Cooperative Motion and Task Planning
Under Temporal Tasks

Meng Guo

Automatic Control Lab, EES
Royal Institute of Technology, KTH, Sweden

Licentiate Seminar
Introduction
Motivation
Background

Nominal Scenario
Problem Formulation
Nominal Solution

Reconfiguration

Motion and Action
Potentially Infeasible Task
Partially-known Workspace

Multi-agent
Dependent Local Tasks
Independent Local Tasks

Summary

Meng Guo Licentiate Seminar
Motivation

Industrial and domestic robots\(^1\) boosted by

- Development of digital processing units
  - more powerful, in speed and capacity
  - more affordable

\(^1\)Gostai, Roomba, Romo, Amigo service robots
Motivation

Industrial and domestic robots\(^1\) boosted by

- Development of **digital processing** units
  - more powerful, in speed and capacity
  - more affordable

- Wireless communication technology
  - potentially connects the robots
  - integrates with other “smart” devices

\(^1\)Gostai, Roomba, Romo, Amigo service robots
Motivation

- Imagine ... several care robots in your house
Motivation

Imagine ... several care robots in your house

- “Amigo, go to the kitchen, find an apple and bring it to me”
Motivation

- Imagine ... several care robots in your house

- “Amigo, go to the kitchen, find an apple and bring it to me”

- “Gostai, keep an eye on the kids in the living room and bedroom”
Motivation

Imagine ... several care robots in your house

- “Amigo, go to the kitchen, find an apple and bring it to me”
- “Gostai, keep an eye on the kids in the living room and bedroom”
- “Amigo, put all the toys in the basket”
- ...
Challenges

Challenges for system engineers (us)

► Provide the non-expert end-users

► Design algorithms for robots that
### Challenges

Challenges for system engineers (us)

- **Provide** the non-expert end-users
  - a formal but flexible way to specify daily tasks
  - task execution status as feedback

- **Design** algorithms for robots that
Challenges

Challenges for system engineers (us)

► Provide the non-expert end-users
  • a formal but flexible way to specify daily tasks
  • task execution status as feedback

► Design algorithms for robots that
  • synthesize and execute a plan to satisfy the task
  • without or with minimal human intervention
Challenges

Challenges for system engineers (us)

» Provide the non-expert end-users
  • a formal but flexible way to specify daily tasks
  • task execution status as feedback

» Design algorithms for robots that
  • synthesize and execute a plan to satisfy the task
  • without or with minimal human intervention
  • accommodate changes in the workspace
  • initiate communication with other devices
  • handle collaborative tasks
Background

- Motion and task planning

---

Background

- Motion and task planning
  - motion plan of dynamic systems\(^2\)

Background

▶ Motion and task planning
  • motion plan of dynamic systems\(^2\)
  • task plan for discrete-event systems\(^3\)

Background

- Motion and task planning
  - motion plan of dynamic systems\(^2\)
  - task plan for discrete-event systems\(^3\)

- Model checking
  - for verification\(^4\)
  - for plan synthesis

---


Multi-agent System

- Multi-agent system to solve a **global** task
  - *decompose* into sub-tasks
  - *top-down*, tightly-coupled

- Multi-agent system with **local** tasks
  - *favour* individual interests
  - *bottom-up*, loosely-coupled
Main Contributions

▶ Reconfiguration for single- and multi-agent systems

▶ Real-time adaptation for single- and multi-agent systems
Introduction
  Motivation
  Background

Nominal Scenario
  Problem Formulation
  Nominal Solution

Reconfiguration

Motion and Action
  Potentially Infeasible Task
  Partially-known Workspace

Multi-agent
  Dependent Local Tasks
  Independent Local Tasks

Summary
  Summary
Motion Abstraction

- Abstracted as the weighted finite transition system (FTS)

\[ \mathcal{T}_c = (\Pi, \rightarrow_c, \Pi_0, AP, L_c, W_c) \]

where the finite regions \( \Pi = \{\pi_1, \pi_2, \cdots, \pi_W\} \)

- Examples:
Motion Abstraction

Abstracted as the weighted finite transition system (FTS)

\[ T_c = (\Pi, \rightarrow_c, \Pi_0, AP, L_c, W_c) \]

where the finite regions \( \Pi = \{\pi_1, \pi_2, \cdots, \pi_W\} \)

- properties of interest \( AP = AP_r \cap AP_p \)
- \( \Pi_0 \) initial states
- \( \rightarrow_c \subseteq \Pi \times \Pi \), control-driven transition
- \( L_c : \Pi \rightarrow 2^{AP} \), labelling function
- \( W_c : \Pi \times \Pi \rightarrow \mathbb{R}^+ \), weight function

Examples:
Abstraction Outcome

[Diagram of a building layout with rooms and corridors, showing connections and distances labeled with numbers 10 and 15.]

Meng Guo  Licentiate Seminar
Temporal Task Specification

How to formally specify the task?

- Language: Linear-time Temporal Logic (LTL) formula
- Syntax:

\[ \varphi ::= True \mid a \mid \varphi_1 \lor \varphi_2 \mid \neg \varphi \mid \Box \varphi \mid \varphi_1 U \varphi_2 \]

- Semantics\(^5\)

- Specified over \( AP \)

Temporal Task Specification

How to formally specify the task?

- Language: Linear-time Temporal Logic (LTL) formula
- Syntax:

\[ \varphi ::= \text{True} \mid a \mid \varphi_1 \lor \varphi_2 \mid \neg \varphi \mid \Diamond \varphi \mid \varphi_1 \mathcal{U} \varphi_2 \]

- Semantics

- Specified over AP

- To specify control tasks:
  - Safety: \( \Box \neg \varphi_1 \).
  - Order: \( \Diamond (\varphi_1 \land \Diamond (\varphi_2 \land \Diamond \varphi_3)) \).
  - Response: \( \varphi_1 \Rightarrow \varphi_2 \).
  - Liveness: \( \Box \Diamond \varphi_1 \).

---


Meng Guo Licentiate Seminar
Task Interpretation

“go to the kitchen, find an apple and bring it to me”

$\varphi = \square((\text{Kit} \land \text{PickApp}) \land \square(\boxdot\text{Liv}))$
Task Interpretation

- “go to the kitchen, find an apple and bring it to me”
- “keep an eye on the kids in the living room and bedroom”

\[ \varphi = \lozenge ((\text{Kit} \land \text{PickApp}) \land \lozenge (\square \text{Liv})) \]

\[ \varphi = \square \lozenge \text{Liv} \land \square \lozenge \text{Bed} \]
Task Interpretation

- “go to the kitchen, find an apple and bring it to me”
- “keep an eye on the kids in the living room and bedroom”
- “put all the toys in the basket”
- …

\[ \varphi = \Diamond ((\text{Kit} \land \text{PickApp}) \land \Diamond (\square \text{Liv})) \]

\[ \varphi = \square \Diamond \text{Liv} \land \square \Diamond \text{Bed} \]

\[ \varphi = \square \Diamond \left( \text{PickToy} \rightarrow (\neg \text{PickToy} \cup (\text{Bas} \land \text{DropToy}))) \right) \]

- …
Problem Formulation

Given the FTS $\mathcal{T}_c$ and the LTL task $\varphi$
Problem Formulation

Given the FTS $\mathcal{T}_c$ and the LTL task $\varphi$

- Synthesize a discrete plan that satisfies $\varphi$

\[
\begin{align*}
  r1 & \quad c1 & \quad c2 & \quad c3 & \quad r3 \\
  c3 & \quad c2 & \quad r2 & \quad c2 & \quad r5 \\
  c2 & \quad r2 & \quad c2 & \quad c1 & (r1) \omega
\end{align*}
\]
Problem Formulation

Given the FTS $\mathcal{T}_c$ and the LTL task $\varphi$

- Synthesize a discrete plan that satisfies $\varphi$
- Construct the hybrid control strategy to execute the derived plan

$$r_1 c_1 c_2 c_3 r_3$$

$$c_3 c_2 r_2 c_2 r_5$$

$$c_2 r_2 c_2 c_1(r_1)^\omega$$
Nominal Solution Outline

- Automata-based model-checking algorithm\(^6\)
- Hybrid controller synthesis\(^7\)

---


\(^7\) G. E. Fainekos et al., Temporal logic motion planning for dynamic robots. *Automatica*, 2009
Step 1. Translation

- Translate $\varphi$ into the Nondeterministic Büchi automaton (NBA) $A_\varphi$ over $2^{AP}$:

$$A_\varphi = (Q, 2^{AP}, \delta, Q_0, \mathcal{F}),$$

- $Q$ is a finite set of states
- $2^{AP}$ are alphabets.
- $\delta \subseteq Q \times 2^{AP} \times Q$.
- $Q_0, \mathcal{F}$ are initial, accepting states.
- $\chi(q_m, q_n) = \{l \in 2^{AP} \mid (q_m, l, q_n) \in \delta\}$. 

---


Meng Guo  Licentiate Seminar
Step 1. Translation

- Translate $\varphi$ into the Nondeterministic Büchi automaton (NBA) $A_\varphi$ over $2^{AP}$:

$$A_\varphi = (Q, 2^{AP}, \delta, Q_0, F),$$

- $Q$ is a finite set of states
- $2^{AP}$ are alphabets.
- $\delta \subseteq Q \times 2^{AP} \times Q$.
- $Q_0, F$ are initial, accepting states.
- $\chi(q_m, q_n) = \{ l \in 2^{AP} | (q_m, l, q_n) \in \delta \}$.

- Automated
- Fast translation algorithm\(^8\)

Step 2. Product Automaton

Construct the weighted product automaton $A_p = \mathcal{T}_c \otimes A_\varphi$:

$$A_p = (Q', \delta', Q'_0, F', W_p)$$

where $Q' = \Pi \times Q$; $Q'_0 = \Pi_0 \times Q_0$; $F' = \Pi \times F$

- $\delta' \subseteq Q \times Q$, where $(\langle \pi_i, q_m \rangle, \langle \pi_j, q_n \rangle) \in \delta'$ iff $(\pi_i, \pi_j) \in \rightarrow_c$ and $(q_m, L_c(\pi_i), q_n) \in \delta$

- $W_p : \delta' \rightarrow \mathbb{R}^+$. $W_p((\langle \pi_i, q_m \rangle, \langle \pi_j, q_n \rangle)) = W_c(\pi_i, \pi_j)$
Step 2. Product Automaton

Construct the weighted product automaton $A_p = \mathcal{T}_c \otimes A_\varphi$:

$$A_p = (Q', \delta', Q'_0, F', W_p)$$

where $Q' = \Pi \times Q$; $Q'_0 = \Pi_0 \times Q_0$; $F' = \Pi \times F$

- $\delta' \subseteq Q \times Q$, where $(\langle \pi_i, q_m \rangle, \langle \pi_j, q_n \rangle) \in \delta'$ iff $(\pi_i, \pi_j) \in \rightarrow_c$ and $(q_m, L_c(\pi_i), q_n) \in \delta$

- $W_p : \delta' \rightarrow \mathbb{R}^+$. $W_p((\langle \pi_i, q_m \rangle, \langle \pi_j, q_n \rangle)) = W_c(\pi_i, \pi_j)$

Algorithms

- static construction
- on-the-fly construction
Plan Structure and Cost

- Accepting run $R$ of $A_p$ with the prefix-suffix structure

$$R = R_{\text{pre}} (R_{\text{suf}})^{\omega} = q'_0 q'_1 \cdots q'_{f-1} \left[ q'_f q'_{f+1} \cdots q'_n \right]^{\omega}$$

- The total cost:

$$\text{Cost}(\ R, \ A_p) = \sum_{i=0}^{f-1} W_p(q'_i, q'_{i+1}) + \gamma \sum_{i=f}^{n-1} W_p(q'_i, q'_{i+1})$$
Step 3. Graph Search

- Algorithm based on Nested-Dijkstra shortest path
  - shortest path from every $q_0' \in Q_0'$ to every $q_f' \in F'$
  - shortest cycle from $q_f'$ and back

```
\begin{figure}
  \centering
  \includegraphics[width=\textwidth]{graph-search-diagram}
  \caption{Graph Search Algorithm Diagram}
\end{figure}
```
Step 3. Graph Search

- Algorithm based on Nested-Dijkstra shortest path
  - shortest path from every $q'_0 \in Q'_0$ to every $q'_f \in F'$
  - shortest cycle from $q'_f$ and back

- The derived accepting run $R_{opt}$ with minimal cost

- Corresponding plan $\tau = R_{opt}|_\Pi$

- Complexity $O(|\delta'| \cdot \log_2 |Q'| \cdot (|Q'_0| + |F'|))$
Step 4. Hybrid Control Strategy

To execute $\tau = R_{opt}|_\Pi$ using $U(x(t), \pi_i, \pi_j)$

- generate an **infinite** execution with **finite** representation
- monitor the execution **status**
- record past motions

```
r1 c1 c2 c3 r3
  c3 c2 r2 c2 r5
  c2 r2 c2 c1(r1)^w
```
Shortcomings of Nominal Solution

▶ Reconfiguration
  • plan as sequence of motion (no actions)
  • feasible task

▶ Real-time adaptation
  • fully-known workspace
  • plan synthesized once off-line
  • executed regardless of the real observation
Shortcomings of Nominal Solution

▶ Reconfiguration
  • plan as sequence of motion (no actions)
  • feasible task

▶ Real-time adaptation
  • fully-known workspace
  • plan synthesized once off-line
  • executed regardless of the real observation

▶ Multi-agent system with local tasks
  • independent or dependent tasks
  • communication
  • collaborative tasks
Introduction
Motivation
Background

Nominal Scenario
Problem Formulation
Nominal Solution

Reconfiguration

Motion and Action
Potentially Infeasible Task
Partially-known Workspace

Multi-agent
Dependent Local Tasks
Independent Local Tasks

Summary

Meng Guo  Licentiate Seminar
Motion and Action Planning

Why is it necessary?
- to automate the choice of actions
- plan as sequence of motion and actions

Why plan motion and action together?
- the purpose of “going somewhere” is to “do something”

Why model motion and action separately?
- robot’s mobility: depend on the workspace structure
- robot’s capable actions: relatively fixed
Model of Mobility and Action

- **Mobility** abstraction is given by (similarly as $\mathcal{T}_c$):

  $$\mathcal{M} = (\Pi_\mathcal{M}, act_\mathcal{M}, \rightarrow_\mathcal{M}, \Pi_{\mathcal{M},0}, \Psi_\mathcal{M}, L_\mathcal{M}, W_\mathcal{M})$$

- Capable action set $Act_B = \{act_B,0, act_B,1, \cdots, act_B,K\}$

- **Precondition** function:

  $$\text{Cond} : Act_B \times 2^{\Psi_p} \times 2^{\Psi_s} \rightarrow \text{True/False}$$

- **Effect** function:

  $$\text{Eff} : Act_B \times (2^{\Psi_s} \times \Psi_b) \rightarrow (2^{\Psi_s} \times \Psi_b),$$

- **Mimic action description language (ADL)**
Complete functionalities by composing $\mathcal{M}$ and $\mathcal{B}$

$$\mathcal{R} = (\Pi_{\mathcal{R}}, \text{Act}_{\mathcal{R}}, \rightarrow_{\mathcal{R}}, \Pi_{\mathcal{R},0}, \Psi_{\mathcal{R}}, L_{\mathcal{R}}, W_{\mathcal{R}}),$$

Automated algorithm
Results

- **Much richer** task specifications

- **Given** $\varphi$ over robot **motion**, **action** and **internal** states

- **Accepting run** with prefix-suffix structure with similar definition for cost
Results

- Much **richer** task specifications
- Given $\varphi$ over robot **motion**, **action** and **internal** states
- Accepting run with prefix-suffix structure with similar definition for cost
- Plan as a sequence of **motion and action**, minimal cost
- Construct the **hybrid** controller accordingly

$r_1 c_1 c_2 c_3 r_3$
$c_3 c_2 r_2 c_2 r_5$
$c_2 r_2 c_2 c_1(r_1)^\omega$

$r_1 c_1 c_2 c_3 r_3$ PickG
$c_3 c_2 r_2$ Drop $c_2 r_5$ PickR
$c_2 r_2$ Drop $c_2 c_1(r_1)^\omega$
Potentially Infeasible Task

- Def. 2.6: $\varphi$ is infeasible for $T_c$ if no accepting run of $A_p$ can be found.

- Closely related to partially-known workspace

- Example:
  - “go to the kitchen, find an apple and bring it to me”
    $\Rightarrow$ infeasible if no apple
  - “keep an eye on the kids in the living room and bedroom”
    $\Rightarrow$ infeasible if not known where the bedroom is
Potentially Infeasible Task

- Def. 2.6: \( \varphi \) is infeasible for \( T_c \) if no accepting run of \( A_p \) can be found.

- Closely related to partially-known workspace

- Example:
  - “go to the kitchen, find an apple and bring it to me”
    \( \Rightarrow \) infeasible if no apple
  - “keep an eye on the kids in the living room and bedroom”
    \( \Rightarrow \) infeasible if not known where the bedroom is

- Nominal solution fails

- Our goal
Potentially Infeasible Task

▶ Def. 2.6: $\varphi$ is infeasible for $T_c$ if no accepting run of $A_p$ can be found.

▶ Closely related to partially-known workspace

▶ Example:
- “go to the kitchen, find an apple and bring it to me”
  $\Rightarrow$ infeasible if no apple
- “keep an eye on the kids in the living room and bedroom”
  $\Rightarrow$ infeasible if not known where the bedroom is

▶ Nominal solution fails

▶ Our goal
  - synthesize the plan satisfying the task partially
  - user-defined balance on satisfiability and cost of the plan
Solution

▶ Relaxed product automaton $A_r = \mathcal{T}_c \times A_\varphi$:

$$A_r = (Q', 2^{AP}, \delta', Q'_0, \mathcal{F}', W_r)$$

where $Q' = \Pi \times Q$; $Q'_0 = \Pi_0 \times Q_0$; $\mathcal{F}' = \Pi \times \mathcal{F}$;
Solution

- **Relaxed product automaton** $A_r = T_c \times A_\varphi$:
  
  $$A_r = (Q', 2^{AP}, \delta', Q'_0, F', W_r)$$
  
  where $Q' = \Pi \times Q$; $Q'_0 = \Pi_0 \times Q_0$; $F' = \Pi \times F$;

  - $\delta' \subseteq Q' \times Q'$. $(\langle \pi_i, q_m \rangle, \langle \pi_j, q_n \rangle) \in \delta'$ iff $(\pi_i, \pi_j) \in \rightarrow_c$ and $\exists l \in 2^{AP}$ such that $(q_m, l, q_n) \in \delta$.

  - $W_r : \delta' \to \mathbb{R}^+$ is the weight function:
    
    $$W_r(\langle \pi_i, q_m \rangle, \langle \pi_j, q_n \rangle) = W_c(\pi_i, \pi_j) + \alpha \cdot \text{Dist}(L(\pi_i), \chi(q_m, q_n)).$$

- $\alpha \geq 0$: user-defined balance
Balanced Cost

- **Accepting run** $R$ with **prefix-suffix** structure:

  $$R = q'_0 q'_1 \cdots [q'_f q'_{f+1} \cdots q'_n] \omega$$
  
  $$= \langle \pi_0, q_0 \rangle \langle \pi_1, q_1 \rangle \cdots [\langle \pi_f, q_f \rangle \cdots \langle \pi_n, q_n \rangle] \omega,$$

- **The balanced cost** of an accepting run:

  $$\text{Cost}(R, A_r) = \sum_{i=0}^{f-1} W_r(q'_i, q'_{i+1}) + \gamma \sum_{i=f}^{n-1} W_r(q'_i, q'_{i+1})$$
  
  $$= \text{cost}_\tau + \alpha \cdot \text{dist}_\varphi$$

  - $\text{cost}_\tau$: implementation cost; $\text{dist}_\varphi$: distance to $\varphi$

  $\text{dist}_\varphi = 0 \Rightarrow R|_{\Pi} \models \varphi$
Results

- The balanced accepting run

\[ R_{\text{bal}} = \min_{R} \text{Cost}(R, A_r) \]

- The balanced plan \( \tau_{\text{bal}} = R_{\text{bal}}|_{\Pi} \)
Results

➤ The balanced accepting run

\[ R_{bal} = \min_R \text{Cost}(R, A_r) \]

➤ The balanced plan \( \tau_{bal} = R_{bal}|\Pi \)

➤ Proposed algorithms:

- synthesize \( R_{bal} \), given \( \gamma \) and \( \alpha \)
- computes the associated \( \text{cost}_\tau \) and \( \text{dist}_\varphi \) for \( \tau_{bal} \)

➤ Theorem 3.1: If \( \varphi \) is feasible over \( \mathcal{T}_c \), the balanced plan \( \tau_{bal} \) satisfies \( \varphi \) if \( \alpha > \bar{\alpha} \).
Feedback by Tuning $\alpha$

- $\alpha$: tunable balance between $\text{cost}_\tau$ and $\text{dist}_\varphi$
  - $\alpha \uparrow \Rightarrow \text{dist}_\varphi \downarrow \Rightarrow$ satisfy $\varphi$ more
  - $\alpha \downarrow \Rightarrow \text{cost}_\tau \downarrow \Rightarrow$ cheaper
Feedback by Tuning $\alpha$

- $\alpha$: tunable balance between $\text{cost}_\tau$ and $\text{dist}_\varphi$
  - $\alpha \uparrow \Rightarrow \text{dist}_\varphi \downarrow \Rightarrow$ satisfy $\varphi$ more
  - $\alpha \downarrow \Rightarrow \text{cost}_\tau \downarrow \Rightarrow$ cheaper

- Example: $\varphi = \Box \Diamond a_1 \land \Box \neg (a_2 \land a_3)$
Soft and Hard Specification

- Specification with two distinctive parts:

\[ \varphi = \varphi^\text{hard} \land \varphi^\text{soft} \]

- \( \varphi^\text{hard} \), for safety or security
  - “do not go to the balcony”, “always alarm if see fire”

- \( \varphi^\text{soft} \), for additional achievement (maybe infeasible)
  - “collect the toys in all rooms”
Soft and Hard Specification

- Specification with two distinctive parts:
  \[ \varphi = \varphi_{\text{hard}} \land \varphi_{\text{soft}} \]

  - \( \varphi_{\text{hard}} \), for safety or security
    - “do not go to the balcony”, “always alarm if see fire”
  - \( \varphi_{\text{soft}} \), for additional achievement (maybe infeasible)
    - “collect the toys in all rooms”

- Nominal solution fails

- Our goal
Soft and Hard Specification

- Specification with two distinctive parts:
  \[ \varphi = \varphi_{\text{hard}} \land \varphi_{\text{soft}} \]

- \( \varphi_{\text{hard}} \), for safety or security
  - “do not go to the balcony”, “always alarm if see fire”

- \( \varphi_{\text{soft}} \), for additional achievement (maybe infeasible)
  - “collect the toys in all rooms”

- Nominal solution fails

- Our goal
  - synthesize the plan satisfies \( \varphi_{\text{hard}} \) completely and \( \varphi_{\text{soft}} \) partially
  - user-defined balance on satisfiability and cost of the plan
Solution

- Relaxed intersection of $\mathcal{A}^{\text{soft}}$ and $\mathcal{A}^{\text{hard}}$ as $\tilde{\mathcal{A}}_\varphi$, by Alg. 9

$$\tilde{\mathcal{A}}_\varphi = (Q, 2^AP, \delta, Q_0, \mathcal{F})$$

- Safety-ensured and relaxed product automaton

$$\tilde{\mathcal{A}}_r = \mathcal{T}_c \times \tilde{\mathcal{A}}_\varphi = (Q', \delta', Q'_0, \mathcal{F}', W_r)$$

where $Q' = \Pi \times Q$, $Q'_0 = \Pi_0 \times Q_0$, $\mathcal{F}' = \Pi \times \mathcal{F}$
Solution

- **Relaxed intersection** of $\mathcal{A}^{\text{soft}}$ and $\mathcal{A}^{\text{hard}}$ as $\tilde{\mathcal{A}}_{\varphi}$, by Alg. 9

$$\tilde{\mathcal{A}}_{\varphi} = (Q, 2^{\mathcal{A}P}, \delta, Q_0, F)$$

- **Safety-ensured and relaxed product automaton**

$$\tilde{\mathcal{A}}_{r} = \mathcal{T}_c \times \tilde{\mathcal{A}}_{\varphi} = (Q', \delta', Q'_0, F', W_r)$$

where $Q' = \Pi \times Q$, $Q'_0 = \Pi_0 \times Q_0$, $F' = \Pi \times F$

- $\delta' : Q' \to 2^{Q'}$. $\langle \pi_j, q_n \rangle \in \delta'(\langle \pi_i, q_m \rangle)$ iff $(\pi_i, \pi_j) \in \rightarrow_c$ and $q_n \in \delta(q_m, L_c(\pi_i))$

- $W_r : \delta' \to \mathbb{R}^+$ is the weight function.

$$W_r(\langle \pi_i, q_m \rangle, \langle \pi_j, q_n \rangle) = W_c(\pi_i, \pi_j) + \alpha \cdot \text{Dist}(L_c(\pi_i), \chi_{\text{soft}}(q_2, \check{q}_2))$$
Results

- **Theorem 3.3:** Assume $R$ is an accepting run of $\tilde{A}_r$.
  \[ \tau = R|_\Pi \text{ is safe for } T_c \text{ and } \varphi. \]

- **Balanced cost** of accepting runs:
  \[
  \text{Cost}(R, \tilde{A}_r) = \text{cost}_\tau + \alpha \cdot \text{dist}_{\varphi\text{soft}}
  \]
Results

- **Theorem 3.3**: Assume $R$ is an accepting run of $\tilde{A}_r$.
  \[ \tau = R|_{\Pi} \text{ is safe for } T_c \text{ and } \varphi. \]

- **Balanced cost** of accepting runs:
  \[ \text{Cost}(R, \tilde{A}_r) = \text{cost}_\tau + \alpha \cdot \text{dist}_{\varphi_{soft}} \]

- The **safe** accepting run
  \[ R_{\text{safe}} = \min_R \text{Cost}(R, \tilde{A}_r) \]

- The **safe** plan $\tau_{\text{safe}} = R_{\text{safe}}|_{\Pi}$

- Similar algorithms to synthesize $R_{\text{safe}}$

- **Lemma 3.2**: $\text{dist}_{\varphi_{soft}} = 0 \Rightarrow \tau_{\text{safe}} \models \varphi$
On-line Planning

Why put planner on-line?

- To handle partially-known workspace
- Plan may not be executed as expected
- Plan could be improved
- Feed real-time observation back to planner
Problem Formulation

- The agent’s FTS at time $t \geq 0$:

$$\mathcal{T}_c^t = (\Pi, \rightarrow_c^t, \Pi_0, AP, L_c^t, W_c^t)$$

- Task specification

$$\varphi = \varphi^{\text{soft}} \land \varphi^{\text{hard}}$$

- Our goal:
  - **model** the robot’s **sensing**
  - **update** the system **model**
  - **guarantee** the plan is always **valid** and **safe**
Solution Framework

Step 1: Initial synthesis
- Initial FTS
- Task specific

Step 2: Validate plan
- Initial plan

Step 3: Revise plan
- Current plan
- Invalid or unsafe plan

Step 4: System update
- Observation

Updated FTS
Initial Synthesis and System Update

- Step 1. **initial synthesis at** \( t = 0 \)
  - \( \tau^0 \) obtained for feasible or infeasible task
  - starts executing \( \tau^0 \)
Initial Synthesis and System Update

- **Step 1. initial synthesis at** $t = 0$
  - $\tau^0$ obtained for feasible or infeasible task
  - starts executing $\tau^0$

- **Step 2. knowledge update at** $t \geq 0$
  - **sensing** information obtained

  $$\text{Sense}^t = \{((\pi, S, S_\neg), E, E_\neg)\}$$
Initial Synthesis and System Update

- **Step 1. initial synthesis at** $t = 0$
  - $\tau^0$ obtained for feasible or infeasible task
  - starts executing $\tau^0$

- **Step 2. knowledge update at** $t \geq 0$
  - **sensing** information obtained
    
    $$\text{Sense}^t = \{((\pi, S, S\neg), E, E\neg)\}$$

- **Step 2. update** $\mathcal{T}_c^t$ based on $\text{Sense}^t$
Plan Verification and Revision

- **Step 3.** validate the current plan
  - validity $\iff$ invalid transitions
  - safety $\iff$ unsafe transitions
Plan Verification and Revision

- **Step 3. validate** the current plan
  - validity $\iff$ invalid transitions
  - safety $\iff$ unsafe transitions

- **Step 4. local revision**, instead of full synthesis
Plan Verification and Revision

▶ **Step 3. validate** the current plan
  - validity $\iff$ invalid transitions
  - safety $\iff$ unsafe transitions

▶ **Step 4. local revision**, instead of full synthesis

▶ Low complexity, suitable for **real-time** applications

▶ Theorem. 3.5: **validity** and **safety** of the revised plan guaranteed
Simulation Example

- Nonholonomic ground vehicle
- Potential filed-based navigation controller\(^9\)
- Surveillance over regions 2, 3, 4
- Detect walls and obstacles in real-time

\(^9\)S. R. Lindemann, I. I. Hussein, S. M. LaValle. Real time feedback control for nonholonomic mobile robots with obstacles. CDC, 2006
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Nominal Scenario</th>
<th>Reconfiguration</th>
<th>Multi-agent</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
<td>Problem Formulation</td>
<td>Motion and Action</td>
<td>Potentially Infeasible Task</td>
<td>Summary</td>
</tr>
<tr>
<td>Background</td>
<td>Nominal Solution</td>
<td>Partially-known Workspace</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Multi-agent**
- Dependent Local Tasks
- Independent Local Tasks

**Summary**
Dependent Local Tasks

- System of $N$ agents, $i = 1, \cdots, N$:
  \[ T_i = (\Pi_i, \longrightarrow_i, \Pi_{i,0}, AP_i, L_i, W_i) \]

- Locally-assigned LTL specification by $\varphi_i$

- $\varphi_i$ may contain requirements on other agents $j \neq i$
  - constraints, e.g., “do not be in the same room with agent 1”
  - collaborations, e.g., “move the desk together with agent 1”
Dependent Local Tasks

- System of $N$ agents, $i = 1, \cdots, N$:

$$\mathcal{T}_i = (\Pi_i, \rightarrow_i, \Pi_{i,0}, AP_i, L_i, W_i)$$

- **Locally-assigned** LTL specification by $\varphi_i$

- $\varphi_i$ may contain **requirements** on other agents $j \neq i$
  - **constraints**, e.g., “do not be in the same room with agent 1”
  - **collaborations**, e.g., “move the desk together with agent 1”

- **Difficulties**:
  - the **joined execution** may not be **mutually feasible** even though the individual one is.
  - the **priority** of each agent plays an important role.
Dependency Cluster

- **Dependency**: Agents $i$ and $j$ are called **dependent** if:

  1. **agent $i$ depends on agent $j$** if $AP_{\varphi_i} \land AP_j \neq \emptyset$, or
  2. **agent $j$ depends on agent $i$** if $AP_{\varphi_j} \land AP_i \neq \emptyset$.  

\[ \Theta_1, \Theta_2, \Theta_3, \Theta_4 \]
 Dependency Cluster

- **Dependency**: Agents $i$ and $j$ are called dependent if:

  1. agent $i$ depends on agent $j$ if $AP_{\varphi_i} \land AP_j \neq \emptyset$, or
  2. agent $j$ depends on agent $i$ if $AP_{\varphi_j} \land AP_i \neq \emptyset$.

- **Dependency graph**: $G_d = (V, E)$, $V = 1, 2 \cdots , N$ and $E \subseteq V \times V$ is the dependency relation

- **Dependency cluster**: $\Theta \subseteq V$, $\forall i, j \in \Theta$ there is a path from $i$ to $j$ in $G_d$. 
Mutual Infeasible

Within one cluster $\Theta = \{1, 2, \cdots, M\}$

- The composed FTS $\mathcal{T}_\Theta = \mathcal{T}_1 \times \cdots \mathcal{T}_M$ is:

  $$\mathcal{T}_\Theta = (\Pi_\Theta, \rightarrow_\Theta, \Pi_{\Theta,0}, AP_{\Theta}, L_\Theta, W_\Theta)$$

- The mutual specification is

  $$\varphi_\Theta = \varphi_1 \land \varphi_2 \cdots \land \varphi_M$$

- Mutually infeasible if $\varphi_\Theta$ is infeasible over $\mathcal{T}_\Theta$. 
Mutual Infeasible

Within one cluster $\Theta = \{1, 2, \cdots, M\}$

- The composed FTS $\mathcal{T}_\Theta = \mathcal{T}_1 \times \cdots \mathcal{T}_M$ is:
  
  $$\mathcal{T}_\Theta = (\Pi_\Theta, \rightarrow_\Theta, \Pi_{\Theta,0}, AP_\Theta, L_\Theta, W_\Theta)$$

- The mutual specification is
  
  $$\varphi_\Theta = \varphi_1 \land \varphi_2 \cdots \land \varphi_M$$

- Mutually infeasible if $\varphi_\Theta$ is infeasible over $\mathcal{T}_\Theta$.

- The relaxed intersection of $A_{\varphi_i}$ of $\{\varphi_i, i \in \Theta\}$:
  
  $$\tilde{A}_{\varphi_\Theta} = (Q, 2^{AP_{\varphi_\Theta}}, \delta, Q_0, \mathcal{F})$$,
Solution

- The relaxed product automaton

\[ \mathcal{A}_{r,\Theta} = T_{\Theta} \times \tilde{A}_{\varphi_{\Theta}} = (Q', \delta', Q'_0, \mathcal{F}', W_r) \]

where \( Q' = \Pi_{\Theta} \times Q \), \( Q'_0 = \Pi_{\Theta,0} \times Q_0 \), \( \mathcal{F}' = \Pi_{\Theta} \times \mathcal{F} \)
Solution

- The relaxed product automaton

\[ \mathcal{A}_{r,\Theta} = T_{\Theta} \times \mathcal{A}_{\varphi_{\Theta}} = (Q', \delta', Q'_0, \mathcal{F}', W_r) \]

where \( Q' = \Pi_{\Theta} \times Q \), \( Q'_0 = \Pi_{\Theta,0} \times Q_0 \), \( \mathcal{F}' = \Pi_{\Theta} \times \mathcal{F} \)

- \( \delta' \subseteq Q' \times Q' \). \((\langle \pi_{\Theta}, q_a \rangle, \langle \pi'_{\Theta}, q_b \rangle) \) \( \in \delta' \) iff \((\pi_{\Theta}, \pi'_{\Theta}) \in \rightarrow_{\Theta} \) and \((q_a, q_b) \in \delta \).

- \( W_r : \delta' \rightarrow \mathbb{R}^+ \) is the weight function:

\[
W_r((\langle \pi_{\Theta}, q_1, \cdots, q_M, t \rangle, \langle \pi'_{\Theta}, q'_1, \cdots, q'_M, t' \rangle)) = W_{\Theta}(\pi_{\Theta}, \pi'_{\Theta}) + \alpha \sum_{i=1}^{M} \beta_i \text{Dist}(L_{\Theta}(\pi_{\Theta}), \chi_i(q_i, q'_i))
\]

- \( \alpha \): penalty on violating \( \varphi_{\Theta} \)

- \( \beta_i \): priority of agent \( i \)' task
Results

- Synthesize the balanced accepting run of $A_r, \Theta$
- Projection onto $T_i$ as the individual plan, $i \in \Theta$
Results

- Synthesize the balanced accepting run of $A_r, \Theta$
- Projection onto $T_i$ as the individual plan, $i \in \Theta$
- Change $\alpha$ and $\beta$
- Example

![Diagram showing potential motion plans and distances to $\phi_1$ and $\phi_2$.]
Independent Local Tasks

- System of $N$ agents coexisting within a partially-known workspace:

$$\mathcal{T}_i^t = (\Pi_i, \Pi_{i,0}, AP_i, L_i^t, W_i^t)$$

- Locally-assigned task specification and independent

$$\varphi_i = \varphi_i^{\text{soft}} \land \varphi_i^{\text{hard}}$$
Independent Local Tasks

- System of $N$ agents **coexisting** within a **partially-known** workspace:

$$\tau_i^t = (\Pi_i, \rightarrow_i^t, \Pi_{i,0}, AP_i, L_i^t, W_i^t)$$

- Locally-assigned task specification and independent

$$\varphi_i = \varphi_{i,\text{soft}} \land \varphi_{i,\text{hard}}$$

- Motivation:
  - agents located at various **locations** within the workspace
  - observe **up-to-date** information
  - beneficial to **communicate**
Knowledge Update and Transfer

- **Knowledge update by**
  - own sensing ability \( \text{Sense}_k^t = \{ \pi, S, S\neg, E, E\neg \} \)
  - communication with others
Knowledge Update and Transfer

- **Knowledge update by**
  - own sensing ability $\text{Sense}_k^t = \{(\pi, S, S\_\neg), E, E\_\neg\}$
  - communication with others

- Communication network: $\mathcal{N}_k \in \mathcal{N}$ (static or dynamic)

- Transfer knowledge:
  - request once: $\text{Request}_{k,g}^t = (k, \varphi_k|_{AP_k})$
  - event-based reply: $\text{Reply}_{h,k}^t = (\pi, S', S'_\neg)$, where $S' = S \cap (\varphi_h|_{AP_h})$ and $S'_\neg = S_\neg \cap (\varphi_h|_{AP_h})$.

- Update $\mathcal{T}_k^t$ based on $\text{Sense}_k^t$ and $\text{Reply}_{g,k}^t$

- Validate and revise the current plan
Results

- Full synthesis or local revision
- Event-based trigger to re-synthesize the plan
Software Implementation

- Robot operating system (ROS)-based
- ROS core + ROS nodes

ROS node for planning
Experiments

- NAO humanoid
- MAS Lab @CVAP
- NEXUS ground vehicle
- Smart Mobility Lab @ACL
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Nominal Scenario</th>
<th>Reconfiguration</th>
<th>Multi-agent</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
<td>Problem Formulation</td>
<td>Motion and Action</td>
<td>Dependent Local Tasks</td>
<td>Summary</td>
</tr>
<tr>
<td>Background</td>
<td>Nominal Solution</td>
<td>Potentially Infeasible Task</td>
<td>Independent Local Tasks</td>
<td>Summary</td>
</tr>
<tr>
<td></td>
<td>Reconfiguration</td>
<td>Partially-known Workspace</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

▪ Motion and task planning
  • Discrete motion and task plan with minimal cost
  • Hybrid control strategy
Summary

▶ Motion and task planning
  • Discrete motion and task plan with minimal cost
  • Hybrid control strategy

▶ Reconfiguration and real-time adaptation
  • Potentially infeasible task
  • Soft and hard specifications
  • Partially-known workspace
  • Motion and action planning
Summary

- Motion and task planning
  - Discrete motion and task plan with minimal cost
  - Hybrid control strategy

- Reconfiguration and real-time adaptation
  - Potentially infeasible task
  - Soft and hard specifications
  - Partially-known workspace
  - Motion and action planning

- Multi-agent systems with local tasks
  - Dependent tasks
  - Independent tasks
  - Software implementation
Future Work

- Automated abstraction
- Natural language to LTL, graphic interface
- Trade-off between computational complexity and optimality
- Robustness and fault tolerance (both motion and action)
- Continuous constraints, coupled dynamics
Thank you!