A Weighted Method for Fast Resolution of Strictly Hierarchical Robot Task Specifications using Exact Penalty Functions

Ajay Suresha Sathya[1], Goele Pipeleers[1], Wilm Decré[1] and Jan Swevers[1]

***Abstract*—Extensive work has been done on efficiently resolving hierarchical robot task specifications that minimize the ℓ-2 norm of linear constraint violations, but not for ℓ-1 norm, in which there has recently been growing interest for sparse control. Both approaches require solving a cascade of quadratic programs (QP) or linear programs (LP). In this letter, we introduce alternate and more efficient approaches to hierarchical ℓ-1 norm minimization by formulating it as a *single* LP that can be solved by any off-the-shelf solver. The first approach is a recursive method that transforms the lexicographic LP (LLP) into a single objective problem using Lagrangian duality. The second approach, which forms the main focus of this letter, is a weighted method based on the exact penalty method, that is equivalent to the original LLP for a well chosen set of weights. We propose methods to compute and adapt these weights. The algorithms were applied on an interesting dual arm robot task. We discuss and benchmark the computational efficiency of these methods. Simplicity of the weighted method makes it a promising approach for tackling challenging prioritized robot control problems involving a control horizon or nonlinear constraints. Within this letter, we take the first step towards that goal by demonstrating the efficacy of the weighted method on a simpler instantaneous robot control problem with linear constraints.**

***Index Terms*—Optimization and Optimal Control; Redundant Robots; Whole-Body Motion Planning and Control**

I. Introduction

CONSTRAINT-BASED task specifications are a popular way to specify continuous robot motion and estimation tasks [1], [2]. They are composable and allow a programmer to specify a set of tasks to be achieved simultaneously to fully utilize any redundant degrees of freedom present in a robot. Various algorithms have been developed over the years to resolve these tasks into the desired robot control input space of either the joint velocities or accelerations [3], [4], [5]. These different task objectives may conflict with each other and demand a strategy for conflict resolution. A common approach is to assign strict priorities, when achieving a task with higher priority is more important than a task of lower priority and a trade-off is either not desirable or difficult to model. For example, maintaining balance of a humanoid robot is more important than collision avoidance of an arm which could in turn be more important than picking up a glass of water. Therefore, an algorithm that can resolve such a hierarchical task specification is highly desirable and has received a considerable amount of attention.

Resolving equality constrained tasks is straight-forward using the Pseudo-inverse and projection onto the nullspace of higher priority tasks [6]. This becomes significantly more complex in the presence of inequality constraints and requires an iterative solver because it is not always possible to predict the optimal active-set of the constraints. There are several approaches in the literature dedicated to solving this problem.

The simplest approach is to have only two priority levels with hard constraints and weighted soft constraints that are quadratic penalties of constraint violations [5]. It results in a single quadratic program (QP) which can be efficiently solved by leveraging mature open-source or commercial QP solvers.The more complex sequential method [7] supports an arbitrary number of priority levels, by solving a cascade of QPs at every priority level. This approach can get computationally expensive when there are several layers of priority and many inequality constraints. A state-of-the-art method for this approach that addresses this computational issue is the hierachical quadratic programming (HQP) algorithm [8]. Here, the inequality constraints are solved using an active-set algorithm implemented on top of a highly efficent stack-of-tasks implementation for equality constrained problems using hierarchical complete orthogonal decomposition (HCOD). It is found to be computationally efficient in practice when appropriately warm started.

There has been a recent interest in using sparsity-inducing ℓ-1 norm penalties [9], [10], [11], [12] because of its tendency to produce parsimonious control policies where, only a subset of the joints are actuated. This is found to result in more intuitive motions [12]. This is in constrast to the QP approaches described in the previous paragraph that minimize the squared ℓ-2 penalties which typically actuates a large number of the joints. In [9] and [10], the ℓ-1 penalty on the constraint violations is formulated in a specific way that leads to sparse control actions when the resulting linear program (LP) is solved using a particular simplex solver. However, it

Manuscript received: October, 15, 2020; Revised January, 24, 2021; Accepted February, 11, 2021.

This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Flanders Make through the SBO project - MULTIROB. Flanders Make is the Flemish strategic research centre for the manufacturing industry.

[1]Ajay Suresha Sathya, Goele Pipeleers, Wilm Decré and Jan Swevers are with the Division Robotics, Automation and Mechatronics, Department of Mechanical Engineering, KU Leuven, C300 BE-3001, Belgium and also with DMMS Lab, Flanders Make, Leuven, Belgium. email:ajay.sathya@kuleuven.be, goele.pipeleers@kuleuven.be, wilm.decre@kuleuven.be, jan.swevers@kuleuven.be

Digital Object Identifier (DOI): see top of this page.

supports only a single level of hierarchy. A more rigorous approach towards sparse control policies is taken in [11], [12] using Lasso regularization. The ℓ-1 penalty is applied only at the lowest priority level for regularization while quadratic objectives are minimized at higher levels using the traditional approach of solving a cascade of QPs. These sequential approaches also do not scale well with higher number of levels and there has been no method proposed in the literature that is similar in computational efficiency to the HQP method for ℓ-1 penalties.

The methods in the literature predominantly deal with instantaneous control problems with linear constraints. Computationally efficient methods like the HQP solver are complex and extending them to solve for prioritized objectives for model predictive control (MPC) or nonlinear constraints is conceptually challenging. Moreover, the HQP solver is efficient only when the optimal active-set does not change significantly between iterations. It is thus unlikely to scale well to the larger MPC problems where this assumption does not hold. HQP solver would be unable to switch to an interior point method (IPM), if IPM was more suitable for such an optimal control problem. In this letter, we explore a different strategy by minimizing the weighted ℓ-1 norm penalties on constraint violations that results in a simple and elegant formulation that is also computationally efficient. We first motivate the reason for this approach through an intuitive graphical example presented in the next subsection.

A. Motivating example

Consider a single dimensional problem with two conflicting equality constraints. Quadratic penalties and ℓ-1 norm penalties on the violation of each constraint are plotted in the fig. 1. The higher priority constraint is given a higher weight to reflect the preference. In the weighted least squares problem, the minimizer of the weighted sum is not the minimizer of the higher priority constraint. Even in this simple problem the weight on the higher priority constraint would have to be infinitely higher than lower priority weight for the combined objective function to have a minima that satisfies priorities strictly and *any* real choice of weights cannot avoid some degree of violation of priority. This is because the gradient of the quadratic penalty vanishes to zero at the minima. However, for a non-smooth penalty function such as the ℓ-1 norm, the minima of the combined objective function is clearly identical to the minima of the higher priority constraint even when the higher priority constraint has a slightly higher and finite weight. Thus, a constraint can be made to behave as a hard constraint by using a non-smooth penalty function with a high enough weight. This is also known as the exact penalty method in the optimization literature [13]. In fact, any p-norm can be used as the exact penalty. But ℓ-1 and ℓ-∞ norms remain a popular choice as they can be reformulated into smooth objectives with the addition of only linear constraints. Choosing ℓ-2 norm results in a second-order conic constraint during smooth reformulation which is harder to solve than a linear constraint. In this letter we investigate the feasibility of extending this idea to multiple priority levels by weighting

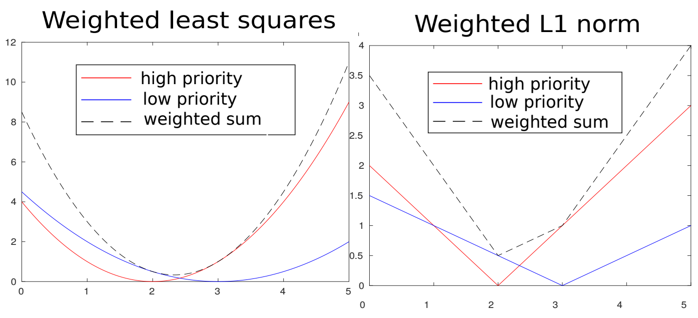


Fig. 1: Comparison of weighted least squares and weighted ℓ-1 norm penalties.

the exact penalty functions high enough that the corresponding constraints effectively behave like hard constraints with respect to the lower priority constraints.

B. Approach and contributions

We present the first work that introduces and successfully demonstrates the application of weighted ℓ-1 norm penalties to resolving hierarchical robot task specification in the robotics literature, to the best of our knowledge. This approach retains the simplicity of weighted least-squares method and can be implemented in any existing control framework within hours. For some problems such as hierarchical parsimonious control, our approach is the only method we are aware of, that does not need to solve more than one LP/QP at every control instance. We contribute primarily towards the modelling of the optimization problem and therefore retain the freedom to choose an optimization solver that is suited for a task. Thus, it can leverage future advances in LP/QP solvers.

There exist classical results in multi-objective optimization literature that use the solution of the sequential method to prove the existence of a set of equivalent weights for the ℓ-1 norm penalties [14], [15]. Though the proof is constructive in nature, it does not provide a recipe to compute them in practice as it can lead to numerically large weights. Therefore we design a tunable parameter for computing the weights that can be either manually tuned or automatically adapted. We propose an efficient procedure to confirm lexicographic optimality at a given level, that is central for adapting the weights automatically. We evaluate the accuracy of this method on a set of challenging tasks with rank-deficient constraints. We also discuss smoothness issues that are critical for implementating the algorithm on a real robot.

Additionally, we introduce an algorithm to formulate the lexicographic problem as a single linear program using the theoretical properties of Lagrangian duality and assess its computational performance. As our approach is based on the exact penalty method which works well for nonlinear programs as well, it has the potential to be generalized to solve prioritized optimal control or MPC problems more efficiently [16] than the existing sequential methods.

The source code for all the algorithms presented in this work is made available publicly at the following link to improve the reproducibility of work presented in this letter: https://github.com/AjSat/hqp_l1

II. Problem Formulation

The lexicographic problem formulation with the ℓ-1 norm penalty objectives at each level is mathematically formalized in this section. Three different equivalent single objective optimization formulations are introduced, the sequential method, a formulation based on Lagrangian duality and the weighted method, which forms the main the focus of this paper.

A. Lexicographic problem formulation

The robot task function approach [1] contributes to time-dependent affine equality and inequality constraints that must be solved as a lexicographic optimization problem. We follow a notation similar to [8], but have replaced the quadratic objective with the ℓ-1 penalty at each level. The problem is formulated as follows:

$$\underset{x,w_1,w_{e1},..,w_p,w_{ep}}{\textbf{lex min}} \quad \{\| \begin{bmatrix} w_1 & w_{e1} \end{bmatrix}^T \|_1, ..., \| \begin{bmatrix} w_p & w_{ep} \end{bmatrix}^T \|_1\}, \tag{1a}$$

$$\textbf{subject to} \quad A_{e0}(t)x = b_0(t), \ \ A_0(t)x \le u_0(t) \tag{1b}$$

$$A_{ek}(t)x = b_k(t) + w_{ek}, \tag{1c}$$

$$A_k(t)x \le u_k(t) + w_k, \ \ k = 1, 2, ..., p$$

where $x \in \mathbb{R}^n$ is the optimization variable which is normally chosen as joint-space velocities or accelerations. $A_k(t) \in \mathbb{R}^{m_k \times n}$ and $u_k(t) \in \mathbb{R}^{m_k}$ define m_k number of linear inequality constraints at the kth level. These matrices are obtained by evaluating and computing the Jacobian of the task function at the kth priority level at time t. For $k \ge 1$, the constraints are relaxed through slack variables $w_k \in \mathbb{R}^{m_k}$ at each level. p is the total number of relaxable priority levels. Equality constraints and their relaxations are similarly defined by the terms $A_{ek}(t)$, $b_k(t)$ and w_{ek}. The optimal solution to the lexicographic optimization problem defined by eq. (1), minimizes $\| \begin{bmatrix} w_k & w_{ek} \end{bmatrix}^T \|_1$ for all kth levels such that $\| \begin{bmatrix} w_q & w_{eq} \end{bmatrix}^T \|_1$ can be reduced no further without increasing the objective at a higher priority level $\| \begin{bmatrix} w_r & w_{er} \end{bmatrix}^T \|_1$, where $r < q$.

1) Dropping terms for simplicity of notation: In the subsequent analysis, we omit the equality constraints and the time dependency to simplify the notation. The following derivations can be easily extended to the equality constraints as well.

2) Normalization of the constraints: We assume without loss of generality that each row vector of $A_k(t)$ in eq. (1c) has a unit norm. Any arbitrary affine inequality constraint can be transformed into this form by multiplying each row of unnormalized A_k and u_k with the appropriate factor. While the normalization step is not essential for the algorithm, it allows the objective at kth level $\|w_k^*\|_1$ to be interpreted as the sum of the distance of the solution point x^* from all the constraint manifolds (halfspaces for inequalities) at the kth priority level.

3) Smooth reformulation: The ℓ-1 penalty functions in eq. (1a) are non-smooth and could cause convergence issues for many solvers. The common approach [17] to resolve this issue is to reformulate into an equivalent problem by adding non-negativity constraints on the slack vectors. The resulting hierarchical linear program (HLP) is given below:

$$\underset{x,w_1,w_2,..w_p}{\textbf{lex min}} \quad \{\mathbf{1}_1^T w_1, \mathbf{1}_2^T w_2, \mathbf{1}_3^T w_3 ... \mathbf{1}_p^T w_p\}, \tag{2a}$$

$$\textbf{subject to} \quad A_0 x \le u_0, \tag{2b}$$

$$A_k x \le u_k + w_k, \ \ w_k \ge 0,$$

$$k = 1, 2, ..., p \tag{2c}$$

where $\mathbf{1}_k \in \mathbb{R}^{m_k}$ refers to a column vector with each element equal to 1.

B. Sequential method

A natural approach to solve the HLP is to solve an LP at each level in a decreasing order of priority similar to the sequential QP approach[7]. While solving the LP at the ith level, all the constraints and slack vectors from lower priority levels $j > i$ are ignored. The optimal objective $\mathbf{1}_k^T w_k^*$ for a higher priority level $k < i$ is already computed in a previous step and is added as a hard constraint to ensure that a higher priority objective does not become worse while solving the lower levels. Let us call this method sequential linear program (SLP). The ith optimization problem in SLP is given below:

$$\underset{x,w_1..w_i}{\textbf{minimize}} \quad \mathbf{1}_i^T w_i \tag{3a}$$

$$\textbf{subject to} \quad A_0 x \le u_0 \tag{3b}$$

$$\mathbf{1}_k^T w_k \le \mathbf{1}_k^T w_k^* \quad k = 1, 2, ..., i-1 \tag{3c}$$

$$A_i x \le u_i + w_i, \ \ w_k \ge 0, \ \ k = 1, 2, ..., i \tag{3d}$$

C. Formulation using Lagrangian Duality

It is a common approach to use Lagrangian duality to efficiently solve bilevel optimization problems such as min-max problems. In this section, we introduce and extend this idea to an arbitrary number of levels to formulate the HLP in eq. (2) as a single-objective LP. Consider the first optimization problem in SLP, where only the first relaxable priority level is considered. The corresponding dual problem [17] is given below.

$$\underset{\lambda_{1,0},\lambda_{1,1}}{\textbf{maximize}} \quad -\lambda_{1,0}^T u_0 - \lambda_{1,1}^T u_1 \tag{4a}$$

$$\textbf{subject to} \quad A_0^T \lambda_{1,0} + A_1^T \lambda_{1,1} = 0, \tag{4b}$$

$$0 \le \lambda_{1,1} \le \mathbf{1}_i, \quad 0 \le \lambda_{1,0} \tag{4c}$$

Let the objective functions of the dual problem be denoted as $\mathbf{d}_1(\lambda_{1,0}, \lambda_{1,1})$. According to duality theory, the dual function is always a lower bound of the primal function at any primal feasible point. For an LP strong duality always holds, which implies the following relations:

$$\mathbf{d}_1(\lambda_{1,0}, \lambda_{1,1}) \le \mathbf{d}_1(\lambda_{1,0}^*, \lambda_{1,1}^*) = \mathbf{1}_1^T w_1^* \le \mathbf{1}_1^T w_1 \tag{5}$$

Now, considering the second optimization problem of SLP that solves for the second level of relaxable constraints, we replace $\mathbf{1}_1^T w_1^*$ with $\mathbf{d}_1(\lambda_{1,0}, \lambda_{1,1})$ in the constraint eq. (3c) of the problem and add the constraints of the dual problem. This new problem would take the form below.

$$\begin{aligned}
\underset{x,w_1,w_2,\lambda_{1,0},\lambda_{1,1}}{\textbf{minimize}} \quad & \mathbf{1}_2^T w_2 && \text{(6a)}\\
\textbf{subject to} \quad & A_0 x \leq u_0, && \text{(6b)}\\
& A_1 x \leq u_1 + w_1, \;\; A_2 x \leq u_2 + w_2, && \text{(6c)}\\
& \mathbf{1}_1^T w_1 \leq -\lambda_{1,0}^T u_0 - \lambda_{1,1}^T u_1 && \text{(6d)}\\
& A_0^T \lambda_{1,0} + A_1^T \lambda_{1,1} = 0, && \text{(6e)}\\
& w_1, w_2, \lambda_{1,0} \geq 0 \quad 0 \leq \lambda_{1,1} \leq \mathbf{1}_1 && \text{(6f)}
\end{aligned}$$

There are two important observations to be made in the above problem:

First, duality "trick" is employed in the constraint in eq. (6d) by enforcing the reverse of the inequality in eq. (5). The only feasible solution to this constraint is where the duality gap zero and $\mathbf{1}_1^T w_1$ is equal to the optima of the first SLP step. So solving this problem would also compute the higher priority objective $\mathbf{1}_1^T w_1$ while computing the optimal $\mathbf{1}_2^T w_2$.

Second, the dual variables from the first SLP problem are part of the primal variables now and that the new problem eq. (6) is still an LP. This implies that the dual of this problem can be computed and a similar duality trick can be enforced on the next level. The process can be continued recursively till the last step of the sequential method and all the constraints in eq. (3c) are reformulated with the corresponding dual functions as the upper bound. This results in a larger, but a single optimization problem with a lot of sparsity that can exploited by state-of-the-art LP solvers.

It is cumbersome to derive the optimization problems by hand for even three priority levels. Therefore, we implemented this algorithm using an automatic dualizer [18] provided by the YALMIP toolbox. The efficacy of this method is investigated in the section V. Though we do not pursue this direction in this letter, we additionally note that this idea is not restricted to ℓ-1 penalty. When applied to ℓ-2 penalties, depending on whether the ℓ-2 norm or the squared ℓ-2 norm is used as the objective, the duality trick would lead to either a second order conic program (SOCP) or a quadratically-constrained quadratic program (QCQP) formulation of the lexicographic problem.

D. Weighted Formulation

We now introduce the weighted formulation for solving the HLP in eq. (2) as follows

$$\begin{aligned}
\underset{x,w_1,w_2,..w_p}{\textbf{minimize}} \quad & \epsilon_1 \mathbf{1}_1^T w_1 + \epsilon_2 \mathbf{1}_2^T w_2 + ... + \epsilon_p \mathbf{1}_p^T w_p && \text{(7a)}\\
\textbf{subject to} \quad & A_0 x \leq u_0, \quad A_k x \leq u_k + w_k, && \text{(7b)}\\
& w_k \geq 0, \quad k = 1, 2, ..., p && \text{(7c)}
\end{aligned}$$

$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 & \epsilon_2, & ... & \epsilon_p \end{bmatrix}^T, \boldsymbol{\epsilon} \in \mathbb{R}_+^p$ is the vector of the weights for each priority level whose numerical values determine whether this formulation returns a lexicographically optimal solution. [14], [15] show that there exists a finite number $M \geq 1$ such that if the weights are chosen to satisfy $\epsilon_{i-1} \geq M\epsilon_i$, the weighted method returns a lexicographically optimal solution. But M depends on the problem at hand and cannot be fixed in advance as it is always possible to derive pathological cases where the weighted method would fail for any given value of M. In the next section, we present some intuitive procedures to compute and adapt these weights vector $\boldsymbol{\epsilon}$. The weighted linear program (WLP) will be referred to as WLP from now.

III. Computation of weights for WLP

A. Basic heuristic

We first present a simple heuristic that computes weights ϵ such that the weighted gradient of a constraint at a level has a greater ℓ-2 norm than the cumulative sum of the norm of the weighted gradients of all the lower priority levels. Then all the lower priority constraints combined would, by themselves, not be able to cause the violation of the constraint at higher level.

$$\epsilon_i \geq \sum_{k=i+1}^{p} \epsilon_k . \sum_{j=0}^{m_k} \|A_{kj}\| \tag{8}$$

where A_{kj} is the jth row of the matrix A_k. Because we assumed that each row of A_k has a unit norm in the subsection II-A, the equation above to simplifies to the formula below:

$$\epsilon_i = \gamma_i \sum_{k=i+1}^{p} \epsilon_k . m_k \tag{9}$$

Where $\gamma_i \geq 1$ is a tunable constant that can be intuitively understood as the relative weight on the ith level. Higher the value of γ_i, higher is the weight on the constraints at ith level relative to all the constraints at lower levels. The analysis behind this heuristic ignores all the constraints at a higher or the same priority level. Lower priority constraints can cause violation of a given constraint with the help of a higher priority constraint. Thus, the weights computed using this tunable parameter are a heuristic and are not guaranteed to provide the correct solution. Effectively, the problem is now transformed from finding absolute weights to that of finding relative weights whose meaning is preserved even when the problem size and the number of constraints change.

B. Adaptive method

When there are few priority levels, it might be practically sufficient to simply select a high value of $\gamma_i, \; i = 1, ..., p$ to successfully compute the lexicographic solutions. But this idea is not viable for a high number of priority levels, as high relative weights would lead to numerical issues. To scale better with the number of levels and to avoid the issue of tuning, an adaptive method is desirable that can detect failure of lexicographic optimality and increase the corresponding relative weights accordingly.

When constraints are violated at ith level, one can check if they are caused by lower priority constraints by resetting all the weights of lower levels $\epsilon_k = 0,\ k = i+1,...,p$. The original solution is used to warm start and solve the new problem very efficiently. Often, when hierarchy is not violated, this simply amounts to the solver verifying optimality of the original solution despite the change in weights. If a better objective $\mathbf{1}_i^T w_i^*$ (without worsening a higher level objective) is found by the verification step, this implies a failure of lexicographic optimality had been caused by the lower levels. The weight ϵ_i is increased and weights $\epsilon_k,\ k = i+1,...,p$ are restored to their original values. The problem is solved again iteratively increasing ϵ_i till the constraint violation at ith level matches the value found when all lower priority constraints had zero weight. Because of the continuous nature of a robot control problem, this step is implemented only when the active set of constraints change or when there is a jump in constraint violations to confirm that the change was lexicographically optimal to save computations.

IV. Uniqueness and Smoothness of the Solution

The solution to the lexicographic problem is not unique when the constraints underdetermine the solution both for our ℓ-1 formulation as well the quadratic penalty formulation in HQP. This occurs when the total rank of all the active constraints is less than the number of decision variables. In this scenario the solver is free to choose any solution in the nullspace of all the constraints that does not violate the hard constraints. This can result in an undesirable drift in the joint velocities or accelerations. To prevent this, a Tikhonov regularization term is added in the HQP algorithm at the lowest priority level which makes the optimization problem at the final step strongly convex with a unique global minimizer. Following a similar approach, a natural option for regularization in WLP or SLP is to add a lowest priority equality constraint $x = 0$. This results in the sparsity-inducing ℓ-1 norm regularization at the lowest priority level similar to the approach in [12] and [11]. The resulting problems, which we call WLP-ℓ-1 and SLP-ℓ-1 still remain a linear program.

It is also possible to alter the objective function of the last level to implement quadratic regularization in our algorithm. To do so, the objective function of only the final problem in the sequential method in the eq. (3) is replaced with $\frac{1}{2}w_p^T Q w_p$ instead of $\mathbf{1}_p^T w_p$, where Q is a symmetric positive definite weighting matrix. It is normally chosen to be the identity matrix or the joint space inertia matrix if a desired physical quantity such as the kinetic energy is to be minimized over the nullspace of the active tasks. Similarly, the objective function of the weighted method in the eq. (7) is modified as $\epsilon_1 \mathbf{1}_1^T w_1 + \epsilon_2 \mathbf{1}_2^T w_2 + ... + \epsilon_{p-1}\mathbf{1}_{p-1}^T w_{p-1} + \frac{1}{2}\epsilon_p w_p^T Q w_p$ instead of $\epsilon_1 \mathbf{1}_1^T w_1 + \epsilon_2 \mathbf{1}_2^T w_2 + ... + \epsilon_p \mathbf{1}_p^T w_{p-1}$. This changes the final step of SLP and the whole WLP into strongly convex QPs with a unique minima. Please note that such a modification of SLP or WLP is allowed only at the last priority level because this objective is not required to be a non-smooth exact penalty function as there are no lower priority levels with respect to which the regularization needs to behave like a hard constraint. While this option makes the solution unique, it also has the disadvantage of solving a QP instead of an LP. Let this regularized problem be called SLP-ℓ-2 and WLP-ℓ-2 respectively.

With ℓ-1 regularization, the optimization problem does not always have a unique minimizer. This can occur when there exists a vector in the nullspace of all the active constraints, that is orthogonal to the subdifferential set of the ℓ-1 norm function at the minima as has been shown in an intuitive example in [7]. At this point, small perturbations in the parameters $A_i(t)$ and $b_i(t)$ can result in choosing a different vertex on the simplex of the LP even if the parameters $A_i(t)$ and $b_i(t)$ vary continuously with time. This issue is present in all approaches that use lasso regularization [12], [11]. One solution to mitigate this issue is to combine both the ℓ-1 and the quadratic regularizations at each level to get a strongly convex problem. The magnitude of Q would determine the smoothness of transition from one vertex of the simplex to another, but might affect sparsity. So we take a different approach to deal with discontinuity that addresses additionally other sources of discontinuities that are also common to HQP algorithm, which occur when the constraint matrices become singular near kinematic singularities or when the optimization problem itself is changed during task switching or priority switching. To address all these situations, we follow the simple idea of *enforcing* continuity on the control actions by bounding its time derivative as a hard constraint. Let x_{prev} be the decision variable from the previous control instance which is retained in memory and used as a parameter in the following constraint below, that is added to the problem eq. (7).

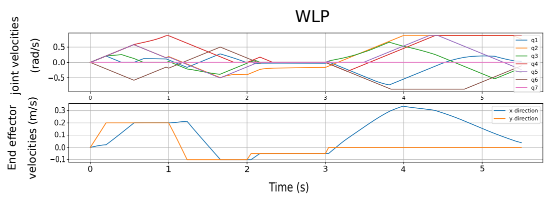
$$\dot{x}_{\min} T_s \leq (x - x_{\text{prev}}) \leq \dot{x}_{\max} T_s \tag{10}$$

where $\dot{x}_{\min}$ and $\dot{x}_{\max}$ are the desired lower and upper bounds on the rate of change of the decision variables and T_s is the sampling time of the controller.

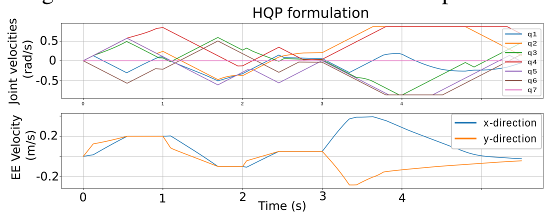
V. Experiments and Discussions

A. Didactical example

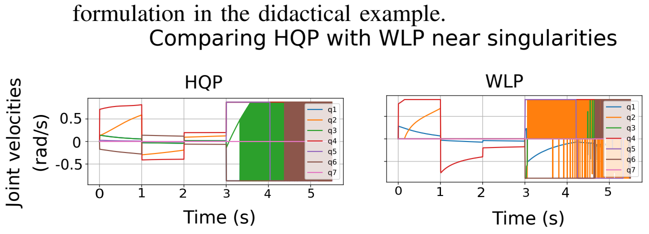
We illustrate and compare the control policies from WLP-ℓ-1 with HQP [8] on a simple problem with conflicting Cartesian velocity constraints on the end-effector a KUKA robot arm. The first set of constraints called S1 are $\dot{x}_{ee} = 0.2$ and $\dot{y}_{ee} = 0.2$, where $\dot{x}_{ee}$ and $\dot{y}_{ee}$ are the end effector velocities in the x and y directions of the base frame of the robot. The second set S2 is $\dot{x}_{ee} = -0.1$ and $\dot{y}_{ee} = -0.1$ and the third set S3, $\dot{x}_{ee} = -0.05$ and $\dot{y}_{ee} = -0.05$. The obtained motions are plotted in fig. 2. At the start, the priority order was chosen to be S1 $>$ S2 $>$ S3. Because of hard acceleration bounds, both algorithms take some time to reach end-effector velocities that satisfy the S1 constraints which has the highest priority. After 1 second, the priorities were switched to S2 $>$ S1 $>$ S3 and both WLP and HQP satisfy S2 that then has the highest priority. Both algorithms are similar when resolving conflicts between levels. After two seconds, S1 and S2 were assigned to the same level with the priority order S1,S2 $>$ S3 causing a conflict within the same level. The choice of norm now affects the solutions. In the HQP formulation, the robot end-effector attains a velocity equal to the average of S1 and S2 constraints because any deviation from the average increases the ℓ-2 norm of the objective at the highest level. In contrast, the ℓ-1 norm cost is the sum of violations at the level S1 and S2 which remains constant for the range of motions between 0.2 m/s and -0.1 m/s. This allows a greater degree of flexibility for the solver to optimize a lower level constraint which is why S3 is satisfied at this stage. This non-uniqueness of optimal constraint violations w^* during conflicts at the same level is an important difference between ℓ-1 norm and ℓ-2 norm methods. This provides additional flexibility to optimize lower level constraints. After 3 seconds only the set $\dot{x}_{ee} = 1.0$ and $\dot{y}_{ee} = 0$ are applied, causing the robot arm to extend in the x direction to reach joint states close to kinematic singularities. Continuously varying joint velocities are computed despite the proximity to singularities as well as the constraint on $\dot{y}_{ee}$ is satisfied. WLP-ℓ-1 is also found to compute sparser solutions due to ℓ-1 regularization which is not possible using the HQP algorithm.



(a) Joint velocities and end effector velocities for the weighted formulation in the didactical example.



(b) Joint velocities and end effector velocities for HQP formulation in the didactical example.



(c) Chattering observed for both formulations on the same task when acceleration bounds are removed.

Fig. 2: Didactical example comparing the behaviour of HLP vs HQP formulation.

B. Accuracy of the weighted formulation

In this subsection, we assess how accurately the methods developed in section III compute the correct weights that enable the weighted formulation to return the lexicographically optimal solution, with the SLP method serving as the ground truth. Pathological cases where the weighted method fails involve constraints that are almost linearly dependent. Thus 300 rank-deficient random tasks were generated with 25 decision variables and 25 equality constraints of rank 15 and 25 inequality constraints of rank 15 and were distributed

TABLE I: Accuracy of the weighted methods with γ in paranthesis and computation time in square brackets

Levels	WLP-ℓ-2f	WLP-ℓ-2a	SLP
2	**1.0**(0.2)[0.029s]	**1.0**(0.2) [0.028s]	[0.024s]
3	**1.0** (500) [0.040s]	**1.0**(0.2) [0.046s]	[0.041s]
4	**1.0** (500) [0.050s]	**1.0**(0.2)[0.054s]	[0.057s]
5	0.9567 (45) [0.054s]	**1.0**(0.2)[0.060s]	[0.071s]
6	0.9067 (10) [0.057s]	**1.0**(0.2)[0.064s]	[0.083s]
7	0.6867 (5) [0.061s]	**1.0**(0.2) [0.069s]	[0.089s]
8	0.54 (3) [0.065s]	**1.0**(0.2)[0.069s]	[0.109s]
9	0.1733 (1) [0.070s]	**0.997**(0.2)[0.074s]	[0.097s]
10	0.1467 (1.0) [0.069s]	**0.997**(0.2)[0.075s]	[0.113s]

uniformly into a number of levels. The remainder of tasks when not perfectly divisible is included in the last level. The fraction of the cases for which the weighted method successfully returned a correct solution for both the basic heuristic (WLP-ℓ-2f) and the adaptive method (WLP-ℓ2a) is presented in the table I. The parenthesis contain the default values the weights γ_is defined in eq. (9) before adaptation (fixed in the case of WLP-ℓ2a). The square brackets show the average computation time of the solver in each case. Please note that significantly lower time would be needed due to warm-starting when implemented for robot control. The simulations were performed using QPOases[19] on a system with Intel i7-8850H CPU @ 2.60GHz running an Ubuntu 18.04 operating system

For problems with two priority levels (including the hard priority level), both the weight computations return the same optimal constraint violations, as the problems are identical to the sequential formulation. Ideally an infinitely high value of γ_is would ensure lexicographic optimality for more than 2 levels. Up to 5 levels (which includes most robotics applications), simply fixing all the γ_is at a high value provides excellent accuracy. This was however, not numerically feasible for more than 5 levels. The adaptive method, which selectively updates weights only at those levels where the lexicographic optimality is violated, clearly scales better numerically to higher number of levels and computed a correct solution with nearly a hundred percent success rate. Thus, we successfully demonstrate that the weighted method combined with adaptation (for high number of priority levels) can reliably return the same solution as a sequential method even on a challenging example with many degenerate constraints.

C. Comparison between the sequential and the duality trick formulations

The SLP and the duality trick (DT) formulations are equivalent formulations and return identical solutions. The most relevant comparison between them would be on the basis of computation times. The benchmarking was done on 300 randomly generated tasks with 20 decision variables and 20 inequality constraints (of rank 10) and 20 equality constraints with rank 10 similarly to the previous subsection. The simulations were performed on MATLAB using Yalmip toolbox (because the toolbox provides an automatic dualizer [18]) and Gurobi LP solver. The results are displayed in fig. 3. Despite its theoretical elegance, the DT method, while being slightly

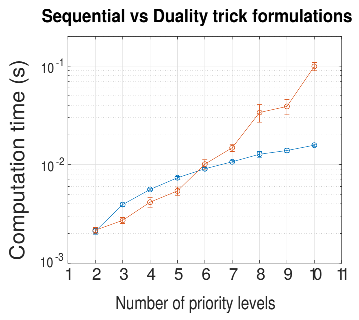


Fig. 3: Comparison of computation times between the sequential (blue graph) and the duality trick formulations (orange graph).

faster than the sequential method for upto 5 levels, is found to not scale well to a higher number of levels. This can be explained by a high-level complexity analysis of DT. In the recursive formulation of the DT method, every primal variable contributes a constraint to its dual problem, which then gets added as a primal constraint at the next priority level. This primal constraint in turn leads to a dual variable, which itself becomes an additional primal variable at the next level. So, the problem size (in the number of decision variables) grows exponentially by atleast doubling for every two additional priority levels. Yalmip's dualization is also suboptimal as it is not tailored for a lexicographic problem and does not eliminate constraints through substitution whenever possible. A custom-built dualizer that addresses this issue and might allow the DT method to scale slightly better but is outside the scope of this letter.

D. Dual arm task

In this subsection, we implement the weighted method on a practical dual arm assembly task with 5 levels of priorities on two Kuka iiwa manipulators that are fixed on a rotating platform that adds an extra degree of freedom and couples the kinematics of the two manipulators. The task involves picking up a couple of assembly parts, that are close to each other, in order to perform bi-manual assembly later. The stack-of-tasks and their priorities are indicated in table II. The problem has 17 decision variables - 15 joints and 2 progress variables (excluding the slack terms), 46 inequality constraints and 12 equality constraints. The WLP-ℓ-1 using the primal simplex solver from MOSEK and WLP-ℓ-2 regularization using QPOases [19] and HQP using the Lexlsi solver [20] were implemented on this task. The robots arms reach for the parts simultaneously resulting in conflict between the collision avoidance constraints and the path progress variables. The right arm, whose progress variable has lower priority stops moving and even back-tracks along its collision-free path to allow the left arm to progress avoiding a potential deadlock situation that would have arisen if there were no priorities. A video of the implementation is available in the github repository mentioned in the Section I. The computation times required by different solvers are shown in fig. 4.

The weighted formulations WLP-ℓ-1 and WLP-ℓ-2 were atleast 4 times faster than their sequential counterparts taking only 55 microseconds and 350 microseconds on average. WLP-ℓ-1 is in particular almost 25 times faster than cascaded QP solved using QPOases. The only solver which outperformed our method was the highly optimized Lexlsi [20] solver, which is the fastest lexicographic QP solver known to us in the literature (several times faster than the original HQP [8] algorithm) and took about 35 microseconds on average. Remarkably, the WLP-ℓ-1 algorithm, which is significantly simpler than the Lexlsi algorithm, using a general purpose off-the-shelf LP solver from MOSEK was not much slower and in some instances even faster than Lexlsi. However, it can compute sparse solutions unlike Lexlsi as shown in table III. The average number of joints indicate the degree of sparsity achieved by the solver and the WLP-ℓ-1 computed the sparsest solutions as expected actuating 10 joints most of the time. Since the task constrained 3 translational directions and 2 rotational directions for each robot, the task itself required 10 degrees of freedom and therefore the achieved sparsity was optimal in this example.

TABLE II: Stack of tasks

Priority	Tasks
0	joint position, velocity and acceleration limits
1	Fix robot end effector to be vertical and collision avoidance
2	Robots stay on a preplanned collision free path
3	Left robot arm progresses on its path
4	Right robot arm progreses on its path
5	Regularization on joint velocities. ℓ-1 with Mosek and ℓ-2 with QPOases

TABLE III: Comparing HQP with ℓ-1 and ℓ-2 regularized WLP

Metric	HQP	WLP-ℓ-1	WLP-ℓ-2
Average number of joints actuated	13.93	9.8299	11.629

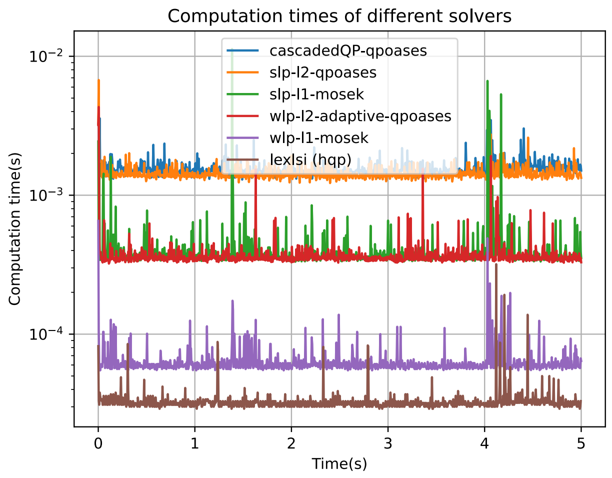


Fig. 4: Computation time comparison between weighted and sequential methods.

VI. CONCLUSIONS AND FUTURE WORK

We explored different formulations for solving the hierarchical robot task specification problems with ℓ-1 norm penalty on constraint violations. All methods that use an off-the-shelf solver use the sequential method which always provides a lexicographically optimal solution, but requires solving an optimization problem at every level. We showed how the sequential method could be transformed into an equivalent single LP using Lagrangian duality theory. However, this method was found to scale poorly and the problem size increased exponentially with the number of priority levels.

We introduced the weighted method which can achieve strict task priorities by solving a single optimization problem for a well chosen set of weights. This idea has no counterpart with methods that use only quadratic penalties and has not been explored before in the robotics literature to the best of our knowledge. It can be used to implement controllers that compute sparse policies faster than any other current methods as they all rely on solving a cascade of optimization problems. As the value of the weights are critical in determining the whether the weighted method computes a lexicographically optimal solution, we designed a simple tunable parameter that can be manually tuned as well as adaptively computed for a given problem. The efficacy of the weighted method was positively confirmed on a set of challenging tasks with rank-deficient constraints.

Regularization and smoothness aspects that are important for deployment on a hardware were discussed and the usefulness of the presented algorithms were demonstrated by implementing them for motion control in an interesting dual arm robot task with prioritized objectives. The weighted method was found to be remarkably fast despite its simplicity. Though slower than the more complicated Lexlsi solver, it can compute sparse solutions unlike the HQP methods. It is simple enough to be implemented on any existing controller that rely on off-the-shelf solvers within hours and still obtain high computational efficiency. It is also a promising strategy for extending lexicographic ordering of priorities to controllers with nonlinear constraints and predictive horizon, as it is based on the exact penalty method which is more generally applicable and this will naturally be the focus of future work.

ACKNOWLEDGMENT

The authors thank Dr Adrien Escande and Prof Pierre-Brice Wieber for providing access to the Lexlsi solver.

REFERENCES

[1] C. Samson, B. Espiau, and M. L. Borgne, *Robot control: the task function approach*. Oxford University Press, Inc., 1991.

[2] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *Int. J. Robot. Res.*, vol. 26, no. 5, pp. 433–455, 2007.

[3] W. Decré, R. Smits, H. Bruyninckx, and J. De Schutter, "Extending itasc to support inequality constraints and non-instantaneous task specification," in *Proc. IEEE Int. Conf. Robot. Autom.* IEEE, 2009, pp. 964–971.

[4] N. Mansard, O. Khatib, and A. Kheddar, "A unified approach to integrate unilateral constraints in the stack of tasks," *IEEE Trans. Robot.*, vol. 25, no. 3, pp. 670–685, 2009.

[5] E. Aertbeliën and J. De Schutter, "etasl/etc: A constraint-based task specification language and robot controller using expression graphs," in *Proc. IEEE/RSJ Int. Conf. Int. Robots. Syst.* IEEE, 2014, pp. 1540–1546.

[6] B. Siciliano and J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. Int. Conf. Adv. Robot.*, 1991, pp. 1211–1216 vol.2.

[7] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 785–792, 2011.

[8] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. J. Robot. Res.*, vol. 33, no. 7, pp. 1006–1028, 2014.

[9] V. M. Gonçalves, P. Fraisse, A. Crosnier, and B. V. Adorno, "Parsimonious kinematic control of highly redundant robots," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 65–72, 2015.

[10] J. J. Quiroz-Omaña and B. V. Adorno, "Whole-body kinematic control of nonholonomic mobile manipulators using linear programming," *J. Intell. Robot. Syst.*, vol. 91, no. 2, pp. 263–278, 2018.

[11] S. Tarbouriech, B. Navarro, P. Fraisse, A. Crosnier, A. Cherubini, and D. Sallé, "Dual-arm relative tasks performance using sparse kinematic control," in *Proc. IEEE/RSJ Int. Conf. Int. Robots. Syst.* IEEE, 2018, pp. 6003–6009.

[12] E. M. Hoffman, M. P. Polverini, A. Laurenzi, and N. G. Tsagarakis, "A study on sparse hierarchical inverse kinematics algorithms for humanoid robots," *IEEE Robot. Autom. Lett.*, vol. 5, no. 1, pp. 235–242, 2020.

[13] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[14] H. D. Sherali, "Equivalent weights for lexicographic multi-objective programs: Characterizations and computations," *Eur. J. Oper. Res.*, vol. 11, no. 4, pp. 367–379, 1982.

[15] H. D. Sherali and A. L. Soyster, "Preemptive and nonpreemptive multi-objective programming: Relationship and counterexamples," *J. Optim. Theory. Appl.*, vol. 39, no. 2, pp. 173–186, 1983.

[16] J. Vada, O. Slupphaug, T. A. Johansen, and B. A. Foss, "Linear mpc with optimal prioritized infeasibility handling: application, computational issues and stability," *Automatica*, vol. 37, no. 11, pp. 1835–1843, 2001.

[17] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[18] J. Lfberg, "Dualize it: software for automatic primal and dual conversions of conic programs," *Optim. Meth. Softw.*, vol. 24, no. 3, pp. 313–325, 2009.

[19] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: A parametric active-set algorithm for quadratic programming," *Math. Prog. Comp.*, vol. 6, no. 4, pp. 327–363, 2014.

[20] D. Dimitrov, A. Sherikov, and P.-B. Wieber, "Efficient resolution of potentially conflicting linear constraints in robotics," Aug. 2015. [Online]. Available: https://hal.inria.fr/hal-01183003