1

DYNAMIC SHORTEST PATH SEARCH AND SYNCHRONIZED TASK SWITCHING

Jay Wagenpfeil, Adrian Trachte

Outline

2

- Shortest Communication-Path Searching
  - Bellmann-Ford algorithm
  - Algorithm for dynamic case
  - Modifications to our algorithm
- Synchronized Task-Switching
  - Combining tasks
  - An algorithm for synchronized task-switching
  - Time complexity
- Summary

3

Bellman-Ford Algorithm

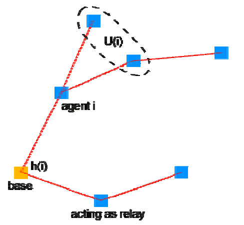
Algorithm

Restrictions in Dynamic Case

Communication

4

- Setup:
  - Network of agents, transmitting data to the base.
  - Communication costs, which increase with increasing distance between agents, should be kept low



➡ Routing protocol needed, to find the shortest path to the base

---

Bellman-Ford Shortest Path [1]

5

- Setup: Set of edges which are connected over vertices.
  Goal: Find shortest path from each agent to base.

Notation:

$c_i =$ communication cost to base of agent i

$N_i =$ set of neighbors of agent i

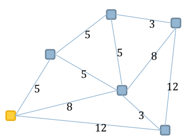
$v_{ij} =$ com. cost from agent i to agent j with $j \in N_i$

$d_i =$ downstream neighbor of agent i

Update rule for every agent:

$c_i = \min_{j \in N_i}(c_j + v_{ij})$

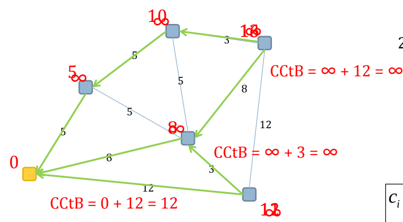
$d_i = \arg(\min_{j \in N_i}(c_j + v_{ij}))$



[1] Richard Bellman - **On a Routing Problem - 1958**

Bellman-Ford - Example

6



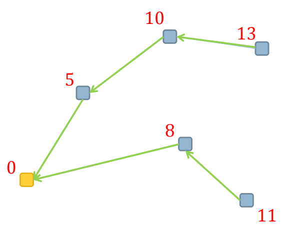
1. Initialization
   a) $c_B = 0$
   b) $c_i = \infty$
2. Agents search for possible new shortest path to base.

$c_i = \min_{j \in N_i}(c_j + v_{ij})$

$d_i = \arg(\min_{j \in N_i}(c_j + v_{ij}))$

Bellman-Ford - Example

7



1. Initialization
   a) $c_B = 0$
   b) $c_i = \infty$
2. Agents search for possible new shortest path to base.
3. Shortest path is found after maximum of N – 1 iterations

8

Bellman-Ford Algorithm

Algorithm

Restrictions in Dynamic Case

Dynamic shortest path search

9

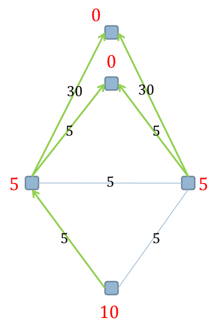
- Changes to static case:
  - Topology of network changes
  - Weightings of vertices vary over time

Certain problems occur in dynamic case

- **looping**
  communication loops occur, connection to base gets lost
- **„longest path“ search**
  because of old information in the network, agents choose wrong path to base

Looping in dynamic case

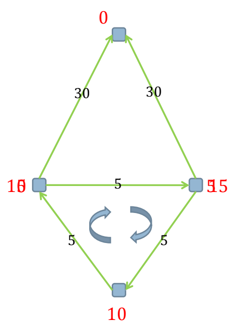
10



1) Agents in steady state
2) Downstream Neighbor moves away
   → communication cost increases
3) Agents search for new shortest path

Looping in dynamic case

11



1) Agents in steady state
2) Downstream Neighbor moves away
   → communication cost increases
3) Agents search for new shortest path

12

„Longest path“-search

13

Dynamic Shortest Path Search

Regular Search

High frequency search

Idea of Dynamic Shortest Path Search

14

- Problem: changing weights and time delayed information propagation leads to loops and wrong pathes
- But: No problem in static case, because here, the communication cost only decreases while converging to shortest path
- Idea: Fix the communication costs and topology between agents and use static computation to find shortest path
- **Advantages:**
  - Finds real shortest path for given setup (no „longest path“, loops)
- **Disadvantages:**
  - Needs time to converge, during this time, not optimal path

Realization of Dynamic Shortest Path Search

15

- **Procedure:**
  1) Fix communication costs to all neigbor agents
  2) Start new static shortest path search
  3) When shortest path search is finished, set new found downstream neigbhor as new d.n.
  4) Go to first step.
- **Two Problems:**
  - How do agents know when to finish the shortest path search and start with a new one with updated communication costs?
  - How assure that new found shortest path downstream neighbor is still in communication range?

Realization of Dynamic Shortest Path Search

16

- **First Problem:**
  Idea: Base is central processing unit and therefore can be used as a quasi synchronisation module to start the new search.
- New search should start, after shortest path to base was found.
  → Wait worst case time for shortest path search.
- New search signal will be propagated over whole communication range from each agent.
  → New search signal is faster than shortest path search.

⇨ Base sends new search signal to all agents in communication range after worst case computation time for shortest path

Realization of Dynamic Shortest Path Search

17

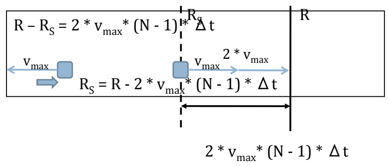
- **Worst case time** for new shortest path to base:
  - Worst case topology is connected chain
  - Worst case update is, if agents farest away from base update first.

Δt	maximum time within communicator updates → all communicators update at least once within timespan Δt
$\Delta t_{total} = (N - 1) * \Delta t$	is worst case time to find shortest path

Realization of Dynamic Shortest Path Search

18

- **Second Problem:**
  Idea: Agents should not move out of communication range while shortest path search
  - assume maximum speed of agent v_{max}
  - worst case if agents move in opposite direction with maximum speed, during whole worst case computation time



$N = 40 + 1$
$\Delta t = 0.02s$
$v_{max} = 5m/s$
$R = 21m$

$R_S = 13m$
$\Delta t_{total} = 0.8s$

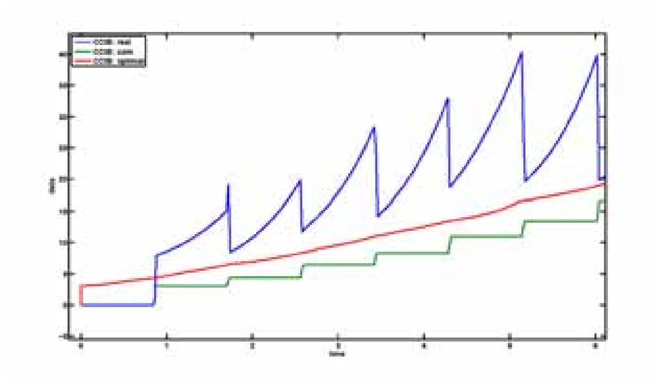
TP:
Li [2] + KTF

19 Dynamic shortest path search

[2] Li et al – **Distributed Cooperative Coverage Control of Sensor Networks - 2005**

Dynamic Shortest Path Search - Analysis

20



Dynamic Shortest Path Search - Limitations

21

- Worst case waiting time: $(N – 1) * \Delta t$
  - Increases with growing number of agents N and computation time Δt .
    - ⇨ Time between new searches becomes to big, and therefore the error increases.
- Limited neighbor range: $R_S = R - 2 * v_{max} * (N - 1) * \Delta t$
  - Decreases with growing number of Agents N, computation time Δt and v_{max}.
    - ⇨ Maybe R_S becomes to small for a proper shortest path search.

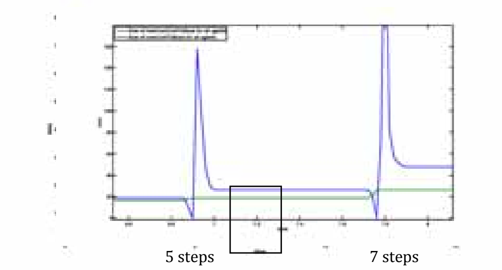
22 Dynamic Shortest Path Search

Regular Search
High frequency search

High search frequency - Motivation

23

- Idea: Increase the frequency with which a new search starts.
  - ⇨ Security Radius R_S would be bigger and time between searches smaller



$N = 40 + 1$
$\Delta t = 0.02s$
$v_{max} = 5m/s$
$R = 21m$

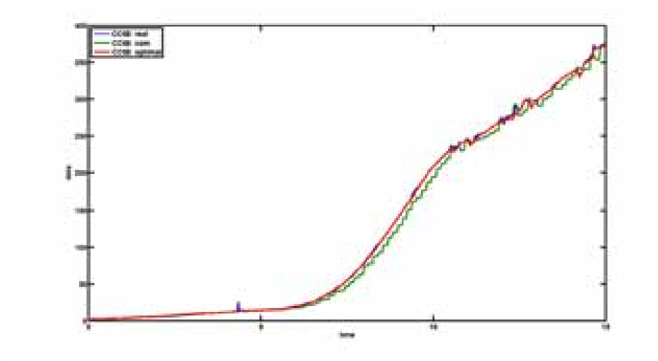
$R_S = 20m$
$\Delta t_{total} = 0.1s$

TP:
Li [2] + KTF

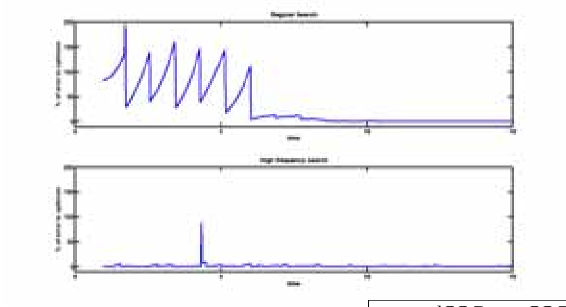
24 Higher new search frequency

[2] Li et al – **Distributed Cooperative Coverage Control of Sensor Networks - 2005**

High search frequency



High search frequency



ortest

$$100 * \frac{realCCtB - optCCtB}{optCCtB}$$

Discussion

Regular search

- **Positive:**
  - Finds shortest path using only local information (no looping etc.)
- **Negative:**
  - Strong dependence on number of agents etc.
  - Error while waiting for worst case convergence time

High frequency search

- **Positive:**
  - Reduces distance to optimum
- **Negative:**
  - Shortest path is not guaranteed to be found in computation time
  - Rough knowledge of topology needed

Synchronized Task-Switching

Combining Tasks

- Coverage control:
  - Maximizing the probability of detecting events.
  - ➔ Most important areas of the mission space are well covered.
- Exploration of the mission space:
  - Use of deployment algorithms.
  - Maximize the area covered by all agents.

Combining Tasks

- Combine both tasks:
  - First explore the mission space.
  - Then cover the most important areas.

➔ Enables the agents to cover areas unreachable if only using coverage control.

- Switch task when the exploration task is finished.

Combining Tasks

31

- How do the agents know that the exploration task is finished?
  - For each agent, only local information is available.
  - But, information about all agents (=global information) is necessary.
- Use of consensus-like algorithms
  - Enables each agent to determine the state of the network.
  - ➡ Task-switch is performed, when all agents agree that the exploration task is finished.

Notations

32

- Bi-directional communication between agent i and its neighbors N_i

  $N_i(k) = \{ j \in \{1,..,n\} \mid \|s_i - s_j\| < R \}$
- Communication topology is undirected graph G:

  A ... Adjacency matrix of G $a_{ij} = a_{ji} = 1$ if $j \in N_i(k)$

  D ... Degree matrix of G $d_{ii} = \sum_{j \neq i} a_{ij}$
- State variables for consensus:

  z ... Task state of agents $z_i = 1$ if agent i has finished first task

  x ... Consensus state

An Algorithm

33

- Assumptions:
  - There exists a time k_0 such that $A(k)=A(k_0)$ for all $k \geq k_0$.
  - There exists a time $k_1 \geq k_0$ s.t. $z(k)=\underline{1}$ for all $k \geq k_1$.
  - If $A(k+1) \neq A(k)$, that is there exists i s.t. $N_i(k+1) \neq N_i(k)$, then $z_i(k+1)=0$ even if agent i has finished first task.
- More Notations:

  Z ... diag(z(k))

  I ... n × n identity matrix

  $\underline{1}$ ... n × 1 vector with all elements equal to 1

  d_i ... $d_i=|N_i|$ is the cardinality of the set N_i

An Algorithm

34

- Algorithm:
  - If $z_i=1$, set each agent i's consensus variable x_i to the average value of the sum of its own task state z_i and the consensus states of its neighbors.
  - Else, set $x_i=0$.
- Update rule:
  - For each agent: $x_i(k+1) = z_i \cdot \frac{1}{d_i + 1} \cdot \left[\sum_{j \in N_i} x_j(k) + 1 \right]$
  - Whole network: $x(k+1) = Z \cdot (D+I)^{-1} \cdot [A \cdot x(k) + \underline{1}]$

An Algorithm

35

- State of the network and task-stwitch:
  - If at least one agent has not finished the first task, $x_i(k)<1$ for all agents.
  - If all agents have finished the first task, $z(k)=\underline{1}$ and for every agent i, $x_i(k) \rightarrow 1$ for $k \rightarrow \infty$.
  - Perform task-switch if x_i is sufficiently close to 1.
- Convergence of Algorithm:
  - For constant z(k), system is an asymptotically stable LTI-system with constant input.
  - There is always one unique equilibrium point x_{EP} and $x_{EP}=1$ for z=1.

Threshold for Task-Switch

36

- When is x_i sufficiently close to 1?
  - Task-switch if $x_i > \delta$, with $\delta < 1$
  - If δ is too small, false task-switch might happen.
- How to determine δ?
  - Derive from static case where no topology changes happen.
  - Show that even under switching topology, $x_i(k)$ is never larger than $\max_i x_i^{wc}$ in the worst case static topology.

Time Complexity

- In [3] the term Time Complexity is introduced:
  - The **Time Complexity TC** is the time an algorithm needs to perform, depending on the *number of agents* ***n***.
  - For the **task-switch**, a sensible notion is the time from when the last agent finishes the first task until the last agent starts with the second task.
- Upper bound:
  - An upper bound to the order of the time complexity is given by:

$$TC_{A1} \in O\left(\frac{\ln(1-\delta(n))}{\ln\left(\frac{n-1}{n}\right)}\right)$$

[3] Martinez, Bullo, Cortes, Frazzoli – „On synchronous robotic networks – Part II: Time complexity of rendezvous and deployment algorithms"

Time Complexity

- Proof:
  - At step k_1, let i be the agent such that $x_{min}(k_1) := x_i(k_1) \leq x_j(k_1)$ for all agents j.
  - Then in the next step for agent i:

$$x_i(k_1+1) = \frac{1}{d_i+1}\cdot\left(\sum_{j\in N_i(k_1)} \underbrace{x_j(k_1)}_{\geq x_i(k_T)} + \underbrace{z_i(k_1)}_{=1}\right) \geq \frac{1}{d_i+1}\cdot(d_i\cdot x_i(k_1)+1)$$

  - The smallest possible value for $x_i(k_1+1)$ is achieved by maximizing the number of neighbors.

$$x_i(k_1+1) \geq \frac{1}{n}\cdot((n-1)\cdot x_i(k_1)+1)$$

Time Complexity

- Proof:
  - The value $x_{min}(k_1+1) := x_i(k_1+1)$ provides a lower bound on the consensus values of all agents j in step k_1+1:

$$x_j(k_1+1) \geq x_{\min}(k_1+1) = \frac{1}{n}\cdot((n-1)\cdot x_{\min}(k_1)+1)$$

    - This can easily be seen:
      Suppose there exists $x_l(k_1+1) < x_{min}(k_1+1)$

$$x_l(k_1+1) = \frac{1}{d_l+1}\cdot\left(\sum_{j\in N_l(k_1)} x_j(k_1)+1\right) \geq \frac{1}{d_l+1}\cdot(d_l\cdot x_{\min}(k_1)+1)$$

$$\geq \frac{1}{N}((n-1)\cdot x_{\min}(k_1)+1) = x_{\min}(k_1+1)$$

Time Complexity

- Proof of Time Complexity:
  - This lower bound on the consensus value of all agents can be generally described by:

$$x_{\min}(k+1) = \frac{1}{n}\cdot((n-1)\cdot x_{\min}(k)+1)$$

  - The solution to this difference equation for $k \geq k_1$ is:

$$x_{\min}(k) = 1-\left(\frac{n-1}{n}\right)^{k-k_1}\cdot(1-x_{\min}(k_T))$$

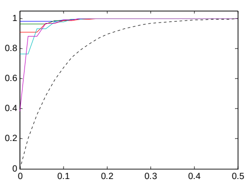
  - With the switching condition $x_i > \delta$ and k_T the step when all agents have switched to the second task it follows:

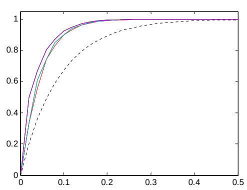
$$TC = k_T - k_1 \geq \frac{\ln(1-\delta(n))}{\ln\left(\frac{n-1}{n}\right)}$$

Time Complexity

- Simulations:
  - Simulation of task-switch for different topologies and numbers of agents in task 1 before switch.
  - Chain topology:
    - Waiting for only one agent:

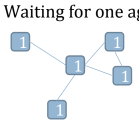
    - Waiting for all agents:



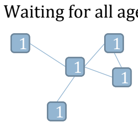


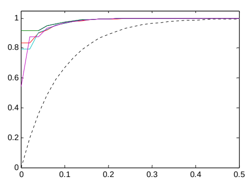
Time Complexity

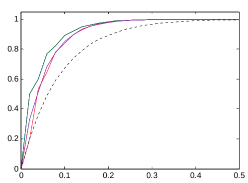
- Simulations:
  - Random topology:
    - Waiting for one agent:



    - Waiting for all agents:



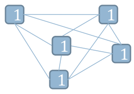


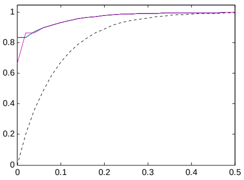


Time Complexity

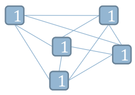
43

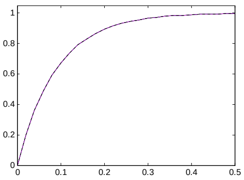
- Simulations:
  - Random topology:
    - Waiting for one agent:





    - Waiting for all agents:





44

Simulation Results: No Exploration

45

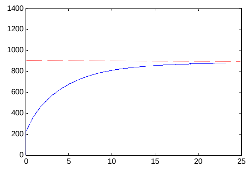
Simulation Results: With Exploration

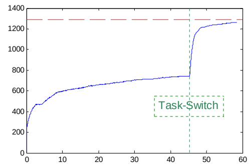
Simulation Results

46

Joint Event Detection Rate

Without Exploration	With Exploration





Summary

Summary

- We discussed problems in searching the shortest communication-path in the **dynamic case**.
- We presented an **algorithm** to compute the shortest path in the dynamic case.
- We introduced an algorithm to **synchronize a task-switch** in a distributed network.
- We discussed the **time-complexity** of the presented algorithm.
- We presented a example simulation to show the **improved performance** of the combined tasks.