Decentralised Multi-UAV Coordination using the Max-Sum Algorithm

Zhe Xu
ARC Centre of Excellence for Autonomous Systems
Overview

• Motivations

• Related work

• The max-sum algorithm

• Our solution

• Results

• Future work
Motivations: Overview

• There is active research interest in decentralised multi-robot coordination

• The max-sum algorithm is the state of the art in multi-agent coordination

• Goal: to apply the max-sum algorithm for the decentralised coordination of UAVs
Motivations: Problem

• Given a team of UAVs, find the set of joint controls that optimises a team utility function in a decentralised manner

• Why coordinate?
  • Utilities of the UAVs are not independent

• Example applications:
  • Search – sub-additive  \[ U(a) + U(b) \geq U(a + b) \]
  • Track – super-additive  \[ U(a) + U(b) \leq U(a + b) \]
Overview

- Motivations
- Related work
- The max-sum algorithm
- Our solution
- Results
- Future work
Related Work

• Distributed Constraint Optimisation Problem
  • Coordinated environmental monitoring

• Cooperative search and track
  • D Cole
  • F Bourgault

• Pursuit evasion
Overview

• Motivations
• Related work
• The max-sum algorithm
  • Characteristics
    • Utility function
    • Mechanics
• Our solution
• Results
• Future work
Max-Sum Algorithm: Characteristics

- Approximate algorithm for the Distributed Constraint Optimisation Problem
- Discrete variables
- Scales exponentially with number of “local” robots
- Based on message-passing
Max-Sum Algorithm: Utility Function

- Team utility function decomposes into sum of individual utility functions, e.g.

\[ F(x_1, x_2, x_3) = f_1(x_1) + f_2(x_1, x_2) + f_3(x_1, x_2, x_3) \]

- Represented as factor graph

- In a robotics context, variable nodes represent controls, function nodes represent utilities
Max-Sum Algorithm: Messages

- Two types of messages

  - Function to Variable - encodes maximum utility of function for each value of the variable

  - Variable to Function - encodes maximum utility of other function nodes for that value of the variable
Max-Sum Algorithm: Messages

• Function to Variable message

\[
r_{m \rightarrow n}(x_n) = \max_{\overline{x}_m} \left( U_m(\overline{x}_m) + \sum_{n' \in N(m) \setminus n} q_{n' \rightarrow m}(x_{n'}) \right)
\]

• Example:

\[
x_i = \{0, 1\}
\]

\[
\begin{align*}
\text{max} & \left( U_m \left( \begin{bmatrix} 0 \\ x_j \\ x_k \end{bmatrix} \right) + \sum_{j \rightarrow n} q_{j \rightarrow n}(x_j) + \sum_{k \rightarrow n} q_{k \rightarrow n}(x_k) \right) \\
\text{max} & \left( U_m \left( \begin{bmatrix} 1 \\ x_j \\ x_k \end{bmatrix} \right) + \sum_{j \rightarrow n} q_{j \rightarrow n}(x_j) + \sum_{k \rightarrow n} q_{k \rightarrow n}(x_k) \right)
\end{align*}
\]
Max-Sum Algorithm: Messages

- Variable to Function messages

\[ q_{n \to m}(x_n) = \alpha_{nm} + \sum_{m' \in M(n) \setminus m} r_{m' \to n}(x_n) \]

- Example:

\[ q_{n \to i} = \left[ \begin{array}{c} \sum_{j} r_{j \to n}(0) + \sum_{k} r_{k \to n}(0) \\ \sum_{j} r_{j \to n}(1) + \sum_{k} r_{k \to n}(1) \end{array} \right] \]

\( x_n = \{0,1\} \)
Max-Sum Algorithm: Decisions

- Stop iterating when message values converge or after \( n \) iterations

- Evaluate the marginal function

\[
Z_n(x_n) = \sum_{m \in M(n)} r_{m \rightarrow n}(x_n)
\]

- Selected control is one with maximum value

\[
x'_n = \text{argmax}(Z_n)
\]
Overview

- Motivations
- Related work
- The max-sum algorithm
  - Utility function formulation
  - Architecture
  - Receding horizon control
- Our solution
- Results
- Future work
Solution: Utility Function

- How to decompose team utility so it is sum of individual utilities?

- Use incremental utilities
  - Establish unique ID for each robot
  - Incremental utility of each robot is the additional utility gain on top of those with lower ID
  - Exploits sparsity of interactions in may cases
Solution: Utility Function

- Example:

\[\begin{align*}
1 & \quad 2 \\
2 & \quad 3 \\
3 & \quad 1
\end{align*}\]

- Example:

\[\begin{align*}
1 & \quad 2 \\
2 & \quad 3 \\
3 & \quad 4
\end{align*}\]
Solution: Architecture

Navigation sensor measurements → Navigation

UAV state estimate → Motion model
  - Predefined trajectories
  - Prediction horizon

Predict sensor measurement → Predict sensor measurement

Sensor model → Build decision tree

Build decision tree → Max-sum algorithm

Max-sum algorithm → Low level control
  - Bank angle command

Actuator commands → Actuator commands

Low level control → Actuator commands

Motion model

Sensor model

Sensor measurement → Decentralised Data Fusion

Decentralised Data Fusion

Sensor model

P(x|Z)

P(z'|x)

Max-sum messages

P(z|x) or P(x|z)

DDF messages

P(z|x) or P(x|z)
Solution: Predefined Controls

- Standard max-sum algorithm requires discrete control space
Solution: Decision Tree

- Build decision tree to compute function to variable messages

\[
    r_{m \rightarrow n}(x_n) = \max_{\vec{x}_m \setminus n} \left( U_m(\vec{x}_m) + \sum_{n' \in N(m) \setminus n} q_{n' \rightarrow m}(x_{n'}) \right)
\]

- Encodes all possible values of \( \vec{x}_m \)

- Use predicted observations to determine which other UAVs should be part of \( \vec{x}_m \)
Solution: Receding Horizon Control

• Further challenges
  • Operating in dynamic environment
  • Imperfect models used for prediction

• Standard solution: receding horizon control
Overview

- Motivations
- Related work
- The max-sum algorithm
- Our solution
- Results
  - System in operation
  - Benchmarks
  - Limited communications
- Future work
Results: System in Operation
Results: Benchmarks

Time for 95% Detection Probability (s)

No Coordination Implicit Coordination Best Response Max Sum

5 UAVs
Results: Limited Communications

- Range limited communications model

![Graph showing performance improvement over no coordination for different communications ranges.]

- Max Sum
- Best Response
- Implicit
Overview

• Motivations

• Related work

• The max-sum algorithm

• Our solution

• Results

• Future work
Future Work

- Extension to tracking of multiple targets
- Real-time demonstration
The Max-Sum Algorithm: Example

- Numerical example
  - Global utility function:
    \[ F(x_1, x_2, x_3) = f_1(x_1) + f_2(x_1, x_2) + f_3(x_2, x_3) \]

- Factor graph

\[ f_1 = \begin{bmatrix} 10 \\ 8 \end{bmatrix} \]
\[ f_2 = \begin{bmatrix} x_2 \\ 3 & 4 \end{bmatrix} x_1 \]
\[ f_3 = \begin{bmatrix} x_3 \\ 4 & 1 \end{bmatrix} x_2 \]