A Simple Formulation for Fast Prioritized Optimal Control of Robots using Weighted Exact Penalty Functions

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

***Abstract*— Prioritization of tasks is a common approach to resolve conflicts in instantaneous control of redundant robots. However, the idea of prioritization has not yet been satisfactorily extended to model predictive control (MPC) to allow for real-time robot control. The standard sequential approach for prioritization is unsuitable because of the computational burden involved in solving a nonlinear problem (NLP) at every priority level. We introduce an alternate promising approach of using weighted exact penalties for the MPC stage costs, where a correctly tuned set of weights can introduce strict prioritization. We prove the existence of a set of equivalent weights that provides the same solution as the sequential approach for a local convex approximation of the original NLP and use this insight to design an algorithm to adaptively tune the weights. The weighted method is validated on a dual arm robot task in simulations and also implemented on a physical robot. We report computational times that are fast enough for prioritized MPC of robot manipulators for the first time, to the best of our knowledge.**

I. Introduction

Constraint-based task specification [1], [2] and solving the resulting constrained optimization problem (COP) is a standard approach to control robots. Specifying multiple tasks to be achieved simultaneously may lead to conflicts between the tasks, which demands a conflict resolution strategy. Prioritization is a simple and well-established approach to resolve conflicts when some objectives are known to be strictly more important than the others and carefully tuning a trade-off is difficult or unnecessary [3], [4]. For example, transporting a glass of water at a desired velocity can be specified as a lower priority task, to be achieved without affecting the critical higher priority task of holding the glass upright, on a redundant robot manipulator. Prioritization can also resolve deadlocks at the continuous control level between multiple robots. Additionally, the presence of disturbances may cause violations of some constraints in a model predictive control (MPC) scheme requiring the relaxation of constraints to recover feasibility of the COP [5]. Prioritization then allows us to retain or recover feasiblity of critical constraints at the expense of less important constraints. There are mature and efficient implementations of hierarchical solvers for instantaneous robot control [6]. But instantaneous control does not compute a control policy that is (at least locally) optimal and feasible over a time horizon. As there are no guarantees, in general, on future constraint satisfaction, the robots need to be operated slowly and conservatively while using instantaneous control for reasons of safety. Therefore, there is a growing interest in using optimal control and MPC in robotics to generate fast dynamic behaviour. [7], [8].

Recently, there have been early attempts in the robotics literature to introduce prioritization of objectives in optimal control [9]–[12]. All these methods rely on the computationally expensive sequential method [13], where a nonlinear program (NLP) is solved at every priority level in a decreasing order of priority. A lower priority task is solved over the locally linearized nullspace of the higher priority task at every level to introduce priorization in [9], [11], [12] . The nullspace of only the cost-to-go function (obtained during the Riccati recursion step of differential dynamic programming (DDP)) is used in [11], [12] to obtain improved computational performance in comparison with the earlier work [9], which used the nullspace over the entire horizon. Faster convergence was reported[12] when using regularization which is known to improve conditioning of the sequential method [13], though at the expense of strict prioritization. However, all the above implementations of robot control were coded in MATLAB, report computational times in the order of minutes, were restricted to quadratic objectives and only validated in simulation. Moreover, they do not support specifying inequality constraints, also known as unilateral constraints. Another recent attempt includes [14] where they approximate the prioritized OCP (pOCP) as a quadratically-constrained quadratic program (QCQP) and solve a single QCQP per MPC iteration. But they use a non-standard definition of prioritization, where they enforce a higher priority objective to be smaller than a lower priority objective in magnitude through a quadratic constraint. This definition allows the lower priority objectives to influence higher priority objectives depending on factors like the choice of weights and units. Also, no computational performance was reported for this formulation to indicate its suitability for MPC.

The weighted method is promising compared to the previous approaches as only one NLP needs to be solved unlike the sequential method. But it is known that the weighted method is not applicable for quadratic penalties because it requires extensive tuning [11] and also the higher priority weights need to be infinitely higher than the lower priority weights [15] to obtain strict prioritization, which is numerically infeasible. However, for exact penalty functions, such as the $\ell 1$ norm, correctly chosen finite valued weights

*The authors gratefully acknowledge support from Flanders Make through the Flanders Make SBO project - MULTIROB and from the Research Foundation Flanders (FWO) through the project G.0C45.15. Flanders Make is the Flemish strategic research centre for the manufacturing industry.

[1]The authors are with the Division of Robotics, Automation and Mechatronics in the Department of Mechanical Engineering, KU Leuven, C300 BE-3001, Belgium and DMMS-M Lab, Flanders Make, Leuven, Belgium. Contact:firstname.lastname@kuleuven.be

are sufficient to enforce strict prioritization. The existence of such equivalent weights was first proved in [16] for lexicographic linear programs (LLP), where the existence of equivalent weights. However, it does not provide an algorithm to compute numerically reasonable weights. A parametric programming algorithm was proposed in [17] for computing optimally small equivalent weights for linear MPC, but this computation can take several minutes and is unsuitable for controlling nonlinear systems like a robot, where the linearization of the system changes at every control instance. Therefore, it is worth exploring whether it is possible to tune weights such that they remain sufficiently high for all the MPC control instances of a task, even if the weights are not optimally small. It was found that a simple combination of manual tuning and weight adaptation worked well for prioritized task-space constraints[15] and compared favourably against the sequential methods in terms of computational performance. In this paper, we extend this idea to control with a predictive horizon.

A. Approach and contributions

We propose to relax the task constraints in a prioritized OCP (pOCP) using weighted exact penalty functions, such as the $\ell 1$ norm in constrast to the previous methods that use quadratic penalties. We also support inequality constraints in our formulation. Using exact penalty functions for robot pOCP introduces other advantages of being able to generate fast near time-optimal motions [18], [19].

Solving the multi-objective NLP above with strict prioritization of objectives is also known as lexicographic optimization [20]. While it would be ideal to obtain global lexicographically optimal solutions for this pOCP, this is generally a difficult and an intractable problem especially for online or real-time control. We therefore follow the commonly followed pragmatic approach in robot control to use locally optimal solutions of the NLP. We prove for a local convex approximation of the original prioritized NLP, that there exist equivalent weights where a weighted formulation and the sequential formulation are equivalent. The focus of this paper is on the formulation of the problem, so readers can use any mature state-of-the-art NLP solver to implement the presented work and not on a specific lexicographic NLP algorithm.

We propose an adaptive tuning step, based on the existence proof, to verify and adapt the weights of the pOCP if the computed solution is not lexicographically optimal. We benchmark the weighted formulation using $\ell 1$ norm with the sequential method and traditional pOCP with quadratic penalties. We validate the weighted method on a dual arm robot task in both simulations and experiments. We report a computational speedup of several times compared to the sequential method because a single NLP is solved. Moreover, the computation times obtained suggest that it is feasible to implement prioritized MPC for robot control for the first time, to the best of our knowledge.

B. Preliminaries

Let the lexicographic nonlinear problem be represented as follows:

$$\underset{x}{\textbf{lex minimize}} \quad \{l_1(x), l_2(x), ..., l_p(x)\} \tag{1a}$$

$$\textbf{subject to} \quad \mathbf{g}(x) \leq 0 \tag{1b}$$

The lexicographic ordering of objectives above that leads to a notion of lexicographic optimality that is defined as follows.

Definition 1.1: [13] x^* is lexicographically optimal if it is feasible and no other feasible x exists such that $l_i(x) < l_i(x^*)$ without worsening at least one higher priority objective objective $j < i$, $l_j(x) > l_j(x^*)$.

1) Sequential Method: The most common approach for solving the problem in eq. (1) is the sequential method. This involves solving a sequence of single objective NLPs for each level in a decreasing order of priority and using additional hard constraints to ensure the lexicographic optimality of the higher priority objectives. The sequential method while solving for the ith objective is formulated as follows:

$$\underset{x}{\textbf{minimize}} \quad l_i(x) \tag{2a}$$

$$\textbf{subject to} \quad l_j(x) \leq l_j(x^*) \quad j = 1, 2, ..., i-1 \tag{2b}$$

$$\mathbf{g}(x) \leq 0 \tag{2c}$$

2) Weighted Method: The weighted method, also known as scalarization, transforms the problem eq. (1) to a single objective problem with the objective being weighted sum of different objectives. The weighted problem is defined as follows

$$\underset{x}{\textbf{minimize}} \quad \epsilon_1 l_1(x) + \epsilon_2 l_2(x) + ... + \epsilon_p l_p(x) \tag{3a}$$

$$\textbf{subject to} \quad \mathbf{g}(x) \leq 0 \tag{3b}$$

$\boldsymbol{\epsilon} = \begin{bmatrix}\epsilon_1 & \epsilon_2, & ... & \epsilon_p\end{bmatrix}^T, \boldsymbol{\epsilon} \in \mathbb{R}^p_+$ is a vector of the weights for each priority level. ϵ_p can be chosen to be 1 since only the relative magnitude of the weights matter. The weighted formulation always returns a pareto-optimal solution to the multi-objective problem [21]. But whether it returns a lexicographically optimal solution depends on the value of the weights as well as the objective functions.

We next mention some concepts from optimization theory that will be relevant for the discussion later in the paper. First order KKT (FO-KKT) conditions is a set of necessary conditions that characterizes the optimal primal-dual values of a constrained problem under certain regularity assumptions over constraints. These constraint qualifications are sufficient conditions that guarantee whether FO-KKT conditions are satisfied at the optimum of a constrained problem [22]. Linear independence constraint qualification (LICQ) is the most well known constraints qualification and requires all the equality constraints and the active constraints of the problem to be linearly independent. The optimal dual

is unique when LICQ is satisfied. Mangasarian-Fromovitz constraint qualification (MFCQ) is a generalization of LICQ that requires only the equality constraints to be linearly independent and that there exists a feasible point where all the inequality constraints are strictly satisfied.

II. Problem Formulation

The prioritized optimal control problem (pOCP) is formulated below in the form of a lexicographic optimization problem. We include only the inequality constraints in the pOCP equations below for the sake of brevity and clarity of notation.

$$\underset{x,u,s}{\textbf{lex minimize}} \quad \{l_1(s_1), l_2(s_2), ..., l_p(s_p)\}, \tag{4a}$$

$$\textbf{subject to} \quad x_0 = x_{\text{init}} \tag{4b}$$

$$x_{i+1} = f(x_i, u_i), \quad i = 0, 1, .., N-1 \tag{4c}$$

$$g_0(x_i, u_i) \leq 0, \quad i = 1, .., N \tag{4d}$$

$$g_k(x_i, u_i) \leq s_{k,i}, \quad i = 1, .., N \quad k = 1, .., p \tag{4e}$$

Where $x_i \in \mathbb{R}^{n_x}$ and $u_i \in \mathbb{R}^{n_u}$ represent the states and controls of the dynamical system at the discrete-time step i. eq. (4b) specifies the initial state constraint of the OCP. $f : \mathbb{R}^{n_x \times n_u} \to \mathbb{R}^{n_x}$ is the function that models the discrete-time forward dynamics of the system and eq. (4c) enforces the system dynamics. and eq. (4d) specifies hard and non-relaxable constraints in the pOCP. The function $g_0 : \mathbb{R}^{n_x \times n_u} \to \mathbb{R}^{m_0}$ in eq. (4d) represents hard constraints (highest priority) in the formulation. $g_k : \mathbb{R}^{n_x \times n_u} \to \mathbb{R}^{m_k}$ for $k \geq 1$ models the relaxable constraints in eq. (4e). The matrix variables $s_k \in \mathbb{R}^{m_k \times N}$ introduce slackness in the relaxable constraints and $s_{k,i}$ refers to the ith column of the matrix s_k.

Penalty functions $l_k : \mathbb{R}^{m_k} \to \mathbb{R}$ penalize the constraint violations at kth priority level. m_k is the number of constraints at the kth priority level. For quadratic penalties $l_k(s_k) = \sum_{i=0}^{N} \|s_{k,i}\|_2^2$. For the exact penalty functions $l_k(s_k) = \sum_{i=0}^{N} \|s_{k,i}\|_c$ where c is the chosen norm.

A. Smooth reformulation

The function $\|s_{k,i}\|_c$ is not a smooth function and can lead to numerical issues with NLP solvers and therefore a smooth reformulation is performed depending on the chosen norm c as follows.

1) $\ell 1$ norm: To reformulate an $\ell 1$ norm penalty, an additional constraint $s \geq 0$ is added to the OCP in eq. (4) and the equivalent penalty function would be $l_k(s_k) = \sum_{i=0}^{N} \sum_{j=1}^{m_k} s_{k,j,i}$, where $s_{k,j,i}$ is the jth element of the column vector $s_{k,i}$.

2) ℓ_∞ norm: For the smooth reformulation of ℓ_∞ norm penalty, further simplification is possible. $s_{k,i} \in \mathbb{R}^{m_k}$ can be replaced by a single number $s_k \in \mathbb{R}$. The constraint eq. (4e), then implies that every element of g_k must be less than the scalar s_k. And the constraint $s \geq 0$ needs to be added to the OCP eq. (4).

3) $\ell 2$ norm: The reformulation for $\ell 2$ norm is trickier. It requires specification of second order cone (SOC) constraints. Additional optimization variables $s'_{k,i} \in \mathbb{R}$, linear inequality constraints $s'_k \geq 0$ and SOC constraints $\|s_{k,i}\|_2 \leq s'_k$ are introduced in the OCP eq. (4) for each level k. The penalties l_k in eq. (4a) are redefined as $l_k(s'_k) = s'_k$. While this is theoretically possible, we are not aware of existing mature optimization software that can solve for such cone constraints in an NLP.

4) Quadratic penalties: Quadratic penalties are already smooth and $l_k(s_k) = \sum_{i=0}^{N} \|s_{k,i}\|_2^2$ can be directly used as the objective function. We note that for a combination of equality constraints of the form $g_k(x_i, u_i) = s_k$ (not included in eq. (4) for the sake of brevity) and quadratic penalties, further simplification in the OCP formulation is possible as slack variables are not necessary for a smooth formulation. One can directly redefine the lexicographic objectives as $l_k(x, u) = \sum_{i=0}^{N} \|g_k(x_i, u_i)\|_2^2$. All the previous literature [9]–[12] on prioritized OCP for robot control, that we are aware of, are limited to such quadratic penalties on equality constraints.

With quadratic penalties are used, the weight on the higher priority penalty needs to be infinitely higher than the weight on a lower priority penalty [15] for the weighted method to obtain a lexicographically optimal solution. This is practically infeasible, hence the existing approaches [9]–[12] use only the sequential method. But when exact penalties are used, the pOCP can also be solved using the weighted method. The weighted method returns a lexicographically optimal solution even with finite-valued weights provided that weights are appropriately tuned. In the next section, we discuss the existence of the equivalent weights for the pOCP problem and also propose an adaptive tuning algorithm.

III. Existence of Equivalent Weights

Suppose that the $\ell 1$ norm is chosen and the smooth reformulation explained in section II-A.1 is performed, the objective function is already linear in the decision variables. The nonlinear constraints are linearized in the same fashion as sequential quadratic programming (SQP) to obtain a local convex approximation. We note that this convexified problem results in a lexicographic linear program. It is known that for such a problem [23], there exist finite weights such that sequential and the weighted methods are equivalent. In this section, we provide a different constructive proof that constructs a set of valid weights based on the sequential method and exact penalty functions. This proof is then used to derive a lower bound on the weights as a function of the number of constraints and the horizon size.

Theorem 1: If there exists a feasible set of solutions to the locally linearized pOCP problem with $\ell 1$ penalties solved using the sequential method, there exist equivalent weights for which the weighted formulation returns a solution that is also a solution to the sequential formulation.

Proof: Let s_k^* be the optimal violations at each level computed by the sequential method. Since the sequential method is feasible, there exists a finite-valued vector of

optimal Lagrange multipliers associated with the pth LP (the last step) of the sequential method, that we call $\boldsymbol{\lambda}^*_{p-1}$. According to the exact penalty method theory, [22, Theorem 17.3], any constraint in this LP can be relaxed with a non-smooth penalty function with a sufficiently large penalty weight ($\epsilon_{p-1} \geq ||\boldsymbol{\lambda}_{p-1}||_\infty$). The solution to this relaxed LP is also optimal for the LP from the sequential method.

We choose to relax only the constraint in eq. (2b) for the $(p-1)$th level with a penalty parameter $\epsilon_{p-1} > ||\boldsymbol{\lambda}_{p-1}||_\infty$ as:

$$\epsilon_{p-1}\max\{0, l_{p-1}(s_{p-1}) - l_{p-1}(s^*_{p-1})\} + l_p(s_p) \quad (5)$$

The relaxation penalizes the constraint violation at level $p-1$ when $l_{p-1}(s_{p-1}) \geq l_{p-1}(s^*_{p-1})$. This objective can be further simplified to:

$$\epsilon_{p-1} l_{p-1}(s_{p-1}) + l_p(s_p) \quad (6)$$

The simplification above is valid for any high $\epsilon_{p-1} \geq ||\boldsymbol{\lambda}^*_{p-1}||_\infty$. Solving the objective above can not compute a solution where $l_{p-1}(s_{p-1}) < l_{p-1}(s^*_{p-1})$ without worsening the constraint violation at higher priority levels, which is prevented by the hard inequality constraints eq. (2b) for $i < p-1$. Otherwise, the $p-1$th LP of the sequential method, being an LP, would have found this solution.

Let us call this resulting LP the $(p-1)$th reverse sequential LP, where we solve for the lowest two priority levels using what is effectively the weighted method. The solution to this LP has an associated optimal dual vector $\boldsymbol{\lambda}^*_{p-2}$. The constraint in eq. (2b) for $(p-2)$th level can now be similarly relaxed with penalty weight $\epsilon_{p-2} > ||\boldsymbol{\lambda}^*_{p-2}||_\infty$. This step can be repeated till level 1 to recursively obtain an equivalent weighted formulation of the convexified pOCP.

■

We now proceed to derive a sufficiently high lower bound, for equivalent penalty weights at each priority level, as a function of the sum of all lower priority penalty weights, the horizon size and the number of constraints.

Proposition 1: There exists a finite real number C_i such that, choosing a sufficiently high $\epsilon_i > C_i \sum_{k=i+1}^{p} \epsilon_k.m_k.N$, results in selecting equivalent weights.

Proof: The objective function of the ith reverse sequential step is

$$\sum_{k=i+1}^{p} \epsilon_k.l_k(s_k) \quad (7)$$

.

Let $\mathcal{A}_i$ be a subset of the linear inequalities of the ith reverse sequential LP, that are strongly active set and satisfy LICQ, that is all the constraints in $\mathcal{A}_i$ are linearly independent. Active set algorithms successfully find such a $\mathcal{A}_i$ prior to termination [22]. The stationarity condition of the KKT conditions is obtained by taking the gradient of the Lagrangian w.r.t the decision variables $\mathbf{x}$ and setting it to zero. Since $l_k(s_k)$ is the sum of all elements of s_k, the elements of $\nabla_{\mathbf{x}} l_k(s_k)$would be equal to 1 at the locations corresponding to s_k in $\mathbf{x}$ and 0 otherwise. Let v_k denote this gradient, it would have $m_k.N$ number of non-zeros. The stationarity condition results in the following equation:

$$\mathcal{A}_i^T \boldsymbol{\lambda}_i = \sum_{k=i+1}^{p} \epsilon_k.v_k \quad (8)$$

Since, the rows in $\mathcal{A}_i$ are linearly independent, the equation above permits a unique solution for $\boldsymbol{\lambda}_i$, which can be computed using the Moore-Penrose pseudo-inverse of $\mathcal{A}_i^T$ denoted as $\mathcal{A}_i^{T+}$.

$$\boldsymbol{\lambda}_i = \mathcal{A}_i^{T+} \sum_{k=i+1}^{p} \epsilon_k.v_k \quad (9)$$

Taking the norm on either side of the equations and relaxing it using a combination of Cauchy-Schwarz inequality and the properties of norms results in the following inequality:

$$||\boldsymbol{\lambda}_i||_\infty \leq ||\boldsymbol{\lambda}_i||_2 \leq ||\mathcal{A}_i^{T+}||_2.(\sum_{k=i+1}^{p} \epsilon_k.m_k.N) \quad (10)$$

where $||\mathcal{A}_i^{T+}||_2$ is the Frobenius norm of $\mathcal{A}_i^{T+}$. Since $\epsilon_i > ||\boldsymbol{\lambda}_i||_\infty$ results in selecting the equivalent weights, the following equation provides a lower bound for choosing equivalent weights.

$$\epsilon_i > \|\mathcal{A}_i^{T+}\|_2.(\sum_{k=i+1}^{p} \epsilon_k.m_k.N) \quad (11)$$

■

A. An adaptive algorithm to compute the weights

Let $\epsilon_i = \gamma_i.(\sum_{k=i+1}^{p} \epsilon_k.m_k.N)$, where γ_i are the relative weights that need to be tuned and let $\gamma \in {\mathbb{R}_+}^{p-1}$ be the vector of all the relative weights γ_i. We now present an Algorithm 1 that checks if the lexicographic optimality of the solution of the weighted method on pOCP is violated at the level i by setting the absolute weights ϵ_k of the lower priority constraints to zero. Therefore, by design the lower priority constraints cannot affect the solution at the level i and thus the optimal l_i^* is obtained. If it is found to be violated, relative weights at ith level are increased till the weighted method returns a solution where l_i^* is optimal.

We next present an outer algorithm 2 that calls algorithm 1 to return a solution that is lexicographically optimal.

Corollary 1: The outer algorithm 2 terminates in a finite number of iterations to the lexicographically optimal solution.

Proof: The outer algorithm calls the algorithm 1 for every priority level in the decreasing order of priority. Within the inner algorithm, the weight ϵ_i at ith level is updated only if the lexicographic optimality is violated at that level for the solution of the weighted method. From Proposition 1, there is a finite number C_i such that, when $\gamma_i > C_i$, the violations at the ith level are lexicographically optimal. There can only be a finite number of such updates upper-bounded by $\sum_{i=1}^{p-1} log_\rho \frac{C_i}{\gamma_i^{\text{initial}}} + 1$. ■

Algorithm 1 Verify and rectify L-opt at level i

Require: $\boldsymbol{\gamma}$, l_k^* for i = 1,..,i-1, $\rho > 1$

Solve the weighted method using $\boldsymbol{\gamma}$ relative-weights to obtain candidate l_i^*.
Set $\epsilon_k = 0$ for $k = i+1, \ldots, p$
while True **do**
  Solve weighted method with the modified $\boldsymbol{\epsilon}$ to get l_ks
  if $\exists k$ such that $l_k > l_k^*, \quad k = 1, \ldots, i-1$ **then**
    $\gamma_k = \gamma_k * \rho$; recompute $\epsilon_k, \quad k = 1, \ldots, i-1$
  else
    if $l_i < l_i^*$ **then**
      $l_i^* = l_i$
    else
      if $l_i = l_i^*$ **then**
        return weighted method solution, $\boldsymbol{\gamma}$
    $\gamma_i = \gamma_i * \rho$; recompute $\boldsymbol{\epsilon}$

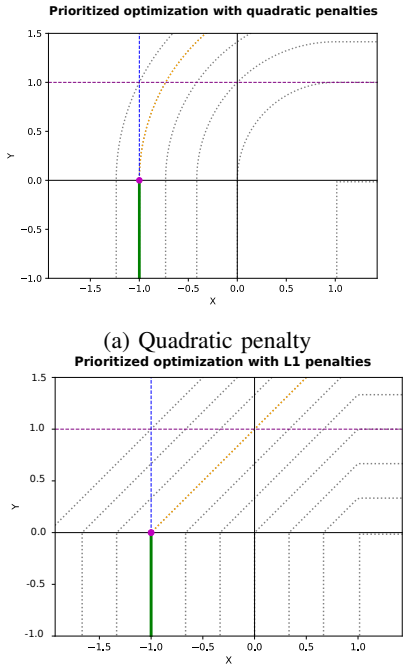
Algorithm 2 Solve hierachical problem

Require: $\boldsymbol{\gamma}$, weighted problem, $\rho > 1$

solution = solve weighted method for initial relative weights $\boldsymbol{\gamma}$
for (k=1,..., p-1) **do**
  if $l_k > 0$ **then**
    solution, $\boldsymbol{\gamma}$ = Verify and rectify L-opt at level k ($\boldsymbol{\gamma}$, solution, ρ)
return solution, $\boldsymbol{\gamma}$

IV. Constraint Qualifications for pOCP

Existence of finite-valued equivalent weights, such that the weighted method solves the original nonlinear pOCP, depends on whether KKT conditions are necessarily satisfied for each NLP in the sequential method [22, Theorem 17.3]. A detailed mathematical treatment of this question is out of the scope of this paper. Nevertheless, we present below a brief illustrative example of a failure case. Consider a prioritized optimization problem over only two decision variables $(x, y) \in \mathbb{R}^2$ plane, with $x = -1$ as the constrain at the priority level 1, two constraints at priority level 2: $x \geq 1$ and $y \leq 0$ and one constraint at the level 3: $y = 1$. The set of minimizers of l_1, l_2 and l_3 are plotted as the dashed blue line, dashed yellow line and the dashed magenta lines in fig. 1. The plots in fig. 1a and fig. 1b show these sets for the quadratic and $\ell 1$ penalties respectively. If the sequential method was used to solve this problem, the solutions to the second sequential problem and the third sequential problem are the green line and the magenta point at $(0, -1)$ respectively. Thus the sequential NLPs are feasible, but the third sequential problem with quadratic (or $\ell 2$) penalties in fig. 1a is a textbook case where the Lagrange multipliers satisfying the KKT conditions do not exist. Neither ∇l_1 nor ∇l_2 have a component in the y direction and solving the linearized problem would return the lexicographically suboptimal $(-1.0, 1.0)$ at the intersection of the purple and blue lines. The linearized nullspace projection approaches in [9], [11], [12] also theoretically suffer from this issue. Quadratic penalties are especially vulnerable to this issue as the gradient often vanishes for these objectives at the optimum. In this example, the solution to the third sequential problem with $\ell 1$ penalties can be shown to satisfy the KKT conditions as l_2 has a subgradient with a non-zero component in y direction.



(a) Quadratic penalty

(b) $\ell 1$ penalty

Fig. 1: Existence of weights for prioritized optimization

The constraint qualifications such as LICQ and MFCQ, that guarantee satisfaction of KKT conditions, are not satisfied in general for the pOCP problems. To see why, for LICQ, consider x^*, a lexicographically optimal minimizer. x^* being a solution to the first sequential problem must also satisfy the condition that the gradient $\nabla_x l_1(x^*)$ is in the columnspace of $\mathcal{A}_1^T(x^*)$ according to constraint are linearly dependent (the stationarity condition of FO-KKT of the first sequential NLP). Because l_1 is a part of the constraints in the second step of the sequential method, it follows that the constraint gradients are no longer linearly independent at x^*. We have assumed here that the lexicographic objective is differentiable for illustrative simplicity, but a similar argument can also by made with subgradients.

The lack of constraint qualifications has an implication on the solvers used for pOCP problems. The sequential method requires reliable NLP solvers that with globalization strategies that are robust to lack of constraint qualifications. Other strategies for making solvers more robust are to use regularization [12] (prevents the gradient of quadratic objectives from vanishing) or to relax the priority constraints in eq. (2b) by a small numerical value [13] (enforces MFCQ).

TABLE I: Constraints and their priorities in the dual arm task.

Priority	Constraints
0	Double sided inequality Joint pos, vel and acc limits.
0	Inequality constraint for Collision avoidance
1	Inequality constraint on end-effector (EE) orientation
2	Equality constraint left arm EE position
3	Equality constraint right arm EE position

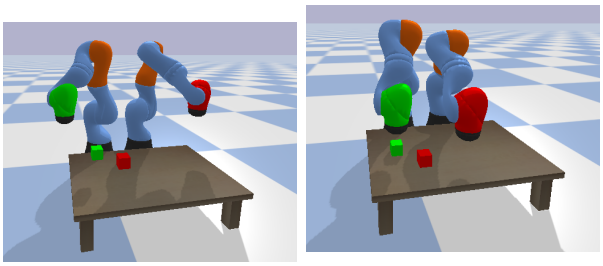
Both of these approaches come at the expense of losing strict prioritization even in those cases where it may have been possible to achieve. In contrast, the MFCQ always holds for the weighted formulation with exact penalties eq. (3), assuming that there exists a feasible x for which $g_0(x) < 0$, and it is not necessary to use large regularization terms or priority relaxation for solvers to compute a solution reliably. But the issue of requiring infinitely large weights to approach strict prioritization still remains for the weighted method if KKT conditions are not satisfied for the sequential method.

V. Experiments and Discussions

A. Dual Arm Pick and Place OCP

We consider a dual arm robot setup, where the arms need to simultaneously pick up objects in the shared workspace without colliding with each other or entering a deadlock. The constraints involved in the task specification are task are shown in table I in a decreasing order of priority, with 0 implying a hard constraint. The pOCP problem with a double integrator system model (considering joint accelerations as the control inputs) was constructed, with 1.5 seconds as the prediction horizon and the OCP sampling time of 0.1 seconds. The optimization problems involved were formulated using CasADi [24] and solved using IPOPT [25] for 120 different test cases for different object locations. One example is shown in fig. 2, where the robots arms are reaching for the parts simultaneously in fig. 2a. This results in activating the collision avoidance constraints and right arm stops to allow left arm, which has a higher priority, to reach its goal position as seen in fig. 2b. Without prioritization between the arms, this situation may have resulted in a deadlock.

We benchmark the performance of four different formulations on this task, namely sequential $\ell 1$ penalty (SLP), weighted $\ell 1$ penalty (WLP), sequential quadratic penalty (SQP, not to be confused with sequential quadratic program) and weighted quadratic penalty (WQP) for different weights on these 120 different dual arm pick up scenarios. A system with Intel i7-8850H CPU @ 2.60GHz running an Ubuntu 18.04 operating system was used for the benchmarking, all the computations were done on a single core. The average (and the standard deviation) of the results are tabulated in table II. They are compared on the basis of computation time, control effort, constraint violations at the three priority levels, whether the higher priority arm reaches its goal location within the OCP horizon and the amount of time it this motion takes (if it reaches). As expected, the weighted methods are



(a) Both the arms start moving towards their respective target objects.

(b) Left arm reaches the goal pose, while the right arm stops early to avoid violating higher priority constraints.

Fig. 2: Weighted method in action for dual arm robot control.

about 2 - 3 times faster than the sequential methods since only a single NLP is solved in WLP instead of the three NLPs in SLP.

The constraint violations at the highest priority level were practically zero for $\ell 1$ penalties with both the weighted and the sequential methods. For quadratic penalties are used, the constraint violation are nearly zero (not zero due to regularization) only with the sequential method. The weighted methods presented non-zero even for high values of weights, thus providing numerical confirmation that it is challenging to attain lexicographic optimality between conflicting objectives using weighted quadratic penalties.

Using the $\ell 1$ penalty appeared to result in faster motions with the higher priority arm reaching its goal position within the 0.67 seconds on average and succeeds in reaching the goal position by the end of the OCP horizon in *every* test case (please note that no hard terminal constraints were used to enforce this goal reaching). In contrast, for the traditional quadratic penalties, the left arm failed to reach the goal position within the OCP horizon (1.5 seconds) in a even single test case for the same numerical value of weights $\epsilon_1 = 20, \epsilon_2 = 10, \epsilon_1 = 2$ as the $\ell 1$ penalties. For a higher value of weights $\epsilon_1 = 100, \epsilon_2 = 25, \epsilon_1 = 2$ on the quadratic penalties WQP(2), the left arm reached the goal position for 25.8% of the cases and took 1.23 seconds on average to do so. The success rate of reaching the goal position increases to 75.8% and takes 0.82 seconds on average for large weights $\epsilon_1 = 1000, \epsilon_2 = 100, \epsilon_1 = 2$. Increasing the weights indefinitely is not an option however as it adversely affects the numerical conditioning of the problem, which in turn affects the convergence rate and computation times.

We report that it was remarkably easy to tune the WLP weights, as one of the first choice of weights, $\epsilon_1 = 20, \epsilon_2 = 10, \epsilon_1 = 2$, provided results close to the lexicographic optimum provided by the ground truth SLP method. It was significantly more challenging to tune the weights for the quadratic penalties, because strict lexicographic optimality is not only impossible in general for finite weights, but also a good choice of weights seemed sensitive to the location of the target objects in different test cases. This is perhaps

TABLE II: Comparison of different methods for the object picking task on the dual arm setup. The results over different metrics are averaged over 120 different test cases (the standard deviations are presented in the paratheses).

Metric	SLP	WLP (1)	SQP	WQP (1)	WQP (2)	WQP (3)
Comp time	0.315 (0.020) s	**0.102 (0.047)** s	0.272 (0.020) s	0.129 (0.077) s	**0.117 (0.067)** s	0.159 (0.171) s
$\int \|\|s1\|\|_1 dt$	$\mathbf{3.54 \times 10^{-10}}$	$\mathbf{5.86 \times 10^{-10}}$	$9.5 \times 10^{-4}(2 \times 10^{-4})$	0.27 (0.078)	0.105 (0.042)	0.041 (0.02)
$\int \|\|s2\|\|_1 dt$	**2.173** (0.81)	**2.174** (0.81)	3.346 (1.02)	3.31 (1.01)	2.64 (0.88)	**2.16 (0.78)**
$\int \|\|s3\|\|_1 dt$	$\mathbf{5.776}(2.29)$	$\mathbf{5.775}(2.29)$	8.24 (1.88)	8.20 (1.85)	8.31 (1.97)	8.36 (2.05)
Motion time	**0.670 (0.1)** s	**0.670 (0.1)** s	N/A	N/A	1.23 (0.048) s	0.82 (0.154) s
Goal reached	**1.0**	**1.0**	0.0	0.0	0.258	0.758
$\int \|\|\dot{q}\|\|_2 dt$	7.949 (1.22)	7.944 (1.22)	**5.494 (1.19)**	**5.439 (1.17)**	6.241 (1.39)	7.122 (1.65)

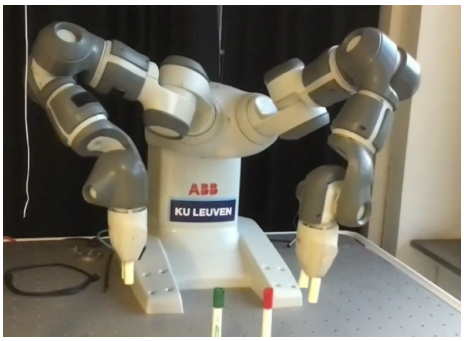


Fig. 3: Experimental deployment of pOCP for the Yumi robot

due to the gradient of a quadratic penalty also depending on the magnitude of constraint violation, unlike the $\ell 1$ norm penalty. The pOCP was also experimentally deployed on the Yumi robot fig. 3.

B. MPC for the Dual arm Picking Task

The above task was implemented in an MPC scheme with the same frequency of 10 Hz and a prediction horizon of 1.5 seconds as the OCP, with IPOPT as the NLP solver. IPOPT solver settings were adjusted to make it relatively better suited for warm-starting. A lower level joint velocity controller running at a frequency of 200 Hz interpolated the MPC reference to control the robot arms in simulation in a PyBullet environment. The mean, standard deviation and the maximum of the MPC computation times for the 120 test cases are shown in table III. Most computations for the WLP took less than 50 ms, except for 7 of the 120 cases. However, the maximum computation time of 88 ms remains under the chosen MPC sampling time of 100 ms. WLP is computationally more expensive than the traditional WQP, mostly because WLP has a higher number of decision variables and constraints (due slack variables introduced for smooth reformulation). The sequential method SLP however did not display a significant speed up due to warm-starting. The best strategy to warmstart the sequential method is also unclear as the optimization problem itself changes at every priority level with the addition of new constraints and slack variables. The current results suggest that the weighted method is more effectively warmstarted and is nearly 10 times faster than the sequential method as a result. This finding needs to be further investigated for other solvers and tasks.

TABLE III: MPC Computation Times.

Time measure	SLP	WLP(1)	WQP(1)
Mean time	0.300	0.031 s	0.020 s
Standard Deviation	0.0509	0.004 s	0.002 s
Maximum time	0.600	0.088 s	0.033 s

C. Adaptive Weight Tuning

Despite the effectiveness of manually tuned set of weights, the adaptive weight tuning described in section III-A is a promising and practical approach to automate this tuning. A preliminary investigation of this was performed for the dual arm task. The computation time required for adaptive tuning is naturally higher than the weighted method as multiple optimization problems need to be solved, but even for poor initial guess of the weight tuning took under 1.5 s on average for the 120 tasks and needed an average of 8.3 weight updates. This suggests that online weight tuning is a feasible approach. The idea is to first tune weights and use the same weights during the entire MPC iterations. Of course, these weights may not be suitable if the MPC motion deviates significantly from the OCP trajectory which did not occur in our simulations.

VI. CONCLUSIONS AND FUTURE WORK

We introduced a new formulation for prioritized optimal control of robots. Our main contribution is to show that strict prioritization of constraints for robot pOCP can be achieved using weighted exact penalty functions or conversely that competing exact penalty functions can introduce the effect of prioritization. Our formulation supports specification of hard constraints and also inequality constraints unlike the previous attempts in prioritized optimal control in robotics. It was shown both theoretically (for a locally linearized approximation) and experimentally that there exist finite values of weights such that weighted method with exact penalty functions and the sequential method are equivalent. A lower bound was computed for the equivalent weights that was found to be a linear function of the lower priority weights and an algorithm, that is guaranteed to terminate for a convexified pOCP, was proposed for automatic tuning of the weights. The weighted method with exact penalties was found to be easily tuned and returned the same solution as the sequential method for a non-trivial dual arm task, while being several times faster than the sequential method. Using exact

penalty functions appeared to result in faster near-time optimal motions compared to the pOCP with traditional quadratic objectives, which is desirable to improve the throughput of industrial robot operations. A mean computation time of 30 ms was observed for pMPC with $\ell 1$ penalties even with a relatively slow solver IPOPT, which suggests that it is computationally feasible to implement prioritzed MPC, for the first time to the best of our knowledge.

In the future, we will investigate this algorithm for other tasks and investigate different optimization solvers that are more efficient than IPOPT to improve the control frequency of the prioritized MPC. Smooth switching and sequencing of prioritized MPC tasks will also form the focus of future work.

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," *The International Journal of Robotics Research*, 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 Transactions on Robotics*, vol. 25, no. 3, pp. 670–685, 2009.

[5] E. Kerrigan and J. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," in *Proceedings of the UKACC International Conference on Control*, vol. 2000. IEE, 2000.

[6] 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.

[7] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 300–306.

[8] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3346–3351.

[9] A. Del Prete, F. Romano, L. Natale, G. Metta, G. Sandini, and F. Nori, "Prioritized optimal control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 2540–2545.

[10] D. Lunni, A. Santamaria-Navarro, R. Rossi, P. Rocco, L. Bascetta, and J. Andrade-Cetto, "Nonlinear model predictive control for aerial manipulation," in *Int. Conf. Unmanned. Aircraft. Sys.*, 2017, pp. 87–93.

[11] F. Romano, A. Del Prete, N. Mansard, and F. Nori, "Prioritized optimal control: A hierarchical differential dynamic programming approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 3590–3595.

[12] M. Geisert, A. Del Prete, N. Mansard, F. Romano, and F. Nori, "Regularized hierarchical differential dynamic programming," *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 819–833, 2017.

[13] E. C. Kerrigan and J. M. Maciejowski, "Designing model predictive controllers with prioritised constraints and objectives," in *Proceedings. IEEE International Symposium on Computer Aided Control System Design*, 2002, pp. 33–38.

[14] J. Lee, S. H. Bang, E. Bakolas, and L. Sentis, "Mpc-based hierarchical task space control of underactuated and constrained robots for execution of multiple tasks," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 5942–5949.

[15] A. S. Sathya, G. Pipeleers, W. Decré, and J. Swevers, "A weighted method for fast resolution of strictly hierarchical robot task specifications using exact penalty functions," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3057–3064, 2021.

[16] H. D. Sherali, "Equivalent weights for lexicographic multi-objective programs: Characterizations and computations," *European Journal of Operational Research*, vol. 11, no. 4, pp. 367–379, 1982.

[17] 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.

[18] R. Verschueren, H. J. Ferreau, A. Zanarini, M. Mercangöz, and M. Diehl, "A stabilizing nonlinear model predictive control scheme for time-optimal point-to-point motions," in *2017 IEEE 56th annual conference on decision and control (CDC)*. IEEE, 2017, pp. 2525–2530.

[19] C. V. Rao and J. B. Rawlings, "Linear programming and model predictive control," *Journal of Process Control*, vol. 10, no. 2-3, pp. 283–289, 2000.

[20] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

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

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

[23] H. D. Sherali and A. L. Soyster, "Preemptive and nonpreemptive multi-objective programming: Relationship and counterexamples," *Journal of Optimization Theory and Applications*, vol. 39, no. 2, pp. 173–186, 1983.

[24] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

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