Planning Problems for Social Robots

Gian Diego Tipaldi    Kai O. Arras
Social Robotics Lab
University of Freiburg, Germany
Motivations

- **Socially compatible robots**
  - Blend into human activities
- **Understand social spaces**
  - Learn patterns of activities
- **Human-aware planning**
  - Look for people around
  - Minimize hindrance to people
Learning Activity Patterns

- Learn spatio-temporal patterns of human activities
- Answer questions like:
  - How probable is an activity performed at a certain time and space?
  - How long do I need to wait for an activity to happen?
  - What is the path that maximize the probability of encountering a certain activity?
Spatial Affordance Map

- Poisson process
  - Non-homogeneous spatial Poisson process with rate function \( \lambda(\vec{x}, t) \)

- Assumption
  - Function approximators are too slow
  - Piecewise homogeneous in space and time

- Learning
  - Using Bayesian learning
  - Gamma distributed
  - Poisson parameter obtained via expectation
    \[ \beta = \mathbb{E}[\lambda] = \int \lambda(\vec{x}; t) \, \text{dx} \, \text{dt} \]
Learning Example
People Simulator

- Real data is hard to collect
- Simulator with 3-layer agent architecture
- Three simulated environments
- Activities learned from questionnaires

Office

Warehouse

House
Maximum Encounter Planning

- Plan paths that **maximize the probability of encountering people**, giving a deadline

- **Example**: Coffee delivery robot
  - Deliver coffee fast
  - Coffee must be still hot (deadline)
  - People may move
Maximum Encounter Planning

- **Finite horizon MDP**
  - **State:** cell in the map
  - **Action:** move to next cell
  - **Reward:** Poisson rate
  - **Horizon:** the deadline

- **Challenges**
  - Horizon reduced in time
  - Time variance of reward

---

Algorithm 1: Encounter Probability Planning

```
In: Rate \( \lambda(x, t) \); time \( t_{\text{max}} \); initial state \( s_0 \);
Out: The best path \( \mathcal{P}^* \);

// Compute the policy
1  Compute the horizon \( N \);
2  \( J_N(s) \leftarrow \lambda_{ij} \forall s \);
3  for \( k \leftarrow N-1 \) to 0 do
4    \( J_k(s) \leftarrow \max_a \left[ R(s, a) + \sum_{s'} p(s'|s, a) J_{k+1}(s') \right] \);
5    \( A_k^*(s) \leftarrow \arg\max_a \left[ R(s, a) + \sum_{s'} p(s'|s, a) J_{k+1}(s') \right] \);
6  end

// Extract the path
7  \( \mathcal{P}^*(0) \leftarrow s_0 \);
8  for \( k \leftarrow 1 \) to \( N \) do
9    \( s \leftarrow \mathcal{P}^*(k-1) \);
10   \( \mathcal{P}^*(k) \leftarrow \mathbb{E} \left[ p(s'|s, A_{k-1}^*(s)) \right] \);
11  end
12  return \( \mathcal{P}^* \);
```
Planning heuristics

- MDP is too complex for real time planning
  - $O(N^3)$ time complexity
    - Too slow
  - $O(N^3)$ space complexity
    - Memory swap for limited resource robots

- MDP behavior
  - Go towards the sink if deadline is enough
  - Use a longer but more probable path

- Heuristics
  - Relax action stochasticity
  - A* towards the local sink
  - A* towards the global sink
Generated Path Analysis

PMDP = 0:26
Generated Path Analysis

\[ PMDP = 0.66 \]
Generated Path Analysis

PMDP = 0.94
Encounter Planning Experiments

- **Experiment setup**
  - 10 simulation days
  - 1000 paths
  - Random starting location
  - Random starting time

- **Metric used**
  - Success rate with respect to the deadline

- **Approaches**
  - MDP
  - Local/global sink
  - Waiting
  - Random walk

![Graph showing the success rate with path length for different approaches]
Minimum Interference Coverage

- Plan paths that **cover** the entire space, **minimizing** the interference with humans

- Example: Autonomous vacuum cleaner
  - Cleans the whole house
  - Cleans room when people are not there
  - Uses the routes with the minimum traffic
Minimum Interference Coverage

- **Time-dependent TSP**
  - Nodes: rooms
  - Edges: doorways
  - Costs: Poisson rates

- **Challenges and properties**
  - **Sparseness**: TSP is usually fully connected
  - **Asymmetry**: presence of node costs
  - **Time dependence**: Poisson rates vary over time
Minimum Interference Coverage

Algorithm: ATDTSP

- Generate the room graph
- Complete the graph (Floyd-Warshall)
- Solve the TSP (dynamic programming)
Minimum Interference Coverage

Algorithm: ATDTSP

- Generate the room graph
- Complete the graph (Floyd-Warshall)
- Solve the TSP (dynamic programming)
Minimum Interference Coverage

Algorithm: ATDTSP

- Generate the room graph
- Complete the graph (Floyd-Warshall)
- Solve the TSP (dynamic programming)
Minimum Interference Coverage

Algorithm: ATDTSP

- Generate the room graph
- Complete the graph (Floyd-Warshall)
- Solve the TSP (dynamic programming)
Preliminary results

- **Experiment setup**
  - 10 simulation days
  - 1000 paths
  - Random starting location
  - Random starting time
  - Coverage/transit times

- **Metric used**
  - Interference time
  - People interfered

- **Approaches**
  - Dynamic programming
  - Greedy/NN heuristic
  - General TSP
**Complexity and Heuristics**

- **Dynamic programming too expensive**
  - $O(N^{2^N})$ in time
  - $O(2^N)$ in space

- **Graph completion also expensive**
  - Floyd-Warshall for every time step $O(N^4)$

- **Heuristics**
  - Greedy $O(N^2 \log^2 N)$
  - Nearest neighbor $O(N^2)$
  - Good search heuristic for asymmetric problems?

- **TSP: good formulation?**
  - No sparseness
  - Complex reduction

- **Alternatives?**
  - Symbolic planning?
  - Temporal planning?
Conclusions

- Novel planning problems for social robots
  - Maximum encounter probability
  - Minimum interference coverage

- Learn and reason about human activities
  - Spatial affordance map

- Simulator engine of populated environments
  - Three realistic scenarios
  - Code available soon (mail me!)

tipaldi@informatik.uni-freiburg.de