Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Optimal Scheduling and Model Predictive Control for Trajectory Planning of Cooperative Robot Manipulators

A. Tika * , **N. Gafur** * , **V. Yfantis** * , **N. Bajcinca** *

* *Department of Mechanical and Process Engineering, Technische Universität Kaiserslautern, Gottlieb-Daimler-Straße 42, 67663 Kaiserslautern, Germany (e-mail: naim.bajcinca@mv.uni-kl.de).*

Abstract: A hierarchical control approach for cooperative pick-and-place tasks in a narrow shared workspace is proposed. A scenario with two robot arms performing pick-and-place tasks with moving objects while ensuring collision-free planning and execution of their respective trajectories is specifically addressed. To this end, we consider a hierarchical architecture with two-layer optimization-based control policies involving task scheduling in the top layer and path planning, along with the motion constraints, at the bottom one. On the one hand, for task allocation, a distance minimization algorithm is introduced, leading to an integer optimization problem with linear constraints and a bilinear cost function. On the other hand, we invoke model-based collision-free minimum-time planning of robot trajectories. Hereby, inverse robot dynamics and time scaling appear to be useful tools. The former accounts for the compensation of nonlinear robot dynamics, while the latter converts the trajectory planning to a fixed-time optimization problem, thus enabling synchronous robot task executions.

Keywords: Trajectory planning, hierarchical control, MPC, scheduling, robot manipulators

1. INTRODUCTION

The use of robots on existing assembly or packing lines aims to increase the flexibility of the production lines, reduce the production cost and to maximize throughput and machine utilization. In order to partially meet these requirements and to minimize the space required for the use of robots or to enable cooperative tasks to be performed, multiple robot arms in the same workspace are increasingly used. Sharing the same workspace by multiple robots can lead to collisions, so that safe operation can no longer be guaranteed. In addition to that, more challenges arise for the task assignments for randomly distributed objects transported by a conveyor belt and trajectory planning for grasping moving objects. One way of addressing the design of operation involving multiple cooperative robots is splitting it into the task of scheduling and collision-free trajectory planning.

The research on robot task scheduling has been primarily focused on minimizing the cycle time by determining the optimal sequence of a set of unordered static task points ("way-points") in the 3D space. A detailed literature review thereof is provided in Section 3.1. There are several approaches in literature regarding trajectory planning of a single robot formulated as an optimization problem. For instance, Schlemmer and Gruebel (1998) considered a parameter optimization problem for planning of a collision-free trajectory. Thereby, the time optimality results from a heuristic adaptation of the final time. For point-to-point trajectory planning, Ardakani et al. (2015) propose a model predictive control (MPC) algorithm as a fixed-time optimization problem, in which the deviation of states and control inputs are penalized. The real-time implementation of the algorithm is applied for a convex optimization problem. A minimum time path planning is also proposed by Zhang and Zhao (2016), where the travel schedule of the set points and the path parameters are optimized simultaneously using a genetic algorithm (GA) in order to obtain a minimum point-to-point transfer time. In Al Homsi et al. (2016), a minimum-time trajectory generation problem involving two robots that share the same working environment is introduced. The authors propose a hierarchical reformulation of a minimum-time mixed-integer optimization problem for online trajectory generation of two Degree of Freedom (DoF) SCARA robots. However, none of the presented methods deals with cooperative collision-free trajectory planning and task assignments of dynamically changing operating points or objects for multiple 6 DoF robots. Therefore, scheduling and a time-optimal MPC (van den Broeck et al. (2009) and Rosmann et al. (2017)) as an optimization algorithm are considered in this work, that incorporate these limitations in constraints.

This paper is organized as follows. Section 2 describes a sample of an associated plant motivated by a practical application. In Section 3, all components of the proposed hierarchical architecture are addressed, including the iterative integer programming-based scheduler, the dynamic model of the robot arms and an MPC-based trajectory generation and control. Simulation results with moving objects, randomly distributed over the conveyor belt, are presented and discussed in Section 4. The paper is finalized by brief concluding remarks in Section 5.

2. PLANT DESCRIPTION

Depending on the flexibility and capacity requirements of a fruit-packing station, fruit-sorting and packing are so far carried out either manually or fully automated. Although the fully automated packing systems are characterized by a high packing throughput, they are very cost-intensive and regarding the fruit varieties inflexible. An appealing solution, offering highly automated and flexible packing plants is attained by facilitating the existing manual production with lightweight robot arms as shown in the scheme with two robot arms in Fig. 1. The limited available

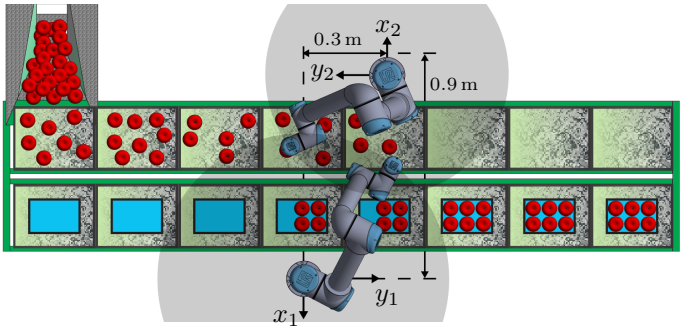


Fig. 1. *Schematic construction of a packing plant for fruit objects with light-weight robot manipulators performing time-varying cooperative pick-and-place tasks in partially overlapping workspaces.*

space in conjunction with the necessarily shared common workspace and the high throughput requirements typically lead to robots operating close to one another. Furthermore, the robots need to be supported by a camera system for measuring / estimating the position, orientation, as well as possible defects of the fruit objects. For this purpose, recently, cascaded architectures based on convolutional neural networks (CNNs) are increasingly getting popular, e.g., see Giefer et al. (2019). The random distribution of the fruit objects on the conveyor belt as delivered by the fruit feeder, together with the continuous transportation of the fruit objects and the trays to be filled, introduces a cooperative pick-and-place problem with highly time-varying targets, necessitating a continuous update of the robot arm trajectories. Moreover, the objects to be sorted may lie very close to each other. This essentially increases the collision potential between the robot arm manipulators. Thus, the coordinated workflow should be organized in such a way that the robots fulfill the pick-and-place tasks in an optimal way by maximizing the packing throughput without colliding with each other and other obstacles in the direct environment. Therefore, two main tasks are of fundamental importance and are elaborated in detail in the present article. Firstly, a resource allocation problem is raised by questioning which object is to be picked by which robot and in which slot of a tray it is to be placed. For this purpose, a scheduling algorithm for cooperative selection of non-stationary objects, while additionally taking into account safety requirements by adding some safety-related constraints, is presented. Secondly, in order to accomplish the assigned tasks, an algorithm for model-based planning and control of collision-free trajectories for robots sharing a common workspace is proposed. Notice that grasping the objects by the end effector itself is not addressed in this study.

3. ROBOT DYNAMICS AND HIERARCHICAL CONTROL ALGORITHM

In order to optimally fill the objects into the trays, a hybrid control algorithm is employed, consisting of two layers. The upper-level deals with the scheduling problem, while the lower-level includes an MPC planning algorithm running on top of the computed torque control law. The scheduler computes the optimal allocation of the objects and trays ("tasks") to the robots ("resources") by minimizing the total traversed euclidean distance between the objects and trays. The scheduling layer then provides the

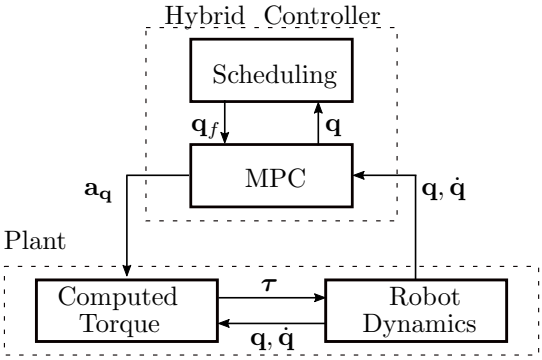


Fig. 2. *Schematic illustration of the used control structure.*

position set-points to the MPC layer, which computes the optimal trajectories of the robot manipulators by minimizing the execution time of the tasks, while accounting for potential collisions. A schematic depiction of the proposed control algorithm is shown in Fig. 2. The following sections address the specific components of this scheme, including the scheduling algorithm, the dynamic model of the robot manipulators and trajectory planning utilizing MPC.

3.1 Iterative scheduling

The goal of the proposed overall control structure is to pick up dynamically moving objects, i.e. the fruit, and place them into dynamically moving slots in a tray in a time optimal manner. The decisions that have to be made by the algorithm include the allocation of the robots to the objects, i.e. by which robot each object is picked up, the subsequent assignment of the objects to slots in the trays, the sequence in which the objects are picked up and the computation of the corresponding collision-free trajectories of the robot manipulators in order to execute these tasks. A highly non-convex mixed-integer nonlinear programming problem would result if the discrete scheduling decisions and the dynamics of the robots were to be included in a monolithic optimization framework, which would severely limit its practical online applicability, as the required computation time would be prohibitively long. In Zhang and Zhao (2016) this problem is modeled as an extension to the traveling salesman problem (TSP), and a GA is applied to solve the modified TSP in order to obtain the optimal traveling schedule for a 3 DoF robot performing drilling/spot welding tasks. The transfer path between the task points is realized by minimizing the transfer time while considering the dynamic performance of the robot without any collision avoidance restrictions. Zacharia and Aspragathos (2005)

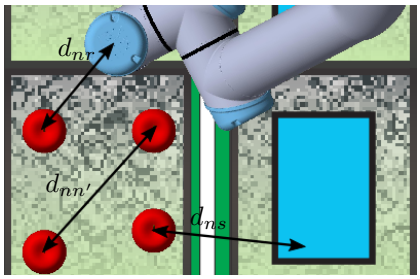


Fig. 3. *Distance parameters of the scheduling problem.*

introduced a GA based method to solve the TSP, which is adapted to take into account the multiple solutions of the inverse kinematics. This idea is extended in Xidias et al. (2010) to the case of a two-robot work cell. The robots are each modeled as first-degree B-spline curves in order to derive collision-free motions. Kinematic and dynamic constraints, as well as collision avoidance with the environment, are not considered. Another idea involving two robots is presented in Sutdhiraksa and Zurawski (1996). Therein, timed Petri nets and the uniform cell decomposition approach are used to model the robot tasks and ensure collision-free operation. The applications found in the literature are mainly concerned with the offline optimization for repetitive tasks in which the cycle time is to be minimized. The reported computation times are still not suitable for an online application with dynamically moving targets. In order to cope with these practical difficulties the control structure is split into two layers, the higher-level scheduling layer and the lower-level time optimal MPC layer. The goal of the scheduling layer is to allocate a set of objects $n \in \mathcal{N}$ to a set of slots $s \in \mathcal{S}$ while minimizing the total euclidean distance covered by the robots $r \in \mathcal{R}$ to execute all necessary tasks. The slots are located on a set of trays $p \in \mathcal{P}$, where $\mathcal{S}_p \subset \mathcal{S}$, denotes the subset of slots located on tray p. The sequence in which the slots are filled is fixed so as to eliminate the possibility of collisions during the placement of the objects into the trays. All slots filled at the same time as slot s belong to the subset $\mathcal{S}_s^c$. Slot $\hat{s}_p$ is the first one filled in tray p.
In reality, a more appropriate objective would be the time necessary to execute all tasks. However, the required time depends on the multiple possible configurations by which a target can be reached, resulting from the non-unique solutions of the inverse kinematics problem. Therefore the exact time can only be modeled by including the dynamics of the robotic manipulators in the formulation of the optimization problem. Thus, at this upper level it is assumed that the time necessary to move between two spots in the 3D workspace is proportional to the traversed distance. The exact timing and execution of this movement is later performed by the MPC layer. The inputs to the scheduling layer are the positions $\mathbf{x}^n$ of each object $n \in \mathcal{N}$, the positions $\mathbf{x}^s$ of each slot $s \in \mathcal{S}$ and the positions $\mathbf{x}^r$ of each robot $r \in \mathcal{R}$ (all in Cartesian coordinates), provided by the camera system. Given these coordinates, the euclidean distances between all objects and all slots, $d_{ns} = \|\mathbf{x}^n - \mathbf{x}^s\|_2$, all pairs of objects, $d_{nn'} = \|\mathbf{x}^n - \mathbf{x}^{n'}\|_2$ and all objects and robots, $d_{nr} = \|\mathbf{x}^n - \mathbf{x}^r\|_2$, can be computed a priori, as shown in Fig. 3. As the MPC layer receives its targets (objects and slots) from the scheduling layer, a poorly chosen sequence of tasks may lead to unavoidable collisions during the execution of these tasks. Therefore, collision avoidance has to be accounted for in the scheduling layer, as well. Collisions while placing the objects into the trays can be easily avoided by predefining the sequence in which the slots are filled. If the slots that are filled at the same time are far enough apart from each other, no collision can occur. As the sequence in which the objects are picked up constitutes the main degree of freedom of the optimization it can not be restricted a priori. Collisions are avoided by introducing a minimum distance $d^{\min}$ between simultaneously picked up objects, which is included in the optimization as a constraint. Note that the scheduling layer only ensures collision avoidance while picking up and while placing an object. The potential collisions during the movements of the robots between the targets provided by the scheduling layer are addressed by the MPC layer. The scheduling model is based on the well established travelling salesman problem (TSP), but includes two sets of binary variables, X_{ns} which indicates the allocation of object n to slot s and W_{rp} which indicates the filling of tray p by robot r. It is assumed that the number of object is always larger than the number of slots ($|\mathcal{S}| \leq |\mathcal{N}|$), since not all trays could be filled otherwise. In order to avoid collisions during the placement of the objects in the trays, each tray can only be filled by a single robot. This can be expressed as a constraint on the binary robot allocation variables,

$$\sum_{\forall r \in \mathcal{R}} W_{rp} = 1, \ \forall p \in \mathcal{P}. \tag{1}$$

Note that a collision during the placement of objects in different trays is avoided due to the fixed sequence in which the slots are filled, as explained above. Similarly, each robot can only fill a single tray,

$$\sum_{\forall p \in \mathcal{P}} W_{rp} = 1, \ \forall r \in \mathcal{R}. \tag{2}$$

In order to meet the product requirements, each tray has to be filled with a specified amount of objects, i.e. each slot in the trays must contain exactly one object. This can be analogously modelled by using the binary object to slot allocation variables X_{ns},

$$\sum_{\forall n \in \mathcal{N}} X_{ns} = 1, \ \forall s \in \mathcal{S}. \tag{3}$$

However, since the number of objects is greater than the number of slots, not every object can be allocated to a single slot. Nevertheless, each object can at most be allocated to one slot, but does not necessarily have to be allocated to a slot at all. This is modelled by an inequality instead of an equality constraint on the binary object to slot allocation variables,

$$\sum_{\forall s \in \mathcal{S}} X_{ns} \leq 1, \ \forall n \in \mathcal{N}. \tag{4}$$

To avoid collisions of the robots while picking up objects, two objects that are picked up at the same time have to have a minimum distance $d^{\min}$ between them. Two objects are picked up at the same time if they are allocated to corresponding slots on the trays, i.e. if they are placed into the trays at the same time. The minimum distance

can be enforced by the following constraint,

$$d_{nn'} + (2 - X_{ns} - X_{n's'}) \cdot d^{\min} \geq d^{\min}, \quad \forall n, n' \in \mathcal{N}, n \neq n', s \in \mathcal{S}, s' \in \mathcal{S}_s^c. \quad (5)$$

If objects n and n' are not allocated to corresponding slots the constraint is trivially satisfied, since at least one of the two binary variables is equal to zero. However, if the objects are allocated to corresponding slots, the second term in the left-hand side of the inequality vanishes and the constraint is only satisfied if the distance between the objects is greater than the minimum distance.
The objective of the optimization is to minimize the total euclidean distance traversed by the robots. The paths of the robots can be divided into the initial movement to the first picked object, the movement from the object position to its assigned slot and the movement from a slot to the next object. The total distance can be expressed as a bilinear function, that is to be minimized,

$$\begin{aligned} \min \; & \underbrace{\sum_{\forall r \in \mathcal{R}} \sum_{\forall p \in \mathcal{P}} W_{rp} \left(\sum_{\forall n \in \mathcal{N}} X_{n\hat{s}_p} \cdot d_{nr} \right)}_{\text{Initial Movement of the Robots}} \\ & + \underbrace{\sum_{\forall n \in \mathcal{N}} \sum_{\forall s \in \mathcal{S}} X_{ns} \cdot d_{ns}}_{\text{Movement from Object to Assigned Slot}} \\ & + \underbrace{\sum_{\forall n' \in \mathcal{N}} \sum_{\forall p \in \mathcal{P}} \sum_{\forall s \in \mathcal{S}_p} X_{n',s+1} \cdot d_{n's}}_{\text{Movement from Slot to Next Object}} \end{aligned} \quad (6)$$

The initial movement depends on the allocation of a robot to a tray and on the object allocated to the first slot of that tray. The movement from an object to its assigned slot can be derived from the binary object to slot allocation variable. Finally, as the sequence in which the slots are filled is fixed, the robot has to move from the current slot s to the object n' assigned to the next slot $s+1$. To perform the described scheduling algorithm the resulting integer bilinear programming (IBLP) problem has to be solved online. After a solution is obtained, the scheduling layer provides the coordinates of the first picked up object and of its assigned slot for each robot to the MPC layer. Since the objects and the trays move on the conveyor belt, their positions change dynamically. Therefore, after the first objects of the sequence have been placed into the trays, the corresponding slots are removed from the set $\mathcal{S}$, new objects are potentially added to the set $\mathcal{N}$ and the optimization is performed again in an iterative fashion. This approach is only feasible, if the underlying optimization problem can be solved within very short computation times. This issue is further discussed in the following section. Note that the scheduling optimization problem does not account for the feasible workspace of the robots. This could lead to a situation where an object leaves the range of the robots before being picked up, resulting in an infeasible optimization problem in the MPC-layer. To avoid this situation a safety distance margin is employed, depending on the radius of the robots' workspace. If an object exceeds this margin, the decision of the scheduling optimization is overwritten and the object is picked up instead.

3.2 Dynamic model

The dynamic model of the considered robot manipulator is derived by means of the Lagrange formalism (Spong et al., 2006). For the sake of simplicity, viscous and static friction terms have been neglected, so that the robot equation of motion in matrix form can be written as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}, \quad (7)$$

where $\boldsymbol{\tau} \in \mathbb{R}^n$ denotes the vector of generalized torques, $\mathbf{q} \in \mathbb{R}^n$ denotes the vector of generalized coordinates (i.e. the joint angular position), $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n \times n}$ represents the coefficients of the centrifugal (proportional to $\dot{q}_i^2$) and Coriolis (proportional to $\dot{q}_i \dot{q}_j, i \neq j$) forces and $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ is the vector of gravitational torques. For the considered robots $n = 6$ holds. Additionally, in order to provide high-quality visualization of the robots and the experimental setup, a dynamic simulation model in SIMSCAPE has been developed using the CAD data provided by the manufacturer. The SIMSCAPE model is mainly used for the simulation of the forward dynamics, whereas the analytically derived equations of motion are used in feedback control to implement a nonlinear dynamic inversion based controller $\boldsymbol{\tau} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t)$ in order to compensate the nonlinear dynamics. Since the inertia matrix $\mathbf{M}(\mathbf{q})$ is symmetric and positive definite, the nonlinear feedback control law can be chosen as

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\mathbf{a}_\mathbf{q} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}). \quad (8)$$

When substituting (8) into (7), the combined control system results in a linear closed loop system of n double integrators

$$\ddot{\mathbf{q}} = \mathbf{a}_\mathbf{q}, \quad (9)$$

with the new input $\mathbf{a}_\mathbf{q} \in \mathbb{R}^n$. The resulting robot model

$$\frac{d}{dt} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{0}^{n \times n} & \mathbf{I}^{n \times n} \\ \mathbf{0}^{n \times n} & \mathbf{0}^{n \times n} \end{bmatrix}}_{=:\tilde{\mathbf{A}}} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{0}^{n \times n} \\ \mathbf{I}^{n \times n} \end{bmatrix}}_{=:\tilde{\mathbf{B}}} \mathbf{a}_\mathbf{q}, \quad (10)$$

with the dynamic matrix $\tilde{\mathbf{A}}$, the input matrix $\tilde{\mathbf{B}}$ and the identity matrix $\mathbf{I}$, consists of a chain of $2n$ decoupled integrators and contains only kinetic variables (Spong et al., 2006).
In the proposed hierarchical control approach, a centralized MPC algorithm is used for cooperative generation of collision-free trajectories. Hence, considering two robots represented by the generalized coordinates $\mathbf{q}_1$ and $\mathbf{q}_2$, by introducing a new state vector

$$\boldsymbol{\xi} = [\mathbf{q}_1^T, \dot{\mathbf{q}}_1^T, \mathbf{q}_2^T, \dot{\mathbf{q}}_2^T]^T$$

the total continuous time dynamics to be considered in the centralized MPC algorithm is given by

$$\dot{\boldsymbol{\xi}} = \underbrace{\begin{bmatrix} \tilde{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{A}} \end{bmatrix}}_{=:\mathbf{A}} \boldsymbol{\xi} + \underbrace{\begin{bmatrix} \tilde{\mathbf{B}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{B}} \end{bmatrix}}_{=:\mathbf{B}} \underbrace{\begin{bmatrix} \mathbf{a}_{\mathbf{q}_1} \\ \mathbf{a}_{\mathbf{q}_2} \end{bmatrix}}_{=:\mathbf{u}}, \quad (11)$$

where $\mathbf{A} \in \mathbb{R}^{n_\xi \times n_\xi}$ and $\mathbf{B} \in \mathbb{R}^{n_\xi \times n_u}$. The model is subject to robot kinematic constraints, such as angular position, velocity, acceleration, and additional collision avoidance constraints, which will be discussed in more detail in Section 3.3.

3.3 Minimum-time and collision-free trajectory planning

The objective of time optimal MPC is to find the optimal trajectory subject to minimal transition time, avoiding self-collision and collisions between the robots. Sharing one common transition time, means a synchronous task performance of the two robots, in which both finish a task at the same final time.

MPC formulation

The prediction of a minimum-time trajectory with a prediction horizon $N \in \mathbb{N}$ is mapped from the time $t \in \mathbb{R}$ to $\tau \in [0, 1]$ by the following time transformation

$$\tau := \frac{t - t_{k,0}}{t_f - t_{k,0}}, \tag{12}$$

where k is the time index, and t_f and $t_{k,0}$ denote the final and current times respectively, yielding a formal reformulation of the optimization problem within a fixed scaled-time interval $[0, 1]$. For simplicity the current time $t_{k,0}$ will not be explicitly written in the MPC formulation, i.e. $\tau := t/t_f$. The resulting centralized MPC problem for recurrent minimum-time trajectory planning with state vector $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ and control input $\mathbf{u} \in \mathbb{R}^{n_u}$ in a discretized form reads

$$\begin{aligned}
\min_{\mathbf{u}} \quad & t_f && (13)\\
s.t. \quad & \boldsymbol{\xi}_k(j+1) = \mathbf{A}_d\boldsymbol{\xi}_k(j) + \mathbf{B}_d\mathbf{u}_k(j) && (13a)\\
& \boldsymbol{\xi}_k(j+1) \leq \mathrm{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\boldsymbol{\xi}_{\max} && (13b)\\
& \boldsymbol{\xi}_k(j+1) \geq \mathrm{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\boldsymbol{\xi}_{\min} && (13c)\\
& \mathbf{u}_{\min} \leq \mathbf{u}_k(j+1) \leq \mathbf{u}_{\max} && (13d)\\
& \|\mathbf{x}_{r_1}(\boldsymbol{\xi}_k(j+1)) - \mathbf{x}_{r_2}(\boldsymbol{\xi}_k(j+1))\|_2 \geq r_{\min} && (13e)\\
& z_{k,r_1}(\boldsymbol{\xi}_k(j+1)) \geq z_{\min},\ z_{k,r_2}(\boldsymbol{\xi}_k(j+1)) \geq z_{\min} && (13f)\\
& \boldsymbol{\xi}_k(0) = \mathrm{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\boldsymbol{\xi}_0 && (13g)\\
& \boldsymbol{\xi}_k(N) = \mathrm{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\boldsymbol{\xi}_f && (13h)
\end{aligned}$$

where $j \in \{0, .., N-1\}$ is the iteration index of the prediction horizon, and $\mathbf{I} \in \mathbb{R}^{n \times n}$ the identity matrix. The position vector in Cartesian space is denoted by $\mathbf{x}_{r_i}$ with an index i for Robot 1 and Robot 2. The limitation of the operating space of robots in the z-axis is given by z_{k,r_i} in (13f). The discrete state matrix $\mathbf{A}_d \in \mathbb{R}^{n_\xi \times n_\xi}$ and input matrix $\mathbf{B}_d \in \mathbb{R}^{n_\xi \times n_u}$ in (13a) have the following form

$$\begin{aligned}
\mathbf{A}_d &= \mathbf{I} + \Delta\tau\mathbf{A}\\
\mathbf{B}_d &= \Delta\tau t_f^2(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A})\mathbf{B}
\end{aligned} \tag{14}$$

with $\Delta\tau := 1/N$. The scheduling algorithm provides initial $\mathbf{x_0}$ and final $\mathbf{x_f}$ positions to the MPC layer, which are converted to initial $\boldsymbol{\xi}_0$ and final $\boldsymbol{\xi}_f$ joint angles and angular velocities for both robots. The desired state $\boldsymbol{\xi}_f$ should be reached by the final time t_f in time scale t or at $\tau = 1$ in τ-scale by the end of the prediction horizon N, formulated in (13h). The control input $\mathbf{u}$ represents the robots' angular acceleration limited to a range $[\mathbf{u}_{\min}, \mathbf{u}_{\max}]$, see (13d).

Static and dynamic collision avoidance

Self-collision constraints are formulated for the relevant joint angles $\mathbf{q}_k$ in (13b), (13c). Due to the fact, that the robots are working in a work space limited to a conveyor belt, the joint angles have to be limited to their working area. These limitations simplify the choice of all possible inverse kinematic configurations. Moreover, to avoid a collision between a robot and the conveyor belt, a constraint with minimum operating height in the z-coordinate for both robots in (13f) is additionally considered. In this paper, for the sake of simplicity, collision avoidance between the robots is formulated with regard to the robot end effectors in the Cartesian space by utilizing the concept of a "distance function", as formulated in (13e). While this has been a common approach in practice and a variety of other works in literature, note that additional constraints addressing the robot links to strengthen the collision avoidance requirements can be in principle implemented. Therefore, convex hulls around additional links can be invoked, e.g. in form of ellipsoids. The time optimal setting guarantees that in each optimization step k, the desired state $\boldsymbol{\xi}_f$ is reached at the end of the prediction horizon $k + N$, i.e. at $\tau = 1$. The length of the prediction horizon determines the grade of collision-free predictions. Indeed, shorter horizons feature coarse prediction capabilities and, therefore, a larger collision potential. And, vice-versa, larger horizons induce planning outcomes with higher accuracy in collision avoidance with the downside of causing higher computational cost. In this sense, a sufficiently large selection of the control horizon N is required to avoid the collisions. The τ-scale is accordingly divided in equidistant τ-intervals: $\tau \in \{0, \frac{1}{N}, \frac{2}{N}, ..., \frac{N-1}{N}, 1\}$. Note that $\Delta\tau$ intervals are mapped to increasingly tighter Δt-intervals as time t propagates, because t_f reduces with each optimization step. Moreover, the first optimal input from $\mathbf{u}^*$ shall be applied over the time interval $\Delta t = t_f^* \Delta\tau$, where t_f^* refers to the underlying optimal solution of (13). The closer the robots approach the target, the smaller the solution t_f^* gets, while the number of control degrees of freedom remains the same. This leads to a finer prediction towards the end and eventually, to an under-determined problem when $\Delta\tau t_f^* < T_s$, where T_s refers to the sample time. Once, the sample time has been undercut (i.e. $\Delta\tau t_f^* < T_s$), the optimization is stopped. The optimal inputs from the optimization are applied until the desired state is reached. If, in this case, further optimization steps are necessary, the prediction horizon can be reduced to satisfy $\Delta\tau t_f^* \geq T_s$, or an equivalent input can be calculated, which is constant over the sampling time.

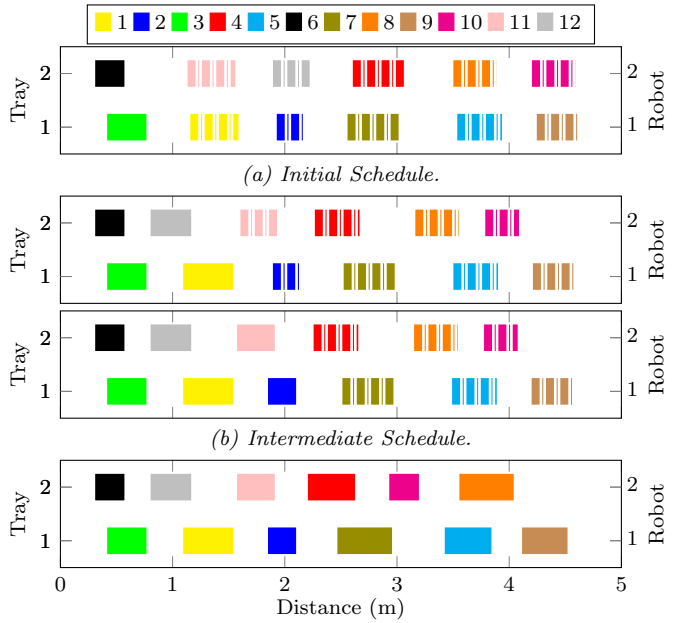
Grasping a moving object

The desired position $\mathbf{x}_f$ changes with time due to the object, that moves along the $y-$axis with a constant velocity $\mathbf{v}_{\text{obj}} = [0, v_b, 0]^T$ changing its position by $\mathbf{x}_{\text{obj}}$, with v_b denoting the velocity of the conveyor belt. The velocity of the robot at the end should be equal to the velocity of the object, in order to be able to grasp the object. At the first optimization step, the desired position of the end effector is equal to the position of the object, i.e. $\mathbf{x}_f = \mathbf{x}_{\text{obj}}$. For this, the transition time t_f is calculated, which will be needed to reach the object if it does not move. But as the object moves, the desired position should be updated by the time that is left to reach the object. In the next optimization step the new desired position changes to $\mathbf{x}_f = \mathbf{x}_{\text{obj}} + t_f^*(1 - \Delta\tau)\mathbf{v}_{\text{obj}}$. However, in a practical implementation, camera-based feedback of the object position is needed, since the object grasping based on the velocity prediction would not be accurate enough.

4. SIMULATION RESULTS

The proposed hierarchical approach is implemented and evaluated on a dynamic simulation model, based on the CAD data of the robots, objects, and the conveyor belt. Along with the robot simulation model, a computed torque controller is designed using the derived equations of motion. Further, for the interaction of the scheduling layer operating in Cartesian space with the trajectory generation layer in the joint space, the inverse and forward kinematics of the considered robot manipulator are derived and implemented.

For evaluating the performance of the scheduling and trajectory generation method, a data set of randomly distributed points is generated, representing the fruits on the conveyor belt. By providing the position of the first 12 objects and the robots to the scheduler, the optimization described in Section 3.1 is performed, and the resulting IBLP problem is solved using the GUROBI solver. The minimum distance between two simultaneously chosen objects is set to $d^{\min} = 0.25\,\mathrm{m}$. According to the initial scheduling results shown in Fig. 4*(a)*, the first tray and the object sequence $\{3, 1, 2, 7, 5, 9\}$ are assigned to Robot 1, and the second tray along with the object sequence $\{6, 12, 11, 4, 10, 8\}$ to Robot 2. From the sequences obtained, only the first element is selected. The remaining objects of the respective sequence, which are displayed as dashed bars in Fig. 4*(a)*, are not further considered since the optimization is performed again after the first two objects have been placed. The selected objects define the desired final point for the underlying trajectory generation problem (13), which is solved by applying the full discretization approach. Following this approach, the unknown states, control inputs, as well as the final time are merged into a vector of optimization variables $\boldsymbol{\zeta} = [\boldsymbol{\xi}^T(0) \cdots \boldsymbol{\xi}^T(N)\ \mathbf{u}^T(0) \cdots \mathbf{u}^T(N-1)\ t_f]^T$. Although this method generally leads to a high-dimensional optimization problem, it has the advantage that the gradient of the cost function and the Jacobian of the constraint functions with respect to $\boldsymbol{\zeta}$, as required for the IPOPT solver used here, can be calculated analytically. The sampling time is set to $T_s = 8\,\mathrm{ms}$ according to the control frequency of the used robot arms UR5 from *Universal Robots*. The prediction horizon and the belt velocity are chosen as $N = 15$ and v_b=0.1 m/s. The control horizon T_c of the obtained optimal inputs has not always the length T_s of the sampling time, see Fig. 6. In the beginning, it is larger due to the larger minimum time resulting from MPC, i.e., T_c=$t_f^* \Delta\tau$, and it is decreasing with time when approaching the targets. To grab the first objects 3 and 6, the end effectors will cover a respective distance of 0.31 m and 0.41 m within the resulting minimum time $t_{f_r} = 0.523\,\mathrm{s}$ (see Fig. 4 and 5*(a)*). All distances related to this scenario are shown in Fig. 7 with reference to the base coordinate frame of the first robot. Once the objects have been grasped, the MPC for trajectory generation is carried out again, targeting to reach the first slots in the corresponding trays in the quickest possible time, see Fig. 5*(b)*. After placing the first two objects, the scheduler is executed again, while maintaining the previous robot-tray assignment. As shown in Fig. 4, the object sequence for Robot 1 is not changing compared to the first iteration. For Robot 2, however, there is a change in the object sequence resulting in a shorter overall traversed distance.



(a) Initial Schedule.

(b) Intermediate Schedule.

(c) Final Schedule.

Fig. 4. *Optimal scheduling for two robots, 12 objects and two trays relative to the traversed distance, which is minimized within the optimization problem. The dashed bars show the planed sequence, which has not been applied yet, since the optimization is performed iteratively after each placed object. The white space between the colored bares indicates the distance covered by the robot to pick up the next object. The final schedule shows the resulting sequence of the objects placed in the filled trays after six optimization iterations.*

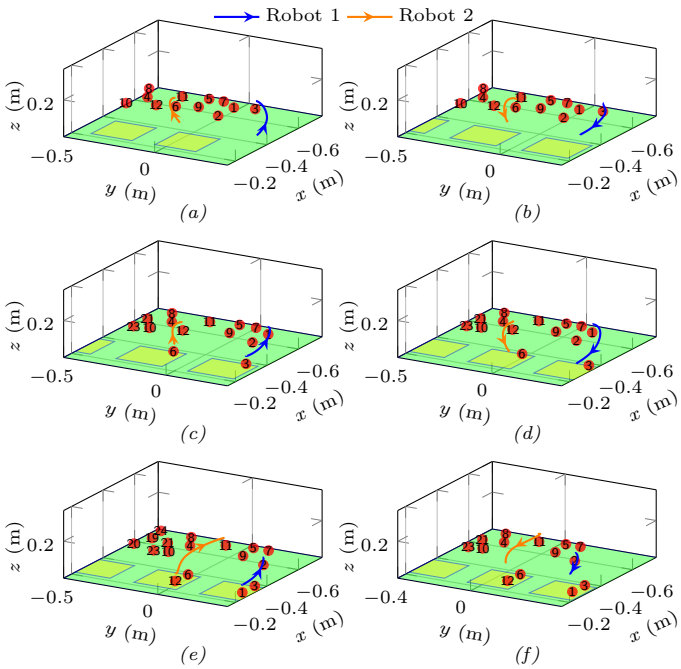


Fig. 5. *Paths of the robots' end effectors. (a): From the starting points to the first objects according to the scheduler results; (b),(d),(f): From the objects to the slots in the corresponding trays; (c),(e): From the slots to the next objects selected by the scheduler. As desired, two simultaneously chosen objects do not lie next to each other.*

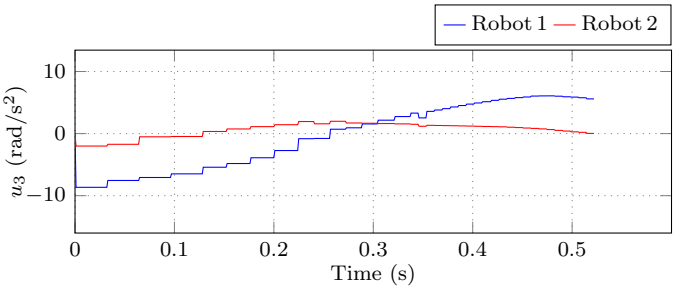


Fig. 6. *Optimal input for the third joints when moving the end effectors from the start position to the first objects. The time-resolution of the input signals is increasing when moving closer to the targets.*

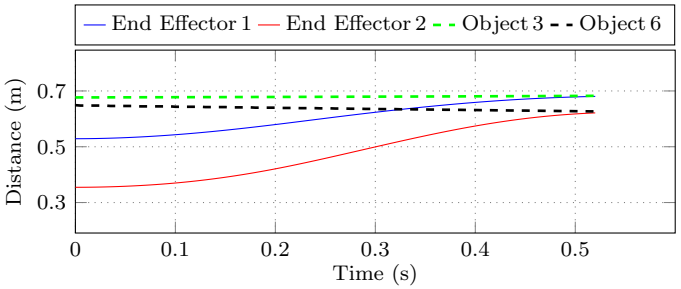


Fig. 7. *Distance of the end effectors and the first objects according to the scheduler. Accordance to the resulting synchronous pick-and-place task performance, the robots reach their targets simultaneously.*

Contrary to the initial plan, the next object to be picked up by Robot 2 is object 12, which is located at a shorter distance than object 11. With the two selected objects, 1 and 12, MPC is performed again, leading the end effectors from the current slots to the objects, see Fig. 5*(c)*. In this way, MPC and scheduler are executed alternately until the trays are filled. The final sequence of selected objects after six optimization iterations is displayed in Fig. 4*(c)*. In the subsequent optimization iteration, each robot is then assigned an object sequence and a tray.

The feasibility of the scheduling model depends mainly on the value of the minimum distance parameter $d^{\min}$ and the related constraints (5). Larger values for $d^{\min}$, as may be preferred for safety reasons, reduce the solution space and the performance of the optimization problem. However, the minimum distance is related to the geometry of the robots and cannot be chosen arbitrarily small. Hence, for larger $d^{\min}$ values, the number of slots $|\mathcal{S}|$ should be much smaller than the number of objects $|\mathcal{N}|$ to reduce the required amount of feasible combinations. The feasibility of the trajectory generation problem depends mainly on the length of the prediction horizon N, as it is required that the robots reach the desired configuration in N steps in the scaled time τ. Shorter prediction horizon would reduce the solution space and may lead to infeasible solutions. On the other hand, a larger prediction horizon would increase the computational effort.

The optimization algorithms are implemented in MATLAB on a standard PC with Intel Core $i5-6600$ Processor and 3.30 GHz clock rate. The GUROBI solver needs, on average, about 0.084 s and 30 simplex iterations to solve for an IBLP problem. On the other hand, minimum-time trajectory computation takes on average about 12 iterations, which corresponds to a CPU time of 0.53 s per MPC step.

5. CONCLUSION

Optimization-based algorithms for cooperative pick-and-place tasks, involving two robots and moving targets are discussed. The hierarchical structure enables an interaction of a scheduling algorithm with a time optimal MPC algorithm for individual robots, that perform the assigned tasks in a synchronous way. For the safe robot tasks execution in a narrow shared workspace, safety-related constraints have been particularly considered in both the scheduling and the trajectory planning layer. Future work will involve the consideration of real-time requirements and validation of the proposed framework in an experimental setup.

REFERENCES

Al Homsi, S., Sherikov, A., Dimitrov, D., and Wieber, P.B. (2016). A hierarchical approach to minimum-time control of industrial robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2368–2374. IEEE.

Ardakani, M.M.G., Olofsson, B., Robertsson, A., and Johansson, R. (2015). Real-time trajectory generation using model predictive control. In Q.S. Jia (ed.), *2015 IEEE Int. Conference on Automation Science and Engineering (CASE)*, 942–948. IEEE, Piscataway, NJ.

Giefer, L., Arango C., J.D., Babr, M.M., and Freitag, M. (2019). Deep learning-based pose estimation of apples for inspection in logistic centers using single-perspective imaging. *Processes*, 7(7), 424.

Rosmann, C., Makarow, A., Hoffmann, F., and Bertram, T. (2017). Time-optimal nonlinear model predictive control with minimal control interventions. In *First Annual IEEE Conference 2017*, 19–24.

Schlemmer, M. and Gruebel, G. (1998). Real-time collision-free trajectory optimization of robot manipulators via semi-infinite parameter optimization. *The International Journal of Robotics Research*, 17(9), 1013–1021.

Spong, M.W., Hutchinson, S., and Vidyasagar, M. (2006). *Robot modeling and control*. Wiley, Hoboken, N.J.

Sutdhiraksa, S. and Zurawski, R. (1996). Scheduling robotic assembly tasks using petri nets. In *Proceedings of IEEE International Symposium on Industrial Electronics*, 459–465. IEEE.

van den Broeck, L., Diehl, M., and Swevers, J. (2009). Time optimal MPC for mechatronic applications. In I. Staff (ed.), *2009 48th IEEE conference on Decision and Control*, 8040–8045. IEEE, Shanghai, China.

Xidias, E.K., Zacharia, P.T., and Aspragathos, N.A. (2010). Time-optimal task scheduling for two robotic manipulators operating in a three-dimensional environment. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 224(7), 845–855.

Zacharia, P.T. and Aspragathos, N.A. (2005). Optimal robot task scheduling based on genetic algorithms. *Robotics and Computer-Integrated Manufacturing*, 21(1), 67–79.

Zhang, Q. and Zhao, M.Y. (2016). Minimum time path planning of robotic manipulator in drilling/spot welding tasks. *Journal of Computational Design and Engineering*, 3(2), 132–139.