Anytime Planning for Decentralized Multi-Robot Active Information Gathering

Brent Schlotfeldt\textsuperscript{1}  Dinesh Thakur\textsuperscript{1}  Nikolay Atanasov\textsuperscript{2}  
Vijay Kumar\textsuperscript{1}  George Pappas\textsuperscript{1}

\textsuperscript{1}GRASP Laboratory
University of Pennsylvania
Philadelphia, PA, USA

\textsuperscript{2}Department of Electrical and Computer Engineering
University of California San Diego
San Diego, CA, USA

May 17, 2018
What is the problem?

Suppose the robot at $x_0$ wants to plan a path to $x_T$ to track a target $y_T$. 
Overview

1. **Active Information Gathering**
   - Problem Statement
   - Reduction to Open-Loop Control, and Optimal Solution
   - \((\epsilon, \delta)\)-Reduced Value Iteration
   - Anytime Reduced Value Iteration

2. **Multiple Robots**
   - Coordinate Descent
   - Distributed Estimation
   - Application: Target Tracking
Consider a team of \( n \) mobile robots, with sensor motion models:

\[
x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \quad i \in \{1, \ldots, n\}
\] (1)

where \( x_{i,t} \in \mathcal{X} \), \( u_{i,t} \in \mathcal{U} \), and \( \mathcal{U} \) is a finite set.
Problem Statement

Consider a team of $n$ mobile robots, with sensor motion models:

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \quad i \in \{1, \ldots, n\}$$

where $x_{i,t} \in \mathcal{X}$, $u_{i,t} \in \mathcal{U}$, and $\mathcal{U}$ is a finite set.

The robots goal is to track a target $y_t$ with target motion model:

$$y_{t+1} = Ay_t + w_t, \quad w_t \sim \mathcal{N}(0, W) \quad t = 0, \ldots, T$$
Problem Statement

Consider a team of \( n \) mobile robots, with sensor motion models:

\[
x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \quad i \in \{1, \ldots, n\}
\]  

(1)

where \( x_{i,t} \in X \), \( u_{i,t} \in U \), and \( U \) is a finite set.

The robots goal is to track a target \( y_t \) with target motion model:

\[
y_{t+1} = Ay_t + w_t, \quad w_t \sim \mathcal{N}(0, W) \quad t = 0, \ldots, T
\]

(2)

The robots are able to sense the target using on-board sensors obeying the following sensor observation model:

\[
z_{i,t} = H(x_{i,t})y_t + v_t(x_{i,t}), \quad v_t \sim \mathcal{N}(0, V(x_{i,t}))
\]

(3)
Consider a team of $n$ mobile robots, with sensor motion models:

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \quad i \in \{1, \ldots, n\}$$

where $x_{i,t} \in \mathcal{X}$, $u_{i,t} \in \mathcal{U}$, and $\mathcal{U}$ is a finite set.

The robots goal is to track a target $y_t$ with target motion model:

$$y_{t+1} = Ay_t + w_t, \quad w_t \sim \mathcal{N}(0, W) \quad t = 0, \ldots, T$$

The robots are able to sense the target using on-board sensors obeying the following sensor observation model:

$$z_{i,t} = H(x_{i,t})y_t + v_t(x_{i,t}), \quad v_t \sim \mathcal{N}(0, V(x_{i,t}))$$

- Note the Linear, Gaussian assumptions on the target and observations, but not on the robot.
- The set $\mathcal{U}$ of admissible controls is finite.
- To simplify notation, we refer to the state of the team as $x_t$ at time $t$. 
Active Information Gathering

Problem (Active Information Acquisition)

Given an initial sensor pose \( x_0 \in X \), a prior distribution over the target state \( y_0 \), and a finite planning horizon \( T \), the task is to choose a sequence of control actions: \( u_0, \ldots, u_{T-1} \in U^T \) which minimize conditional entropy \( h \) between the target state: \( y_{1:T} \) and the measurement set \( z_{1:T} \):

\[
\min_{u \in U^T} J_T^{(n)}(u) := \sum_{t=1}^{T} h(y_t | z_{1:t}, x_{1:t})^\dagger
\]

\[
s.t. \quad x_{t+1} = f(x_t, u_t), \quad t = 0, \ldots, T - 1
\]
\[
y_{t+1} = Ay_t + w_t \quad t = 0, \ldots, T - 1
\]
\[
z_t = H(x_t)y_t + v_t \quad t = 0, \ldots, T
\]

† Note that many other information measures are possible, such as mutual information, minimum mean-squared error, and others.¹

Reduction to Open-Loop Control

- Prior $y_0$ is Gaussian $\mathcal{N}(y_0, \Sigma_0)$, $\Rightarrow$ Kalman filter is optimal.
- Objective proportional to log det, i.e. $h(y_t|z_{1:t}, x_{1:t}) \sim \log \det(\Sigma_t)\) $ 
- $\Sigma_t$ independent of realizations $z_{1:t}$ $\Rightarrow$ open loop control optimal!

Problem (Active Information Acquisition)

$$ \min_{\mu \in \mathcal{U}^T} \quad J^{(n)}_T(\mu) := \sum_{t=1}^{T} \log \det(\Sigma_t) $$

s.t.  

$$ x_{t+1} = f(x_t, u_t), \quad t = 0, \ldots, T - 1 $$

$$ \Sigma_{t+1} = \rho_{t+1}^p(\rho_t^e(\Sigma_t), x_{t+1}), \quad t = 0, \ldots, T - 1 $$

Update: 

$$ \rho_t^e(\Sigma, x) := \Sigma - K_t(\Sigma, x)H_t(x)\Sigma $$

$$ K_t(\Sigma, x) := \Sigma H_t(x)^T (H_t(x)\Sigma H_t(x)^T + V_t(x))^{-1} $$

Predict: 

$$ \rho_t^p(\Sigma) := A_t \Sigma A_t^T + W_t $$

---

Optimal Solution

Deterministic optimal control problem can be solved with Forward Value Iteration, which builds a search tree from $(x_0, \Sigma_0)$.

- Full search has complexity $O(||U||^T)$.
- Can we do better?
What if trajectories cross?

- Want: Conditions on $\Sigma_t$ to eliminate nodes if trajectories cross at level $t$ of the tree.

There is a convex combination of $\Sigma_1$, and $\Sigma_3$ that dominates $\Sigma_2$, i.e. $\Sigma_2 \succ a\Sigma_1 + (1-a)\Sigma_3$. It can be safely removed from the tree.
Want: Conditions on $\Sigma_t$ to \textit{eliminate} nodes if trajectories cross:\footnote{Michael P Vitus et al. “On efficient sensor scheduling for linear dynamical systems”. In: \textit{Automatica} 48.10 (2012), pp. 2482–2493.}

\begin{definition} ($\epsilon$-Algebraic Redundancy )

Let $\epsilon \geq 0$ and let $\{\Sigma_i\}_{i=1}^K \subset S^n_+$ be a finite set. A matrix $\Sigma \in S^n_+$ is $\epsilon$-algebraically redundant with respect to $\{\Sigma_i\}$ if there exist nonnegative constants $\{\alpha_i\}_{i=1}^K$ such that:

$$\sum_{i=1}^K \alpha_i = 1 \quad \text{and} \quad \Sigma + \epsilon I \succeq \sum_{i=1}^K \alpha_i \Sigma_i$$

\end{definition}

Example: Suppose $a\Sigma_1 + (1-a)\Sigma_3$ is a convex combination of $\Sigma_1$, and $\Sigma_2$. $\Sigma_1$ and $\Sigma_3$ cannot strictly eliminate $\Sigma_2$, because it is not contained in $a\Sigma_1 + (1-a)\Sigma_3$. However if we relax the inequality by $\epsilon I$, we can remove $\Sigma_2$. 

$\Sigma_1$ $\Sigma_2$ $\Sigma_3$

$\Sigma_2 + \epsilon I$

$a\Sigma_1 + (1-a)\Sigma_3$
$(\epsilon, \delta)$- Reductions

- Can we remove even more trajectories?
- What if the trajectories do not exactly cross at time $t$?

Definition (Trajectory $\delta$-Crossing)

Trajectories $\pi^1, \pi^2 \in \mathcal{X}^T$ $\delta$-cross at time $t \in [1, T]$ if $d_{\mathcal{X}}(\pi^1_t, \pi^2_t) \leq \delta$ for $\delta \geq 0$. 

\[
\Sigma_1 \quad \Sigma_2 + \epsilon I \succeq a\Sigma_1 + (1 - a)\Sigma_3
\]
(ε, δ) Reduced Value Iteration

Idea: Prune trajectories close in space and similar in informativeness

Expanding Nodes in Search Tree ⇒

(ε, δ)-Pruning ⇒

Algorithm 1 (ε, δ) Reduced Value Iteration

1: J₀ ← 0, S₀ ← {(x₀, Σ₀, J₀)}. Sₜ ← ∅ for t = 1, ... T
2: for t = 1 : T do
3:   for all (x, Σ, J) ∈ Sₜ₋₁ do
4:     for all u ∈ U do
5:       xₜ ← f(x, u), Σₜ ← ρₜ₊₁(ρₜ(Σ, x, u), xₜ))
6:       Jₜ ← J + log det(Σₜ)
7:       Sₜ ← Sₜ ∪ {(xₜ, Σₜ, Jₜ)}
8:   end for
9: end for
10: Sort Sₜ in ascending order according to log det(·)
11: Sₜ′ ← Sₜ[1]
12: for all (x, Σ, J) ∈ Sₜ \ Sₜ[1] do
13:   % Find all nodes in Sₜ′, which δ-cross x:
14:   Q ← {((Σ′)|(x′, Σ′, J′)) ∈ Sₜ′, dₓ(x, x′) ≤ δ
15:   if isempty(Q) or not(Σ is ε-alg-red wrt Q) then
16:     Sₜ′ ← Sₜ′ ∪ (x, Σ, J)
17:   end if
18: end for
19: Sₜ ← Sₜ′
20: end for
21: return min {J | (x, Σ) ∈ Sₜ}
Performance Guarantee

Search tree with \((\epsilon, \delta)\)-pruning is \((\epsilon, \delta)\)-**suboptimal** with cost \(J_T^{\epsilon, \delta}\) such that the bound holds\(^4\)

\[
0 \leq J_T^{\epsilon, \delta} - J_T^* \leq (\zeta_T - 1) \left[ J_T^* - \log \det(W) \right] + \epsilon \Delta_T
\]  

(6)

where

\[
\zeta_T := \prod_{\tau=1}^{t-1} \left( 1 + \sum_{s=1}^{\tau} L_f^s L_m \delta \right) \geq 1
\]  

(7)

and \(\Delta_T, L_f^s\) and \(L_m\) are constants resulting from continuity assumptions.

- Note the tradeoff in performance: \((\epsilon, \delta) \to (0, 0) \implies J_T^{\epsilon, \delta} \to J_T^*\).
- If \((\epsilon, \delta) \to (\infty, \infty)\), we get a **greedy** policy with no guarantee.
- This bound is **monotone** in \(\epsilon, \delta\).

---

How do $\epsilon, \delta$ affect execution time?

What if problem dimension changes? (e.g. more targets)

What if team changes? (e.g. more robots)

All of these can be addressed by modifying the search algorithm to \textit{iteratively} construct the tree, adding progressively more trajectories until execution time runs out, always returning a feasible solution. This is called an \textit{anytime} algorithm.
Anytime Reduced Value Iteration

- Idea: First build a greedy search tree, then iteratively \textit{ImprovePath} until time expires
- To efficiently re-use computations, we mark all computed nodes $S_t$ as open and closed, i.e. $O_t$, $C_t$ at all levels $t$ in the tree\(^5\)

**Algorithm 2** Anytime Reduced Value Iteration ($x_0$, $\Sigma_0$, $T_{ARVI}$)

1: $J_0 \leftarrow 0$, $S_0 \leftarrow \{(x_0, \Sigma_0, J_0)\}$. $S_t \leftarrow \emptyset$ for $t = 1, \ldots, T$
2: $C_0 \leftarrow \emptyset$
3: $O_t \leftarrow S_t$ for $t = 0, \ldots, T$
4: $S \leftarrow \{S_0, \ldots, S_T\}$, $O \leftarrow \{O_0, \ldots, O_T\}$, $C \leftarrow \{C_0, \ldots, C_T\}$
5: $\epsilon \leftarrow \infty$, $\delta \leftarrow \infty$
6: $\{S, O, C\} \leftarrow \text{ImprovePath}(S, O, C, \epsilon, \delta)$
7: Publish best solution from $O[T]$
8: \textbf{while} Time Elapsed $\leq T_{ARVI}$ \textbf{do}
9: Decrease ($\epsilon$, $\delta$)
10: $\{S, O, C\} \leftarrow \text{ImprovePath}(S, O, C, \epsilon, \delta)$
11: Publish best solution $J$ from $O[T]$
12: \textbf{end while}

Tree Interpretation

\[(ε, δ)\]

- \(t = 0\): \(x_0, Σ_0\)
- \(t = 1\): \(x_1, Σ_1\)
- \(t = 2\): \(x_2, Σ_2\)
- \(t = 3\): \(x_3, Σ_3\)

Node not yet computed

Node ∈ \(S_t\)

Node ∈ \(O_t\)

Node ∈ \(C_t\)
AlGORITHM 3 ImprovePath (S, O, C, ϵ, δ)

1: for t = 1 : T do
2: for all (x, Σ, J) ∈ Ot−1 \ Ct−1 do
3: Ct−1 ← Ct−1 ∪ {x, Σ}
4: for all u ∈ U do
5: x_t ← f(x, u), Σ_t ← ρ_{x_t}(Σ)
6: J_t ← J + log det(Σ_t)
7: S_t ← S_t ∪ {(x_t, Σ, J_t)}
8: end for
9: end for
10: Sort S_t in ascending order according to log det(·)
11: Ot ← Ot ∪ S_t[1]
12: for all (x, Σ, J) ∈ S_t \ Ot do
13: % Find all nodes in Ot, which δ-cross x:
14: Q ← {Σ'|(x', Σ', J')} ∈ Ot, d_X(x, x') ≤ δ
15: Ot ← Ot ∪ (x, Σ, J)
16: end for
17: O[t] = Ot
18: end for
19: return {S, O, C}
Main Result

The following holds regarding the performance of ARVI:

**Theorem (ARVI)**

The following are satisfied by the ARVI algorithm:

(i) When \( \text{ImprovePath}(\epsilon, \delta) \) returns with finite \( (\epsilon, \delta) \), the returned solution is guaranteed to be \( (\epsilon, \delta) \)-sub-optimal.

(ii) The cost \( J^{(n)}_T \) of the returned solution decreases monotonically over time.

(iii) Given infinite time, \( C = \emptyset \subseteq S \), and \( O[T] \) will contain the optimal solution to the planning problem.

---

What about multiple robots?

- Algorithm still scales exponentially in the number of robots.
- Consider a *coordinate descent* approach\(^7\). Each robot solves their own planning problem with fixed trajectories from prior robots.

\[
u_{1,0}^c : (T-1) \in \arg \min_{\hat{\mu} \in U_1^T} J_T^{(1)}(\hat{\mu})
\]

This continues:

\[
u_{2,0}^c : (T-1) \in \arg \min_{\hat{\mu} \in U_2^T} J_T^{(2)}(u_{1,0}^c : (T-1), \hat{\mu})
\]

\[
\vdots
\]

Coordinate Descent reduces the planning complexity from exponential to linear in number of robots.

Algorithm achieves no less than 50% of the optimal performance, in the case of submodular objective functions.

Results can also be extended to (approximately) sub-modular objective functions.

---

What about Estimation?

The robots can use a distributed information filter\(^9\), which uses the inverse covariance \(\Omega_{i,t} = \Sigma_{i,t}^{-1}\) and \(\omega_{i,t} = \Sigma_{i,t}^{-1} y_{i,t}\):

\[
\text{Update Step: } \quad \omega_{i,t+1} = \sum_{j \in \mathcal{N}_i \cup \{i\}} \kappa_{i,j} \omega_{j,t} + H_i^T V_i^{-1} z_i(t) \\
\Omega_{i,t+1} = \sum_{j \in \mathcal{N}_i \cup \{i\}} \kappa_{i,j} \Omega_{j,t} + H_i^T V_i^{-1} H_i
\]

\(\hat{y}_i(t) := \Omega_{i,t}^{-1} \omega_{i,t}\)

\[
\text{Predict Step: } \quad \Omega_{i,t+1} = (A \Omega_{i,t+1} A^T + W)^{-1} \\
\omega_{i,t+1} = \Omega_{i,t+1} A \hat{y}_{i,t+1}
\]

We use a discretized unicycle model for $f(x, u)$:

$$
\begin{pmatrix}
    x^1_{t+1} \\
    x^2_{t+1} \\
    \theta_{t+1}
\end{pmatrix} =
\begin{pmatrix}
    x^1_t \\
    x^2_t \\
    \theta_t
\end{pmatrix} +
\begin{pmatrix}
    \nu \operatorname{sinc} \left( \frac{\omega \tau}{2} \right) \cos(\theta_t + \frac{\omega \tau}{2}) \\
    \nu \operatorname{sinc} \left( \frac{\omega \tau}{2} \right) \sin(\theta_t + \frac{\omega \tau}{2}) \\
    \tau \omega
\end{pmatrix}
$$

(11)
Application: Target Tracking

We use a discretized unicycle model for \( f(x, u) \):

\[
\begin{pmatrix}
    x_{t+1}^1 \\
    x_{t+1}^2 \\
    \theta_{t+1}
\end{pmatrix} =
\begin{pmatrix}
    x_t^1 \\
    x_t^2 \\
    \theta_t
\end{pmatrix} +
\begin{pmatrix}
    \nu \operatorname{sinc}\left(\frac{\omega \tau}{2}\right) \cos(\theta_t + \frac{\omega \tau}{2}) \\
    \nu \operatorname{sinc}\left(\frac{\omega \tau}{2}\right) \sin(\theta_t + \frac{\omega \tau}{2}) \\
    \tau \omega
\end{pmatrix}
\] (11)

The target model is a double integrator:

\[
y_{t+1} = A \begin{bmatrix}
    l_2 & \tau l_2 \\
    0 & l_2
\end{bmatrix} y_t + w_t,
\quad w_t \sim \mathcal{N}\left(0, q \begin{bmatrix}
    \tau^3/3 l_2 & \tau^2/2 l_2 \\
    \tau^2/2 l_2 & \tau l_2
\end{bmatrix}\right)
\] (12)
Application: Target Tracking

We use a discretized unicycle model for $f(x, u)$:

$$
\begin{pmatrix}
  x^1_{t+1} \\
  x^2_{t+1} \\
  \theta_{t+1}
\end{pmatrix} =
\begin{pmatrix}
  x^1_t \\
  x^2_t \\
  \theta_t
\end{pmatrix} +
\begin{pmatrix}
  \nu \text{sinc}(\frac{\omega t}{2}) \cos(\theta_t + \frac{\omega t}{2}) \\
  \nu \text{sinc}(\frac{\omega t}{2}) \sin(\theta_t + \frac{\omega t}{2})
\end{pmatrix}
$$

(11)

The target model is a double integrator:

$$
y_{t+1} = A \begin{bmatrix} l_2 & \tau l_2 \\ 0 & l_2 \end{bmatrix} y_t + w_t, \quad w_t \sim \mathcal{N} \left( 0, q \begin{bmatrix} \tau^3/3 l_2 & \tau^2/2 l_2 \\ \tau^2/2 l_2 & \tau l_2 \end{bmatrix} \right)
$$

(12)

The robots have range and bearing sensors.

$$
z_{t,m} = h(x_t, y_{t,m}) + v_t, \quad v_t \sim \mathcal{N} \left( 0, V(x_t, y_{t,m}) \right)
$$

$$
h(x, y) = \begin{bmatrix} r(x, y) \\ \alpha(x, y) \end{bmatrix} := \begin{bmatrix} \sqrt{(y^1 - x^1)^2 + (y^2 - x^2)^2} \\ \tan^{-1}((y^2 - x^2)(y^1 - x^1)) - \theta \end{bmatrix}
$$

(13)
Linearization, and MPC

Due to the non-linear observation model, we linearize about a predicted target trajectory $y$ and apply MPC:

$$\nabla_y h(x, y) = \frac{1}{r(x, y)} \begin{bmatrix} (y^1 - x^1) \\ -\sin(\theta + \alpha(x, y)) \\ \cos(\theta + \alpha(x, y)) \end{bmatrix} \begin{bmatrix} (y^2 - x^2) \\ 0_{1x2} \end{bmatrix}$$  \hspace{1cm} (14)
Linearization, and MPC

Due to the non-linear observation model, we linearize about a predicted target trajectory $y$ and apply MPC:

$$\nabla_y h(x, y) = \frac{1}{r(x, y)} \begin{bmatrix} (y_1 - x_1) & (y_2 - x_2) & 0_{1x2} \\ -\sin(\theta + \alpha(x, y)) & \cos(\theta + \alpha(x, y)) & 0_{1x2} \end{bmatrix}$$

(14)

Algorithm 5 Anytime Multi-Robot Target Tracking

1: **Input:** $T_{\text{max}}, x_0, \hat{\omega}_0, \hat{\Omega}_0, f, U, H, V, A, W, T, T_{\text{ARVI}}$
2: **while** $t = 1 : T_{\text{max}}$ **do**
3: Send $\omega_{i,t}$ and $\Omega_{i,t}$ to neighboring robots.
4: Receive measurements $z_{i,t}$ and perform distributed update step with any neighbor $\omega_{i,t}, \Omega_{i,t}$ received.
5: Predict a target trajectory of length $T$: $\hat{y}_t, ... \hat{y}_T$, and linearize observation model: $H_t \leftarrow \nabla_y h(\cdot, \hat{y}_t)$
6: Plan $T$-step trajectories with ARVI (Alg. 2) and coordinate descent.
7: Apply first control input to move each robot.
8: **end while**
Figure: Simulation results with 9 robots tracking 16 moving targets. The top row sweeps over communication radius, and the bottom row sweeps planning time.
Future Work:

- Resilient Active Information Acquisition (IROS 2018)
- Unknown Target Models
- Quadrotor Motion Model, Camera observation Model, Continuous Actions, etc.