Weierstrass Institute for
Applied Analysis and Stochastics

Leibniz Association

Automatic reconfiguration of robotic workcells

Chantal Landry

Joint work with: M. Gerdts*, R. Henrion, D. Hömberg, M. Skutella** and W. Welz**

Industrial partner: Rücker EKS GmbH

* University of the Federal Armed Forces Munich, Germany.

** Technical University of Berlin, Germany.

Berlin, 08.05.2014

Content

1 **Workcell Problem**
- Motivation
- Definition
- Workcell algorithm
- Numerical result

2 **Time-optimal kinodynamic motion planning**
- Model
- Numerical method
- Initialization
- Numerical results

3 **Conclusions**

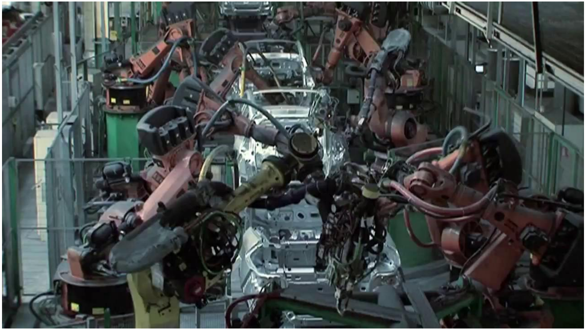
1 **Workcell Problem**
- Motivation
- Definition
- Workcell algorithm
- Numerical result

2 Time-optimal kinodynamic motion planning

3 Conclusions

Facts:

- High degree of automation
- Production line consists of workcells
- A workcell:
  - 1 workpiece
  - several robots
  - tasks
  - some obstacles



Example: automotive industry

Challenges

- optimal configuration of the workcells
- automatic reconfiguration

Optimal configuration of a workcell

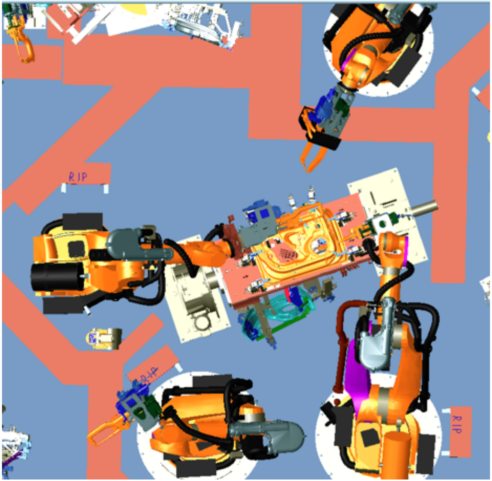


Image courtesy of Rücker EKS GmbH

To be determined:

- Which robot does perform which tasks?

  → **Task assignment**

- In which order?

  → **Sequencing**

- How does a robot move between 2 task locations?

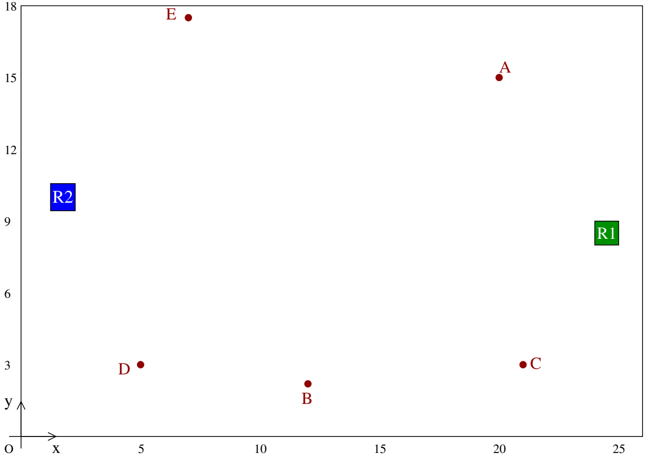
  → **Motion planning**

Such that:

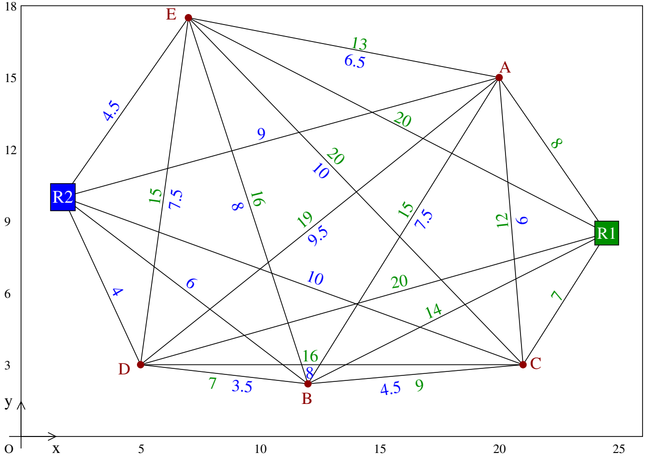
the total time taken to complete all the tasks is **minimal**

Who does what? In which order? What is the fastest trajectory?

2 robots: R1 and R2 / 5 tasks: A to E / R2 moves twice faster than R1



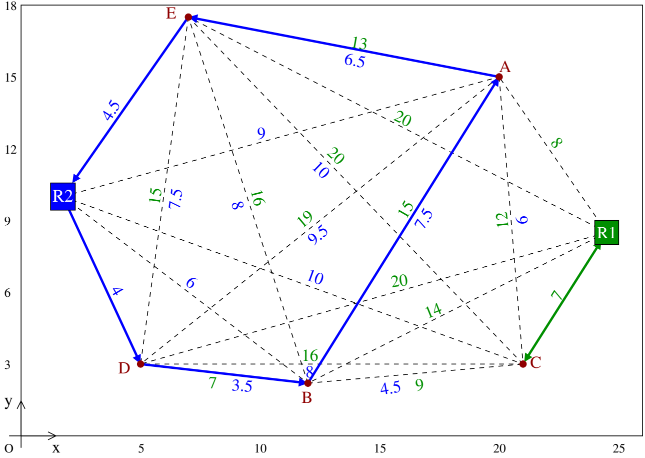
2 robots: R1 and R2 / 5 tasks: A to E / R2 moves twice faster than R1



we need the **travel times**

Who does what? In which order? What is the fastest trajectory?

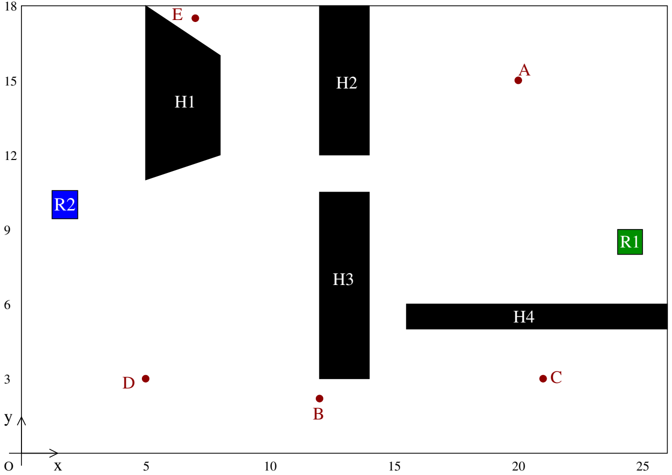
2 robots: R1 and R2 / 5 tasks: A to E / R2 moves twice faster than R1



discrete optimization: vehicle routing problem

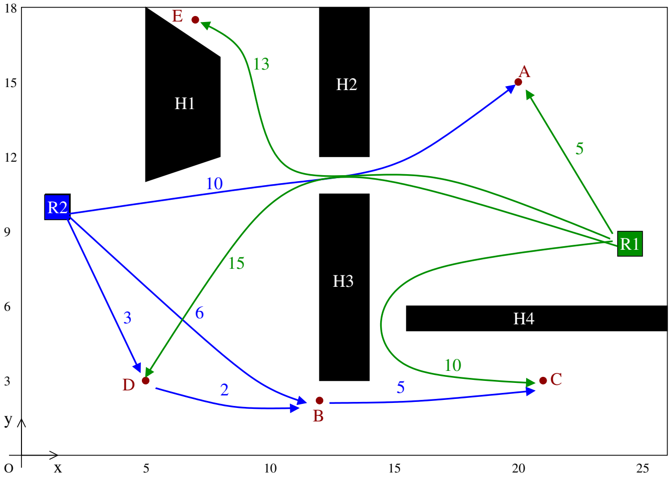
total time: max(4+3.5+7.5+6.5+4.5, 7+7)=26 s

Add the obstacles $H1$ to $H4$ into the workcell



Who does what? In which order? What is the fastest trajectory?

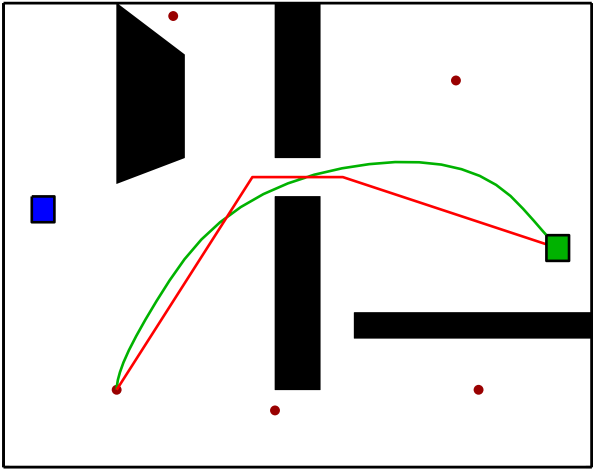
Trajectories are more difficult to compute!



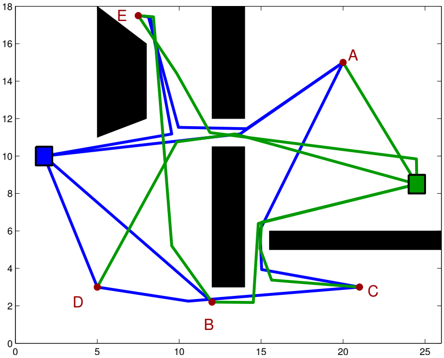
Who does what? In which order? What is the fastest trajectory?

Calculating all the fastest collision-free trajectories is expensive

⇒ at first use approximated trajectories

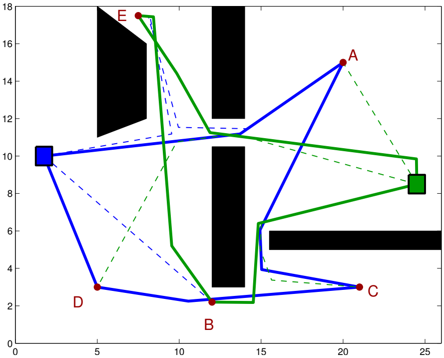


1. Compute the approximated trajectories



Workcell Algorithm

1. Compute the approximated trajectories
2. Find the optimal sequences

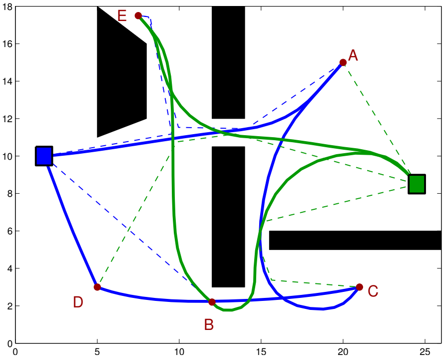


Robot R1: Start → E → B → Start

Robot R2: Start → A → C → D → Start

Workcell Algorithm

1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet

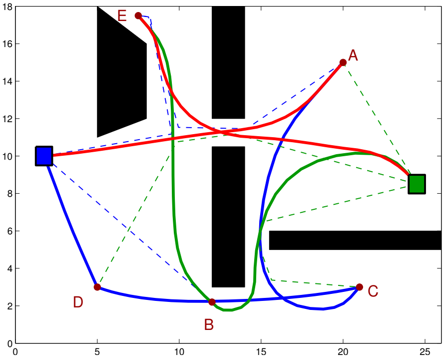


Robot R1: Start → E → B → Start

Robot R2: Start → A → C → D → Start

Workcell Algorithm

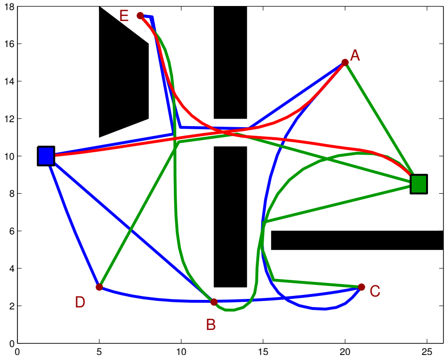
1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet
4. **If** there is a collision, **then**
   (i) New constraint: these two trajectories cannot be used simultaneously



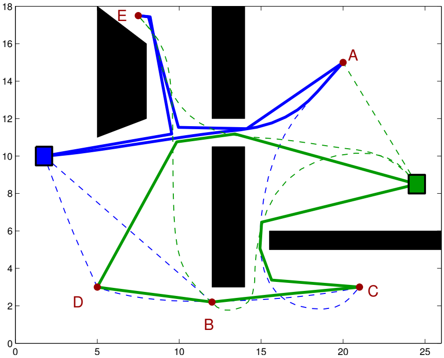
Robot R1: Start → E → B → Start

Robot R2: Start → A → C → D → Start

1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet
4. **If** there is a collision, **then**
   - **(i)** New constraint: these two trajectories cannot be used simultaneously
   - **(ii)** go to 2



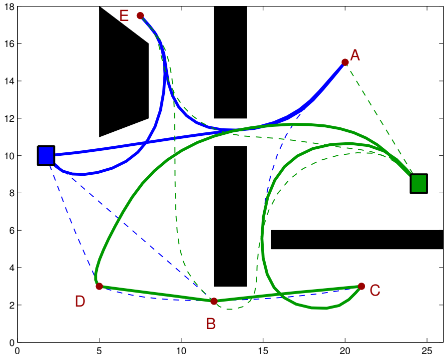
1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet
4. **If** there is a collision, **then**
   (i) New constraint: these two trajectories cannot be used simultaneously
   (ii) go to 2



Robot R1: Start → C → B → D → Start

Robot R2: Start → A → E → Start

1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet
4. **If** there is a collision, **then**
   (i) New constraint: these two trajectories cannot be used simultaneously
   (ii) go to 2



Robot R1: Start → C → B → D → Start

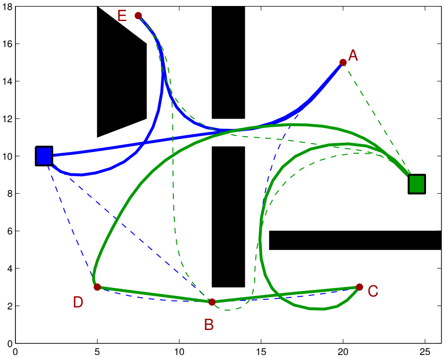
Robot R2: Start → A → E → Start

Workcell Algorithm

1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet
4. **If** there is a collision, **then**
   - (i) New constraint: these two trajectories cannot be used simultaneously
   - (ii) go to 2

   Else

   - (iii) Compute the optimal sequences with the exact trajectories



Robot R1: Start $\longrightarrow$ C $\longrightarrow$ B $\longrightarrow$ D $\longrightarrow$ Start

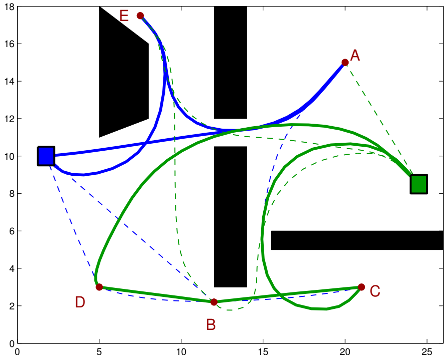
Robot R2: Start $\longrightarrow$ A $\longrightarrow$ E $\longrightarrow$ Start

Workcell Algorithm

1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet
4. **If** there is a collision, **then**
   - (i) New constraint: these two trajectories cannot be used simultaneously
   - (ii) go to 2

   Else
   - (iii) Compute the optimal sequences with the exact trajectories
   - (iv) **If** same output, **then** RETURN
     Else go to 3
     End if

   End if



Robot R1: Start → C → B → D → Start

Robot R2: Start → A → E → Start

1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet
4. **If** there is a collision, **then**
   - (i) New constraint: these two trajectories cannot be used simultaneously
   - (ii) go to 2

   Else
   - (iii) Compute the optimal sequences with the exact trajectories
   - (iv) **If** same output, **then** RETURN
     Else go to 3
     End if

   End if

- **Steps 2 and 4-(iii)**:
  Discrete optimization - vehicle routing problem
  M. Skutella and W. Welz
  Next talk

1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet
4. **If** there is a collision, **then**
   (i) New constraint: these two trajectories cannot be used simultaneously
   (ii) go to 2

   Else

   (iii) Compute the optimal sequences with the exact trajectories
   (iv) **If** same output, **then** RETURN
   Else go to 3
   End if

   End if

- **Steps 2 and 4-(iii)**:
  Discrete optimization - vehicle routing problem
  M. Skutella and W. Welz
  Next talk

- **Step 3**: Optimal control problem
  C. Landry, M. Gerdts, D. Hömberg and R. Henrion
  Next part of the talk

1. Compute the approximated trajectories
2. Find the optimal sequences
3. Compute the exact trajectories that have not been computed yet
4. **If** there is a collision, **then**
   - (i) New constraint: these two trajectories cannot be used simultaneously
   - (ii) go to 2

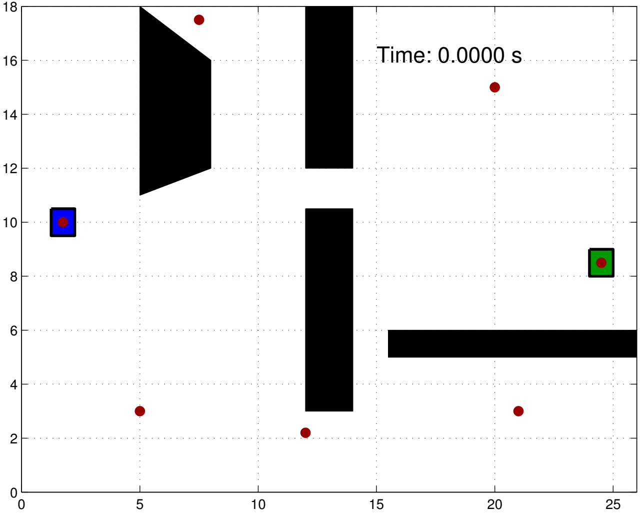
   Else
   - (iii) Compute the optimal sequences with the exact trajectories
   - (iv) **If** same output, **then** RETURN
     Else go to 3
     End if

   End if

- **Steps 2 and 4-(iii)**:
  Discrete optimization - vehicle routing problem
  M. Skutella and W. Welz
  Next talk
- **Step 3**: Optimal control problem
  C. Landry, M. Gerdts, D. Hömberg and R. Henrion
  Next part of the talk
- **Step 4**: Dynamic collision detection
  C. Landry and N. Feyeux
  based on F. Schwarzer, M. Saha and J. Latombe (2005)

Solution

R2 moves twice faster than R1 and R1 cannot reach the upper right point



1 Workcell Problem

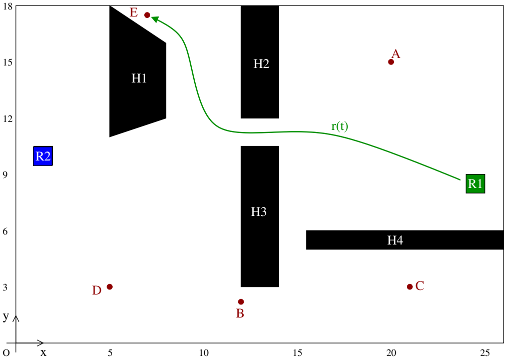
2 Time-optimal kinodynamic motion planning
- Model
- Numerical method
- Initialization
- Numerical results

3 Conclusions

Input data

Input:

- P_0: original position
- P_F: goal position
- characteristics of the robot
- obstacles



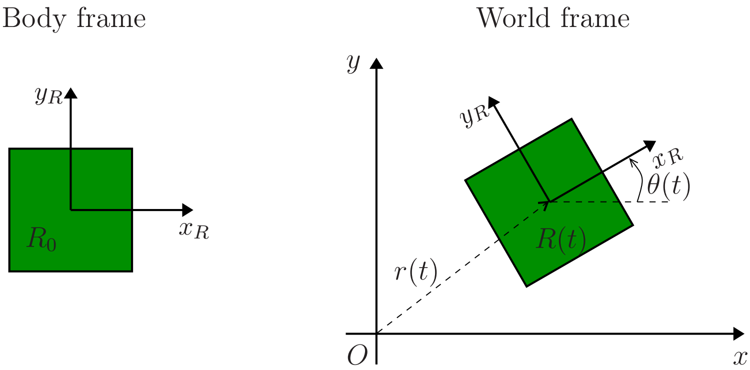
Goal:

Find the fastest and collision-free trajectory that goes from P_0 to P_f

Notation:

- r: position
- v: velocity
- a: acceleration
- θ: rotation angle
- μ: velocity of the rotation angle
- t_f: travel time

Frames



$R_0 \in \mathbb{R}^{n\times 2}$

$$R(t) = R(r(t), \theta(t)) = R_0 \begin{pmatrix} \cos(\theta(t)) & \sin(\theta(t)) \\ -\sin(\theta(t)) & \cos(\theta(t)) \end{pmatrix} + e_n \cdot r(t)^T,$$

with n = number of vertices and $e_n^T = (1, \ldots, 1) \in \mathbb{R}^n$

((OCP)):

Find $r, v, a : [0, t_f] \to \mathbb{R}^2$, $\theta, \mu : [0, t_f] \to \mathbb{R}$ and t_f

such that: $\min \ t_f$ subject to

- equations of motion:

$$\begin{aligned} r'(t) &= v(t) \\ v'(t) &= a(t) \\ \theta'(t) &= \mu(t), \text{a.e. in } [0, t_f] \end{aligned}$$

- collision avoidance (at safety level ε):

$$\min_{\ell=1,\dots,4} \operatorname{dist}(R(r(t), \theta(t)), H_\ell) \geq \varepsilon, \text{ a.e. in } [0, t_f]$$

- boundary conditions:

$$r(0) = P_0,\ v(0) = 0,\ \theta(0) = 0, r(t_f) = P_f,\ v(t_f) = 0,\ \theta(t_f) = 0$$

- box constraints:

$$\underline{a} \leq a(t) \leq \bar{a}, \underline{\mu} \leq \mu(t) \leq \bar{\mu}, \text{ a.e. in } [0, t_f]$$

(OCP):

Find $r, v, a : [0, t_f] \to \mathbb{R}^2$, $\theta, \mu : [0, t_f] \to \mathbb{R}$ and t_f such that: $\min \ t_f$ subject to

- equations of motion:

$$\begin{aligned} r'(t) &= v(t) \\ v'(t) &= a(t) \\ \theta'(t) &= \mu(t), \text{a.e. in } [0, t_f] \end{aligned}$$

- collision avoidance (at safety level ε):

$$\min_{\ell=1,\dots,4} \text{dist}(R(r(t), \theta(t)), H_\ell) \geq \varepsilon, \text{ a.e. in } [0, t_f]$$

- boundary conditions:

$$r(0) = P_0, \ v(0) = 0, \ \theta(0) = 0, r(t_f) = P_f, \ v(t_f) = 0, \ \theta(t_f) = 0$$

- box constraints:

$$\underline{a} \leq a(t) \leq \bar{a}, \underline{\mu} \leq \mu(t) \leq \bar{\mu}, \text{ a.e. in } \ [0, t_f]$$

Optimal Control Problem with:

- state variables: r, v, θ
- control variables: a, μ

First Discretize ...

- Grid: $\mathbb{G}_N := \{t_k = kh | k = 0, \dots, N\}$ with $h = t_f/N$.
- Approximation of the controls by B-splines of order j:

$$a_h(t) = \sum_{i=0}^{N+j-2} a_i \, B_{ij}(t) \text{ and } \mu_h(t) = \sum_{i=0}^{N+j-2} \mu_i \, B_{ij}(t),$$

where $a_i \in \mathbb{R}^2$ and $\mu_i \in \mathbb{R}$.

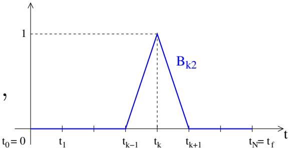


Fig.: B-spline of order 2

- Define the vector of unknowns: $z = (a_0, \mu_0, \dots, a_{N+j-2}, \mu_{N+j-2}, t_f)^T$
- Solve ODEs by a 1-step explicit method (e.g. explicit Runge-Kutta method)

$$\begin{pmatrix} r_{k+1} \\ v_{k+1} \\ \theta_{k+1} \end{pmatrix} (z) = \begin{pmatrix} r_k \\ v_k \\ \theta_k \end{pmatrix} (z) + h \, \phi(t_k, r_k(z), v_k(z), \theta_k(z), a_h(t_k), \mu_h(t_k), t_f, h),$$

where ϕ is the increment function

- Eliminate ODEs from the optimal control problem (OCP)

Nonlinear Optimization Problem

$$\min_{z} t_f \quad \text{subject to}$$

$$\begin{aligned} \min_{\ell=1,\dots,4} \text{dist}(R(r_k(z),\theta_k(z)),H_\ell) &\geq \varepsilon, \qquad k=0,\dots,N,\\ r_0 = P_0,\ v_0 = 0,\ \theta_0 &= 0,\\ r_N(z) = P_f,\ v_N(z) = 0,\ \theta_N(z) &= 0,\\ \underline{z} \leq z &\leq \bar{z}, \end{aligned}$$

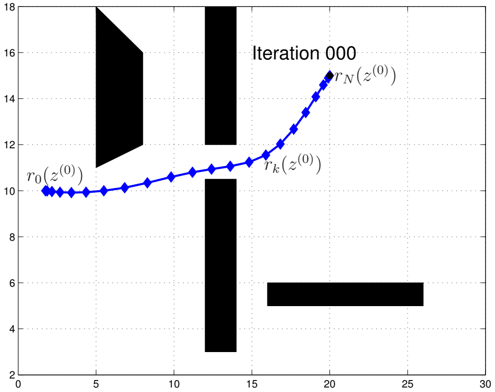
where $z = (a_0, \mu_0, \dots, a_{N+j-2}, \mu_{N+j-2}, t_f)^T$ and $\underline{z} = (\underline{a}, \underline{\mu}, \dots, \underline{a}, \underline{\mu}, \underline{t})^T$
$\bar{z} = (\bar{a}, \bar{\mu}, \dots, \bar{a}, \bar{\mu}, \bar{t})^T$.

- Solver: Sequential Quadratic Programming method (SQP)
  Iterative method: $z^{(m+1)} = z^{(m)} + d_z$ where d_z is the solution of the m^{th}-QP

- Use Matthias Gerdts's package **OCPID-DAE1**

A sequence of trajectories

Initialization: $z^{(0)} \xrightarrow{\text{ODEs integration}} r_0(z^{(0)}), \ldots, r_k(z^{(0)}), \ldots, r_N(z^{(0)})$

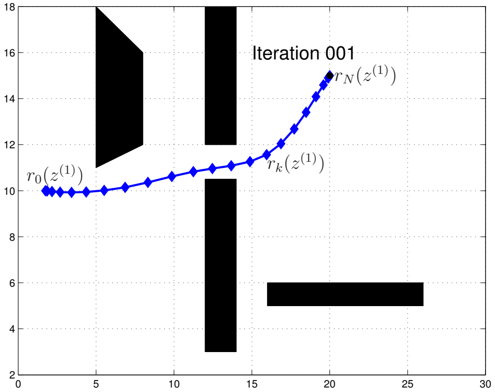


A sequence of trajectories

$$\text{Initialization: } z^{(0)} \xrightarrow{\text{ODEs integration}} r_0(z^{(0)}), \ldots, r_k(z^{(0)}), \ldots, r_N(z^{(0)})$$

$\downarrow$ **QP**

$$\text{Iteration 1: } z^{(1)} \xrightarrow{\text{ODEs integration}} r_0(z^{(1)}), \ldots, r_k(z^{(1)}), \ldots, r_N(z^{(1)})$$



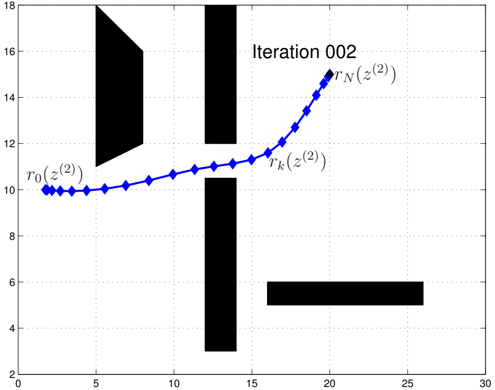
$$\text{Initialization: } z^{(0)} \xrightarrow{\text{ODEs integration}} r_0(z^{(0)}), \ldots, r_k(z^{(0)}), \ldots, r_N(z^{(0)})$$

$\downarrow$ **QP**

$$\text{Iteration 1: } z^{(1)} \xrightarrow{\text{ODEs integration}} r_0(z^{(1)}), \ldots, r_k(z^{(1)}), \ldots, r_N(z^{(1)})$$

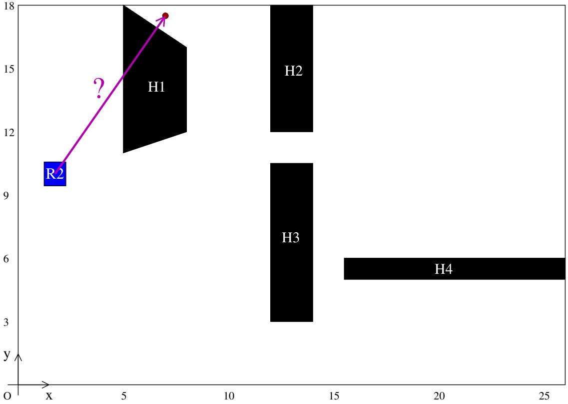
$\downarrow$ **QP**

$$\text{Iteration 2: } z^{(2)} \xrightarrow{\text{ODEs integration}} r_0(z^{(2)}), \ldots, r_k(z^{(2)}), \ldots, r_N(z^{(2)})$$



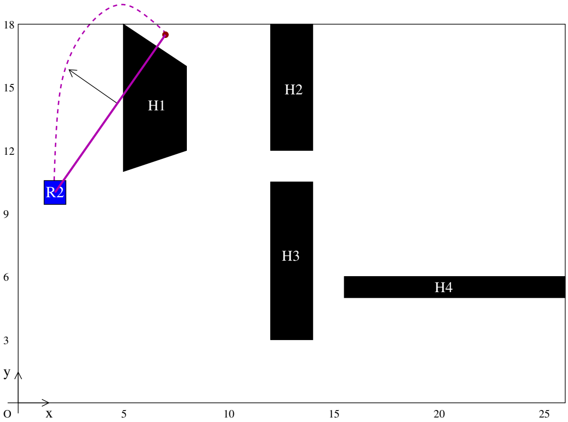
How to initialize $z^{(0)}$?

- Reminder: $z^{(0)} = (a_0^{(0)}, \mu_0^{(0)}, \dots, a_{N+j-2}^{(0)}, \mu_{N+j-2}^{(0)}, t_f^{(0)})^T$
- **Without a good initialization of $z^{(0)}$, SQP method might not converge**
- **1st attempt:** $z^{(0)}$ such that the initial trajectory describes the segment $[P_0, P_f]$



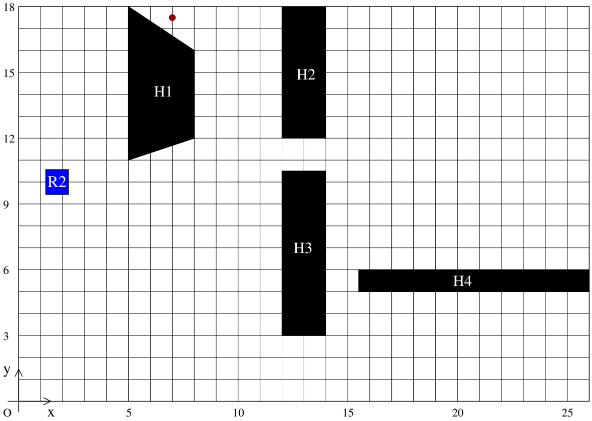
How to initialize $z^{(0)}$?

- Reminder: $z^{(0)} = (a_0^{(0)}, \mu_0^{(0)}, \ldots, a_{N+j-2}^{(0)}, \mu_{N+j-2}^{(0)}, t_f^{(0)})^T$
- **Without a good initialization of $z^{(0)}$, SQP method might not converge**
- **1st attempt:** $z^{(0)}$ such that the initial trajectory describes the segment $[P_0, P_f]$



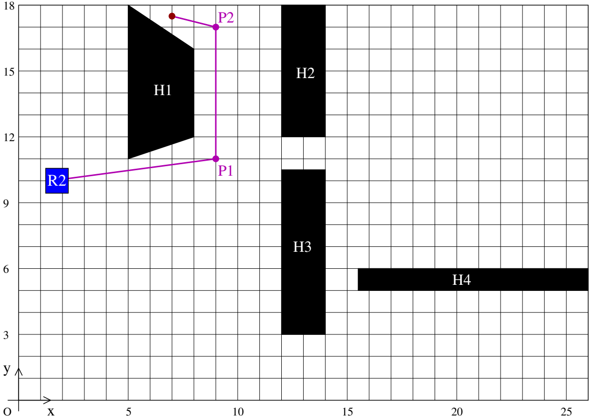
Initialization of $z^{(0)}$

- **New attempt:** grid on the workspace and use of the shortest path algorithm



Initialization of $z^{(0)}$

- **New attempt:** grid on the workspace and use of the shortest path algorithm



→ $z^{(0)}$ **s.t. the initial trajectory passes through the neighborhood of P_1 and P_2**

→ computation of the approximated trajectories in Workcell Algorithm

Initial trajectory

Look for the **fastest** trajectory between P_0 and P_f that passes through the neighborhood of the **via points** $P_i, i = 1, \dots, n_I$

$$\|P_i - r(\xi_i)\|_2 \leq \varepsilon_r, \; i = 1, \dots, n_I,$$

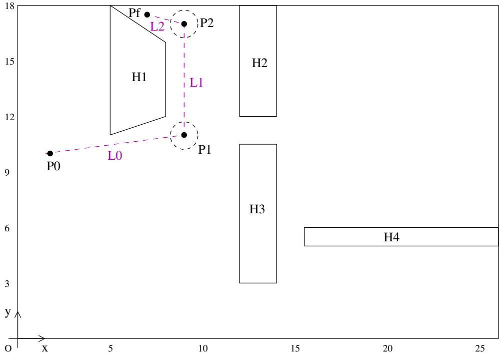
where ξ_i is chosen proportional to the travelled distance:

$$\xi_i := \left\lfloor \frac{\sum_{k=0}^{i} L_k}{L} N \right\rfloor$$

with $\varepsilon_r > 0$ small and

$$L := \sum_{i=0}^{n_I} L_i = \sum_{i=0}^{n_I} \|P_i P_{i+1}\|_2$$

where $P_{n_I+1} = P_f$.



Initial trajectory - Optimal Control Problem

(OCP_I): Find $r, v, a : [0, t_f] \to \mathbb{R}^2$, $\theta, \mu : [0, t_f] \to \mathbb{R}$ and t_f such that:

$$\min\ t_f + \sum_{i=1}^{n_I} \max(\|P_i - r(\xi_i)\|_2 - \varepsilon_r, 0)^2$$

s.t.

- ODEs:

$$\begin{aligned} r'(t) &= v(t) \\ v'(t) &= a(t) \\ \theta'(t) &= \mu(t), \text{ a.e. in } [0, t_f] \end{aligned}$$

- boundary conditions:

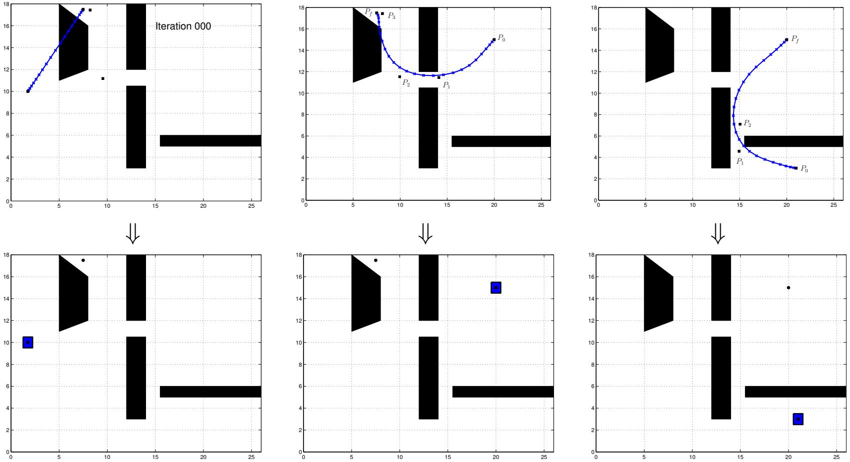
$$r(0) = P_0,\ v(0) = 0,\ \theta(0) = 0, r(t_f) = P_f,\ v(t_f) = 0,\ \theta(t_f) = 0$$

- box constraints: $\underline{a} \leq a(t) \leq \bar{a}, \underline{\mu} \leq \mu(t) \leq \bar{\mu}, \text{ a.e. in } [0, t_f],$

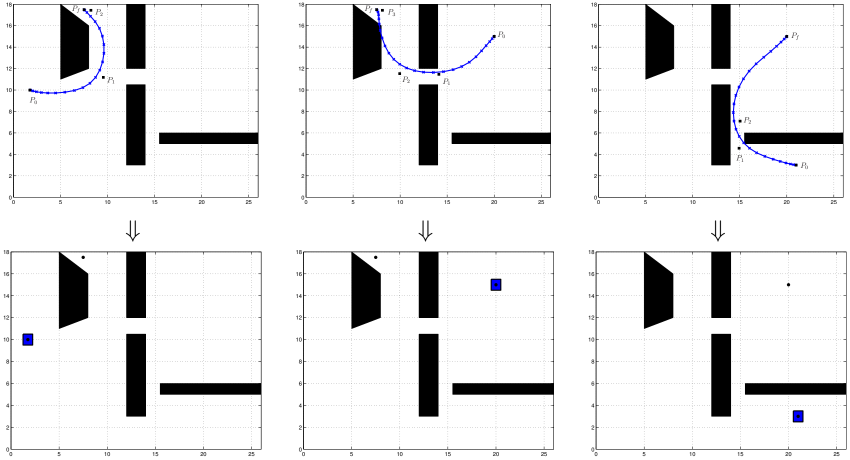
Remarks:

- Similar to (OCP), but without obstacle
- Use the same direct method as for (OCP)

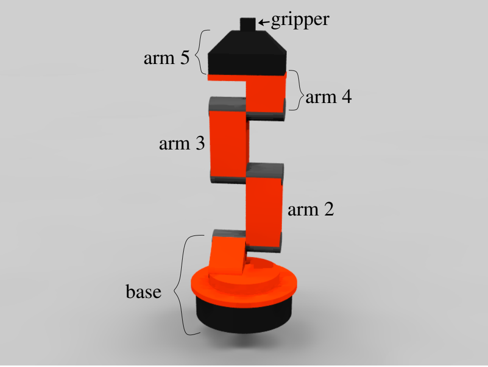
Initial trajectory - Numerical examples



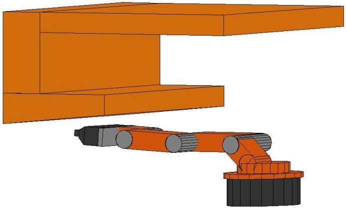
Initial trajectory - Numerical examples



- **Variable**: $q \in \mathbb{R}^4$: joint angles



- **Variable**: $q \in \mathbb{R}^4$: joint angles
- **Same strategy**
  1. Via points:
     shortest path algorithm
  2. Initial trajectory:
     solve (OCP_I)
  3. Exact trajectory:
     solve (OCP)



1 Workcell Problem

2 Time-optimal kinodynamic motion planning

3 Conclusions

Conclusions

- Conclusions
  - Configuration of robotic workcells: efficient interplay between discrete and continuous optimization
  - 2D model to compute the fastest and collision-free trajectory between P_0 and P_f
  - Computation of a good initialization is crucial
  - Combination of discrete optimization with 2 optimal control problems
- Current work
  - Adaptation to a 3D robotic workcell