# Adaptive Path Planning for Effective Information Collection

IROS Workshop on AI and Robotics, 2014

**Ayan Dutta, Raj Dasgupta**

C-MANTIC Lab

Computer Science Department

University of Nebraska at Omaha


# Outline

- What is informative path planning?
- Background
- Model
- Algorithms
- Experiments.
- Conclusion.

# Adaptive Path Planning

- We consider a multi-robot scenario, where robots are **heterogeneous** in terms of available sensors.
- Sensing same location with same sensors earn no information gain.
- Sensing  same location with different sensors earn higher information gain.
- Paths of robots should be adjusted (or adapted) depending on these criteria.
- Each robot has a specific battery budget – path length is limited.

# Informative Path Planning

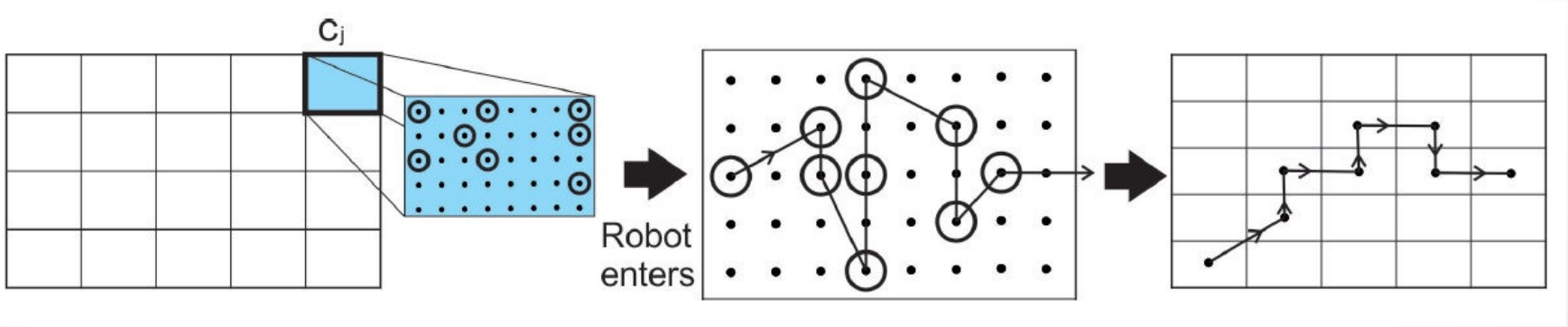
- Information Collection from an environment by a multi-robot system (MIPP).
- No fixed goal point for navigation purpose – robots can roam around within battery budget.
- Performance of multi-robot information collection with multiple sensors can be improved, if a robot's path is not only updated using its self-sampled information, but, the information collected by other robots as well as the sensor capabilities of those robots is incorporated dynamically into robots' path plan.

# Background

- **Guestrin *et al.(2005)*** proposed a greedy algorithm for sensor placement for information collection.
- **Singh *et al.(2009)*** have proposed a recursive, branch and bound algorithm to solve the MIPP problem that finds the best, budget-limited path through a graph of possible waypoints.
- MIPP problem with periodic connectivity between robots has been studied by **Hollinger and Singh *(2010)*.**
- Path hybridization algorithm for a single robot has been proposed by **Raveh *et al. (2011).***
- All of these previous works on MIPP problem has been studied in a scenario, where every robot has same set of sensors and dynamic uncertainty is not involved in information collection.

# Model

- Set of *N* robots are considered – each robot $r_i$ has set of sensors $S_i$.
- Each sensor has a specific, pre-defined reliability value.
- Map of the environment is assumed to be known by the robot and it is decomposed into cells (using cellular decomposition method).
- Each cell $c_j$ consists of a set of information points.
- Upon entering a cell, a robot dynamically selects a subset of information points in the cell to visit and take readings from, using sensors.
- Information collected by $r_i$ from cell $c_j$ is: $I_{i,j} = -\sum_{p_{k,j} \in P_j} P(p_{k,j}) \log_b P(p_{k,j}) \times \sum_{s \in S_i} \rho_s$



# Contd..

- Total information collected from a robot's path: $I(\Pi_i) = \sum_{c_j \in \Pi_i} I_{i,j}$
- Cost of visiting cell $c_j$: $cost_{i,j} = cost^{i,j}_{sense} + cost^{i,j}_{travel}$
- Total cost incurred in a robot's path: $cost(\Pi_i) = \sum_{c_j \in \Pi_i} cost_{i,j}$
- Utility of a single robot's path: $U(\Pi_i) = I(\Pi_i) - cost(\Pi_i)$
- Total utility earned from two different paths:

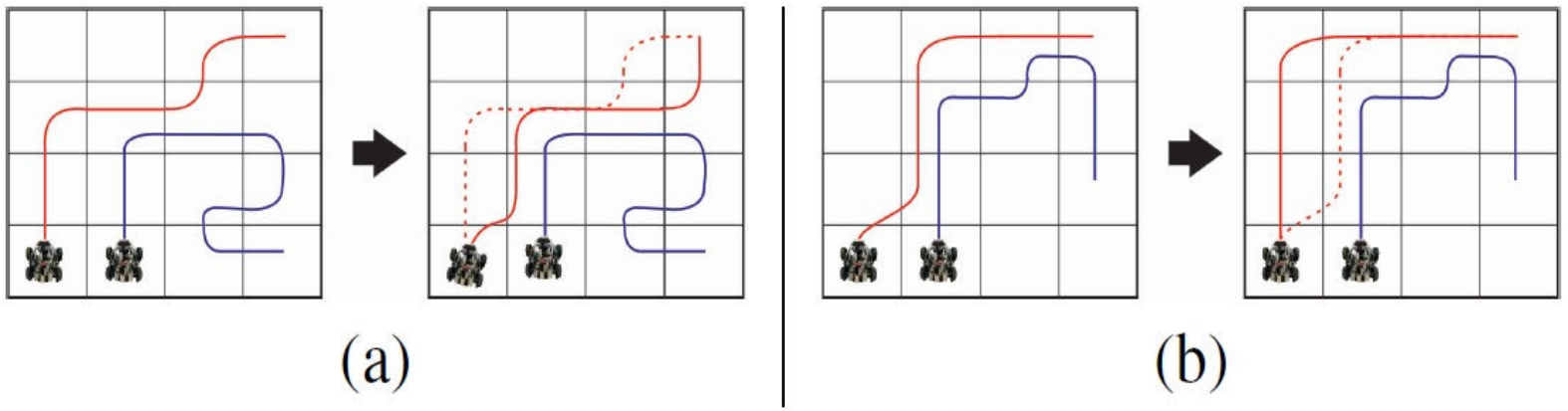
$$U(\Pi_{ij} = \begin{cases} U(\Pi_i) + U(\Pi_j) \quad - \sum_{c_k \in \Pi_i \cap \Pi_j} I_{i,k} \\ \quad \text{if } S_i \cap S_j \neq \{\emptyset\} \\ U(\Pi_i) + U(\Pi_j) \quad +( \sum_{c_k \in \Pi_i \cap \Pi_j} I_{i,k} \times \sum_{s \in S_i \cup S_j} \rho_s) \\ \quad \text{if } S_i \cap S_j = \{\emptyset\} \end{cases}$$

# Path Generation Algorithm

- Initially all the robots will generate paths for moving through the environment to collect information.
- We have used path generation algorithm for information collection, similar to as proposed by Guestrin *et al.(2005).*
- We add 2 new cells to the current path in one cycle, addition of which to the current path yields highest amount of utility.

# Path Merging Algorithm

- The objective of path merging algorithm is to merge paths of robots, having different set of sensors to increase the information gain.



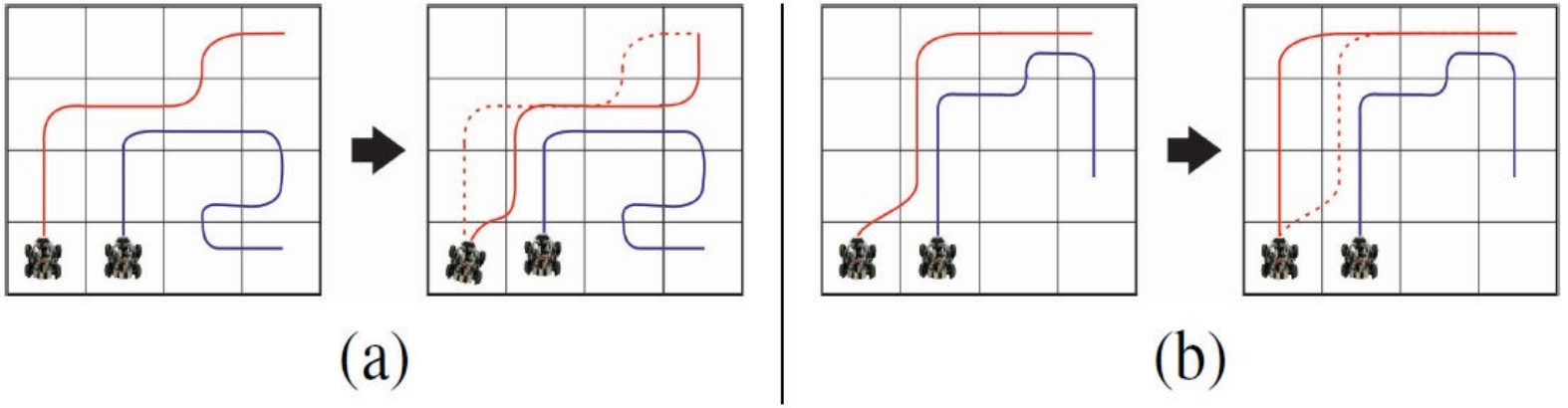
1 *pathMerge()*
**Input**: $\mathcal{K}_{sort}$: Sorted set of all paths.
**Output**: $\mathcal{K}^*$: A set of new merged paths.
2 $\mathcal{K}^* \leftarrow \{\emptyset\}$.
3 $\mathcal{R}_{sort}$: Sorted set of all robots, acc. to $\mathcal{K}_{sort}$.
4 Each $r_i \in \mathcal{R}_{sort}$ will do the following:
5 $BEST_U \leftarrow U(\Pi_i)$
6 **for** *all* $(\Pi_i, \Pi_j)pair, \exists \Pi_j \in \mathcal{K}_{sort}, i \neq j, S_i \cap S_j! = \{\emptyset\}$ **do**
7  **for** *each* $c_k \in \Pi_i$ **do**
8  **if** *(*$\exists c_l \in \Pi_j$, $c_l \notin \Pi_i$ *and replaceable(*$c_k, c_l$*) ==*
 *TRUE)* **then**
9  $\Pi_i \leftarrow \Pi_i \cup \{c_l\} \setminus \{c_k\}$. //replace $c_k$ with $c_l$.
10 $\mathcal{K}^* \leftarrow \Pi_i$.
11 Send $\mathcal{K}_{sort}$ and $\mathcal{K}^*$ to $r_{i+1} \in \mathcal{R}_{sort}$.

*replaceable(*$c_k, c_l$*)*
**Input**: $c_k \in \Pi_i$ and $c_l \in \Pi_j$.
**Output**: TRUE or FALSE.
12 $\Pi_i' \leftarrow \Pi_i \cup \{c_l\} \setminus \{c_k\}$.
13 **if**
$(U(\Pi_i') + U(\Pi_j) + I_{i,l} \times \sum_{s \in S_i \cup S_j} \rho_s > BEST_U + U(\Pi_j))$
**then**
14  return TRUE;
15 **else**
16  return FALSE;

This part checks whether replacing 2 cells from 2 Paths, yield more utility or not.

# Path Decoupling Algorithm

- If two or more robots, having same set of sensors visit same cell to collect information, then they do not gain any new information.
- Higher cost will be incurred without new information – low utility.
- To avoid this situation, robots having same set of sensors should not visit same cells.



1 *pathDecoupling()*
**Input**: $K$: Set of all paths.
**Output**: $\Pi_i$: Modified path of robot $r_i$.
2 $K_i \subseteq K$: Sorted set of *similar* paths with robot $r_i$'s path $\Pi_i$.
3 $R_{sort}$: Sorted set of robots according to $K_i$.
4 Each robot $r_k \in R_{sort}$ will do the following:
5 **for** *each path* $\Pi_k \in K_i$ **do**
6 **for** *each cell* $c_j \in (k \cap \Pi_i)$ **do**
7 $c_l \leftarrow$ Highest utility-earning *OK* cell in $neighbor(c_{j-1}) \setminus c_j$. (Details in text)
8 $\Pi_i \leftarrow \Pi_i \setminus c_j \cup c_l$.
9 Update $K_i$.
10 return $\Pi_i$.

If cells are similar in both paths, choose a different neighbor of its parent cell.

# Path Planning under Comm. Constraints

- Estimated utility of a path might change when robots actually visit the cell and sense the environment – uncertain (or dynamic) environment.
- Paths of the robots will be changed accordingly.
- New path information should be exchanged among robots to adapt to these new paths.
- Constant communication among robots is costly and non-effective.
- Re-planning and re-adapting should be done in certain time intervals.

1 **Each robot $r_i$ will follow these steps:**
2 Generate path $\Pi_i$, using *pathGeneration()* algorithm.
3 Exchange path information with in-range robots.
4 Decouple $\Pi_i$ from similar paths, using *pathDecoupling()* algorithm.
5 Merge $\Pi_i$ with appropriate robots' paths, using *pathMerge()* algorithm.
6 Send new path information to the next robot in $\mathcal{R}_{sort}$.
7 Start following $\Pi_i$.
8 Calculate actual utility earned and actual amount of battery spent.
9 **if** $(|U(\Pi_i) - U_{act}(\Pi_i)|) > THRES\_U \bigvee (|EstB(\Pi_i) - ActB(\Pi_i)|) > THRES\_B$ *at time interval* $T^{\alpha}$ **then**
10 Generate a new path $\Pi_i'$ and follow the path.
11 **else**
12 Follow path $\Pi_i$.
13 **if** $r_i$ *comes in comm. range of* $r_k$ *for the first time in time interval* $T^{\alpha}$ *and* $T^{\alpha+1}$ **then**
14 **if** $S_i \cap S_k ! = \{\emptyset\}$ **then**
15 Follow steps in line 4 and $6 - END$.
16 **else**
17 Follow steps in line $4, 5$ and $7 - END$.
18 **else**
19 Follow path $\Pi_i$.
20 **Termination Condition:** Generated path has been completely visited OR spent budgeted amount of battery.

If difference between expected and actual utility/battery cross a threshold, then generate a new path

If one robot comes in comm. Range with other for first time, in between 2 time intervals, then merge/decouple paths.

# Time Complexity

- ***Path merging algorithm:*** $O(nl^2)$, where $n$ and $l$ denote the number of robots and maximum length of a path.
- Comparable path hybridization algorithm (Raveh *et al. (2011)*) has complexity $O(n^2l^2)$.
- ***Path decoupling algorithm:*** $O(n^2bl)$, where $b$ denotes the number of neighbors of a cell (we assumed 8-connectivity in our experiments).
- ***Communication:*** Worst case will be $O(n^2)$. Note that, in real scenarios, robots will be scattered all over the place; thus communication complexity will be much less.

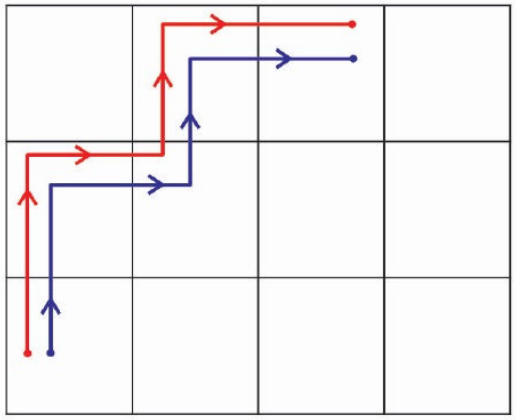
# Experiments

- Path adaptation algorithms are tested in simulated environments.
- An environment with 100 cells has been used.
- Maximum number of cells in a robot's path has been varied between [5,50].
- Reliability values of sensors are drawn from $U(0,1)$.
- Sensor cost (in term of power needed) is also normalized to [0,1].
- Each test is run 5 times – deviation in results were minimal.

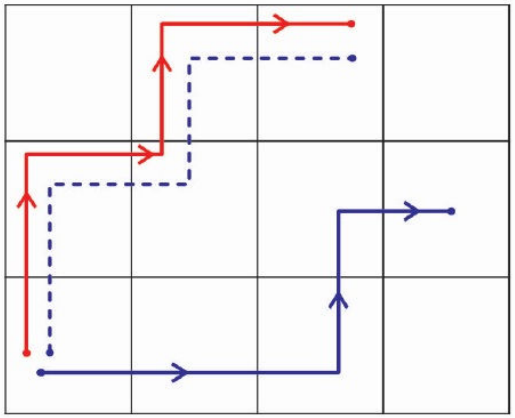
# Simulation Results: Path Decoupling

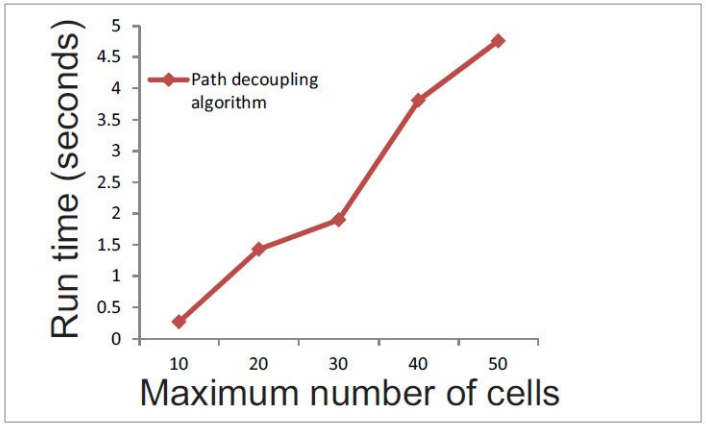
- Implemented on 2 robots, with same set of sensors.
- Run time was nominal.
- A scenario with maximum 5 cells and 2 robots has been implemented.

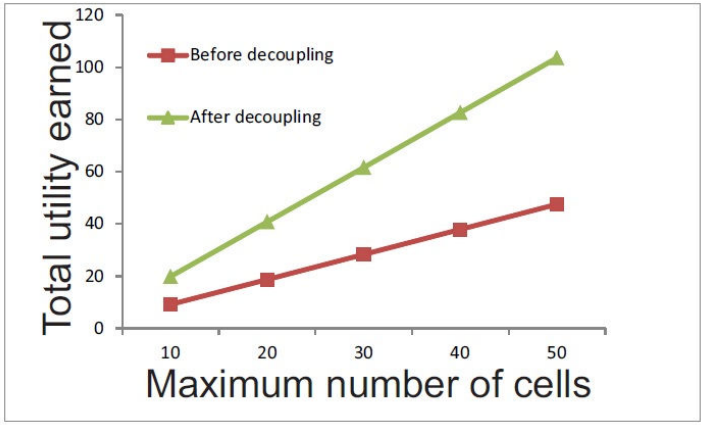
Before Decoupling:



After Decoupling:

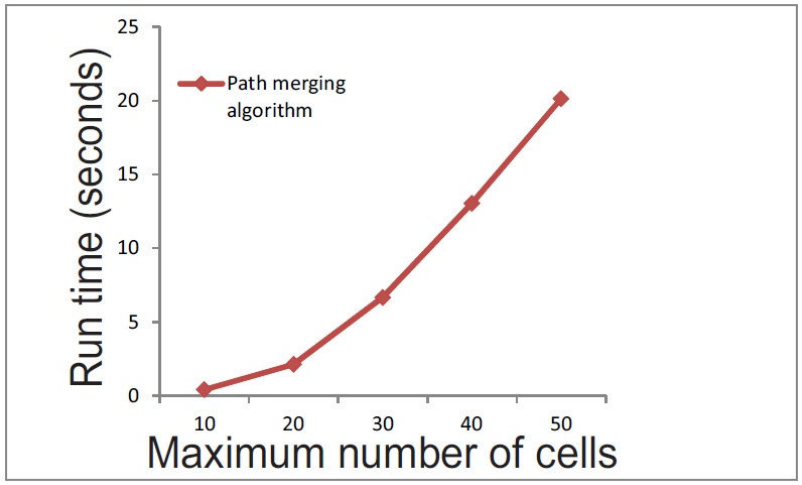




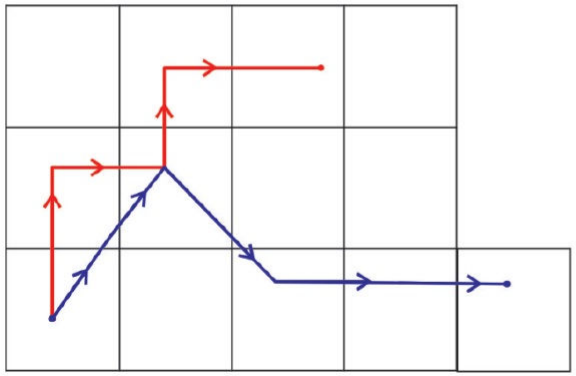



# Simulation Results: Path Merging

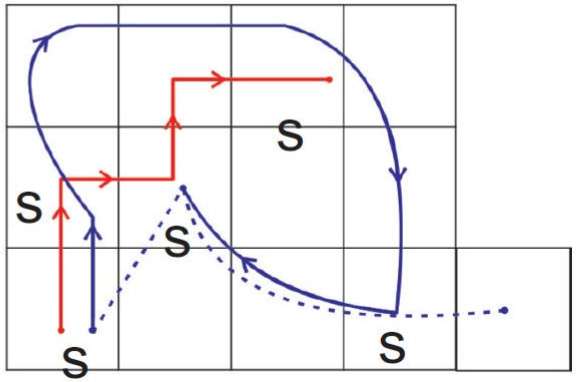
- Implemented on 2 robots, with different set of sensors.
- Run time was higher than decoupling algorithm.
- A scenario with maximum 5 cells and 2 robots has been implemented.

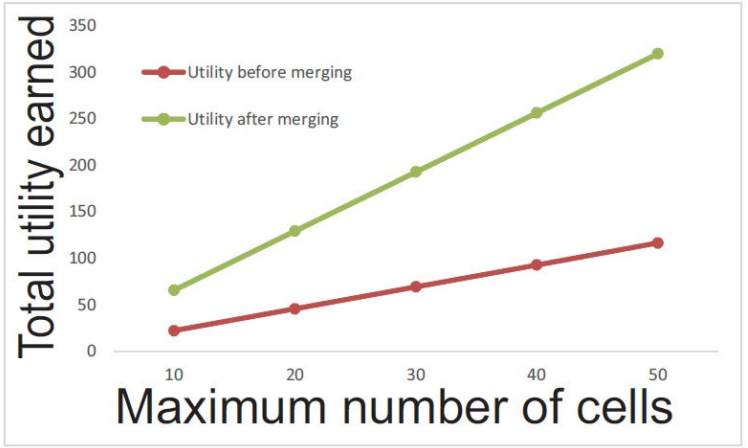



Before Merging:



After Merging:



# Conclusion and Future Work

- Proposed path adaptation algorithms for information collection.
- Communication constraints and uncertain, dynamic environment has ben taken into account.
- For future work, we will use more sophisticated **path generation algorithm**, which will earn higher utility.
- We will also **compare** our algorithms with other existing similar work.
- We will test our algorithms with **more number of robots**.
- We will implement our proposed algorithms on **real robots**, in real environment.


# Thank You!

Questions…