable to converge and return solutions whose objective function value differs from the best one by more than 100%. Also, the objective function values returned by SCA-H and SCA-G are sometimes slightly different, due to numerical issues related to the choice of the tolerance parameters, but such differences are mild ones and never exceed 1%. Therefore, these approaches appear to be fast and robust. It is also worthwhile to remark that SCA approaches are compatible with online planning requirements within the context of the LGV application. According to Haschke *et al.* [18] (see also [36]), in "highly unstructured, unpredictable, and dynamic environments," there is a need to replan in order to adapt the motion in reaction to unforeseen events or obstacles. How often to replan depends strictly on the application. Within the context of the LGV application (where the environment is structured), replanning every 100–150 ms

TABLE I

AVERAGE (MINIMUM AND MAXIMUM) COMPUTING TIMES (IN SECONDS) FOR SCA-H, SCA-G, AND IPOPT OVER EXPERIMENTS 1 AND 2

Exp.	n		SCA-H	SCA-G	IPOPT
1	100	min	0.012	0.042	0.03
		mean	0.016	0.072	0.132
		max	0.026	0.138	0.305
1	500	min	0.042	0.21	0.352
		mean	0.064	0.276	1.01
		max	0.104	0.456	1.828
1	1000	min	0.1	0.426	1.432
		mean	0.149	0.626	3.289
		max	0.237	0.828	7.137
2	100	min	0.012	0.036	0.052
		mean	0.02	0.047	0.113
		max	0.038	0.073	0.263
2	500	min	0.049	0.102	0.534
		mean	0.093	0.172	0.886
		max	0.212	0.237	1.457
2	1000	min	0.083	0.228	1.733
		mean	0.242	0.386	2.487
		max	0.709	0.539	3.74

is acceptable, and thus, the computing times of the SCA approaches at $n = 100$ are suitable. Of course, computing times increase with n, but we notice that the computing times of SCA-H still meet the requirement at $n = 500$. Moreover, a relevant feature of SCA-H and SCA-G is that, at each iteration, a feasible solution is available. Thus, we could stop them as soon as a time limit is reached. At $n = 500$, if we impose a time limit of 150 ms, which is still quite reasonable for the application, SCA-G returns slightly worse feasible solutions, but these do not differ from the best ones by more than 2%.

B. Experiment 3

In our third experiment, we compared the performance of the two proposed approaches (SCA-H and SCA-G), over two possible automated driving scenarios, as the number n of samples increases. As a first example, we considered a continuous curvature path composed of a line segment, a clothoid, a circle arc, a clothoid, and a final line segment (see Fig. 4). The minimum-time velocity planning on this path, whose total length is $s_f = 90$ m, is addressed with the following data. The problem constants are compatible with a typical urban driving scenario. The maximum squared velocity is 225 m^2s^{-2} (corresponding to 54 km h^{-1}), the longitudinal acceleration limit is $A = 1.5$ ms^{-2}, and the maximal normal acceleration is $A_N = 1$ ms^{-2}, while, for the jerk constraints, we set $J = 1$ ms^{-3}. Next, we considered a path of length $s_f = 60$ m (see Fig. 5) whose curvature was defined according to the following function:

$$k(s) = \frac{1}{5}\sin\left(\frac{s}{10}\right), \quad s \in [0, s_f]$$

and parameter A, A_N, and J were set equal to 1.39 ms^{-2}, 4.9 ms^{-2}, and 0.5 ms^{-3}, respectively. The maximum squared velocity is still equal to 225 m^2s^{-2}. The computational results are reported in Figs. 6 and 7 for values of n that grows from 100 to 1000. They show that the performance of SCA-H and SCA-G depends on the path. In particular, it seems that the heuristic performs in a poorer way when the number of points of the upper bound vector at which PAR constraints are violated tends to be large, which is the case for the second instance. We can give two possible motivations: 1) the directions computed by the heuristic procedure are not necessarily good descent directions, so routine `computeUpdate` slowly converges to a solution and 2) the heuristic procedure often fails, and it is in any case necessary to call GUROBI. Note that the computing times of IPOPT on these two paths are larger than those of SCA-H and SCA-G, and, as usual, the gap increases with n. Moreover, for the second path, IPOPT was unable to converge for $n = 100$ and returned a solution, which differed by more than 35% with respect to those returned by SCA-H and SCA-G.

As a final remark, we notice that the computed traveling times along the paths only slightly vary with n. For the first path, they vary between 14.44 and 14.45 s while, for the second path, between 20.65 and 20.66 s. The differences are very mild, but we should point out that this is not always

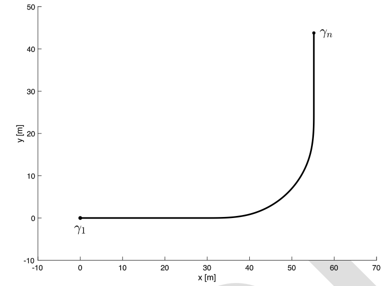

Fig. 4. Experiment 3—first path.

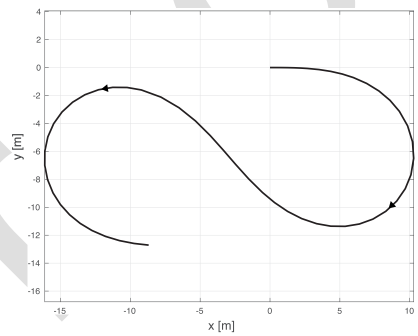

Fig. 5. Experiment 3—second path.

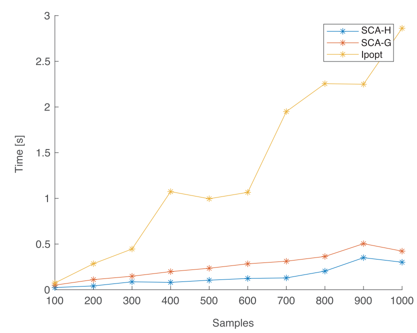

Fig. 6. Computing times (in seconds) for the path in Fig. 4.

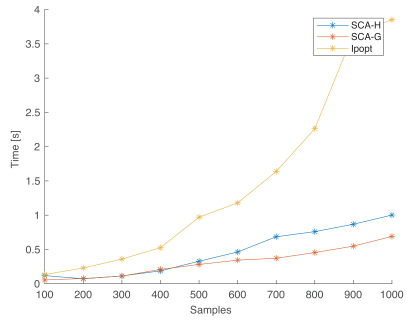

Fig. 7. Computing times (in seconds) for the path in Fig. 5.

TABLE II

MINIMUM, AVERAGE, AND MAXIMUM COMPUTING TIMES (IN SECONDS) AND RELATIVE PERCENTAGE DIFFERENCE BETWEEN THE TRAVELING TIMES COMPUTED BY THE HEURISTIC PRESENTED IN [23] AND THE SCA APPROACHES WITH $n = 100$ FOR THE INSTANCES OF EXPERIMENT 1

Heuristic from [23]	min	mean	max
Time	0.016	0.048	0.2049
Relative percentage difference	5.5%	12.1%	31.2%

the case. We further comment on this point when presenting Experiment 5.

C. Experiment 4

In this experiment, we compared the performance of our approach with the heuristic procedure recently proposed in [23]. In Table II, we report the computing times and the relative percentage difference $[(f_{\text{HEUR}} - f_{\text{SCA}})/f_{\text{SCA}}] * 100\%$ between the traveling times computed by the heuristic and the SCA approaches for the instances of Experiment 1 with $n = 100$. Algorithms SCA-H and SCA-G have comparable computing times (actually, better for what concerns SCA-H) with respect to that heuristic, and the quality of the final solutions is, on average, larger than 10% (these observations also extend to other experiments). Such difference between the quality of the solutions returned by algorithm SCA and those returned by the heuristic is best explained through the discussion of a representative instance, taken from Experiment 1 with $n = 100$. In this instance, we set $A = 2.78$ ms^{-2}, while, for the jerk constraints, we set $J = 2$ ms^{-3}. The total length of the path is $s_f = 60$ m. The maximum velocity profile is the piecewise constant black line in Fig. 8. In the same figure, we report in red the velocity profile returned by the heuristic and in blue the one returned by algorithm SCA. The computing time for the heuristic is 45 ms, while, for algorithm SCA-H, it is 39 ms. The final objective function value (i.e., the traveling time along the given path) is 15.4 s for the velocity profile returned by the heuristic and 14.02 s for the velocity profile returned by algorithm SCA. From the qualitative point of view, it can be observed in this instance (and similar observations hold for the other instances that we tested) that the heuristic produces velocity profiles whose local minima coincide with those of the maximum velocity profile. For instance, in the interval between 10 and 20 m, we notice that the velocity profile returned by the heuristic coincides

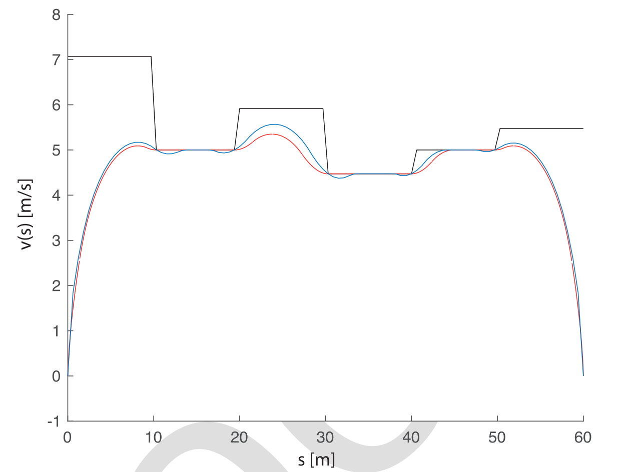

Fig. 8. Velocity profile returned by the heuristic proposed in [23] (red line) and by algorithm SCA (blue line). The black line is the maximum velocity profile.

with the maximum velocity profile in that interval. Instead, the velocity profile generated by algorithm SCA generates velocity profiles that fall below the local minima of the maximum velocity profile, but, this way, they are able to keep the velocity higher in the regions preceding and following the local minima of the maximum velocity profile. Again, referring to the interval between 10 and 20 m, we notice that the velocity profile computed by algorithm SCA falls below the maximum velocity profile in that region and, thus, below the velocities returned by the heuristic, but, this way, velocities in the region before 10 m and in the one after 20 m are larger with respect to those computed by the heuristic.

D. Experiment 5

As a final experiment, we planned the speed law of an autonomous guided vehicle operating in a real-life automated warehouse. Paths and problem data have been provided by packaging company Ocme S.r.l., based in Parma, Italy. We generated 50 random paths from a general layout. Fig. 9 shows the warehouse layout and a possible path. In all paths, we set maximum velocity to 2 m s^{-1}, maximum longitudinal acceleration to $A = 0.28$ m/s^2, maximum normal acceleration to 0.2 m/s^2, and maximum jerk to $J = 0.025$ m/s^3. Table III shows computation times for algorithms SCA-H, SCA-G, and IPOPT for a number of sampling points $n \in \{100, 500, 1000\}$. SCA-H is quite fast although it sometimes returns slightly worse solutions (the largest percentage error, at a single instance with $n = 1000$, is 8%). IPOPT is clearly slower than SCA-H and SCA-G for $n = 500$ and 1000, while, for $n = 100$, it is slower than SCA-H but quite similar to SCA-G. However, for these paths, the difference in terms of traveling times as n increases is much more significant with respect to the other experiments (see also the discussion at the end of Experiment 3). More precisely, the percentage difference between the traveling times of solutions at $n = 100$ and $n = 1000$ is 0.5% on average for Experiment 1 with a maximum of 2.1%, while, for Experiment 2, the average difference is 0.3% with a maximum of 0.4%. Instead, for the current experiment,

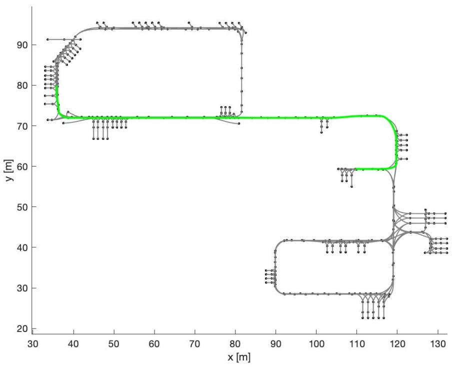

Fig. 9. Warehouse layout considered in Example 5 and a possible path.

TABLE III
AVERAGE, MINIMUM, AND MAXIMUM COMPUTING TIMES (IN SECONDS) FOR SCA-H, SCA-G, AND IPOPT OVER EXPERIMENT 5

n		SCA-H	SCA-G	IPOPT
100	min	0.009	0.033	0.029
	mean	0.013	0.043	0.037
	max	0.026	0.062	0.052
500	min	0.032	0.104	0.222
	mean	0.068	0.146	0.289
	max	0.174	0.224	0.423
1000	min	0.078	0.249	0.744
	mean	0.201	0.385	1.25
	max	0.501	0.65	3.359

the average difference is 2.7% with a maximum of 7.9%. However, the average falls to 0.2% and the maximum to 0.6% if we consider the percentage difference between the traveling times of solutions at $n = 500$ and $n = 1000$. Thus, for this experiment, it is advisable to use a finer discretization or, equivalently, a larger number of sampling points. A tentative explanation for such different behavior is related to the lower velocity limits of Experiment 5 with respect to the other experiments. Indeed, the objective function is much more sensitive to small changes at low speeds so that a finer grid of sampling points is able to reduce the impact of approximation errors. However, this is just a possible explanation. A further possible explanation is that, in Experiments 1–4, curves are composed of segments with constant and linear curvature, whereas curves on industrial LGV layouts typically have curvatures that are highly nonlinear with respect to arc length.

VI. CONCLUSION

In this article, we considered a speed planning problem under jerk constraints. The problem is a nonconvex one, and we proposed a sequential convex approach, where we exploited the special structure of the convex subproblems to solve them very efficiently. The approach is fast and is theoretically guaranteed to converge to a stationary point of the nonconvex problem. As a possible topic for future research, we would like to investigate ways to solve Problem 9, currently the bottleneck of the proposed approach, alternative to the solver GUROBI, and the heuristic mentioned in Remark 6. Moreover, we suspect that the stationary point to which the proposed approach converges is, in fact, a global minimizer of the nonconvex problem, and proving this fact is a further interesting topic for future research.

ACKNOWLEDGMENT

The authors are really grateful to the Associate Editor and the three anonymous reviewers for their careful reading and very useful suggestions.

REFERENCES

[1] P. Pharpatara, B. Hérissé, and Y. Bestaoui, “3-D trajectory planning of aerial vehicles using RRT*,” *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 1116–1123, May 2017.

[2] K. Kant and S. W. Zucker, “Toward efficient trajectory planning: The path-velocity decomposition,” *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, Sep. 1986.

[3] L. Consolini, M. Locatelli, A. Minari, and A. Piazzi, “An optimal complexity algorithm for minimum-time velocity planning,” *Syst. Control Lett.*, vol. 103, pp. 50–57, May 2017.

[4] F. Cabassi, L. Consolini, and M. Locatelli, “Time-optimal velocity planning by a bound-tightening technique,” *Comput. Optim. Appl.*, vol. 70, no. 1, pp. 61–90, May 2018.

[5] F. Pfeiffer and R. Johanni, “A concept for manipulator trajectory planning,” *IEEE J. Robot. Autom.*, vol. RA-3, no. 2, pp. 115–123, Apr. 1987.

[6] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, “Time-optimal path tracking for robots: A convex optimization approach,” *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.

[7] M. Yuan, Z. Chen, B. Yao, and X. Zhu, “Time optimal contouring control of industrial biaxial gantry: A highly efficient analytical solution of trajectory planning,” *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 1, pp. 247–257, Feb. 2017.

[8] E. Velenis and P. Tsiotras, “Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation,” *J. Optim. Theory Appl.*, vol. 138, no. 2, pp. 275–296, 2008.

[9] M. Frego, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, “Semi-analytical minimum time solutions with velocity constraints for trajectory following of vehicles,” *Automatica*, vol. 86, pp. 18–28, Dec. 2017.

[10] K. Hauser, “Fast interpolation and time-optimization with contact,” *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1231–1250, 2014.

[11] H. Pham and Q.-C. Pham, “A new approach to time-optimal path parameterization based on reachability analysis,” *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 645–659, Jun. 2018.

[12] L. Consolini, M. Locatelli, A. Minari, A. Nagy, and I. Vajk, “Optimal time-complexity speed planning for robot manipulators,” *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 790–797, Jun. 2019.

[13] T. Lipp and S. Boyd, “Minimum-time speed optimisation over a fixed path,” *Int. J. Control*, vol. 87, no. 6, pp. 1297–1311, 2014.

[14] F. Debrouwere *et al.*, “Time-optimal path following for robots with convex–concave constraints using sequential convex programming,” *IEEE Trans. Robot.*, vol. 29, no. 6, pp. 1485–1495, Dec. 2013.

[15] A. K. Singh and K. M. Krishna, “A class of non-linear time scaling functions for smooth time optimal control along specified paths,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 5809–5816.

[16] A. Palleschi, M. Garabini, D. Caporale, and L. Pallottino, “Time-optimal path tracking for jerk controlled robots,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3932–3939, Oct. 2019.

[17] S. Macfarlane and E. A. Croft, “Jerk-bounded manipulator trajectory planning: Design for real-time applications,” *IEEE Trans. Robotics Autom.*, vol. 19, no. 1, pp. 42–52, Feb. 2003.

[18] R. Haschke, E. Weitnauer, and H. Ritter, “On-line planning of time-optimal, jerk-limited trajectories,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 3248–3253.

[19] H. Pham and Q.-C. Pham, “On the structure of the time-optimal path parameterization problem with third-order constraints,” in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 679–686.

[20] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, “Time-optimal control of robotic manipulators along specified paths,” *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, Sep. 1985.

[21] K. G. Shin and N. D. McKay, “Minimum-time control of robotic manipulators with geometric path constraints,” *IEEE Trans. Autom. Control*, vol. AC-30, no. 6, pp. 531–541, Jun. 1985.

[22] J. Villagra, V. Milanés, J. Pérez, and J. Godoy, "Smooth path and speed planning for an automated public transport vehicle," *Robot. Auton. Syst.*, vol. 60, no. 2, pp. 252–265, 2012.

[23] M. Raineri and C. G. L. Bianco, "Jerk limited planner for real-time applications requiring variable velocity bounds," in *Proc. IEEE Int. Conf. Automat. Sci. Eng. (CASE)*, Aug. 2019, pp. 1611–1617.

[24] J. Dong, P. M. Ferreira, and J. A. Stori, "Feed-rate optimization with jerk constraints for generating minimum-time trajectories," *Int. J. Mach. Tools Manuf.*, vol. 47, nos. 12–13, pp. 1941–1955, Oct. 2007.

[25] K. Zhang, X. S. Gao, H. B. Li, and C. M. Yuan, "A greedy algorithm for feedrate planning of CNC machines along curved tool paths with confined jerk," *Robot. Comput.-Integr. Manuf.*, vol. 28, no. 4, pp. 472–483, Aug. 2012.

[26] Y. Zhang *et al.*, "Speed planning for autonomous driving via convex optimization," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 1089–1094.

[27] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Singapore: Wiley, 2000.

[28] L. Consolini, M. Laurini, and M. Locatelli, "Graph-based algorithms for the efficient solution of optimization problems involving monotone functions," *Comput. Optim. Appl.*, vol. 73, no. 1, pp. 101–128, May 2019.

[29] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, vol. 80. Philadelphia, PA, USA: SIAM, 2002.

[30] L. Consolini, M. Locatelli, and A. Minari, "A sequential approach for speed planning under jerk constraints," 2021, *arXiv:2105.15095*. [Online]. Available: http://arxiv.org/abs/2105.15095

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[32] Gurobi Optimization Inc. (2016). *Gurobi Optimizer Reference Manual*. [Online]. Available: http://www.gurobi.com

[33] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, May 2006.

[34] L. L. Hoberock, "A survey of longitudinal acceleration comfort studies in ground transportation vehicles," *J. Dyn. Syst., Meas., Control*, vol. 99, no. 2, pp. 76–84, Jun. 1977.

[35] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 1025–1035, Dec. 2004.

[36] D. Verscheure, M. Diehl, J. De Schutter, and J. Swevers, "Recursive log-barrier method for on-line time-optimal robot path tracking," in *Proc. Amer. Control Conf.*, St. Louis, MO, USA, Apr. 2009, pp. 4134–4140.

Luca Consolini (Member, IEEE) received the Laurea degree *(cum laude)* in electronic engineering and the Ph.D. degree from the University of Parma, Parma, Italy, in 2000 and 2005, respectively.

From 2005 to 2009, he held a post-doctoral position at the University of Parma, where he was an Assistant Professor from 2009 to 2014. Since 2014, he has been an Associate Professor with the University of Parma. His main current research interests are nonlinear control, motion planning, and control of mechanical systems.

Marco Locatelli is currently a Full Professor of Operations Research with the University of Parma, Parma, Italy. He published more than 90 articles in international journals and coauthored, with F. Schoen, the book *Global Optimization: Theory, Algorithms, and Applications* [Society for Industrial and Applied Mathematics (SIAM)]. His main research interests are the theoretical, practical, and applicative aspects of optimization.

Dr. Locatelli has been nominated as a EUROPT Fellow in 2018. He is also on the Editorial Board of the journals *Computational Optimization and Applications, Journal of Global Optimization*, and *Operations Research Forum*.

Andrea Minari received the B.S. and M.S. degrees *(cum laude)* in computer engineering and the Ph.D. degree from the University of Parma, Parma, Italy, in 2013, 2016, and 2020, respectively. He presented a dissertation about optimization-based algorithms applied to the speed planning problem for mobile robots and industrial manipulators.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your article. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

Carefully check the page proofs (and coordinate with all authors); additional changes or updates WILL NOT be accepted after the article is published online/print in its final form. Please check author names and affiliations, funding, as well as the overall article for any errors prior to sending in your author proof corrections.

AQ:1 = Please confirm or add details for any funding or financial support for the research of this article.

AQ:2 = Please confirm the postal code for Università di Parma.

AQ:3 = Please specify the section numbers for the phrase “next sections.”

A Sequential Algorithm for Jerk Limited Speed Planning

Luca Consolini, *Member, IEEE*, Marco Locatelli, and Andrea Minari

***Abstract*—In this article, we discuss a sequential algorithm for the computation of a minimum-time speed profile over a given path, under velocity, acceleration, and jerk constraints. Such a problem arises in industrial contexts, such as automated warehouses, where LGVs need to perform assigned tasks as fast as possible in order to increase productivity. It can be reformulated as an optimization problem with a convex objective function, linear velocity and acceleration constraints, and nonconvex jerk constraints, which, thus, represents the main source of the difficulty. While existing nonlinear programming (NLP) solvers can be employed for the solution of this problem, it turns out that the performance and robustness of such solvers can be enhanced by the sequential line-search algorithm proposed in this article. At each iteration, a feasible direction, with respect to the current feasible solution, is computed, and a step along such direction is taken in order to compute the next iterate. The computation of the feasible direction is based on the solution of a linearized version of the problem, and the solution of the linearized problem, through an approach that strongly exploits its special structure, represents the main contribution of this work. The efficiency of the proposed approach with respect to existing NLP solvers is proven through different computational experiments.**

***Note to Practitioners*—This article was motivated by the needs of LGV manufacturers. In particular, it presents an algorithm for computing the minimum-time speed law for an LGV along a preassigned path, respecting assigned velocity, acceleration, and jerk constraints. The solution algorithm should be: 1) *fast*, since speed planning is made continuously throughout the workday, not only when an LGV receives a new task but also during the execution of the task itself, since conditions may change, e.g., if the LGV has to be halted for security reasons and 2) *reliable*, i.e., it should return solutions of high quality, because a better speed profile allows to save time and even small percentage improvements, say a 5% improvement, has a considerable impact on the productivity of the warehouse, and, thus, determines a significant economic gain. The algorithm that we propose meets these two requirements, and we believe that it can be a useful tool for LGV manufacturers and users. It is obvious that the proposed method also applies to the speed planning problem for vehicles other than LGVs, e.g., road vehicles.**

AQ:1 Manuscript received June 29, 2021; accepted August 24, 2021. This article was recommended for publication by Associate Editor M. Robba and Editor C. Seatzu upon evaluation of the reviewers' comments. This work was supported in part by the Programme "FIL-Quota Incentivante" of the University of Parma and in part by the Fondazione Cariparma. *(Corresponding author: Marco Locatelli.)*

AQ:2 The authors are with the Dipartimento di Ingegneria e Architettura, Università di Parma, 43121 Parma, Italy (e-mail: luca.consolini@unipr.it; marco.locatelli@unipr.it; andrea.minari2@studenti.unipr.it).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TASE.2021.3111758.

Digital Object Identifier 10.1109/TASE.2021.3111758

***Index Terms*—Optimization, sequential line-search method, speed planning.**

I. Introduction

An IMPORTANT problem in motion planning is the computation of the minimum-time motion of a car-like vehicle from a start configuration to a target one while avoiding collisions (obstacle avoidance) and satisfying kinematic, dynamic, and mechanical constraints (for instance, on velocities, accelerations, and maximal steering angle). This problem can be approached in two ways.

1) As a minimum-time trajectory planning, where both the path to be followed by the vehicle and the timing law on this path (i.e., the vehicle's velocity) are simultaneously designed. For instance, one could use the RRT* algorithm (see [1]).
2) As a (geometric) path planning followed by a minimum-time speed planning on the planned path (see [2]).

In this article, following the second paradigm, 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. The problem can be reformulated as an optimization problem, and it is quite relevant from the practical point of view. In particular, in automated warehouses, the speed of LGVs needs to be planned under acceleration and jerk constraints. As previously mentioned, the solution algorithm should be both *fast* and *reliable*. In our previous work [3], we proposed an optimal time-complexity algorithm for finding the time-optimal speed law that satisfies constraints on maximum velocity and tangential and normal acceleration. In the subsequent work [4], we included a bound on the derivative of the acceleration with respect to the arc length. In this article, we consider the presence of jerk constraints (constraints on the time derivative of the acceleration). The resulting optimization problem is nonconvex and, for this reason, is significantly more complex than the ones that we discussed in [3] and [4]. The main contribution of this work is the development of a line-search algorithm for this problem based on the sequential solution of convex problems. The proposed algorithm meets both requirements of being fast and reliable. The former is met by heavily exploiting the special structure of the optimization problem, the latter by the theoretical guarantee that the returned solution is a first-order stationary point (in practice, a local minimizer) of the optimization problem.

A. Problem Statement

Here, we introduce the problem at hand more formally. Let $\boldsymbol{\gamma}:[0, s_f] \to \mathbb{R}^2$ be a smooth function. The image set $\boldsymbol{\gamma}([0, s_f])$ is the path to be followed, $\boldsymbol{\gamma}(0)$ the initial configuration, and $\boldsymbol{\gamma}(s_f)$ the final one. Function $\boldsymbol{\gamma}$ has arc-length parameterization, such that $(\forall \lambda \in [0, s_f])$, $\|\boldsymbol{\gamma}'(\lambda)\| = 1$. In this way, s_f is the path length. We want to compute the speed-law that minimizes the overall transfer time (i.e., the time needed to go from $\boldsymbol{\gamma}(0)$ to $\boldsymbol{\gamma}(s_f)$). To this end, let $\lambda:[0, t_f] \to [0, s_f]$ be a differentiable monotone increasing function that represents the vehicle's arc-length position along the curve 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 derivative of the vehicle arc-length position, which corresponds to the norm of its velocity vector at position s. The position of the vehicle as a function of time is given by $\mathbf{x}:[0, t_f] \to \mathbb{R}^2$, $\mathbf{x}(t) = \boldsymbol{\gamma}(\lambda(t))$. The velocity and acceleration are given, respectively, by

$$\dot{\mathbf{x}}(t) = \boldsymbol{\gamma}'(\lambda(t))v(\lambda(t))$$
$$\ddot{\mathbf{x}}(t) = a_T(t)\boldsymbol{\gamma}'(\lambda(t)) + a_N(t)\boldsymbol{\gamma}'^{\perp}(\lambda(t))$$

where $a_T(t) = v'(\lambda(t))v(\lambda(t))$ and $a_N(t) = k(\lambda(t))v(\lambda(t))^2$ are, respectively, the tangential and normal components of the acceleration (i.e., the projections of the acceleration vector $\ddot{\mathbf{x}}$ on the tangent and the normal to the curve). Moreover $\boldsymbol{\gamma}'^{\perp}(\lambda)$ is the normal to vector $\boldsymbol{\gamma}'(\lambda)$ and the tangent of $\boldsymbol{\gamma}'$ at λ. Here, $k:[0, s_f] \to \mathbb{R}$ is the scalar curvature, defined as $k(s) = <\boldsymbol{\gamma}''(s), \boldsymbol{\gamma}'(s)^{\perp}>$. Note that $|k(s)| = \|\boldsymbol{\gamma}''(s)\|$. In the following, we assume that $k(s) \in \mathcal{C}^1([0, s_f], \mathbb{R})$. The total maneuver time, for a given velocity profile $v \in C^1([0, s_f], \mathbb{R})$, is returned by the functional

$$\mathcal{F} : C^1\big([0, s_f], \mathbb{R}\big) \to \mathbb{R}, \quad \mathcal{F}(v) = \int_0^{s_f} v^{-1}(s)ds. \tag{1}$$

In our previous work [3], we considered the problem

$$\min_{v \in \mathcal{V}} \mathcal{F}(v) \tag{2}$$

where the feasible region $\mathcal{V} \subset C^1([0, s_f], \mathbb{R})$ is defined by the following set of constraints:

$$v(0) = 0, \quad v(s_f) = 0 \tag{3a}$$
$$0 \le v(s) \le v_{\max}, \quad s \in [0, s_f] \tag{3b}$$
$$|v'(s)v(s)| \le A, \quad s \in [0, s_f] \tag{3c}$$
$$|k(s)|v(s)^2 \le A_N, \quad s \in [0, s_f] \tag{3d}$$

where $v_{\max}$, A, and A_N are upper bounds for the velocity, the tangential acceleration, and the normal acceleration, respectively. Constraints (3a) are the initial and final interpolation conditions, while constraints (3b)–(3d) limit velocity and the tangential and normal components of acceleration. In [3], we presented an algorithm, with linear-time computational complexity with respect to the number of variables, which provides an optimal solution of (2) after spatial discretization. One limitation of this algorithm is that the obtained velocity profile is Lipschitz[1] but not differentiable so that the vehicle's acceleration is discontinuous. With the aim of obtaining a smoother velocity profile, in the subsequent work [4], we required that the velocity be differentiable, and we imposed a Lipschitz condition (with constant J) on its derivative. In this way, after setting $w = v^2$, the feasible region of the problem $\mathcal{W} \subset C^1([0, s_f], \mathbb{R})$ is defined by the set of functions $w \in C^1([0, s_f], \mathbb{R})$ that satisfy the following set of constraints:

$$w(0) = 0, \quad w(s_f) = 0 \tag{4a}$$
$$0 \le w(s) \le v_{\max}^2, \quad s \in [0, s_f] \tag{4b}$$
$$\frac{1}{2}|w'(s)| \le A, \quad s \in [0, s_f] \tag{4c}$$
$$|k(s)|w(s) \le A_N, \quad s \in [0, s_f] \tag{4d}$$
$$|w'(s_1) - w'(s_2)| \le J|s_1 - s_2|, \quad s_1, s_2 \in [0, s_f]. \tag{4e}$$

Then, we end up with the problem

$$\min_{w \in \mathcal{W}} G(w) \tag{5}$$

where the objective function is

$$G : C^1\big([0, s_f], \mathbb{R}\big) \to \mathbb{R}, \quad G(w) = \int_0^{s_f} w^{-1/2}(s)ds. \tag{6}$$

The objective function (6) and constraints (4a)–(4d) correspond to the ones in problem (2) after the substitution $w = v^2$. Note that this change of variable is well known in the literature. It has been first proposed in [5], while, in [6], it is observed that Problem (2) becomes convex after this change of variable. The added set of constraints (4e) is a Lipschitz condition on the derivative of the squared velocity w. It is used to enforce a smoother velocity profile by bounding the second derivative of the squared velocity with respect to arc length. Note that constraints (4) are linear, and the objective function (6) is convex. In [4], we proposed an algorithm for solving a finite-dimensional approximation of Problem (4). The algorithm exploited the particular structure of the resulting convex finite-dimensional problem. This article extends the results of [4]. It considers a nonconvex variation of Problem (4), in which constraint (4e) is substituted with a constraint on the time derivative of the acceleration $|\dot{a}(t)| \le J$, where $a(t) = (d/dt)v(\lambda(t)) = v'(\lambda(t))v(\lambda(t)) = (1/2)w'(\lambda(t))$. Then, we set

$$j_L(t) = \dot{a}(t) = \frac{1}{2}w''(s(t))\sqrt{(w(s(t)))}.$$

This quantity is commonly called "jerk." Bounding the absolute value of jerk allows to avoid sudden changes of acceleration and obtain a smoother motion. Then, we end up with the following minimum-time problem.

Problem 1 (Smooth Minimum-Time Velocity Planning Problem: Continuous Version):

$$\min_{w \in C^2} \int_0^{s_f} w(s)^{-1/2}\, ds$$
$$w(0) = 0, \quad w(s_f) = 0$$
$$0 \le w(s) \le \mu^+(s), \quad s \in [0, s_f]$$
$$\frac{1}{2}|w'(s)| \le A, \quad s \in [0, s_f] \tag{7}$$
$$\frac{1}{2}|w''(s)\sqrt{w(s)}| \le J \quad s \in [0, s_f] \tag{8}$$

[1] A function $f:\mathbb{R} \to \mathbb{R}$ is *Lipschitz* if there exists a real positive constant L such that $(\forall x, y \in \mathbb{R})$ $|f(x) - f(y)| \le L|x - y|$.

where μ^+ is the square velocity upper bound depending on the shape of the path, i.e.,

$$\mu^+(s) = \min\left\{v_{\max}^2, \frac{A_N}{|k(s)|}\right\}$$

where $v_{\max}$, A_N, and k are the maximum vehicle velocity, the maximum normal acceleration, and the path curvature, respectively. Parameters A and J are the bounds representing the limitations on the (tangential) acceleration and the jerk, respectively. For the sake of simplicity, we consider constraints (7) and (8) symmetric and constant. However, the following development could be easily extended to the nonsymmetric and nonconstant case. Note that the jerk constraint (8) is nonconvex. The continuous problem is discretized as follows. We subdivide the path into $n-1$ intervals of equal length, i.e., we evaluate function w at points $s_i = ((i-1)s_f)/(n-1)$, $i = 1, \dots, n$, so that we have the following n-dimensional vector of variables:

$$\mathbf{w} = (w_1, w_2, \dots, w_n) = (w(s_1), w(s_2), \dots, w(s_n)).$$

Then, the finite dimensional version of the problem is given as follows.

Problem 2 (Smooth Minimum-Time Velocity Planning Problem: Discretized Version):

$$\min_{\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1}} + \sqrt{w_i}} \tag{9}$$

$$0 \le \mathbf{w} \le \mathbf{u} \tag{10}$$

$$w_{i+1} - w_i \le 2hA, \quad i = 1, \dots, n-1 \tag{11}$$

$$w_i - w_{i+1} \le 2hA, \quad i = 1, \dots, n-1 \tag{12}$$

$$(w_{i-1} - 2w_i + w_{i+1})\sqrt{\frac{\ell_i(\mathbf{w})}{4}} \le 2h^2 J$$
$$i = 2, \dots, n-1 \tag{13}$$

$$-(w_{i-1} - 2w_i + w_{i+1})\sqrt{\frac{\ell_i(\mathbf{w})}{4}} \le 2h^2 J$$
$$i = 2, \dots, n-1 \tag{14}$$

where

$$\ell_i(\mathbf{w}) = w_{i+1} + w_{i-1} + 2w_i \tag{15}$$

while $u_i = \mu^+(s_i)$, for $i = 1, \dots, n$, and, in particular, $u_1 = 0$ and $u_n = 0$ since we are assuming that the initial and final velocities are equal to 0. The objective function (9) is an approximation of (6) given by the Riemann sum of the intervals obtained by dividing each interval $[s_i, s_{i+1}]$, for $i = 1, \dots, n-1$, in two subintervals of the same size. Constraints (11) and (12) are obtained by a finite difference approximation of w'. Constraints (13) and (14) are obtained by using a second-order central finite difference to approximate w'', while w is approximated by a weighted arithmetic mean of three consecutive samples. Due to jerk constraints (13) and (14), Problem 2 is nonconvex and cannot be solved with the algorithm presented in [4].

B. Main Result

The main contribution of this article is the development of a new solution algorithm for finding a local minimum of the nonconvex Problem 2. As detailed in next sections, we propose to solve Problem 2 by a line-search algorithm based on the sequential solution of convex problems. The algorithm is an iterative one where the following operations are performed at each iteration. AQ:3

1) Constraint Linearization: We first define a convex problem by linearizing constraints (13) and (14) through a first-order Taylor approximation around the current point $\mathbf{w}^{(k)}$. Different from other sequential algorithms for nonlinear programming (NLP) problems, we keep the original convex objective function. The linearized problem is introduced in Section II.

2) Computation of a Feasible Descent Direction: The convex problem (actually, a relaxation of such problem) is solved in order to compute a feasible descent direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$. The main contribution of this article lies in this part. The computation requires the minimization of a suitably defined objective function through a further iterative algorithm. At each iteration of this algorithm, the following operations are performed:

C. Objective Function Evaluation

Such evaluation requires the solution of a problem with the same objective function but subject to a subset of the constraints. The special structure of the resulting subproblem is heavily exploited in order to solve it efficiently. This is the topic of Section III.

D. Computation of a Descent Step

Some Lagrange multipliers of the subproblem define a subgradient for the objective function. This can be employed to define a linear programming (LP) problem that returns a descent step for the objective function. This is the topic of Section IV.

Line Search: Finally, a standard line search along the half-line $\mathbf{w}^{(k)} + \alpha\boldsymbol{\delta}\mathbf{w}^{(k)}$, $\alpha \ge 0$, is performed.

Sections II–IV detail all what we discussed above. Next, in Section V, we present different computational experiments.

E. Comparison With Existing Literature

Although many works consider the problem of minimum-time speed planning with acceleration constraints (see [7]–[9]), relatively few consider jerk constraints. Perhaps, this is also due to the fact that the jerk constraint is nonconvex so that its presence significantly increases the complexity of the optimization task. One can use a general-purpose NLP solver (such as SNOPT or IPOPT) for finding a local solution of Problem 2, but the required time is, in general, too large for the speed planning application. As outlined in Section I-D, in this work, we tackle this problem through an approach based on the solution of a sequence of convex subproblems. There are different approaches in the literature based on the sequential solution of convex subproblems. In [10], it is first observed that the problem with acceleration constraints but no jerk constraints for robotic manipulators can be reformulated as a convex one with linear constraints, and it is solved by a sequence of LP problems obtained by linearizing the objective function at the current point, i.e., the objective function is replaced by its supporting hyperplane at the current point, and by introducing a trust region centered at the current point. In [11] and [12], it is further observed that this problem can be solved very efficiently through the solution of a sequence of 2-D LP problems. In [13], an interior point barrier method is used to solve the same problem based on Newton's method. Each Newton step requires the solution of a KKT system, and an efficient way to solve such systems is proposed in that work. Moving to approaches also dealing with jerk constraints, we mention [14]. In this work, it is observed that jerk constraints are nonconvex but can be written as the difference between two convex functions. Based on this observation, the authors solve the problem by a sequence of convex subproblems obtained by linearizing at the current point the concave part of the jerk constraints and by adding a proximal term in the objective function that plays the same role as a trust region, preventing from taking too large steps. In [15] a slightly different objective function is considered. Rather than minimizing the traveling time along the given path, the integral of the squared difference between the maximum velocity profile and the computed velocity profile is minimized. After representing time-varying control inputs as products of parametric exponential and polynomial functions, the authors reformulate the problem in such a way that its objective function is convex quadratic, while nonconvexity lies in difference-of-convex functions. The resulting problem is tackled through the solution of a sequence of convex subproblems obtained by linearizing the concave part of the nonconvex constraints. In [16], the problem of speed planning for robotic manipulators with jerk constraints is reformulated in such a way that nonconvexity lies in simple bilinear terms. Such bilinear terms are replaced by the corresponding convex and concave envelopes, obtaining the so-called McCormick relaxation, which is the tightest possible convex relaxation of the nonconvex problem. Other approaches dealing with jerk constraints do not rely on the solution of convex subproblems. For instance, in [17], a concatenation of fifth-order polynomials is employed to provide smooth trajectories, which results in quadratic jerk profiles, while, in [18], cubic polynomials are employed, resulting in piecewise constant jerk profiles. The decision process involves the choice of the phase durations, i.e., of the intervals over which a given polynomial applies. A very recent and interesting approach to the problem with jerk constraints is [19]. In this work, an approach based on numerical integration is discussed. Numerical integration has been first applied under acceleration constraints in [20] and [21]. In [19], jerk constraints are taken into account. The algorithm detects a position s along the trajectory where the jerk constraint is singular, that is, the jerk term disappears from one of the constraints. Then, it computes the speed profile up to s by computing two maximum jerk profiles and then connecting them by a minimum jerk profile, found by a shooting method. In general, the overall solution is composed of a sequence of various maximum and minimum jerk profiles. This approach does not guarantee reaching a local minimum of the traversal time. Moreover, since Problem 4 has velocity and acceleration constraints, the jerk constraint is singular for all values of s so that the algorithm presented in [19] cannot be directly applied to Problem 4.

Some algorithms use heuristics to quickly find sub-optimal solutions of acceptable quality. For instance, Villagra *et al.* [22] propose an algorithm that applies to curves composed of clothoids, circles, and straight lines. The algorithm does not guarantee the local optimality of the solution. Raineri and Guarino Lo Bianco [23] present an efficient heuristic algorithm. Also, this method does not guarantee global nor local optimality. Various works in the literature consider jerk bounds in the speed optimization problem for robotic manipulators instead of mobile vehicles. This is a slightly different problem but mathematically equivalent to Problem (1). In particular, paper [24] presents a method based on the solution of a large number of nonlinear and nonconvex subproblems. The resulting algorithm is slow due to a large number of subproblems; moreover, the authors do not prove its convergence. Zhang *et al.* [25] propose a similar method that gives a continuous-time solution. Again, the method is computationally slow since it is based on the numerical solution of a large number of differential equations; moreover, this article does not contain proof of convergence or local optimality. Some other works replace the jerk constraint with *pseudo-jerk*, that is, the derivative of the acceleration with respect to arc length, obtaining a constraint analogous to (4e) and ending up with a convex optimization problem. For instance, Zhang *et al.* [26] add to the objective function a pseudo-jerk penalizing term. This approach is computationally convenient, but substituting (8) with (4e) may be overly restrictive at low speeds.

F. Statement of Contribution

The method presented in this article is a sequential convex one that aims at finding a local optimizer of Problem 2. To be more precise, as usual with nonconvex problems, only convergence to a stationary point can usually be proved. However, the fact that the sequence of generated feasible points is decreasing with respect to the objective function values usually guarantees that the stationary point is a local minimizer, except in rather pathological cases (see [27, p. 19]). Moreover, in our experiments, even after running a local solver from different starting points, we have never been able to detect local minimizers better than the one returned by the method we propose. Thus, a possible, nontrivial, topic for future research could be that of proving the global optimality of the solution. To the best of our knowledge and as detailed in the following, this algorithm is more efficient than the ones existing in the literature since it leverages the special structure of the subproblems obtained as local approximations of Problem 2. We discussed this class of problems in our previous work [28]. This structure allows computing very efficiently a feasible descent direction for the main line-search algorithm; it is one of the key elements that allow us to outperform generic NLP solvers. In summary, the main contributions of this work are: 1) on the theoretical side, the development of an approach for which a rigorous mathematical analysis has been

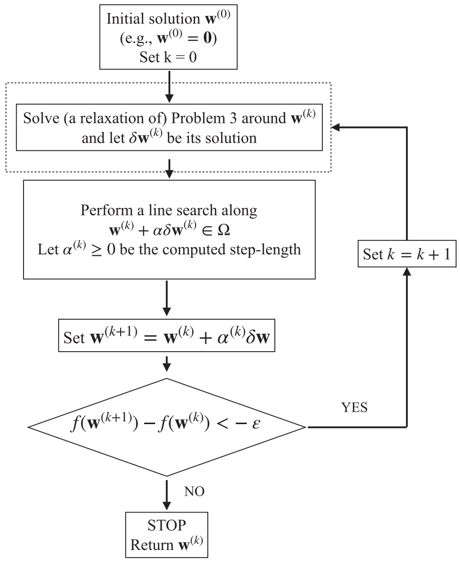

Fig. 1. Flowchart of algorithm SCA. The dashed block corresponds to a call of the procedure `ComputeUpdate`, proposed to solve Problem 3, which represents the main contribution of this article.

performed, proving a convergence result to a stationary point (see Section II) and 2) on the computational side, to exploit heavily the structure of the problem in order to implement the approach in a fairly efficient way (see Sections III and IV) so that its computing times outperform those of nonlinear solvers and are competitive with heuristic approaches that are only able to return suboptimal solutions of lower quality (see Section V).

II. Sequential Algorithm Based on Constraint Linearization

To account for the nonconvexity of Problem 2, we propose a line-search method based on the solution of a sequence of special structured convex problems. Throughout this article, we call this algorithm Sequential Convex Algorithm (SCA), and its flowchart is shown in Fig. 1. It belongs to the class of Sequential Convex Programming algorithms, where, at each iteration, a convex subproblem is solved. In what follows, we denote by $\boldsymbol{\Omega}$ the feasible region of Problem 2. At each iteration k, we replace the current point $\mathbf{w}^{(k)} \in \boldsymbol{\Omega}$ with a new point $\mathbf{w}^{(k)} + \alpha^{(k)}\boldsymbol{\delta}\mathbf{w}^{(k)} \in \boldsymbol{\Omega}$, where the step size $\alpha^{(k)} \in [0, 1]$ is obtained by a *line search* along the descent direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$, which, in turn, is obtained through the solution of a convex problem. The constraints of the convex problem are linear approximations of (10)–(14) around $\mathbf{w}^{(k)}$, while the objective function is the original one. Then, the problem that we consider to compute the direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ is given in the following (superscript k of $\mathbf{w}^{(k)}$ is omitted):

Problem 3:

$$\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1}+\delta w_{i+1}}+\sqrt{w_i+\delta w_i}} \tag{16}$$

$$\mathbf{l_B} \leq \boldsymbol{\delta}\mathbf{w} \leq \mathbf{u_B} \tag{17}$$

$$\delta w_{i+1} - \delta w_i \leq b_{A_i}, \quad i = 1, \dots, n-1 \tag{18}$$

$$\delta w_i - \delta w_{i+1} \leq b_{Di}, \quad i = 1, \dots, n-1 \tag{19}$$

$$\delta w_i - \eta_i \delta w_{i-1} - \eta_i \delta w_{i+1} \leq b_{N_i}, \quad i = 2, \dots, n-1 \tag{20}$$

$$\eta_i \delta w_{i-1} + \eta_i \delta w_{i+1} - \delta w_i \leq b_{P_i}, \quad i = 2, \dots, n-1 \tag{21}$$

where $\mathbf{l_B} = -\mathbf{w}$ and $\mathbf{u_B} = \mathbf{u} - \mathbf{w}$ (recall that $\mathbf{u}$ has been introduced in (10), and its components have been defined immediately in Problem 2), while parameters $\boldsymbol{\eta}$, $\mathbf{b_A}$, $\mathbf{b_D}$, $\mathbf{b_N}$, and $\mathbf{b_P}$ depend on the point $\mathbf{w}$ around which the constraints (10)–(14) are linearized. More precisely, we have

$$\begin{aligned} b_{A_i} &= 2hA - w_{i+1} + w_i \\ b_{Di} &= 2hA - w_i + w_{i+1} \\ \eta_i &= \frac{3w_{i+1} + 3w_{i-1} + 2w_i}{2(w_{i+1} + w_{i-1} + 6w_i)} \\ b_{P_i} &= \sqrt{\ell_i(\mathbf{w})}\frac{8h^2 J + (w_{i-1} - 2w_i + w_{i+1})\sqrt{\ell_i(\mathbf{w})}}{2(w_{i+1} + w_{i-1} + 6w_i)} \\ b_{N_i} &= \sqrt{\ell_i(\mathbf{w})}\frac{8h^2 J - (w_{i-1} - 2w_i + w_{i+1})\sqrt{\ell_i(\mathbf{w})}}{2(w_{i+1} + w_{i-1} + 6w_i)} \end{aligned} \tag{22}$$

where ℓ_i is defined in (15). The following proposition is an immediate consequence of the feasibility of $\mathbf{w}$.

Proposition 1: All parameters $\boldsymbol{\eta}$, $\mathbf{b_A}$, $\mathbf{b_D}$, $\mathbf{b_N}$, and $\mathbf{b_P}$ are nonnegative.

The proposed approach follows some standard ideas of sequential quadratic approaches employed in the literature about nonlinearly constrained problems. However, a quite relevant difference is that the true objective function (9) is employed in the problem to compute the direction, rather than a quadratic approximation of such function. This choice comes from the fact that the objective function (9) has some features (in particular, convexity and being decreasing), which, combined with the structure of the linearized constraints, allows for an efficient solution of Problem 3. Problem 3 is a convex problem with a nonempty feasible region ($\boldsymbol{\delta}\mathbf{w} = \mathbf{0}$ is always a feasible solution) and, consequently, can be solved by existing NLP solvers. However, such solvers tend to increase computing times since they need to be called many times within the iterative algorithm SCA. The main contribution of this article lies in the routine `computeUpdate` (see dashed block in Fig. 1), which is able to solve Problem 3 and efficiently returns a descent direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$. To be more precise, we solve a *relaxation* of Problem 3. Such relaxation, as well as the routine to solve it, is detailed in Sections III and IV. In Section III, we present efficient approaches to solve some subproblems, including proper subsets of the constraints. Then, in Section IV, we address the solution of the relaxation of Problem 3.

Remark 1: It is possible to see that, if one of the constraints (13) and (14) is active at $\mathbf{w}^{(k)}$, then, along the direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ computed through the solution of the linearized Problem 3, it holds that $\mathbf{w}^{(k)} + \alpha\boldsymbol{\delta}\mathbf{w}^{(k)} \in \boldsymbol{\Omega}$ for any sufficiently small $\alpha > 0$. In other words, small perturbations of the current solution $\mathbf{w}^{(k)}$ along direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ do not lead outside the feasible region $\boldsymbol{\Omega}$. This fact is illustrated in Fig. 2. Let us

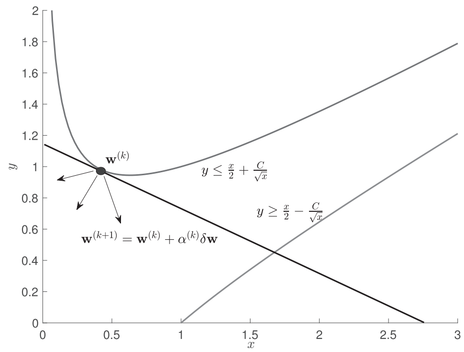

Fig. 2. Constraints (13) and (14) and their linearization ($C = 4h^2 J$).

rewrite constraints (13) and (14) as follows:

$$\left|(x - 2y)\sqrt{x}\right| \leq C \tag{23}$$

where $x = \ell_i(\mathbf{w})$, $y = 2w_i$, and $C = 4h^2 J$ is a constant. The feasible region associated with constraint (23) is reported in Fig. 2. In particular, it is the region between the blue and red curves. Suppose that constraint $y \leq (x/2)+(C/2\sqrt{x})$ is active at $\mathbf{w}^{(k)}$ (the case when $y \geq (x/2) - (C/2\sqrt{x})$ is active can be dealt with in a completely analogous way). If we linearize such constraint around $\mathbf{w}^{(k)}$, then we obtain a linear constraint (black line in Fig. 2), which defines a region completely contained into the one defined by the nonlinear constraint $y \leq (x/2)+(C/2\sqrt{x})$. Hence, for each direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ feasible with respect to the linearized constraint, we are always able to perform sufficiently small steps, without violating the original nonlinear constraints, i.e., for $\alpha > 0$ small enough, it holds that $\mathbf{w}^{(k)} + \alpha\boldsymbol{\delta}\mathbf{w}^{(k)} \in \boldsymbol{\Omega}$.

Constraints (13) and (14) can also be rewritten as follows:

$$w_{i-1} + w_{i+1} - 2w_i - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}} \leq 0 \tag{24}$$
$$2w_i - w_{i-1} - w_{i+1} - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}} \leq 0. \tag{25}$$

Note that the functions on the left-hand side of these constraints are concave. Now, we can define a variant of Problem 3 where constraints (20) and (21) are replaced by the following linearizations of constraints (24) and (25):

$$-\beta_i\delta w_{i-1} - \beta_i\delta w_{i+1} + \delta w_i \leq b'_{N_i} \tag{26}$$
$$\theta_i\delta w_{i-1} + \theta_i\delta w_{i+1} - \delta w_i \leq b'_{P_i} \tag{27}$$

where

$$\begin{aligned}\theta_i &= \frac{1 + 2h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}{2 - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}\\ \beta_i &= \frac{1 - 2h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}{2 + 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}\\ b'_{N_i} &= \frac{6h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}}}{2 + 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}\\ b'_{P_i} &= \frac{6h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}}}{2 - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}.\end{aligned} \tag{28}$$

The following proposition states that constraints (26) and (27) are tighter than constraints (20) and (21).

Proposition 2: For all $i = 2, \ldots, n-1$, it holds that $\beta_i \leq \eta_i \leq \theta_i$. Equality $\eta_i = \theta_i$ holds if the corresponding nonlinear constraint (24) is active at the current point $\mathbf{w}$. Similarly, $\eta_i = \beta_i$ holds if the corresponding nonlinear constraint (25) is active at the current point $\mathbf{w}$.

Proof: We only prove the results about θ_i and η_i. Those about β_i and η_i are proved in a completely analogous way. By definition of η_i and θ_i, we need to prove that

$$\frac{3w_{i+1} + 3w_{i-1} + 2w_i}{w_{i+1} + 6w_i + w_{i-1}} \leq \frac{1 + 2h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}{2 - 4h^2 J(\ell_i(\mathbf{w}))^{-\frac{3}{2}}}.$$

After few simple computations, this inequality can be rewritten as

$$4h^2 J(\ell_i(\mathbf{w}))^{-\frac{1}{2}} \geq (w_{i-1} - 2w_i + w_{i+1})$$

which holds in view of feasibility of $\mathbf{w}$ and, moreover, holds as an equality if constraint (24) is active at the current point $\mathbf{w}$, as we wanted to prove. □

In view of this result, by replacing constraints (20) and (21) with (26) and (27), we reduce the search space of the new displacement $\boldsymbol{\delta}\mathbf{w}$. On the other hand, the following proposition states that, with constraints (26) and (27), no line search is needed along the direction $\boldsymbol{\delta}\mathbf{w}$, i.e., we can always choose the step length $\alpha = 1$.

Proposition 3: If constraints (26) and (27) are employed as a replacement of constraints (20) and (21) in the definition of Problem 3, then, for each feasible solution $\boldsymbol{\delta}\mathbf{w}$ of this problem, it holds that $\mathbf{w} + \boldsymbol{\delta}\mathbf{w} \in \Omega$.

Proof: For the sake of convenience, let us rewrite Problem 2 in the following more compact form:

$$\begin{aligned}\min\ & f(\mathbf{w} + \boldsymbol{\delta}\mathbf{w})\\ & \mathbf{c}(\mathbf{w} + \boldsymbol{\delta}\mathbf{w}) \leq 0\end{aligned} \tag{29}$$

where the vector function $\mathbf{c}$ contains all constraints of Problem 2 and the nonlinear ones are given as in (24) and (25) (recall that, in that case, vector $\mathbf{c}$ is a vector of concave functions). Then, Problem 3 can be written as follows:

$$\min\ f(\mathbf{w} + \delta\mathbf{w}) \quad \mathbf{c}(\mathbf{w}) + \nabla\mathbf{c}(\mathbf{w})\delta\mathbf{w} \leq 0. \tag{30}$$

Now, it is enough to observe that, by concavity,

$$\mathbf{c}(\mathbf{w} + \boldsymbol{\delta}\mathbf{w}) \leq \mathbf{c}(\mathbf{w}) + \nabla\mathbf{c}(\mathbf{w})\boldsymbol{\delta}\mathbf{w}$$

so that each feasible solution of (30) is also feasible for (29). □

The above proposition states that the feasible region of Problem 3, when constraints (26) and (27) are employed in its definition, is a subset of the feasible region $\boldsymbol{\Omega}$ of the original Problem 2. As a final result of this section, we state the following theorem, which establishes convergence of algorithm SCA to a stationary (KKT) point of Problem 2.

Theorem 1: If algorithm SCA is run for an infinite number of iterations and there exists some positive integer value K such that for all iterations $k \geq K$, constraints (26) and (27) are always employed in the definition of Problem 3, and then, the sequence of points $\{\mathbf{w}^{(k)}\}$ generated by the algorithm converges to a KKT point of Problem 2.

In order to prove the theorem, we first need to prove some lemmas.

Lemma 1: The sequence $\{f(\mathbf{w}^{(k)})\}$ of the function values at points generated by algorithm SCA converges to a finite value.

Proof: The sequence is nonincreasing and bounded from below, e.g., by the value $f(\mathbf{u}_B)$, in view of the fact that the objective function f is monotonic decreasing. Thus, it converges to a finite value. □

Next, we need the following result based on strict convexity of the objective function f.

Lemma 2: For each $\delta > 0$ sufficiently small, it holds that

$$\min\left\{\max\{f(\mathbf{x}), f(\mathbf{y})\} - f\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right)\right.$$
$$\left. : \mathbf{x}, \mathbf{y} \in \Omega,\ \|\mathbf{x}-\mathbf{y}\| \ge \delta\right\} \ge \varepsilon_\delta > 0. \quad (31)$$

Proof: Due to strict convexity, it holds that, $\forall \mathbf{x} \neq \mathbf{y}$,

$$\max\{f(\mathbf{x}), f(\mathbf{y})\} - f\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) > 0.$$

Moreover, the function is a continuous one. Next, we observe that the region

$$\{\mathbf{x}, \mathbf{y} \in \Omega : \ \|\mathbf{x}-\mathbf{y}\| \ge \delta\}$$

is a compact set. Thus, by the Weierstrass theorem, the minimum in (31) is attained, and it must be strictly positive, as we wanted to prove. □

Finally, we prove that also the sequence of points generated by algorithm SCA converges to some point, feasible for Problem 2.

Lemma 3: It holds that

$$\|\delta\mathbf{w}^{(k)}\| \to 0.$$

Proof: Let us assume, by contradiction, that, over some infinite subsequence with index set $\mathcal{K}$, it holds that $\|\delta\mathbf{w}^{(k)}\| \ge 2\rho > 0$ for all $k \in \mathcal{K}$, i.e.,

$$\|\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}\| \ge 2\rho > 0 \quad (32)$$

where $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \delta\mathbf{w}^{(k)}$. Over this subsequence, it holds, by strict convexity, that

$$f\left(\mathbf{w}^{(k+1)}\right) \le f\left(\mathbf{w}^{(k)}\right) - \xi \quad \forall k \in \mathcal{K} \quad (33)$$

for some $\xi > 0$. Indeed, it follows by optimality of $\mathbf{w}^{(k)} + \delta\mathbf{w}^{(k)}$ for Problem 3 and convexity of f that

$$f\left(\mathbf{w}^{(k+1)}\right) \le f\left(\frac{\mathbf{w}^{(k+1)} + \mathbf{w}^{(k)}}{2}\right) \le f\left(\mathbf{w}^{(k)}\right)$$

so that

$$\max\left\{f\left(\mathbf{w}^{(k)}\right), f\left(\mathbf{w}^{(k+1)}\right)\right\} = f\left(\mathbf{w}^{(k)}\right).$$

Then, it follows from (32) and Lemma 2 that we can choose $\xi = \varepsilon_\rho > 0$. Thus, since (33) holds infinitely often, we should have $f(\mathbf{w}^{(k)}) \to -\infty$, which, however, is not possible in view of Lemma 1. □

Now, we are ready to prove Theorem 1.

Proof: As a consequence of Lemma 3, it also holds that

$$\mathbf{w}^{(k)} \to \bar{\mathbf{w}} \in \Omega. \quad (34)$$

Indeed, all points $\mathbf{w}^{(k)}$ belong to the compact feasible region Ω so that the sequence $\{\mathbf{w}^{(k)}\}$ admits accumulation points. However, due to Lemma 3, the sequence cannot have distinct accumulation points.

Now, let us consider the compact reformulation (29) of Problem 2 and the related linearization (30), equivalent to Problem 3 with the linearized constraints (26) and (27). Since the latter is a convex problem with linear constraints, its local minimizer $\delta\mathbf{w}^{(k)}$ (unique in view of strict convexity of the objective function) fulfills the following KKT conditions:

$$\begin{aligned}
&\nabla f\left(\mathbf{w}^{(k)} + \delta\mathbf{w}^{(k)}\right) + \boldsymbol{\mu}_k^\top \nabla\mathbf{c}\left(\mathbf{w}^{(k)}\right) = \mathbf{0}\\
&\mathbf{c}\left(\mathbf{w}^{(k)}\right) + \nabla\mathbf{c}\left(\mathbf{w}^{(k)}\right)\delta\mathbf{w}^{(k)} \le 0\\
&\boldsymbol{\mu}_k^\top\left(\mathbf{c}\left(\mathbf{w}^{(k)}\right) + \nabla\mathbf{c}\left(\mathbf{w}^{(k)}\right)\delta\mathbf{w}^{(k)}\right) = 0\\
&\boldsymbol{\mu}_k \ge \mathbf{0}
\end{aligned} \quad (35)$$

where $\boldsymbol{\mu}_k$ is the vector of Lagrange multipliers. Now, by taking the limit of system (35), possibly over a subsequence, in order to guarantee convergence of the multiplier vectors $\boldsymbol{\mu}_k$ to a vector $\bar{\boldsymbol{\mu}}$, in view of Lemma 3 and (34), we have that

$$\begin{aligned}
&\nabla f(\bar{\mathbf{w}}) + \bar{\boldsymbol{\mu}}^\top \nabla\mathbf{c}(\bar{\mathbf{w}}) = \mathbf{0}\\
&\mathbf{c}(\bar{\mathbf{w}}) \le 0\\
&\bar{\boldsymbol{\mu}}^\top \mathbf{c}(\bar{\mathbf{w}}) = 0\\
&\bar{\boldsymbol{\mu}} \ge \mathbf{0}
\end{aligned}$$

or, equivalently, the limit point $\bar{\mathbf{w}}$ is a KKT point of Problem 2, as we wanted to prove. □

Remark 2: In algorithm SCA at each iteration, we solve to optimality Problem 3. This is indeed necessary for the final iterations to prove the convergence result stated in Theorem 1. However, during the first iterations, it is not necessary to solve the problem to optimality: finding a feasible descent direction is enough. This does not alter the theoretical properties of the algorithm and allows to reduce the computing times.

In the rest of this article, we refer to constraints (18) and (19) as acceleration constraints, while constraints (20) and (21) [or (26) and (27)] are called (linearized) negative acceleration rate (NAR) and positive acceleration rate (PAR) constraints, respectively. Also, note that, in the different subproblems discussed in the following, we always refer to the linearization with constraints (20) and (21) and, thus, with parameters η_i, but the same results also hold for the linearization with constraints (26) and (27) and, thus, with parameters θ_i and β_i.

III. Subproblem With Acceleration and NAR Constraints

In this section, we propose an efficient method to solve Problem 3 when PAR constraints are removed. The solution of this subproblem becomes part of an approach to solve a suitable relaxation of Problem 3 and, in fact, under very mild assumptions, to solve Problem 3 itself. This is clarified in Section IV. We discuss: 1) the subproblem including only (17) and the acceleration constraints (18) and (19); 2) the subproblem including only (17) and the NAR constraints (20);

and 2) the subproblem including all constraints (17)–(20). Throughout the section, we need the results stated in the following two propositions. Let us consider problems with the following form, where $N = \{1, \dots, n\}$ and $M_j = \{1, \dots, m_j\}$, $j \in N$:

$$\begin{aligned}&\min\ g(x_1, \dots, x_n)\\ &\qquad x_j \le a_{i,j}x_{j-1} + b_{i,j}x_{j+1} + c_{i,j}, \quad i \in M_j,\ \ j \in N\\ &\qquad \ell_j \le x_j \le u_j, \quad j \in N \end{aligned} \tag{36}$$

where g is a monotonic decreasing function; $a_{i,j}, b_{i,j}, c_{i,j} \ge 0$, for $i \in M_j$ and $j \in N$; $a_{i,1} = 0$ for $i \in M_1$; and $b_{i,n} = 0$ for $i \in M_n$. The following result is proven in [28]. Here, we report the proof in order to make this article self-contained. We denote by P the feasible polytope of problem (36). Moreover, we denote by $\mathbf{z}$ the componentwise maximum of all feasible solutions in P, i.e., for each $j \in N$, $z_j = \max_{\mathbf{x}\in P} x_j$ (note that the above maximum value is attained since P is a polytope).

Proposition 4: The unique optimal solution of (36) is the componentwise maximum $\mathbf{z}$ of all its feasible solutions.

Proof: If we are able to prove that the componentwise maximum $\mathbf{z}$ of all feasible solutions is itself a feasible solution, by monotonicity of g, it must also be the unique optimal solution. In order to prove that $\mathbf{z}$ is feasible, we proceed as follows. For $j \in N$, let $\mathbf{x}^{*j}$ be the optimal solution of $\max_{\mathbf{x}\in P} x_j$ so that $z_j = x_j^{*j}$. Since $\mathbf{x}^{*j} \in P$, then it must hold that $\ell_j \le z_j \le u_j$. Moreover, let us consider the generic constraint

$$x_j \le a_{i,j}x_{j-1} + b_{i,j}x_{j+1} + c_{i,j}$$

for $i \in M_j$. It holds that

$$\begin{aligned} z_j = x_j^{*j} &\le a_{i,j}x_{j-1}^{*j} + b_{i,j}x_{j+1}^{*j} + c_{i,j}\\ &\le a_{i,j}z_{j-1} + b_{i,j}z_{j+1} + c_{i,j}\end{aligned}$$

where the first inequality follows from feasibility of $\mathbf{x}^{*j}$, while the second follows from nonnegativity of a_{ij} and b_{ij} and the definition of $\mathbf{z}$. Since this holds for all $j \in N$, the result is proven. □

Now, consider the problem obtained from (36) by removing some constraints, i.e., by taking $M'_j \subseteq M_j$ for each $j \in N$

$$\begin{aligned}&\min\ g(x_1, \dots, x_n)\\ &\qquad x_j \le a_{i,j}x_{j-1} + b_{i,j}x_{j+1} + c_{i,j}, \quad i \in M'_j,\ \ j \in N\\ &\qquad \ell_j \le x_j \le u_j, \quad j \in N. \end{aligned} \tag{37}$$

Later, we also need the result stated in the following proposition.

Proposition 5: The optimal solution $\bar{\mathbf{x}}^\star$ of problem (37) is an upper bound for the optimal solution $\mathbf{x}^\star$ of problem (36), i.e., $\bar{\mathbf{x}}^\star \ge \mathbf{x}^\star$.

Proof: It holds that $\mathbf{x}^\star$ is a feasible solution of problem (37) so that, in view of Proposition 4, $\bar{\mathbf{x}}^\star \ge \mathbf{x}^\star$ holds. □

A. Acceleration Constraints

The simplest case is the one where we only consider the acceleration constraints (18) and (19), besides constraints (17) with a generic upper bound vector $\mathbf{y} \ge \mathbf{0}$. The problem to be solved is

Problem 4:

$$\begin{aligned}&\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1}+\delta w_{i+1}} + \sqrt{w_i + \delta w_i}}\\ &\qquad \mathbf{l_B} \le \boldsymbol{\delta}\mathbf{w} \le \mathbf{y}\\ &\qquad \delta w_{i+1} - \delta w_i \le b_{A_i}, \quad i = 1, \dots, n-1\\ &\qquad \delta w_i - \delta w_{i+1} \le b_{D_i}, \quad i = 1, \dots, n-1.\end{aligned}$$

It can be seen that such a problem belongs to the class of problems (36). Therefore, in view of Proposition 4, the optimal solution of Problem 4 is the componentwise maximum of its feasible region. Moreover, in [3], it has been proven that Algorithm 1, based on a forward and a backward iteration and with $O(n)$ computational complexity, returns an optimal solution of Problem 4.

Algorithm 1 Routine `SolveAcc` for the Solution of the Problem With Acceleration Constraints

input : Upper bound $\mathbf{y}$
output: $\boldsymbol{\delta}\mathbf{w}$
$\delta w_1 = 0,\ \delta w_n = 0$;
for $i = 1$ **to** $n-1$ **do**
  $\delta w_{i+1} = \min\{\delta w_i + b_{A_i}, y_{i+1}\}$
for $i = n-1$ **to** 1 **do**
  $\delta w_i = \min\{\delta w_{i+1} + b_{A_i}, y_i\}$
return $\boldsymbol{\delta}\mathbf{w}$

B. NAR Constraints

Now, we consider the problem only including NAR constraints (20) and constraints (17) with upper bound vector $\mathbf{y}$

Problem 5:

$$\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1}+\delta w_{i+1}} + \sqrt{w_i + \delta w_i}}$$

$$\mathbf{0} \le \boldsymbol{\delta}\mathbf{w} \le \mathbf{y} \tag{38}$$

$$\delta w_i \le \eta_i(\delta w_{i-1} + \delta w_{i+1}) + b_{N_i}, \quad i = 2, \dots, n-1 \tag{39}$$

where $y_1 = y_n = 0$ because of the boundary conditions. Also, this problem belongs to the class of problems (36) so that Proposition 4 states that its optimal solution is the componentwise maximum of its feasible region. Problem 5 can be solved by using the graph-based approach presented in [4] and [28]. However, Cabassi *et al.* [4] show that, by exploiting the structure of a simpler version of the NAR constraints, it is possible to develop an algorithm more efficient than the graph-based one. Our purpose is to extend the results presented in [4] to a case with different and more challenging NAR constraints in order to develop an efficient algorithm outperforming the graph-based one.

Now, let us consider the restriction of Problem 5 between two generic indexes s and t such that $1 \le s < t \le n$, obtained by fixing $\delta w_s = y_s$ and $\delta w_t = y_t$ and by considering only the

NAR and upper bound constraints at $s+1,\dots,t-1$. Let $\boldsymbol{\delta}\mathbf{w}^*$ be the optimal solution of the restriction. We first prove the following lemma.

Lemma 4: The optimal solution $\boldsymbol{\delta}\mathbf{w}^*$ of the restriction of Problem 5 between two indexes s and t, $1 \le s < t \le n$, is such that, for each $j \in \{s+1,\dots,t-1\}$, either $\delta w_j^* \le y_j$ or $\delta w_j^* \le \eta_j(\delta w_{j+1}^* + \delta w_{j-1}^*) + b_{N_j}$ holds as an equality.

Proof: It is enough to observe that, in case both inequalities were strict for some j, then, in view of the monotonicity of the objective function, we could decrease the objective function value by increasing the value of δw_j^*, thus contradicting optimality of $\boldsymbol{\delta}\mathbf{w}^*$. □

Note that the above result also applies to the full Problem 5, which corresponds to the special case $s = 1$, $t = n$ with $y_1 = y_n = 0$. In view of Lemma 4, we have that there exists an index j, with $s < j \le t$, such that: 1) $\delta w_j^* = y_j$; 2) the upper bound constraint is not active at $s+1,\dots,j-1$; and 3) all NAR constraints $s+1,\dots,j-1$ are active. Then, j is the lowest index in $\{s+1,\dots,t-1\}$ where the upper bound constraint is active If index j were known, then the following observation allows returning the components of the optimal solution between s and j. Let us first introduce the following definitions of matrix $\mathbf{A}$ and vector $\mathbf{q}$:

$$\mathbf{A} = \begin{bmatrix} 1 & -\eta_{s+1} & 0 & \cdots & 0 \\ -\eta_{s+2} & 1 & -\eta_{s+2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -\eta_{j-1} & 1 \end{bmatrix}$$

$$\mathbf{q} = \begin{bmatrix} b_{N\,s+1} + \eta_{s+1} y_s \\ b_{N\,s+2} \\ \vdots \\ b_{N\,j-2} \\ b_{N\,j-1} + \eta_{j-1} y_j \end{bmatrix}. \tag{40}$$

Note that $\mathbf{A}$ is the square submatrix of the NAR constraints restricted to rows $s+1$ up to $j-1$ and the related columns.

Observation 1: Let $\boldsymbol{\delta}\mathbf{w}^*$ be the optimal solution of the restriction of Problem 5 between s and t and let $s < j$. If constraints $\delta w_s^* \le y_s$, $\delta w_j^* \le y_j$, and $\delta w_i^* \le \eta_i(\delta w_{i+1}^* + \delta w_{i-1}^*) + b_{N_i}$, for $i = s+1,\dots,j-1$, are all active, then $\delta w_{s+1}^*,\dots,\delta w_{j-1}^*$ are obtained by the solution of the following tridiagonal system:

$$\begin{aligned} &\delta w_s = y_s \\ &\qquad \delta w_r - \eta_r \delta w_{r+1} - \eta_r \delta w_{r-1} = b_{N\,r}, \quad r = s+1,\dots,j-1 \\ &\qquad \delta w_j = y_j \end{aligned}$$

or, equivalently, as

$$\begin{aligned} &\delta w_{s+1} - \eta_{s+1}\bar{x}_{s+2} \\ &\ = b_{N\,s+1} + \eta_{s+1} y_s \\ &\quad \delta w_r - \eta_r \delta w_{r+1} - \eta_r \delta w_{r-1} = b_{N\,r}, \quad r = s+2,\dots,j-2 \\ &\quad \delta w_{s+1} - \eta_{s+1}\bar{x}_{s+2} = b_{N\,s+1} + \eta_{s+1} y_s. \end{aligned} \tag{41}$$

In the matrix form, the above tridiagonal linear system can be written as

$$\mathbf{A}\boldsymbol{\delta}\mathbf{w}_{s+1,j-1}^* = \mathbf{q} \tag{42}$$

where matrix $\mathbf{A}$ and vector $\mathbf{q}$ are defined in (40) and $\boldsymbol{\delta}\mathbf{w}_{s+1,j-1}^*$ is the restriction of vector $\boldsymbol{\delta}\mathbf{w}$ to its components between $s+1$ and $j-1$.

Tridiagonal systems

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \quad i = 1,\dots,m$$

with $a_1 = c_m = 0$ can be solved through so-called Thomas algorithm [29] with $O(m)$ operations. In order to detect the lowest index $j \in \{s+1,\dots,t-1\}$ such that the upper bound constraint is active at j, we propose Algorithm 2, also called `SolveNAR` and described in what follows. We initially set $j = t$. Then, at each iteration, we solve the linear system (42). Let $\bar{\mathbf{x}} = (\bar{x}_{s+1},\dots,\bar{x}_{j-1})$ be its solution. We check whether it is feasible and optimal or not. Namely, if there exists $k \in \{s+1,\dots,j-1\}$ such that either $\bar{x}_k < 0$ or $\bar{x}_k > y_k$, then $\bar{\mathbf{x}}$ is unfeasible, and consequently, we need to reduce j by 1. If $\bar{x}_k = y_k$ for some $k \in \{s+1,\dots,j-1\}$, then we also reduce j by 1 since j is not in any case the lowest index of the optimal solution where the upper bound constraint is active. Finally, if $0 \le \bar{x}_k < y_k$, for $k = s+1,\dots,j-1$, then we need to verify if $\bar{\mathbf{x}}$ is the best possible solution over the interval $\{s+1,\dots,j-1\}$. We are able to check that after proving the following result.

Proposition 6: Let matrix $\mathbf{A}$ and vector $\mathbf{q}$ be defined as in (40). The optimal solution $\boldsymbol{\delta}\mathbf{w}^*$ of the restriction of Problem 5 between s and t satisfies

$$\delta w_s^* = y_s, \quad \delta w_r^* = \bar{x}_r, \ r = s+1,\dots,j-1, \ \delta w_j^* = y_j \tag{43}$$

if and only if the optimal value of the LP problem

$$\begin{aligned} \max_{\boldsymbol{\epsilon}} \quad & \mathbf{1}^T \boldsymbol{\epsilon} \\ & \mathbf{A}\boldsymbol{\epsilon} \le \mathbf{0} \\ & \boldsymbol{\epsilon} \le \bar{\mathbf{y}} - \bar{\mathbf{x}} \end{aligned} \tag{44}$$

is strictly positive or, equivalently, if the following system admits no solution:

$$\mathbf{A}^T \boldsymbol{\lambda} = \mathbf{1}, \quad \boldsymbol{\lambda} \ge \mathbf{0}. \tag{45}$$

Proof: Let us first assume that $\boldsymbol{\delta}\mathbf{w}^*$ does not fulfill (43). Then, in view of Lemma 4, j is not the lowest index such that the upper bound is active at the optimal solution, and consequently, $\delta w_k^* = y_k > \bar{x}_k$ for some $k \in \{s+1,\dots,j-1\}$. Such optimal solution must be feasible, and in particular, it must satisfy all NAR constraints between $s+1$ and $j-1$ and the upper bound constraints between $s+1$ and j, i.e.,

$$\begin{aligned} &\delta w_{s+1}^* - \eta_{s+1}\delta w_{s+2}^* \\ &\le b_{N\,s+1} + \eta_{s+1} y_s \\ &\quad \delta w_r^* - \eta_r \delta w_{r+1}^* - \eta_r \delta w_{r-1}^* \le b_{N\,r}, \quad r = s+2,\dots,j-2 \\ &\quad \delta w_{j-1}^* - \eta_{j-1}\delta w_{j-2}^* - \eta_{j-1}\delta w_j^* \le b_{N\,j-1} \\ &\quad \delta w_r^* \le y_r, \quad r = s+1,\dots,j. \end{aligned}$$

In view of $\delta w_j^* \le y_j$ and $\eta_{j-1} \ge 0$, $\boldsymbol{\delta}\mathbf{w}^*$ also satisfies the following system of inequalities:

$$\begin{aligned} &\delta w_{s+1}^* - \eta_{s+1}\delta w_{s+2}^* \\ &\le b_{N\,s+1} + \eta_{s+1} y_s \\ &\quad \delta w_r^* - \eta_r \delta w_{r+1}^* - \eta_r \delta w_{r-1}^* \le b_{N\,r}, \quad r = s+2,\dots,j-2 \end{aligned}$$

$$\delta w^*_{j-1} - \eta_{j-1}\delta w^*_{j-2} \le b_{N\,j-1} + \eta_{j-1} y_j$$
$$\delta w^*_r \le y_r, \quad r = s+1, \ldots, j-1.$$

After making the change of variables $\delta w^*_r = \bar{x}_r + \epsilon_r$ for $r = s+1, \ldots, j-1$, and recalling that $\bar{\mathbf{x}}$ solves system (41), the system of inequalities can be further rewritten as

$$\begin{aligned}
&\epsilon_{s+1} - \eta_{s+1}\epsilon_{s+2} \le 0\\
&\epsilon_r - \eta_r \epsilon_{r+1} - \eta_r \epsilon_{r-1} \le 0, \quad r = s+2, \ldots, j-2\\
&\epsilon_{j-1} - \eta_{j-1}\epsilon_{j-2} \le 0\\
&\epsilon_r \le y_r - \bar{x}_r, \quad r = s+1, \ldots, j-1.
\end{aligned}$$

Finally, recalling the definition of matrix $\mathbf{A}$ and vector $\mathbf{q}$ given in (40), this can also be written in a more compact form as

$$\begin{aligned}
\mathbf{A}\boldsymbol{\epsilon} &\le \mathbf{0}\\
\boldsymbol{\epsilon} &\le \bar{\mathbf{y}} - \bar{\mathbf{x}}.
\end{aligned}$$

If $\delta w^*_k = y_k > \bar{x}_k$ for some $k \in \{s+1, \ldots, j-1\}$, then the system must admit a solution with $\epsilon_k > 0$. This is equivalent to prove that problem (44) has an optimal solution with at least one strictly positive component, and the optimal value is strictly positive. Indeed, in view of the definition of matrix $\mathbf{A}$, problem (44) has the structure of the problems discussed in Proposition 4. More precisely, to see that, we need to remark that maximizing $\mathbf{1}^T\boldsymbol{\epsilon}$ is equivalent to minimizing the decreasing function $-\mathbf{1}^T\boldsymbol{\epsilon}$. Then, observing that $\boldsymbol{\epsilon} = \mathbf{0}$ is a feasible solution of problem (44), by Proposition 4, the optimal solution $\boldsymbol{\epsilon}^*$ must be a nonnegative vector, and since at least one component, namely, component k, is strictly positive, then the optimal value must also be strictly positive.

Conversely, let us assume that the optimal value is strictly positive, and $\boldsymbol{\epsilon}^*$ is an optimal solution with at least one strictly positive component. Then, there are two possible alternatives. Either the optimal solution $\boldsymbol{\delta}\mathbf{w}^*$ of the restriction of Problem 5 between s and t is such that $\delta w^*_j < y_j$, in which case (43) obviously does not hold, or $\delta w^*_j = y_j$. In the latter case, let us assume by contradiction that (43) holds. We observe that the solution that is defined as follows:

$$\begin{aligned}
x'_s &= y_s\\
x'_r &= \bar{x}_r + \epsilon^*_r = \delta w^*_r + \epsilon^*_r, \quad r = s+1, \ldots, j-1\\
x'_j &= y_j = \delta w^*_j\\
x'_r &= \delta w^*_r, \quad r = j+1, \ldots, t
\end{aligned}$$

is feasible for the restriction of Problem 5 between s and t. Indeed, by feasibility of $\boldsymbol{\epsilon}^*$ in problem (44), all upper bound and NAR constraints between s and $j-1$ are fulfilled. Those between, $j+1$ and t, are also fulfilled by the feasibility of $\boldsymbol{\delta}\mathbf{w}^*$. Then, we only need to prove that the NAR constraint at j is satisfied. By feasibility of $\boldsymbol{\delta}\mathbf{w}^*$ and in view of $\epsilon^*_{j-1}, \eta_j \ge 0$, we have that

$$\begin{aligned}
x'_j = \delta w^*_j &\le \eta_j \delta w^*_{j-1} + \eta_j \delta w^*_{j+1} + b_{N\,j}\\
&\le \eta_j\left(\delta w^*_{j-1} + \epsilon_{j-1}\right) + \eta_j \delta w^*_{j+1} + b_{N\,j}\\
&= \eta_j x'_{j-1} + \eta_j x'_{j+1} + b_{N\,j}.
\end{aligned}$$

Thus, $\mathbf{x}'$ is feasible such that $\mathbf{x}' \ge \boldsymbol{\delta}\mathbf{w}^*$ with at least one strict inequality (recall that at least one component of $\boldsymbol{\epsilon}^*$ is strictly positive), which contradicts the optimality of $\boldsymbol{\delta}\mathbf{w}^*$ (recall that the optimal solution must be the componentwise maximum of all feasible solutions).

In order to prove the last part, i.e., problem (44) has a positive optimal value if and only if (45) admits no solution, and we notice that the optimal value is positive if and only if the feasible point $\boldsymbol{\epsilon} = \mathbf{0}$ is not an optimal solution, or equivalently, the null vector is not a KKT point. Since, at $\boldsymbol{\epsilon} = \mathbf{0}$, constraints $\boldsymbol{\epsilon} \le \bar{\mathbf{y}} - \bar{\mathbf{x}}$ cannot be active, then the KKT conditions for problem (44) at this point are exactly those established in (45), where vector $\boldsymbol{\lambda}$ is the vector of Lagrange mutlpliers for constraints $\mathbf{A}\boldsymbol{\epsilon} \le \mathbf{0}$. This concludes the proof. □

Then, if (45) admits no solution, (43) does not hold, and again, we need to reduce j by 1. Otherwise, we can fix the optimal solution between s and j according to (43). After that, we recursively call the routine `SolveNAR` on the remaining subinterval $\{j, \ldots, t\}$ in order to obtain the solution over the full interval.

Remark 3: In Algorithm 2, routine `isFeasible` is the routine used to verify if, for $k = s+1, \ldots, j-1$, $0 \le \bar{x}_k < y_k$, while `isOptimal` is the procedure to check optimality of $\bar{\mathbf{x}}$ over the interval $\{s+1, \ldots, j-1\}$, i.e., (43) holds.

Now, we are ready to prove that Algorithm 2 solves Problem 5.

Proposition 7: The call `solveNAR(`$\mathbf{y}$`, 1, `n`)` of Algorithm 2 returns the optimal solution of Problem 5.

Proof: After the call `solveNAR(`$\mathbf{y}$`, 1, `n`)`, we are able to identify the portion of the optimal solution between 1 and some index j_1, $1 < j_1 \le n$. If $j_1 = n$, then we are done. Otherwise, we make the recursive call `solveNAR(`$\mathbf{y}$`, `j_1`, `n`)`, which enables to identify also the portion of the optimal solution between j_1 and some index j_2, $j_1 < j_2 \le n$. If $j_2 = n$, then we are done. Otherwise, we make the recursive call `solveNAR(`$\mathbf{y}$`, `j_2`, `n`)` and so on. After at most n recursive calls, we are able to return the full optimal solution. □

Algorithm 2 `SolveNAR`$(\mathbf{y}, s, t)$

input : Upper bound $\mathbf{y}$ and two indices s and t with $1 \le s < t \le n$
output: $\boldsymbol{\delta}\mathbf{w}^*$

```
Set j = t;
δw* = y;
while j ≥ s + 1 do
    Compute the solution x̄ of the linear system (42);
    if isFeasible(x̄) and isOptimal(x̄) then
        Break;
    else
        Set j = j − 1;
for i = s + 1, ..., j − 1 do
    Set δw*_i = x̄_i;
return δw* = min{δw*, SolveNAR(δw*, j, t)};
```

Remark 4: Note that Algorithm 2 involves solving a significant amount of linear systems, both to compute $\bar{\mathbf{x}}$ and verify its optimality [see (42) and (45)]. Some tricks can be employed to reduce the number of operations. Some of these are discussed in [30].

The following proposition states the worst case complexity of `solveNAR(`$\mathbf{y}$`,1,`n`)`.

Proposition 8: Problem 5 can be solved with $O(n^3)$ operations by running the procedure `SolveNAR`$(\mathbf{y}, 1, n)$ and by using the Thomas algorithm for the solution of each linear system.

Proof: In the worst case, at the first call, we have $j_1 = 2$ since we need to go all the way from $j = n$ down to $j = 2$. Since, for each j, we need to solve a tridiagonal system, which requires at most $O(n)$ operations, the first call of `SolveNAR` requires $O(n^2)$ operations. This is similar for all successive calls, and since the number of recursive calls is at most $O(n)$, the overall effort is at most of $O(n^3)$ operations. □

In fact, what we observed is that the practical complexity of the algorithm is much better, namely, $\Theta(n^2)$.

C. Acceleration and NAR Constraints

Now, we discuss the problem with acceleration and NAR constraints, with upper bound vector $\mathbf{y}$, i.e.,

Problem 6:

$$\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{n-1} \frac{2h}{\sqrt{w_{i+1}+\delta w_{i+1}}+\sqrt{w_i+\delta w_i}}$$

$$\mathbf{l_B} \le \boldsymbol{\delta}\mathbf{w} \le \mathbf{y}$$

$$\delta w_{i+1} - \delta w_i \le b_{A_i}, \quad i = 1,\ldots,n-1$$

$$\delta w_i - \delta w_{i+1} \le b_{Di}, \quad i = 1,\ldots,n-1$$

$$\delta w_i - \eta_i \delta w_{i-1} - \eta_i \delta w_{i+1} \le b_{N_i}, \quad i = 2,\ldots,n-1.$$

We first remark that Problem 6 has the structure of problem (36) so that, by Proposition 4, its unique optimal solution is the componentwise maximum of its feasible region. As for Problem 5, we can solve Problem 6 by using the graph-based approach proposed in [28]. However, Cabassi *et al.* [4] show that, if we adopt a very efficient procedure to solve Problems 4 and 5, then it is worth splitting the full problem into two separated ones and use an iterative approach (see Algorithm 3). Indeed, Problems 4–6 share the common property that their optimal solution is also the componentwise maximum of the corresponding feasible region. Moreover, according to Proposition 5, the optimal solutions of Problems 4 and 5 are valid upper bounds for the optimal solution (actually, also for any feasible solution) of the full Problem 6. In Algorithm 3, we first call the procedure `SolveACC` with input the upper bound vector $\mathbf{y}$. Then, the output of this procedure, which, according to what we have just stated, is an upper bound for the solution of the full Problem 6, satisfies $\boldsymbol{\delta}\mathbf{w}_{\text{Acc}} \le \mathbf{y}$, and becomes the input for a call of the procedure `SolveNAR`. The output $\boldsymbol{\delta}\mathbf{w}_{\text{NAR}}$ of this call is again an upper bound for the solution of the full Problem 6, and it satisfies $\boldsymbol{\delta}\mathbf{w}_{\text{NAR}} \le \boldsymbol{\delta}\mathbf{w}_{\text{Acc}}$. This output becomes the input of a further call to the procedure `SolveACC`, and we proceed in this way until the distance between two consecutive output vectors falls below a prescribed tolerance value ε. The following proposition states that the sequence of output vectors generated by the alternate calls to the procedures `SolveACC` and `SolveNAR` converges to the optimal solution of the full Problem 6.

Proposition 9: Algorithm 3 converges to the the optimal solution of Problem 6 when $\varepsilon = 0$ and stops after a finite number of iterations if $\varepsilon > 0$.

Proof: We have observed that the sequence of alternate solutions of Problems 4 and 5, here denoted by $\{\mathbf{y}_t\}$, is: 1) a sequence of valid upper bounds for the optimal solution of Problem 6; 2) componentwise monotonic nonincreasing; and 3) componentwise bounded from below by the null vector. Thus, if $\varepsilon = 0$, an infinite sequence is generated, which converges to some point $\bar{\mathbf{y}}$, which is also an upper bound for the optimal solution of Problem 6 but, more precisely, by continuity, is also a feasible point of the problem and, is thus, also the optimal solution of the problem. If $\varepsilon > 0$, due to the convergence to some point $\bar{\mathbf{y}}$, at some finite iteration, the exit condition of the while loop must be satisfied. □

Algorithm 3 Algorithm `SolveACCNAR` for the Solution of Problem 6

input : The upper bound $\mathbf{y}$ and the tolerance ε
output: The optimal solution $\boldsymbol{\delta}\mathbf{w}^*$ and the optimal value f^*

```
δw_Acc = SolveACC(y);
δw_NAR = SolveNAR(δw_Acc, 1, n);
while ‖δw_NAR − δw_Acc‖ > ε do
    δw_Acc = SolveACC(δw*);
    δw_NAR = SolveNAR(δw_Acc, 1, n);
δw* = δw_NAR;
return δw*, evaluateObj(δw*)
```

IV. Descent Method for the Case of Acceleration, PAR, and NAR Constraints

Unfortunately, PAR constraints (21) do not satisfy the assumptions requested in Proposition 4 in order to guarantee that the componentwise maximum of the feasible region is the optimal solution of Problem 3. However, in Section III, we have shown that Problem 6, i.e., Problem 3 without the PAR constraints, can be efficiently solved by Algorithm 3. Our purpose then is to separate the acceleration and NAR constraints from the PAR constraints.

Definition 1: Let $f:\mathbb{R}^n \to \mathbb{R}$ be the objective function of Problem 3, and let $\mathcal{D}$ be the region defined by the acceleration and NAR constraints (the feasible region of Problem 6). We define the function $F:\mathbb{R}^n \to \mathbb{R}$ as follows:

$$F(\mathbf{y}) = \min\{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{D}, \mathbf{x} \le \mathbf{y}\}.$$

Namely, F is the optimal value function of Problem 6 when the upper bound vector is $\mathbf{y}$.

Proposition 10: Function F is a convex function.

Proof: Since Problem 6 is convex, then the optimal value function F is convex (see [31, Sec. 5.6.1]). □

Now, let us introduce the following problem:

Problem 7:

$$\min_{\mathbf{y}\in\mathbb{R}^n} \quad F(\mathbf{y}) \tag{46}$$

$$\eta_i(y_{i-1}+y_{i+1}) - y_i \le b_{P_i}, \quad i=2,\ldots,n-1 \tag{47}$$

$$\mathbf{l_B} \le \mathbf{y} \le \mathbf{u_B}. \tag{48}$$

Such a problem is a relaxation of Problem 3. Indeed, each feasible solution of Problem 3 is also feasible for Problem 7, and the value of F at such solution is equal to the value of the objective function of Problem 3 at the same solution. We solve Problem 7 rather than Problem 3 to compute the new displacement $\boldsymbol{\delta}\mathbf{w}$. More precisely, if $\mathbf{y}^*$ is the optimal solution of Problem 7, then we set

$$\boldsymbol{\delta}\mathbf{w} = \arg\min_{\mathbf{x}\in\mathcal{D},\mathbf{x}\le\mathbf{y}^*} f(\mathbf{x}). \tag{49}$$

In the following proposition, we prove that, under a very mild condition, the optimal solution of Problem 7 computed in (49) is feasible and, thus, optimal for Problem 3 so that, although we solve a relaxation of the latter problem, we return an optimal solution for it.

Proposition 11: Let $\mathbf{w}^{(k)}$ be the current point. If

$$\ell_j(\boldsymbol{\delta}\mathbf{w}) \le \ell_j\left(\mathbf{w}^{(k)}\right)\left(3+\min\left\{0,\xi\left(\mathbf{w}^{(k)}\right)\right\}\right), \quad j=2,\ldots,n-1 \tag{50}$$

where $\boldsymbol{\delta}\mathbf{w}$ is computed through (49) and

$$\xi\left(\mathbf{w}^{(k)}\right) = \frac{\sqrt{\ell_j\left(\mathbf{w}^{(k)}\right)}\left(w_{j-1}^{(k)}+w_{j+1}^{(k)}-2w_j^{(k)}\right)}{2h^2J} \ge -2$$

(the inequality follows from feasibility of $\mathbf{w}^{(k)}$), then $\boldsymbol{\delta}\mathbf{w}$ is feasible for Problem 3, both if the nonlinear constraints are linearized as in (20) and (21), and if they are linearized as in (26) and (27).

Proof: First, we notice that, if we prove the result for the tighter constraints (26) and (27), then it must also hold for constraints (20) and (21). Thus, we prove the result only for the former. By definition (49), $\boldsymbol{\delta}\mathbf{w}$ satisfies the acceleration and NAR constraints so that

$$\begin{aligned}
\delta w_j &\le \delta w_{j+1} + b_{D_j}\\
\delta w_j &\le \delta w_{j-1} + b_{A_{j-1}}\\
\delta w_j &\le \beta_j\left(\delta w_{j+1}+\delta w_{j-1}\right) + b'_{N_j}\\
\delta w_j &\le y_j^*.
\end{aligned}$$

At least one of these constraints must be active; otherwise, δw_j could be increased, thus contradicting optimality. If the active constraint is $\delta w_j \le \beta_j(\delta w_{j+1}+\delta w_{j-1}) + b'_{N_j}$, then constraint (27) can be rewritten as follows:

$$\begin{aligned}
4h^2J\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{3}{2}}&\left(\delta w_{j+1}+2\delta w_j+\delta w_{j-1}\right)\\
&\le 12h^2J\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{1}{2}}
\end{aligned}$$

or, equivalently,

$$\ell_j(\boldsymbol{\delta}\mathbf{w}) \le 3\ell_j\left(\mathbf{w}^{(k)}\right)$$

implied by (50), and thus, the constraint is satisfied under the given assumption. If $\delta w_j = y_j^*$, then

$$\theta_j\left(\delta w_{j-1}+\delta w_{j+1}\right) \le \theta_j\left(y_{j-1}^*+y_{j+1}^*\right) \le y_j^* + b'_{P_j} = \delta w_j + b'_{P_j}$$

where the second inequality follows from the fact that $\mathbf{y}^*$ satisfies the PAR constraints. Now, let $\delta w_j = \delta w_{j+1} + b_{D_j}$ (the case when $\delta w_j \le \delta w_{j-1} + b_{A_{j-1}}$ is active can be dealt with in a completely analogous way). First, we observe that $\delta w_j \ge \delta w_{j-1} - b_{D_{j-1}}$. Then,

$$2\delta w_j \ge \delta w_{j+1} + \delta w_{j-1} + b_{D_j} - b_{D_{j-1}}.$$

In view of the definitions of b_{D_j} and $b_{D_{j-1}}$, this can also be written as

$$2\delta w_j \ge \delta w_{j+1} + \delta w_{j-1} + w_{j+1}^{(k)} - 2w_j^{(k)} + w_{j-1}^{(k)}. \tag{51}$$

Now, after recalling the definitions of θ_j and b'_{P_j} given in (28), and setting $\Delta = h^2J$, (27) can be rewritten as

$$\begin{aligned}
2\delta w_j \ge \delta w_{j+1} + \delta w_{j-1} &+ 2\Delta\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{3}{2}}\ell_j(\boldsymbol{\delta}\mathbf{w})\\
&-6\Delta\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{1}{2}}.
\end{aligned}$$

Taking into account (51), such inequality certainly holds if

$$\begin{aligned}
w_{j+1}^{(k)} - 2w_j^{(k)} + w_{j-1}^{(k)} &\ge 2\Delta\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{3}{2}}\ell_j(\boldsymbol{\delta}\mathbf{w})\\
&\quad -6\Delta\left(\ell_j\left(\mathbf{w}^{(k)}\right)\right)^{-\frac{1}{2}}
\end{aligned}$$

which is equivalent to

$$\ell_j(\boldsymbol{\delta}\mathbf{w}) \le \ell_j\left(\mathbf{w}^{(k)}\right)\left(3+\xi\left(\mathbf{w}^{(k)}\right)\right).$$

This is also implied by (50). □

Assumption (50) is mild. In order to fulfill it, one can impose restrictions on $\delta w_{j-1}, \delta w_j$ and δw_{j+1}. In fact, in the computational experiments, we did not impose such restrictions unless a positive step-length along the computed direction $\boldsymbol{\delta}\mathbf{w}$ could not be taken (which, however, never occurred in our experiments).

Now, let us turn our attention toward the solution of Problem 7. In order to solve it, we propose a descent method. We can exploit the information provided by the dual optimal solution $\boldsymbol{\nu} \in \mathbb{R}_+^n$ associated with the upper bound constraints of Problem 6. Indeed, from the sensitivity theory, we know that the dual solution is related to the gradient of the optimal value function F (see Definition 1) and provides information about how it changes its value for small perturbations of the upper bound values (for further details, see [31, Secs. 5.6.2 and 5.6.5]). Let $\mathbf{y}^{(t)}$ be a feasible solution of Problem 7 and $\boldsymbol{\nu} \in \mathbb{R}_+^n$ be the Lagrange multipliers of the upper bound constraints of Problem 6 when the upper bound is $\mathbf{y}^{(t)}$. Let

$$\varphi_i = b_{P_i} - \eta_i\left(y_{i-1}^{(t)} + y_{i+1}^{(t)}\right) + y_i^{(t)}, \quad i=2,\ldots,n-1.$$

Then, a *feasible descent direction* $\mathbf{d}^{(t)}$ can be obtained by solving the following LP problem:

Problem 8:

$$\min_{\mathbf{d}\in\mathbb{R}^n} -\boldsymbol{\nu}^T\mathbf{d} \tag{52}$$

$$\eta_i(d_{i-1}+d_{i+1}) - d_i \le \varphi_i, \quad i=2,\ldots,n-1 \tag{53}$$

$$\mathbf{l_B} \le \mathbf{y}^{(t)} + \mathbf{d} \le \mathbf{u_B} \tag{54}$$

where the objective function (52) imposes that $\mathbf{d}^{(t)}$ is a descent direction, while constraints (53) and (54) guarantee feasibility with respect to Problem 7. Problem 8 is an LP problem, and consequently, it can easily be solved through a standard LP solver. In particular, we employed GUROBI [32]. Unfortunately, since the information provided by the dual optimal solution $\boldsymbol{\nu}$ is local and related to small perturbations of the upper bounds, it might happen that $F(\mathbf{y}^{(t)}+\mathbf{d}^{(t)}) \geq F(\mathbf{y}^{(t)})$. To overcome this issue, we introduce a trust-region constraint in Problem 8. Thus, let $\sigma^{(t)} \in \mathbb{R}_+$ be the radius of the trust region at iteration t; then, we have

Problem 9:

$$\min_{\mathbf{d}\in\mathbb{R}^n} -\boldsymbol{\nu}^T \mathbf{d} \tag{55}$$

$$\eta_i(d_{i-1}+d_{i+1}) - d_i \leq \varphi_i, \quad i = 2, \dots, n-1 \tag{56}$$

$$\bar{\mathbf{l}}_{\mathbf{B}} \leq \mathbf{d} \leq \bar{\mathbf{u}}_{\mathbf{B}} \tag{57}$$

where $\bar{l}_{B_i} = \max\{l_{B_i} - y_i^{(t)}, -\sigma^{(t)}\}$ and $\bar{u}_{B_i} = \min\{u_{B_i} - y_i^{(t)}, \sigma^{(t)}\}$ for $i = 1, \dots, n$. After each iteration of the descent algorithm, we change the radius $\sigma^{(t)}$ according to the following rules.

1) If $F(\mathbf{y}^{(t)}+\mathbf{d}^{(t)}) \geq F(\mathbf{y}^{(t)})$, then we set $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)}$, and we tight the trust region by decreasing $\sigma^{(t)}$ by a factor $\tau \in (0, 1)$.
2) If $F(\mathbf{y}^{(t)}+\mathbf{d}^{(t)}) < F(\mathbf{y}^{(t)})$, then we set $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)}+\mathbf{d}^{(t)}$ and enlarge the radius $\sigma^{(t)}$ by a factor $\rho > 1$.

The proposed descent algorithm is sketched in Fig. 3, which reports the flowchart of the procedure `ComputeUpdate` used in algorithm SCA. We initially set $\mathbf{y}^{(0)} = \mathbf{0}$. At each iteration t, we evaluate the objective function $F(\mathbf{y}^t)$ by solving Problem 6 with upper bound vector $\mathbf{y}^{(t)}$ through a call of the routine `solveACCNAR` (see Algorithm 3). Then, we compute the Lagrange multipliers $\boldsymbol{\nu}^{(t)}$ associated with the upper bound constraints. After that, we compute a candidate descent direction $\mathbf{d}^{(t)}$ by solving Problem 9. If $\mathbf{d}^{(t)}$ is a descent step, then we set $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} + \mathbf{d}^{(t)}$ and enlarge the radius of the trust region; otherwise, we do not move to a new point, and we tight the trust region and solve again Problem 9. The descent algorithm stops as soon as the radius of the trust region becomes smaller than a fixed tolerance ε_1.

Remark 5: Note that we initially set $\mathbf{y}^{(0)} = \mathbf{0}$. However, any feasible solution of Problem 9 does the job, and actually, starting with a good initial solution may enhance the performance of the algorithm.

Remark 6: Problem 9 is an LP and can be solved by any existing LP solver. However, a suboptimal solution to Problem 9, obtained by a heuristic approach, is also acceptable. Indeed, we observe that: 1) an *optimal* descent direction is not strictly required and 2) a heuristic approach allows to reduce the time needed to get a descent direction. In this article, we employed a possible heuristic, whose description can be found in [30], but the development of further heuristic approaches is a possible topic for future research.

V. Computational Experiments

In this section, we present various computational experiments performed in order to evaluate the approaches proposed in Sections III and IV.

In particular, we compared solutions of Problem 2 computed by algorithm SCA to solutions obtained with commercial NLP solvers. Note that, with a single exception, we did not carry out a direct comparison with other methods specifically tailored to Problem 2 for the following reasons.

1) Some algorithms (such as [22] and [23]) use heuristics to quickly find suboptimal solutions of acceptable quality but do not achieve local optimality. Hence, comparing their solution times with SCA would not be fair. However, in one of our experiments (see Experiment 4), we made a comparison between the most recent heuristic proposed in [23] and algorithm SCA, both in terms of computing times and in terms of the quality of the returned solution.
2) The method presented in [26] does not consider the (nonconvex) jerk constraint but solves a convex problem whose objective function has a penalization term that includes pseudojerk. Due to this difference, a direct comparison with SCA is not possible.
3) The method presented in [24] is based on the numerical solution of a large number of nonlinear and nonconvex subproblems and is, therefore, structurally slower than SCA, whose main iteration is based on the efficient solution of the convex Problem 3.

In the first two experiments, we compare the computational time of IPOPT, a general-purpose NLP solver [33], with that of algorithm SCA over some randomly generated instances of Problem 2. In particular, we tested two different versions of the algorithm SCA. The first version, called SCA-H in what follows, employs the heuristic mentioned in Remark 6. Since the heuristic procedure may fail in some cases, in such cases, we also need an LP solver. In particular, in our experiments, we used GUROBI whenever the heuristic did not produce either a feasible solution to Problem 9 or a descent direction. In the second version, called SCA-G in what follows, we always employed GUROBI to solve Problem 9. For what concerns the choice of the NLP solver IPOPT, we remark that we chose it after a comparison with two further general-purpose NLP solvers, SNOPT and MINOS, which, however, turned out to perform worse than IPOPT on this class of problems.

In the third experiment, we compare the performance of the two implemented versions of algorithm SCA applied to two specific paths and see their behavior as the number n of discretized points increases.

In the fourth experiment, we compare the solutions returned by algorithm SCA with those returned by the heuristic recently proposed in [23].

Finally, in the fifth experiment, we present a real-life speed planning task for an LGV operating in an industrial setting, using real problem bounds and paths layouts, provided by an automation company based in Parma, Italy.

We remark that, according to our experiments, the special purpose routine `solveACCNAR` (Algorithm 3) strongly outperforms general-purpose approaches, such as the graph-based approach proposed in [28], and GUROBI, when solving Problem 6 (which can be converted into an LP as discussed in [28]).

Finally, we remark that we also tried to solve the convex Problem 3 arising at each iteration of the proposed

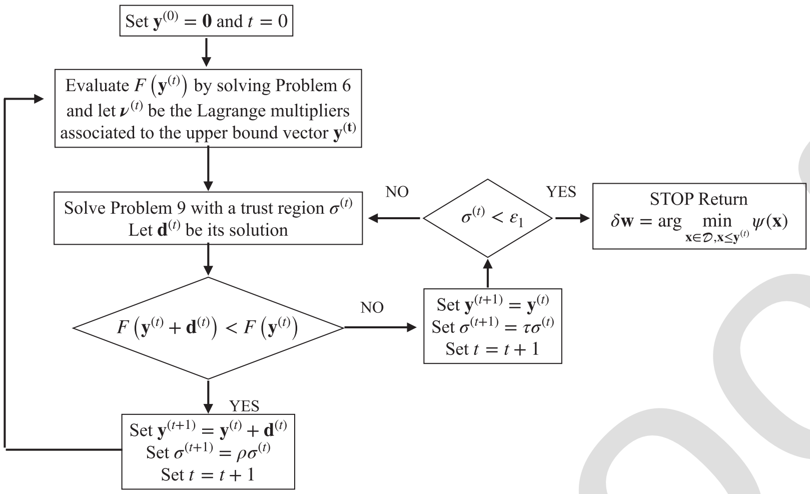

Fig. 3. Flowchart of the routine ComputeUpdate.

method with an NLP solver in place of the procedure ComputeUpdate, presented in this article. However, the experiments revealed that, in doing this, the computing times become much larger even with respect to the single call to the NLP solver for solving the nonconvex Problem 2.

All tests have been performed on an IntelCore i7-8550U CPU at 1.8 GHz. Both for IPOPT and algorithm SCA, the null vector was chosen as a starting point. The parameters used within algorithm SCA were $\varepsilon = 1e^{-8}$, $\varepsilon_1 = 1e^{-6}$ (tolerance parameters), $\rho = 4$, and $\tau = 0.25$ (trust-region update parameters). The initial trust region radius $\sigma^{(0)}$ was initialized to 1 in the first iteration $k = 0$ but adaptively set equal to the size of the last update $\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\|_\infty$ in all subsequent iterations (this adaptive choice allowed to reduce computing times by more than a half). We remark that algorithm SCA has been implemented in MATLAB, so we expect better performance after a C/C++ implementation.

A. Experiments 1 and 2

In Experiment 1, we generated a set of 50 different paths, each of which was discretized setting $n = 100$, $n = 500$, and $n = 1000$ sample points. The instances were generated by assuming that the traversed path was divided into seven intervals over which the curvature of the path was assumed to be constant. Thus, the n-dimensional upper bound vector $\mathbf{u}$ was generated as follows. First, we fixed $u_1 = u_n = 0$, i.e., the initial and final speeds must be equal to 0. Next, we partitioned the set $\{2, \ldots, n-1\}$ into seven subintervals I_j, $j \in \{1, \ldots, 7\}$, which corresponds to intervals with constant curvature. Then, for each subinterval, we randomly generated a value $u_j \in (0, \tilde{u}]$, where $\tilde{u}$ is the maximum upper bound (which was set equal to 100 m^2s^{-2}). Finally, for each $j \in \{1, \ldots, 7\}$, we set $u_k = \tilde{u}_j \ \forall k \in I_j$. The maximum acceleration parameter A is set equal to 2.78 ms^{-2} and the maximum jerk J to 0.5 ms^{-3}, while the path length is $s_f =$ 60 m. The values for A and J allow a comfortable motion for a ground transportation vehicle (see [34]).

In Experiment 2, we generated a further set of 50 different paths, each of which was discretized using $n = 100$, $n = 500$, and $n = 1000$ variables. These new instances were randomly generated such that the traversed path was divided into up to five intervals over which the curvature could be zero, linear with respect to the arc length or constant. We chose this kind of path since they are able to represent the curvature of a road trip (see [35]). The maximum squared speed along the path was fixed equal to 192.93 m^2s^{-2} (corresponding to a maximum speed of 50 kmh^{-1}, a typical value for an urban driving scenario). The total length of the paths was fixed to $s_f = 1000$ m, while parameter A was set equal to 0.25 ms^{-2}, J to 0.025 ms^{-3}, and A_N to 4.9 ms^{-2}.

The results are reported in Table I, in which we show the average (minimum and maximum) computational times for SCA-H, SCA-G, and IPOPT. They show that algorithm SCA-H is the fastest one, while SCA-G is slightly faster than IPOPT at $n = 100$ but clearly faster for a larger number of sample points n. In general, we observe that both SCA-H and SCA-G tend to outperform IPOPT as n increases. Moreover, while the computing times for IPOPT at $n = 100$ are not much worse than those of SCA-H and SCA-G, we should point out that, at this dimension, IPOPT is sometimes unable to converge and return solutions whose objective function value differs from the best one by more than 100%. Also, the objective function values returned by SCA-H and SCA-G are sometimes slightly different, due to numerical issues related to the choice of the tolerance parameters, but such differences are mild ones and never exceed 1%. Therefore, these approaches appear to be fast and robust. It is also worthwhile to remark that SCA approaches are compatible with online planning requirements within the context of the LGV application. According to Haschke *et al.* [18] (see also [36]), in “highly unstructured, unpredictable, and dynamic environments,” there is a need to replan in order to adapt the motion in reaction to unforeseen events or obstacles. How often to replan depends strictly on the application. Within the context of the LGV application (where the environment is structured), replanning every 100–150 ms

TABLE I

AVERAGE (MINIMUM AND MAXIMUM) COMPUTING TIMES (IN SECONDS) FOR SCA-H, SCA-G, AND IPOPT OVER EXPERIMENTS 1 AND 2

Exp.	n		SCA-H	SCA-G	IPOPT
		min	0.012	0.042	0.03
1	100	mean	0.016	0.072	0.132
		max	0.026	0.138	0.305
		min	0.042	0.21	0.352
1	500	mean	0.064	0.276	1.01
		max	0.104	0.456	1.828
		min	0.1	0.426	1.432
1	1000	mean	0.149	0.626	3.289
		max	0.237	0.828	7.137
		min	0.012	0.036	0.052
2	100	mean	0.02	0.047	0.113
		max	0.038	0.073	0.263
		min	0.049	0.102	0.534
2	500	mean	0.093	0.172	0.886
		max	0.212	0.237	1.457
		min	0.083	0.228	1.733
2	1000	mean	0.242	0.386	2.487
		max	0.709	0.539	3.74

is acceptable, and thus, the computing times of the SCA approaches at $n = 100$ are suitable. Of course, computing times increase with n, but we notice that the computing times of SCA-H still meet the requirement at $n = 500$. Moreover, a relevant feature of SCA-H and SCA-G is that, at each iteration, a feasible solution is available. Thus, we could stop them as soon as a time limit is reached. At $n = 500$, if we impose a time limit of 150 ms, which is still quite reasonable for the application, SCA-G returns slightly worse feasible solutions, but these do not differ from the best ones by more than 2%.

B. Experiment 3

In our third experiment, we compared the performance of the two proposed approaches (SCA-H and SCA-G), over two possible automated driving scenarios, as the number n of samples increases. As a first example, we considered a continuous curvature path composed of a line segment, a clothoid, a circle arc, a clothoid, and a final line segment (see Fig. 4). The minimum-time velocity planning on this path, whose total length is $s_f = 90$ m, is addressed with the following data. The problem constants are compatible with a typical urban driving scenario. The maximum squared velocity is 225 $\mathrm{m}^2\mathrm{s}^{-2}$ (corresponding to 54 km h^{-1}), the longitudinal acceleration limit is $A = 1.5$ ms^{-2}, and the maximal normal acceleration is $A_N = 1$ ms^{-2}, while, for the jerk constraints, we set $J = 1$ ms^{-3}. Next, we considered a path of length $s_f = 60$ m (see Fig. 5) whose curvature was defined according to the following function:

$$k(s) = \frac{1}{5}\sin\left(\frac{s}{10}\right), \quad s \in [0, s_f]$$

and parameter A, A_N, and J were set equal to 1.39 ms^{-2}, 4.9 ms^{-2}, and 0.5 ms^{-3}, respectively. The maximum squared velocity is still equal to 225 $\mathrm{m}^2\mathrm{s}^{-2}$. The computational results are reported in Figs. 6 and 7 for values of n that grows from 100 to 1000. They show that the performance of SCA-H and SCA-G depends on the path. In particular, it seems that the heuristic performs in a poorer way when the number of points of the upper bound vector at which PAR constraints are violated tends to be large, which is the case for the second instance. We can give two possible motivations: 1) the directions computed by the heuristic procedure are not necessarily good descent directions, so routine `computeUpdate` slowly converges to a solution and 2) the heuristic procedure often fails, and it is in any case necessary to call GUROBI. Note that the computing times of IPOPT on these two paths are larger than those of SCA-H and SCA-G, and, as usual, the gap increases with n. Moreover, for the second path, IPOPT was unable to converge for $n = 100$ and returned a solution, which differed by more than 35% with respect to those returned by SCA-H and SCA-G.

As a final remark, we notice that the computed traveling times along the paths only slightly vary with n. For the first path, they vary between 14.44 and 14.45 s while, for the second path, between 20.65 and 20.66 s. The differences are very mild, but we should point out that this is not always

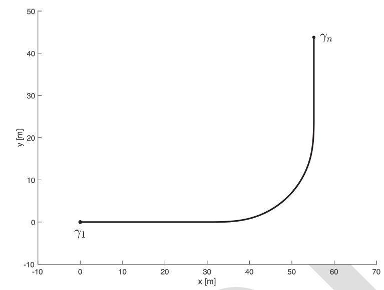

Fig. 4. Experiment 3—first path.

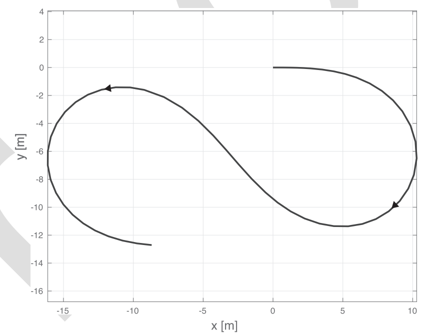

Fig. 5. Experiment 3—second path.

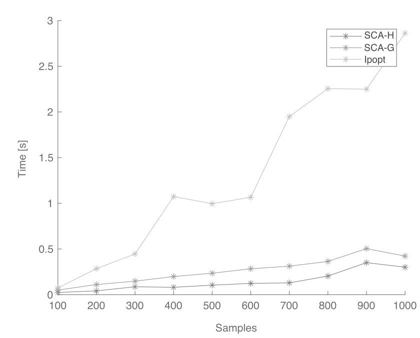

Fig. 6. Computing times (in seconds) for the path in Fig. 4.

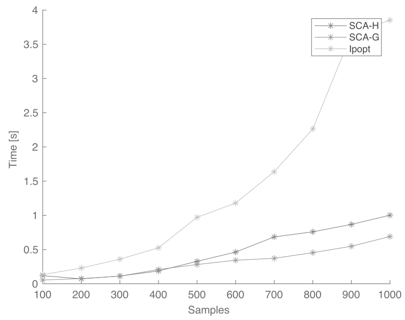

Fig. 7. Computing times (in seconds) for the path in Fig. 5.

TABLE II

MINIMUM, AVERAGE, AND MAXIMUM COMPUTING TIMES (IN SECONDS) AND RELATIVE PERCENTAGE DIFFERENCE BETWEEN THE TRAVELING TIMES COMPUTED BY THE HEURISTIC PRESENTED IN [23] AND THE SCA APPROACHES WITH $n = 100$ FOR THE INSTANCES OF EXPERIMENT 1

Heuristic from [23]	min	mean	max
Time	0.016	0.048	0.2049
Relative percentage difference	5.5%	12.1%	31.2%

the case. We further comment on this point when presenting Experiment 5.

C. Experiment 4

In this experiment, we compared the performance of our approach with the heuristic procedure recently proposed in [23]. In Table II, we report the computing times and the relative percentage difference $[(f_{\text{HEUR}} - f_{\text{SCA}})/f_{\text{SCA}}] * 100\%$ between the traveling times computed by the heuristic and the SCA approaches for the instances of Experiment 1 with $n = 100$. Algorithms SCA-H and SCA-G have comparable computing times (actually, better for what concerns SCA-H) with respect to that heuristic, and the quality of the final solutions is, on average, larger than 10% (these observations also extend to other experiments). Such difference between the quality of the solutions returned by algorithm SCA and those returned by the heuristic is best explained through the discussion of a representative instance, taken from Experiment 1 with $n = 100$. In this instance, we set $A = 2.78$ ms^{-2}, while, for the jerk constraints, we set $J = 2$ ms^{-3}. The total length of the path is $s_f = 60$ m. The maximum velocity profile is the piecewise constant black line in Fig. 8. In the same figure, we report in red the velocity profile returned by the heuristic and in blue the one returned by algorithm SCA. The computing time for the heuristic is 45 ms, while, for algorithm SCA-H, it is 39 ms. The final objective function value (i.e., the traveling time along the given path) is 15.4 s for the velocity profile returned by the heuristic and 14.02 s for the velocity profile returned by algorithm SCA. From the qualitative point of view, it can be observed in this instance (and similar observations hold for the other instances that we tested) that the heuristic produces velocity profiles whose local minima coincide with those of the maximum velocity profile. For instance, in the interval between 10 and 20 m, we notice that the velocity profile returned by the heuristic coincides

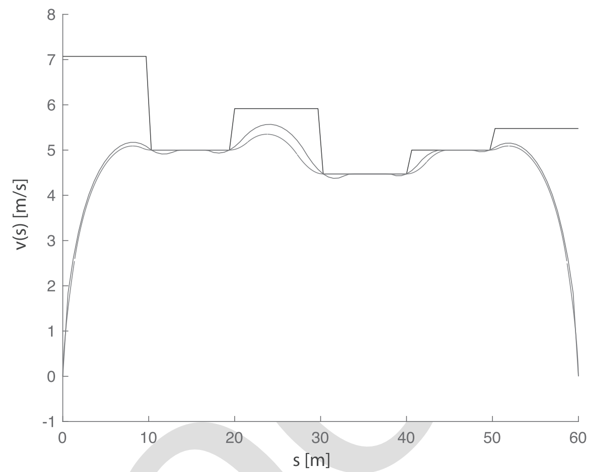

Fig. 8. Velocity profile returned by the heuristic proposed in [23] (red line) and by algorithm SCA (blue line). The black line is the maximum velocity profile.

with the maximum velocity profile in that interval. Instead, the velocity profile generated by algorithm SCA generates velocity profiles that fall below the local minima of the maximum velocity profile, but, this way, they are able to keep the velocity higher in the regions preceding and following the local minima of the maximum velocity profile. Again, referring to the interval between 10 and 20 m, we notice that the velocity profile computed by algorithm SCA falls below the maximum velocity profile in that region and, thus, below the velocities returned by the heuristic, but, this way, velocities in the region before 10 m and in the one after 20 m are larger with respect to those computed by the heuristic.

D. Experiment 5

As a final experiment, we planned the speed law of an autonomous guided vehicle operating in a real-life automated warehouse. Paths and problem data have been provided by packaging company Ocme S.r.l., based in Parma, Italy. We generated 50 random paths from a general layout. Fig. 9 shows the warehouse layout and a possible path. In all paths, we set maximum velocity to 2 m s^{-1}, maximum longitudinal acceleration to $A = 0.28$ m/s^2, maximum normal acceleration to 0.2 m/s^2, and maximum jerk to $J = 0.025$ m/s^3. Table III shows computation times for algorithms SCA-H, SCA-G, and IPOPT for a number of sampling points $n \in \{100, 500, 1000\}$. SCA-H is quite fast although it sometimes returns slightly worse solutions (the largest percentage error, at a single instance with $n = 1000$, is 8%). IPOPT is clearly slower than SCA-H and SCA-G for $n = 500$ and 1000, while, for $n = 100$, it is slower than SCA-H but quite similar to SCA-G. However, for these paths, the difference in terms of traveling times as n increases is much more significant with respect to the other experiments (see also the discussion at the end of Experiment 3). More precisely, the percentage difference between the traveling times of solutions at $n = 100$ and $n = 1000$ is 0.5% on average for Experiment 1 with a maximum of 2.1%, while, for Experiment 2, the average difference is 0.3% with a maximum of 0.4%. Instead, for the current experiment,

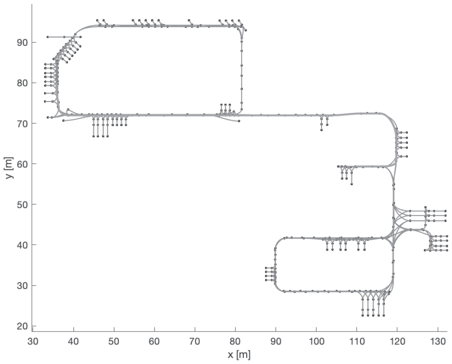

Fig. 9. Warehouse layout considered in Example 5 and a possible path.

TABLE III
AVERAGE, MINIMUM, AND MAXIMUM COMPUTING TIMES (IN SECONDS) FOR SCA-H, SCA-G, AND IPOPT OVER EXPERIMENT 5

n		SCA-H	SCA-G	IPOPT
100	min	0.009	0.033	0.029
	mean	0.013	0.043	0.037
	max	0.026	0.062	0.052
500	min	0.032	0.104	0.222
	mean	0.068	0.146	0.289
	max	0.174	0.224	0.423
1000	min	0.078	0.249	0.744
	mean	0.201	0.385	1.25
	max	0.501	0.65	3.359

the average difference is 2.7% with a maximum of 7.9%. However, the average falls to 0.2% and the maximum to 0.6% if we consider the percentage difference between the traveling times of solutions at $n = 500$ and $n = 1000$. Thus, for this experiment, it is advisable to use a finer discretization or, equivalently, a larger number of sampling points. A tentative explanation for such different behavior is related to the lower velocity limits of Experiment 5 with respect to the other experiments. Indeed, the objective function is much more sensitive to small changes at low speeds so that a finer grid of sampling points is able to reduce the impact of approximation errors. However, this is just a possible explanation. A further possible explanation is that, in Experiments 1–4, curves are composed of segments with constant and linear curvature, whereas curves on industrial LGV layouts typically have curvatures that are highly nonlinear with respect to arc length.

VI. CONCLUSION

In this article, we considered a speed planning problem under jerk constraints. The problem is a nonconvex one, and we proposed a sequential convex approach, where we exploited the special structure of the convex subproblems to solve them very efficiently. The approach is fast and is theoretically guaranteed to converge to a stationary point of the nonconvex problem. As a possible topic for future research, we would like to investigate ways to solve Problem 9, currently the bottleneck of the proposed approach, alternative to the solver GUROBI, and the heuristic mentioned in Remark 6. Moreover, we suspect that the stationary point to which the proposed approach converges is, in fact, a global minimizer of the nonconvex problem, and proving this fact is a further interesting topic for future research.

ACKNOWLEDGMENT

The authors are really grateful to the Associate Editor and the three anonymous reviewers for their careful reading and very useful suggestions.

REFERENCES

[1] P. Pharpatara, B. Hérissé, and Y. Bestaoui, “3-D trajectory planning of aerial vehicles using RRT*,” *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 1116–1123, May 2017.

[2] K. Kant and S. W. Zucker, “Toward efficient trajectory planning: The path-velocity decomposition,” *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, Sep. 1986.

[3] L. Consolini, M. Locatelli, A. Minari, and A. Piazzi, “An optimal complexity algorithm for minimum-time velocity planning,” *Syst. Control Lett.*, vol. 103, pp. 50–57, May 2017.

[4] F. Cabassi, L. Consolini, and M. Locatelli, “Time-optimal velocity planning by a bound-tightening technique,” *Comput. Optim. Appl.*, vol. 70, no. 1, pp. 61–90, May 2018.

[5] F. Pfeiffer and R. Johanni, “A concept for manipulator trajectory planning,” *IEEE J. Robot. Autom.*, vol. RA-3, no. 2, pp. 115–123, Apr. 1987.

[6] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, “Time-optimal path tracking for robots: A convex optimization approach,” *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.

[7] M. Yuan, Z. Chen, B. Yao, and X. Zhu, “Time optimal contouring control of industrial biaxial gantry: A highly efficient analytical solution of trajectory planning,” *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 1, pp. 247–257, Feb. 2017.

[8] E. Velenis and P. Tsiotras, “Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation,” *J. Optim. Theory Appl.*, vol. 138, no. 2, pp. 275–296, 2008.

[9] M. Frego, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, “Semi-analytical minimum time solutions with velocity constraints for trajectory following of vehicles,” *Automatica*, vol. 86, pp. 18–28, Dec. 2017.

[10] K. Hauser, “Fast interpolation and time-optimization with contact,” *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1231–1250, 2014.

[11] H. Pham and Q.-C. Pham, “A new approach to time-optimal path parameterization based on reachability analysis,” *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 645–659, Jun. 2018.

[12] L. Consolini, M. Locatelli, A. Minari, A. Nagy, and I. Vajk, “Optimal time-complexity speed planning for robot manipulators,” *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 790–797, Jun. 2019.

[13] T. Lipp and S. Boyd, “Minimum-time speed optimisation over a fixed path,” *Int. J. Control*, vol. 87, no. 6, pp. 1297–1311, 2014.

[14] F. Debrouwere *et al.*, “Time-optimal path following for robots with convex–concave constraints using sequential convex programming,” *IEEE Trans. Robot.*, vol. 29, no. 6, pp. 1485–1495, Dec. 2013.

[15] A. K. Singh and K. M. Krishna, “A class of non-linear time scaling functions for smooth time optimal control along specified paths,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 5809–5816.

[16] A. Palleschi, M. Garabini, D. Caporale, and L. Pallottino, “Time-optimal path tracking for jerk controlled robots,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3932–3939, Oct. 2019.

[17] S. Macfarlane and E. A. Croft, “Jerk-bounded manipulator trajectory planning: Design for real-time applications,” *IEEE Trans. Robotics Autom.*, vol. 19, no. 1, pp. 42–52, Feb. 2003.

[18] R. Haschke, E. Weitnauer, and H. Ritter, “On-line planning of time-optimal, jerk-limited trajectories,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 3248–3253.

[19] H. Pham and Q.-C. Pham, “On the structure of the time-optimal path parameterization problem with third-order constraints,” in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 679–686.

[20] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, “Time-optimal control of robotic manipulators along specified paths,” *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, Sep. 1985.

[21] K. G. Shin and N. D. McKay, “Minimum-time control of robotic manipulators with geometric path constraints,” *IEEE Trans. Autom. Control*, vol. AC-30, no. 6, pp. 531–541, Jun. 1985.

[22] J. Villagra, V. Milanés, J. Pérez, and J. Godoy, "Smooth path and speed planning for an automated public transport vehicle," *Robot. Auton. Syst.*, vol. 60, no. 2, pp. 252–265, 2012.

[23] M. Raineri and C. G. L. Bianco, "Jerk limited planner for real-time applications requiring variable velocity bounds," in *Proc. IEEE Int. Conf. Automat. Sci. Eng. (CASE)*, Aug. 2019, pp. 1611–1617.

[24] J. Dong, P. M. Ferreira, and J. A. Stori, "Feed-rate optimization with jerk constraints for generating minimum-time trajectories," *Int. J. Mach. Tools Manuf.*, vol. 47, nos. 12–13, pp. 1941–1955, Oct. 2007.

[25] K. Zhang, X. S. Gao, H. B. Li, and C. M. Yuan, "A greedy algorithm for feedrate planning of CNC machines along curved tool paths with confined jerk," *Robot. Comput.-Integr. Manuf.*, vol. 28, no. 4, pp. 472–483, Aug. 2012.

[26] Y. Zhang *et al.*, "Speed planning for autonomous driving via convex optimization," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 1089–1094.

[27] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Singapore: Wiley, 2000.

[28] L. Consolini, M. Laurini, and M. Locatelli, "Graph-based algorithms for the efficient solution of optimization problems involving monotone functions," *Comput. Optim. Appl.*, vol. 73, no. 1, pp. 101–128, May 2019.

[29] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, vol. 80. Philadelphia, PA, USA: SIAM, 2002.

[30] L. Consolini, M. Locatelli, and A. Minari, "A sequential approach for speed planning under jerk constraints," 2021, *arXiv:2105.15095*. [Online]. Available: http://arxiv.org/abs/2105.15095

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[32] Gurobi Optimization Inc. (2016). *Gurobi Optimizer Reference Manual*. [Online]. Available: http://www.gurobi.com

[33] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, May 2006.

[34] L. L. Hoberock, "A survey of longitudinal acceleration comfort studies in ground transportation vehicles," *J. Dyn. Syst., Meas., Control*, vol. 99, no. 2, pp. 76–84, Jun. 1977.

[35] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 1025–1035, Dec. 2004.

[36] D. Verscheure, M. Diehl, J. De Schutter, and J. Swevers, "Recursive log-barrier method for on-line time-optimal robot path tracking," in *Proc. Amer. Control Conf.*, St. Louis, MO, USA, Apr. 2009, pp. 4134–4140.

Luca Consolini (Member, IEEE) received the Laurea degree *(cum laude)* in electronic engineering and the Ph.D. degree from the University of Parma, Parma, Italy, in 2000 and 2005, respectively.

From 2005 to 2009, he held a post-doctoral position at the University of Parma, where he was an Assistant Professor from 2009 to 2014. Since 2014, he has been an Associate Professor with the University of Parma. His main current research interests are nonlinear control, motion planning, and control of mechanical systems.

Marco Locatelli is currently a Full Professor of Operations Research with the University of Parma, Parma, Italy. He published more than 90 articles in international journals and coauthored, with F. Schoen, the book *Global Optimization: Theory, Algorithms, and Applications* [Society for Industrial and Applied Mathematics (SIAM)]. His main research interests are the theoretical, practical, and applicative aspects of optimization.

Dr. Locatelli has been nominated as a EUROPT Fellow in 2018. He is also on the Editorial Board of the journals *Computational Optimization and Applications, Journal of Global Optimization*, and *Operations Research Forum*.

Andrea Minari received the B.S. and M.S. degrees *(cum laude)* in computer engineering and the Ph.D. degree from the University of Parma, Parma, Italy, in 2013, 2016, and 2020, respectively. He presented a dissertation about optimization-based algorithms applied to the speed planning problem for mobile robots and industrial manipulators.