2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Towards Integrated OR/CP Energy Optimization for Robot Cells

Oskar Wigström and Bengt Lennartson

***Abstract*— This paper concerns the energy optimization of systems with interacting robots. Pseudo spectral optimal control is used to collocate the continuous time dynamics of each robot. The resulting model is a multi stage, multi robot, trajectory planning problem with timing constraints. In the case of a given task sequence, the mathematical formulation is that of a Nonlinear Programming problem. We present two example problems where three 3-DoF robots work together. For the case the problem consists of undetermined sequences, we outline an algorithm which integrates Operations Research methods with those of Constraint Programming.**

I. Introduction

Today, efficient energy use seems to be an ever present topic within many branches of engineering. Contributing factors are increasing demands on companies adopting environmental policies, as well as steadily increasing prices on oil and electricity. In the field of manufacturing, much work has been done on energy optimization of individual mechatronic devices [2], [3]. Also energy efficient trajectory planning for individual robots [4], [5], [6], as well as energy consumption of individual robots [7], [8] are well investigated.

Even though there is much literature on energy efficiency of individual devices, a wholistic energy minimizing approach for systems consisting of multiple interactive devices is often missing. In this paper, we will show how energy efficient trajectory planning with free paths can be performed from a systems perspective, taking into account multiple robots and their timings. We present two examples where the the task sequence of a number of robots is predetermined, the path of each robot is free. Secondly we outline how to construct an integrated Operations Research (OR) and Constraint Programming (CP) algorithm for the case where the sequence is undetermined.

Even though the literature on energy efficient multi device systems is sparse, there are a few examples. In [9], it is suggested that several robots may be coupled to a common DC-bus. Another approach is [10], where predetermined makespan minimizing task sequences are post processed by two velocity/acceleration balancing methods.

A natural extension of the velocity balancing algorithm in [10] is to consider the energy consumption at the same time as the task sequence is determined. The constraints of a task sequencing (scheduling) problem can be modeled as a Mixed Integer Linear Program (MILP) [11] or a Constraint Program [12]. In the MILP case, real valued variables represent task starting times and durations; boolean variables are used for modeling scheduling disjunctions. In [13], each task trajectory is subject to a linear scaling parameter, allowing the energy cost of each task to be parameterized by its duration. The total energy consumption, the sum of nonlinear parameterized costs, turns the normal MILP scheduling problem into a Mixed Integer Nonlinear Program.

In [14], we extended the method in [13] by preprocessing each trajectory using Dynamic Programming. This preprocessing generates parameterized functions of the optimal energy cost as a function of execution time, in contrast to using a linear scaling parameter, which does not guarantee the optimality of each individual trajectory. Note however, that the individual trajectories in both cases are still constrained to a given path. That is, the robots will follow the same spatial path for each task, but with different velocity and acceleration profiles. Another drawback with these two methods is that the robots stop after each completed task.

In this paper we further extend the results in [14]. The contribution of this paper is: (i) we present a method for using pseudo spectral optimal control in order to minimize energy consumption in a multi robot system; (ii) the presented method considers the entire cell at once while allowing free paths for the robot trajectories; (iii) it is outlined how to construct an integrated OR and CP algorithm which solves the problem with unknown sequences.

We note that there exist works where multiple robots cooporate, as in for example [15], where two robots work together lifting a work piece. However, most work including cooporating robots only consider one task at a time, excluding the multi stage perspective from the problem.

This paper is structured as follows. Section II provides a short background to pseudo spectral optimal control. Next, Section III includes the model for instances with known sequences and provides two illustrative examples. In Section IV an integrated OR/CP method for the problem with unknown sequences is presented. Finally, Section Section V concludes the paper.

II. Optimal Control

An optimal control problem calls for finding a control input for a given system such that an optimality criterion is minimized, all while upholding possible state constraints. A general optimal control problem can be stated as follows: find the states $\mathbf{x}(t) \in \mathbb{R}^n$ and controls $\mathbf{u}(t) \in \mathbb{R}^m$ on the interval $t \in [t_0, t_f]$ that minimizes a cost

This work was carried out at the Wingquist Laboratory VINN Excellence Center within the Area of Advance – Production at Chalmers, supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) and the Swedish Research Council. The support is gratefully acknowledged.

O. Wigström, B. Lennartson, Automation Research Group, Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden, oskar.wigstrom@chalmers.se

$$J = \Phi(\mathbf{x}(t_f), tf) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt, \qquad (1)$$

where $\Phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is the terminal cost, and $g : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ is the integrated cost. The states are subject to the first-order dynamic constraints,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \qquad (2)$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$. The states may also be subject to boundary conditions

$$\phi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) = 0, \qquad (3)$$

where $\phi : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^q$. Finally, the states may be subject to algebraic path constraints

$$b(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0, \qquad (4)$$

where $b : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^r$.

A. Numerical methods Optimal Control

Most often optimal control problems tend to be nonlinear, and therefore generally lack analytic solutions. This has lead to a number of numerical methods for solving optimal control problems. These methods fall into two categories: indirect methods and direct methods. Indirect methods apply calculus of variations to determine the first-order necessary optimality conditions. Direct methods approximate the continuous time problem by a finite-dimensional nonlinear program (NLP). Optimization of a hybrid systems by Mixed Integer Nonlinear Programing (MINLP) or Generalized Disjunctive Programing is suggested in for example [16].

In this paper we will use direct methods to model the continuous states of the problem. For a detailed introduction to indirect and direct methods, we refer to [17]. The reason for using direct methods in this paper is simply that while indirect methods provide an accurate solution and a good optimality measure, they suffer from several disadvantages. For example, the implementation requires derivation of complicated first-order optimality conditions; also a very good initial guess is required. Within the class of direct methods, there are several approaches available, we have opted for the Gauss pseudospectral method [18]. The following will provide a brief description.

The Gauss pseudospectral method differs from other pseudospectral methods in that the differential equation is not collocated at the boundary points. Also, the approximating polynomial, $\mathbf{X}(t)$, is of degree N, such that $N + 1$ points are required to determine its derivative.

Let $L_k(t)$, $(k = 0, ..., N)$, be a basis of Lagrange polynomials [19] on the interval from $[-1, 1]$ given by

$$L_k(t) = \prod_{\substack{i=0 \\ i \neq k}}^{N} \frac{t - t_i}{t_k - t_i}, \qquad k = 0, ..., N. \qquad (5)$$

We see that

$$L_k(t_i) = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{if } k \neq i \end{cases} \qquad (6)$$

The state vector $\mathbf{x}(t)$ is approximated by $\mathbf{X}(t)$ as

$$\mathbf{x}(t) \approx \mathbf{X}(t) = \sum_{k=0}^{N} \mathbf{x}(t_k) \cdot L_k(t). \qquad (7)$$

Recall that the polynomial is defined on the interval [-1,1]. As such, the interpolation points used are the boundary point, -1, and the N Gaussian quadrature points, t_k, $k = 1, \cdots, N$. Thus, by (6)

$$\mathbf{X}(t_k) = \mathbf{x}(t_k), \quad (k = 0, ..., N). \qquad (8)$$

Differentiation of (7) yields the following expression for the approximated state derivatives $\dot{\mathbf{X}}(t_i)$

$$\dot{\mathbf{x}}(t_i) \approx \dot{\mathbf{X}}(t_i) = \mathbf{x}(t_0) \cdot \bar{D}_i + \sum_{k=1}^{N} \mathbf{x}(t_k) \cdot D_{ik}, \qquad (9)$$

where $D_{ik} = \dot{L}_k(t_i)$ and $\bar{D}_i = \dot{L}_0(t_i)$.

The NLP resulting from transcription of the optimal control problem in (1)-(4) is as follows. The cost (1) is approximated using a Gauss quadrature such that

$$J = \Phi(\mathbf{X}(t_f), t_f) + \frac{(t_f - t_0)}{2} \sum_{k=1}^{N} w_k \cdot g(\mathbf{X}(t_k), \mathbf{U}(t_k), t_k), \qquad (10)$$

where $\mathbf{U}(t_k)$ is an $N - 1$ degree approximation of the controls $\mathbf{u}(t)$ collocated at the Gauss points; w_k are the gauss weights; $\frac{(t_f - t_0)}{2}$ comes from transforming the problem from $t \in [-1, 1]$ into $[t_0, t_f]$. The first-order dynamics in (2) are then approximated by

$$\frac{2}{(t_f - t_0)} \bar{D}_i \cdot \mathbf{X}(t_0) + \frac{2}{(t_f - t_0)} \sum_{k=1}^{N} D_{ik} \cdot \mathbf{X}(t_k) = \mathbf{f}(\mathbf{X}(t_k), \mathbf{U}(t_k), t_k), \quad i = 1, \cdots, N. \qquad (11)$$

The boundary constraints (3) is enforced as

$$\phi(\mathbf{X}(t_0), t_0, \mathbf{X}(t_f), t_f) = 0 \qquad (12)$$

and the algebraic path constraints (4) as

$$b(\mathbf{X}(t_k), \mathbf{U}(t_k), t_k) \leq 0, \quad k = 1, \cdots, N. \qquad (13)$$

Finally, the terminal states are enforced by quadrature approximation of the dynamics as

$$\mathbf{X}(t_f) = \mathbf{X}(t_0) + \frac{(t_f - t_0)}{2} \sum_{k=1}^{N} w_k \cdot \mathbf{f}(\mathbf{X}(t_k), \mathbf{U}(t_k), t_k). \qquad (14)$$

The decision variables for the problem are the states, $\mathbf{X}(t_k)$, controls $\mathbf{U}(t_k)$, initial time t_0 and final time t_f.

III. KNOWN SEQUENCES

We will now describe how to pose the NLP formulation for systems with given sequences. To clarify, given sequences imply that each robot has a predetermined order in which is should perform a set of tasks. The robots' tasks may be coupled by timing constraints; either by equalities or inequalities. There may also be coupling in the form of boundary constraints on the continuous states.

For the examples in this paper, we have chosen to consider three link manipulators moving in 3D space. The first link revolves around the z-axis while the second two are in a vertical plane. Each joint is modeled as a second-order integrator with acceleration as control input. That is, the dynamics of each robot is given by

$$\begin{bmatrix} \dot{x}_{1j} \\ \dot{x}_{2j} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_{1j} \\ x_{2j} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_j, \quad j = 1, 2, 3, \tag{15}$$

where x_{1j}, x_{2j} and $\dot{x}_{2j}$ are the angular position, velocity and acceleration of joint j, and u_j is the control input. While it is possible to include more intricate dynamics, e.g. inertia, friction, gravity, we consider this outside the scope of this paper.

The overall objective is to minimize the weighted integral of squared inputs over all tasks. The cost of each individual task is expressed as

$$\int_{t_0}^{t_f} \left(\sum_{i=1}^{3} u_j \right) dt, \tag{16}$$

where q_j is a positive weight, $j = 1, \cdots, 3$.

State boundary conditions are posed for sequenced tasks processed by the same robot. Consider two tasks a and b, where b should be performed after a. We will denote the state vector at the collocation points for the two tasks $\mathbf{X}^a(t_k^a)$ and $\mathbf{X}^b(t_k^b)$; where t_k^a and $t_k^b$the time indices for tasks a and b. The boundary condition for the two tasks is expressed by

$$\mathbf{X}^a(t_f^a) = \mathbf{X}^b(t_0^b). \tag{17}$$

There are of course also timing constraints added such that

$$t_f^a = t_0^b. \tag{18}$$

In the case of two sequenced tasks processed by different robots, the timing constraint may also take the form of an inequality, and in which case no boundary constraints on the states are added

$$t_f^a \leq t_0^b. \tag{19}$$

The tool center point and other link positions can be computed via forward kinematics. Let us denote the spatial position of the end point of link j, $j = 1, \cdots, 3$, at time t_k during task a, performed by robot h by the nonlinear transition function $T_j^h(\mathbf{X}^a(t_k))$. Using the transition function we can impose boundary constraints on the tool center point by

$$T_3^h(\mathbf{X}^a(t_f^a)) \in \mathcal{S}, \tag{20}$$

where $\mathcal{S}$ is some desired boundary set, for example the entrance to a common zone or the position of some object to be processed.

Furthermore, it may be the case that a task takes a robot close to a forbidden zone, in which case we constrain all link positions to be within a safe set $\mathcal{S}$ during the task

$$T_j^h(\mathbf{X}^a(t_k^a)) \in \mathcal{S}, \quad k = 1, \cdots, N, \quad j = 1, 2, 3. \tag{21}$$

Also, in a case where one robot should hand an object to another robot via a common zone, and the hand off position of the object is undecided, we impose a constraint which sets the tool center point from the hand off task and pick up task equal. For two robots h and m, which should be at the same position at the end of tasks a and b, we impose

$$T_3^h(\mathbf{X}^a(t_f^a)) = T_3^m(\mathbf{X}^b(t_f^b)). \tag{22}$$

Finally, there are linear constraints constraining the cycle time and defining initial and final conditions on the continuous states.

A. Example 1 and 2: 3 robots in line formation

Our first example consists of three robots, [R1,R2,R3], placed on a straight line in the x/y-plane. The objective is for the robots is to move three objects from their initial positions next to R1, to their final positions next to R3. The objects are to be passed via R2. We present two versions of this example: in the first, Example 1, robots must pass the objects by direct interation; in the second, Example 2, the objects must be placed in mutual exclusion zones located between the robots.

In Example 1, the robots can be regarded as a number of synchronized systems with equality constraints on certain switching times and equality constraints on the tool center points. The direct hand off of the object may occur anywhere reachable by both robots. Example 2 on the other hand contains inequality constraints on switching times, to maintain mutual exclusion, and equality constraints on the tool center points. The objects must be placed within the common zones. The instances are similar, but in the first version, the synchronization between systems is much tighter.

Also, the number of tasks for Example 2 is much higher than for Example 1. This is because movement from one point via a common zone barrier means that to and end point must be modeled as two tasks. To somewhat limit the number of tasks for Example 2, the second object is not passed by R2 to R3, but instead placed in front of R2. In Example 1, all robots have 7 tasks, two for each object and one additional to move into an idea position. For Example 3, the middle most robot, R2, has as many as 16 tasks. This is because it moves in and out of two common zones. To exemplify, R2 requires 6 tasks to model its interactions with object one: (i) moving into zone one; (ii) picking up object one; (iii) moving out of zone one; (iv) moving into zone two; (v) placing object one; (vi) moving out of zone two.

B. Example 3: 3 robots in triangular formation

In the third example, Example 3, the three robots are placed in a pattern pattern with two mutual exclusion zones. One accesible by all three robots, and one accesible by only R2 and R3. This time, four objects should be passed from initial to final locations. In constrast to the previous example, all robots have an initial and final object location next to it. This example can be regarded as hybrid systems where one particular, feasible, discrete sequence is to be optimized.

In short, the four objects take the following routes through the system, [R1→Z1→R2→Z2], [R3→Z1→R1], [R2→Z1→R1] and [R3→Z2], where Z1 and Z2 indicate common zones one and two. The known sequence can be summed up as R3 preceding R1, followed by R2 into Z1, and R2 preceeding R3 in Z3.

C. Benchmarks

Based on the examples above, we have generated a number of benchmarks to illustrate the computational properties of the model. All optimization was run on a Windows 7 64bit system with a 2.66 [GHz] Intel Core2 Quad CPU and 4 [GB] of RAM with Ipopt (v3.8.1) as NLP solver. As previously mentioned the NLP model is sensitive to the initial guess. Also, it is sensitive to the scaling of the control signal. Most instances will run fine as is, while a handfull, presumably due to faulty initial step sizes, would get stuck in local minima or turn out infeasible. We noticed that a uniform scaling of the control signal avoids this problem in most cases. This, for some instances, minor tweaks of the control signal scaling were performed. To provide a rough idea of the problem size: at $N = 10$, the number of varibles and constraints were $[1812, 1341]$, $[4524, 3217]$ and $[2896, 2079]$ for Example $1-3$.

With the cycle time fixed to 4 [s] the solution time varied with the number of collocation points as shown in Figure 1, and the fraction of the true optimal cost can be seen in Figure 2. Note that the solution for a low number of collocation points underestimates the true solution for a low number of collocation points. This is because no guarantees are given for the system behaviour inbetween collocation points. The largest problem (constraint-wise) is Example 2, it is also the hardest to solve. However, it has approximatly the same accuracy at $N = 4$, as Example 1 and Example 3 has at $N = 5$ and $N = 6$. This suggests that predictable accuracy of the solution might be achieved by choosing the number of collocation points relative the distance the manipulator travels performing a task.

Comparing fixed cycle time and cost on control with weighted cost on cycle time and control, with $N = 9$ for Example 3, resulted in the following benchmark. Four different scaling factors were used for the control input and for the weighted cycle time, an initial guess of 4[s] was given. A total of 25 istances with varied fixed cycle time and varied weights were run for each scaling factor, making a total of 100 instances with cycle times ranging between $[0.5, 12]$. With fixed time, the mean and median solution times were 27[s] and 22[s], while the weighted cycle time performance

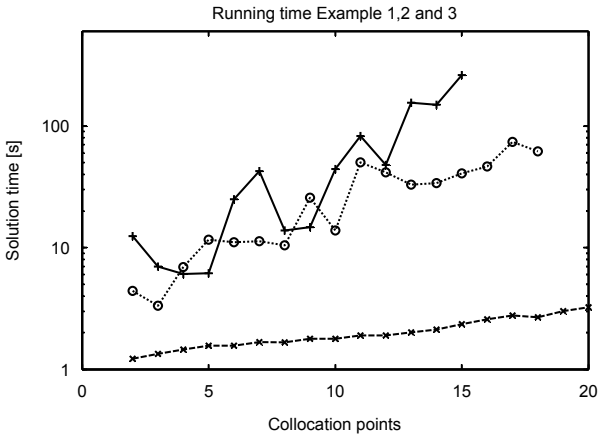

Fig. 1. Running time using varied number of collocation ponts for examples Example 1 (x), 2 (+) and 3 (o).

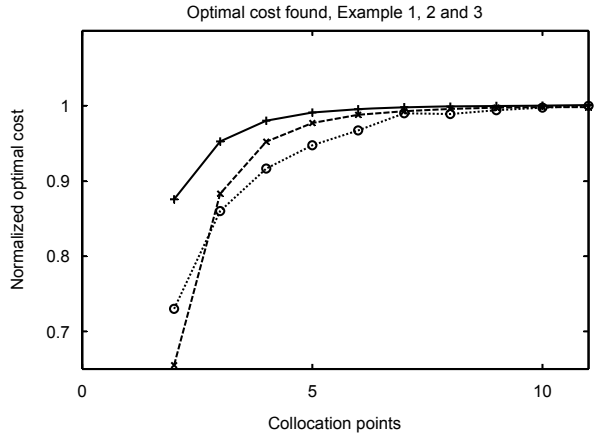

Fig. 2. Fraction of actuall normalized optimal solution found for Example 1 (x), 2 (+) and 3 (o). Bounds may be violated inbetween collocation points, and because of this, a low number of collocation points may yield solutions with lower cost.

was 26[s] and 30[s]. The fixed cycle time model failed to find the optimal solution in 7 cases while the free cycle time failed conciderably more often. The free cycle time model worked sufficiently well when the final time was close to the initial guess of 4 [s]. However, most instances with an optimal cycle time over 7 [s] failed.

For the fixed cycle time, given a sufficiently good initial guess, the correct optimal solution is found in most cases. The initial guesses used for the two examples were made in the following way. Rough estimates for the angular positions of the initial and final time of each task were made. The angular positions of the intermediate collocation points were made by linear interpolation between the initial and final positions. The angular velocities and accelerations were set to zero. Adding an automatic generation of initial guesses, including feasible values for speed and acceleration, is expected to increase performance.

IV. Unknown sequences

A scheduling problem where the task sequences are unknown can be considered as having two parts, one discrete and one continuous. The discrete part consists of determining the order of tasks which have been defined as disjunct, i.e. one must end before the other may start. The continuous part is the matter of determining the timing and evolution of continuous dynamics in each operation such that an optimality criterion is fulfilled. That is, the discrete part entails generating a feasible sequence, and the continuous part solving the resulting 'known sequence' problem in Section III.

In classic scheduling applications, this criterion is for example the minimum makespan or minimum tardiness, and the continuous dynamics are nonexistent. In other words, it is enough to determine the starting time of each operation; the duration is most often fixed. This type of problem can either be solved by Mixed Integer Linear Programming [11] or Constraint Programming [12].

In [14], we showed how the scheduling problem can be extended to take into account the trajectories of each individual process (robot). The trajectory of each task was preprocessed by Dynamic Programming such that a nonlinear energy cost as a function of execution time was created. The optimality criterion for the scheduling problem was then to minimize the sum energy cost for the active operations. The resulting problem falls in the category of MINLP. A benchmark of state of the art MINLP methods suggest that LP/NLP-based Branch and Bound and Extended Cutting Plane methods should be used for optimization of this type of nonlinear scheduling problem [20].

The MINLP framework does have some limitations when it comes to scheduling. Relaxations are weak and infeasibilities stemming from the discrete part may not be detected until deep down in the branches and bound search tree. When the size of the node subproblems (NLPs) grow, this becomes a large problem; especially when optimality is to be proven. In [21], inspired by [22], [23], we showed how OR methods can be integrated with CP such as to eliminate infeasible branches at an early stage in MINLP problems.

In this paper the scheduling problem has been further extended by including collocated robot dynamics. For the collocation approach to provide a good solution, or any at all, a good initial guess is required for each node in the MINLP branch and bound tree. Even though an initial guess may be quickly determined by heuristics, including initial guess heuristics into "out of the box" MINLP algorithms is not always straight forward. This, in combination with the remarks in the previous paragraph leads us to suggest extending the integrated framework presented in [21] such as to solve the scheduling problem with collocated dynamics. The following will outline the structure of such an integrated algorithm.

A. Integrated Algorithm

As metioned, the algorithm is inspired by the search-infer-and-relax framework in SIMPL (Programming Language for Solving Integrated Models), a system for integrating optimization techniques [23]. While the possibility of integrating CP with NLP is briefly mentioned in SIMPL, it is not discussed in any detail.

In short, the algorithm will cycle through three phases, search, infer and relax. Pseudocode for the algorithm can be seen in Algorithm 1. During the search phase, the algorithm branches on the disjunctive scheduling constraints. Each branching results in two new nodes which are processed by CP inference methods. There are a large number of inference methods available for scheduling problems, but none for continuous time optimal control. Therefore, the CP inference part will focus solely on the scheduling constraints doing so, providing domain reduction, and possibly an infeasibility proof. If the node turns out to be feasible after inference, the algorithm proceeds with the relax phase. During this phase, the disjunctive scheduling constraints are removed, and a sub-problem is created. This sub-problem includes: (i) linear scheduling constraints, e.g. sequencing constraints, cycle time constraints; (ii) restrictions from the search phase, in the form of sequencing constraints; (iii) the collocated dynamics of the system. Solving the sub-problem in a branch node yields a lower bound for the current branch, while solving in a terminal node gives an upper bound for the global cost.

Note that, the subproblems in branch nodes may not necessarily be NLP, but also LP. It is suggested in [20], that for scheduling problems with nonlinear cost and scheduling constraints, LP/NLP-based Branch and Bound (LP/NLP-BB) is the most efficient MINLP solution method. The LP/NLP-BB algorithm uses successive linearization and will either solve an NLP or an LP depending on which ruleset is selected. Our implementation of the integrated algorithm in [21] mimics this behaviour; it solves linearized problems in branch nodes and NLPs in terminal nodes. An impor-

Algorithm 1 Integrated Algorithm

```
1:  UB ← ∞
2:  NodeSet ← Top node
3:  while 0 < |NodeSet| do                ▷ While any nodes left
4:      n ← NodeSet                       ▷ Get the next node
5:      n ← Inference(n)                  ▷ Use CP for inference
6:      if n is feasible then
7:          s ← GenerateSubproblem(n)
8:          cost ← Solve(s)               ▷ Solve LP or NLP
9:          if cost < UB then
10:             if n is branch node then
11:                 NodeSet ← Branch(n)
12:             end if
13:             if n is leaf node then
14:                 UB = cost             ▷ New best found
15:             end if
16:         end if
17:     end if
18: end while
```

tant feature of the integrated method is that the linearized problems, in contrast to the LP/NLP-BB, will only include linearizations which are relevant to the current node. When the LP/NLP-BB linearizes its MINLP in one node, these linearizations will still be present in all future nodes unless explicitly removed.

B. Implementation

As in [21], we propose extending an existing CP solver by adding the OR parts by means of a custom constraint propagator.

In CP, specialized scheduling constraints and propagators are used to model and solve scheduling problems. Two constraints, present in many CP solvers, are the disjunctive and alternative constraint. A basic disjunctive scheduling constraint [22], is

$$\text{disjunctive}(\mathbf{s}, \mathbf{p}), \tag{23}$$

where $\mathbf{s} = (s_1, ..., s_n)$ is a tuple of variables s_j indicating the start time for each task j. The parameter $p = (p_1, ..., p_n)$ is a tuple of processing times p_j for each task. The constraint enforces that for any pair of tasks, one must finish before the other. That is, it imposes the disjunctive condition

$$(s_k + p_k \leq s_l) \vee (s_l + p_l \leq s_k) \tag{24}$$

for all k,l where $k \neq l$.

The minimum processing time p_j for each task should be precomputed/inferred. In some problems there may not exist an explicit minimum time. This could be the case if for example no constraints on acceleration or control signal are present. Once a candidate solution is found however, it may be possible to deduce bounds on the minimum task time for which the cost obviously becomes too large to be feasible.

The second constraint we mentioned is the alternative constraint, a basic version can be stated as follows

$$\text{alternative}(s_0, \mathbf{s}, n), \tag{25}$$

where s_0 is some start time, $\mathbf{s} = (s_1, ..., s_n)$ is defined as before and n is integer, $1 \leq n < |s|$. The alternative constraint expresses that if start time s_0 is present, i.e. it has not been disabled by some other alternative, then n out of the $\mathbf{s}$ start times should be present. If a start time is not present, then any constraints associated to it should be void, or modified accordingly.

The CP model fed into the algorithm will take the following form:

$$\begin{aligned} \min_{\mathbf{t}_0, \mathbf{t}_f} \quad & \eta \\ \text{s.t.} \quad & \text{disjunctive}(\mathbf{s}_d^i, \mathbf{p}^i) \quad i = 1, \cdots, n_d \\ & \text{alternative}(s_0^j, \mathbf{s}_a^j) \quad j = 1, \cdots, n_a \\ & A_1 \mathbf{t}_0 + A_2 \mathbf{t}_f \leq b \\ & \min_{\mathbf{x}, \mathbf{t}_0, \mathbf{t}_f} \; SP \leq \eta \end{aligned} \tag{26}$$

where $\mathbf{s}_d^i$ and $\mathbf{p}^i$ are n_d subsets of starting times with corresponding minimum durations modeling scheduling disjunctions; s_0^j and $\mathbf{s}_a^j$ are n_a starting times and subsets of starting times modeling scheduling alternatives; A_1, A_2 and b form linear constraints expressing task duration, maximum cycle time and preset task sequencing; finally min $SP \leq \eta$ requires the cost η to be larger than the solution of the OR sub-problem, SP. A solution to SP is given by forming and minimizing either an LP or an NLP.

In constraint programming, the propagation phase will schedule constraints for propagation in an order based on the computational cost of propagation as well as its impact on variable domains. Because the subproblem generally only affects the cost η, and is relatively computationally expensive to run, it is scheduled last. The CP engine is set to branch on the disjunctive constraints, fixating the order of disjunctive tasks in each branch.

The NLP generation is performed as desribed in Section III. Any restrictions from the search phase branching on the disjunctive and alternative constraints are taken into account. Based on possible alternatives between tasks, boundary constraints between each task should be added to ensure state continuity. The LP-problems are similar, but the nonlinear collocation constraints are instead iterativley linearized. Because of the non-convex nature of the collocation constraints, extra care must be taken with the linearizations.

As previously mentioned, the collocation method may be sensitive to initial conditions. For a robotic application, we suggest pre-computing approximate angles at boundary points and using linear interpolation for the initial guess of intermediate points. From our experience, this is enough to yield a solution. Solutions from previous nodes may very well be used, but only as long as the boundary conditions are approximately the same, and only the timing between robots have changed.

C. Example 4: Alternative sequences

Consider Example 3 from Section III-B, where four objects, [O1 O2 O3 O4], are to be passed through a system using a known sequence. A corresponding unknown sequence problem also requires specifying in which order the robots should be allowed into the common zones. Furthermore, the problem may alternative sequences, where a change of route for the objects through the system would be an option.

In Example 3, the utilization of Z1 can be summed up as [R3, R1, R2, R1]. In more detail: R3 leaves O2; R1 leaves O1 and picks up O2; R2 leaves O3 and picks up O1; R1 picks up O3. Instead, in this example Z1 is utlilized as [R2,R1,R2,R3,R1], where: R2 leaves O3; R1 picks up O3 and leaves O1; R2 picks up O1; R3 leaves O2; R1 picks up O2. That is, R2 will enter the zone an additional time as it first exists the zone to wait for its object (O1) to be dropped off by R1. A cost comparison between the two examples is shown in Figure 3, where the sequence in this example has an average 30% higher cost.

To evaluate all feasible sequences, the finalized algorithm requires an automatic procedure for generation of initial

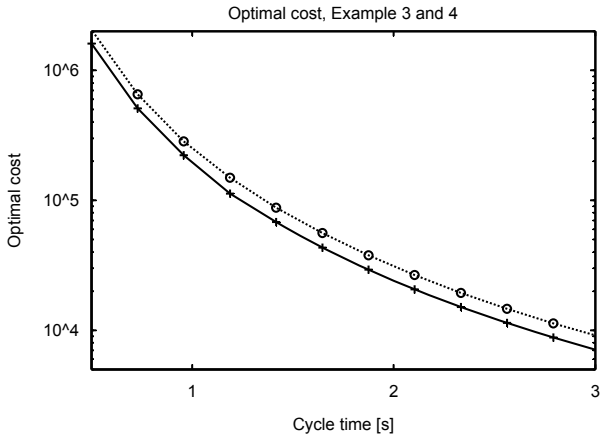

Fig. 3. Optimal cost for Example 3 (+) and 4 (o) for varied fixed cycle time. Example 3 has approximatley 30% lower cost overall.

conditions in each subproblem. One option is to solve a simple trajectory planning problem for each task in order to get a good initial point. A feasible scheduling solution from the CP engine may be used to provide initial guesses for the timing of tasks. The algorithmic performance is expected to be improved with the quality of the initial guess.

V. Discussion and Conclusion

In this paper we have presented an optimization model for the multi robot, fixed sequence trajectory planning problem. The model is based on collocation and takes the form of a nonlinear programming problem. Two examples are presented, and it is shown that considering all robots at the same time while planning trajectories is a tractable approach. It is also outlined how the same problem but with unknown sequences can be solved using an integrated CP/OR algorithm.

One unfortunate drawback of the model, is its sensitivity to an initial guess and scaling of decision variables. Future work will include a robust method for generating NLPs such that the presented integrated algorithm can be implemented. It is also important to compare the Gauss pseudospectral method to other collocations methods, such as to identify the best method specifically for the multi robot, fixed stage, trajectory planning.

References

[1] Philip E Gill, Walter Murray, and Margaret H Wright. Practical optimization. 1981.

[2] R. Saidur. A review on electrical motors energy use and energy savings. *Renewable and Sustainable Energy Reviews*, 14(3):877 – 898, 2010.

[3] R. Visinka. *Ch. 2 - Energy Efficent Three-Phase AC Motor Drives for Appliance and Industrial Applications*. Goldberg and Middleton, 2002.

[4] A. Gasparetto and V. Zanotto. Optimal trajectory planning for industrial robots. *Advances in Engineering Software*, 41(4):548 – 556, 2010.

[5] J. S. Park. Motion profile planning of repetitive point-to-point control for maximum energy conversion efficiency under acceleration conditions. *Mechatronics*, 6(6):649 – 663, 1996.

[6] E.S. Sergaki, G.S. Stavrakakis, and A.D. Pouliezos. Optimal robot speed trajectory by minimization of the actuator motor electromechanical losses. *J. Intell. Robotics Syst.*, 33:187–207, Feb. 2002.

[7] A. Rassolkin, H. Hoimoja, and R. Teemets. Energy saving possibilities in the industrial robot irb 1600 control. In *Compatibility and Power Electronics (CPE), 2011 7th International Conference-Workshop*, pages 226–229, 2011.

[8] Z. Kolíbal and A. Smetanová. Experimental implementation of energy optimization by robot movement. In *Robotics in Alpe-Adria-Danube Region (RAAD), 2010 IEEE 19th International Workshop on*, pages 333–339, 2010.

[9] D. Meike and L. Ribickis. Energy efficient use of robotics in the automobile industry. In *Advanced Robotics (ICAR), 2011 15th International Conference on*, pages 507–511, 2011.

[10] A. Kobetski and M. Fabian. Velocity balancing in flexible manufacturing systems. In *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on*, pages 358 –363, may 2008.

[11] Alan S. Manne. On the job-shop scheduling problem. *Operations Research*, 8(2):pp. 219–223, 1960.

[12] Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-based scheduling: applying constraint programming to scheduling problems*, volume 39. Springer, 2001.

[13] Alberto Vergnano, Carl Thorstensson, Bengt Lennartson, Petter Falkman, Marcello Pellicciari, Francesco Leali, and Stephan Biller. Modeling and optimization of energy consumption in cooperative multi-robot systems. *Automation Science and Engineering, IEEE Transactions on*, 9(2):423–428, 2012.

[14] O. Wigström, B. Lennartson, A. Vergnano, and C. Breitholtz. High level scheduling of energy optimal trajectories. *IEEE Transactions on Automation Science and Engineering*, 2013.

[15] Devendra P. Garg and Manish Kumar. Optimization techniques applied to multiple manipulators for path planning and torque minimization. *Engineering Applications of Artificial Intelligence*, 15(3-4):241 – 252, 2002.

[16] Jan Oldenburg and Wolfgang Marquardt. Disjunctive modeling for optimal control of hybrid systems. *Computers and Chemical Engineering*, 32(10):2346 – 2364, 2008.

[17] Divya Garg. *Advances in global pseudospectral methods for optimal control*. PhD thesis, University of Florida, 2011.

[18] David Benson. *A Gauss pseudospectral transcription for optimal control*. PhD thesis, Massachusetts Institute of Technology, 2005.

[19] P.J. Davis. *Interpolation and approximation*. Dover publications, 1975.

[20] O. Wigström and B Lennartson. Sustainable production automation - energy optimization of robot cells. In *Robotics and Automation, 2013 IEEE International Conference on, to be published*, 2013.

[21] O. Wigström and B Lennartson. Integrated or/cp optimization for discrete event systems with nonlinear cost. In *Decision and Control, 2013 IEEE Conference on, to be published*, 2013.

[22] John Hooker. *Integrated methods for optimization*, volume 100. Springer, 2007.

[23] Ionut D. Aron, John N. Hooker, and Tallys H. Yunes. Simpl: A system for integrating optimization techniques. In Jean-Charles Régin and Michel Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, First International Conference, CPAIOR 2004, Nice, France, April 20-22, 2004, Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2004.