Bees, collaborative algorithms & NodeJS

What a combo !

Context

- Reflexions on a biological algorithms and a programming framework,
- Cooperative algorithm.
- Proof of concept
    - coded in 4 full time days !

Biological based models

- Mathematical models that are motivated by observation of living organisms, for examples:
  - Trees,
  - brain,
  - Ants, …

Ants model

- Find the shortest path in complex graph,
- Use local information exchange,
- Pheromone analogy,
- Very powerful in time changing graph.

Bees

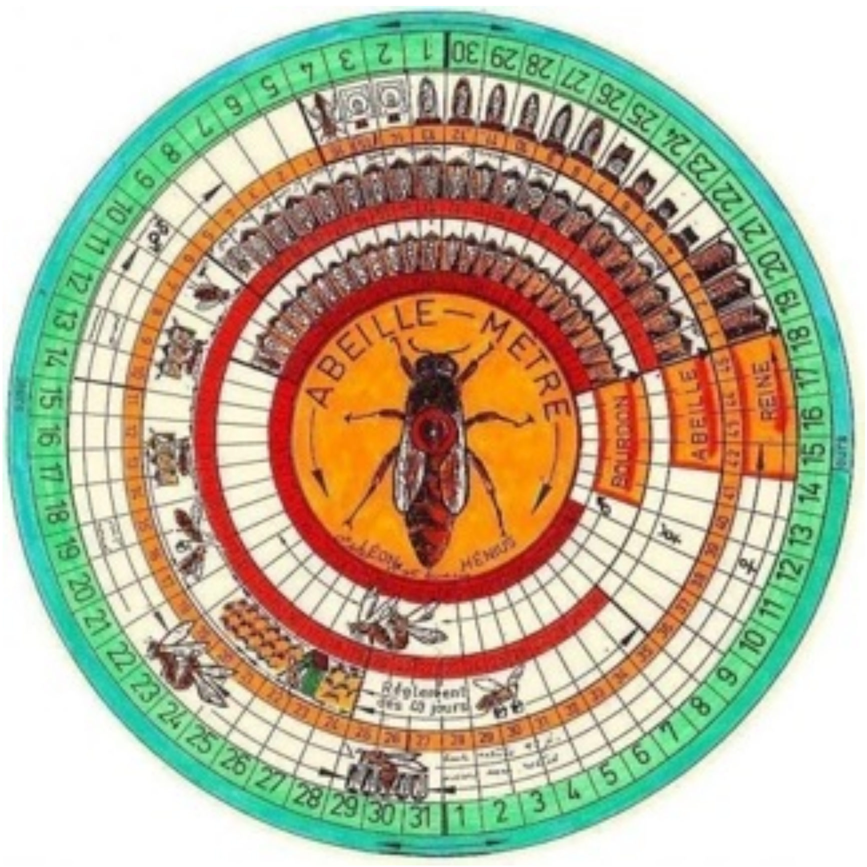
- Bees have a complex social behaviour.
- Karl von Frisch received the Nobel Prize in Physiology or Medicine in 1973, along with Nikolaas Tinbergen and Konrad Lorenz.

Karl von Frisch (1886-1982)

Bees’ way of life

- Each worker bee takes several position inside the colony during its life: cleaner, nurse, wall worker, handler, ventilator and guard.

- The oldest are scouts, juste before this important positions they are pollen collector.



Knowledge sharing

- Each bee gathers the pollen and shares its discoveries with others.
- The description is made through a dance inside the hive by the bee (Karl von Frisch) and presents:
    - the place: coordinates in space, based on the position of the sun and electromagnetic hearth field,
    - the quality of the finding flowers.

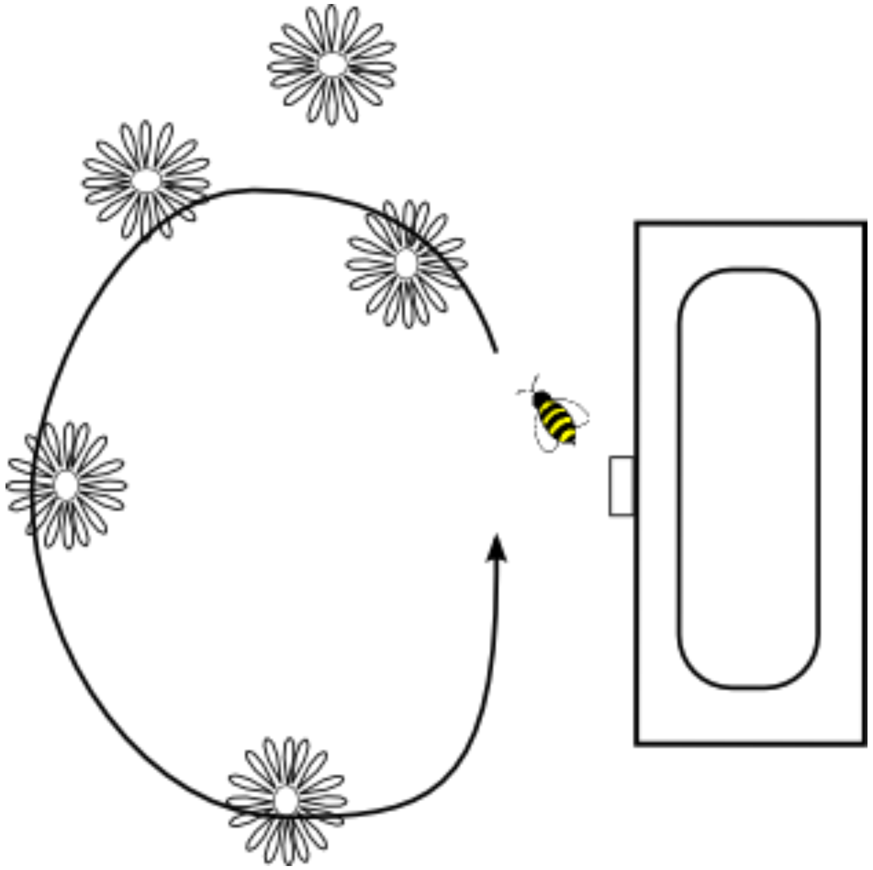
Pollen gathering

- Two steps process

1. pollen finding

2. knowledge sharing

   -

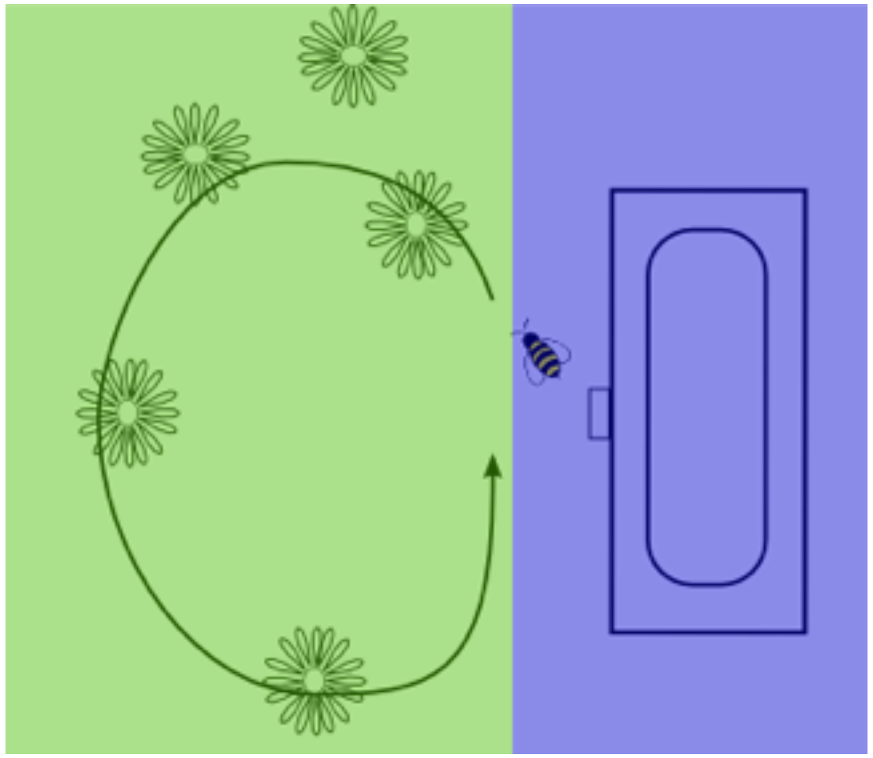


Pollen Gathering

Pollen gathering

- Two steps process

1. pollen finding
   - long process
2. knowledge sharing
   - Each bee present its results to all.
   - short process



General idea:
long and short process

Goal of the bees colony

- The colony must collect the maximal amount of honey during a minimum of time (the winter is coming).
- This process is, after all, a path finding algorithm in a vast and time changing environment (*Artificial Intelligence, A Modern Approach, S. Russel & P. Norvig*).

What about ?

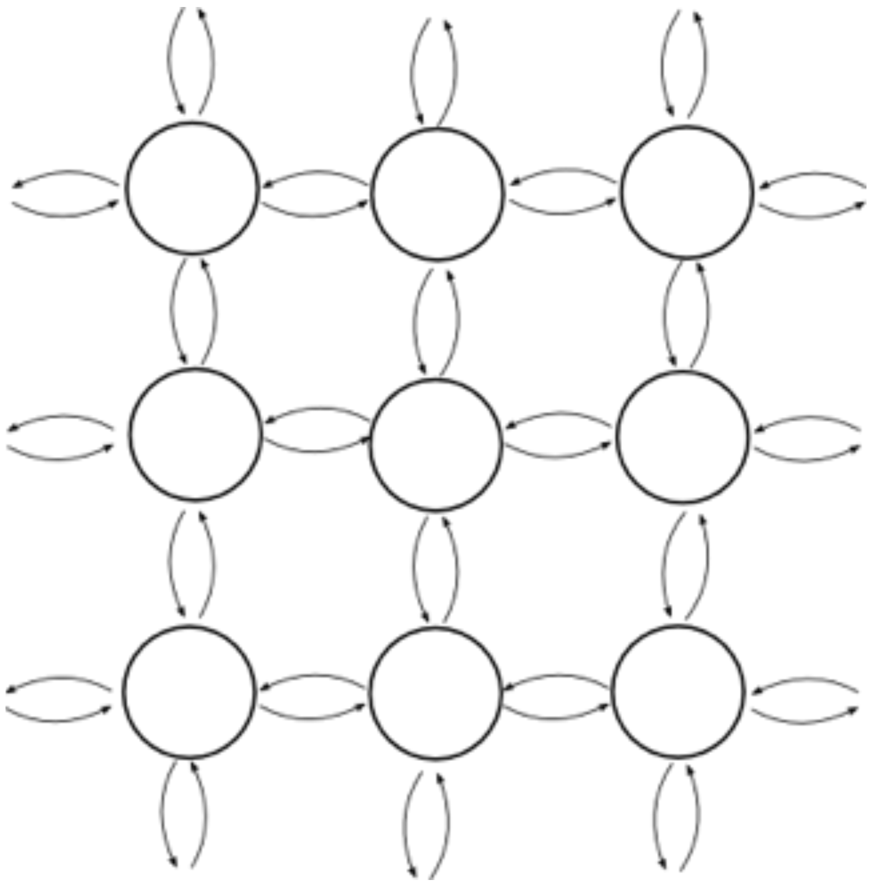
- Doing a simple path finding algorithm based on that idea ?
- Coding it very rapidly ?

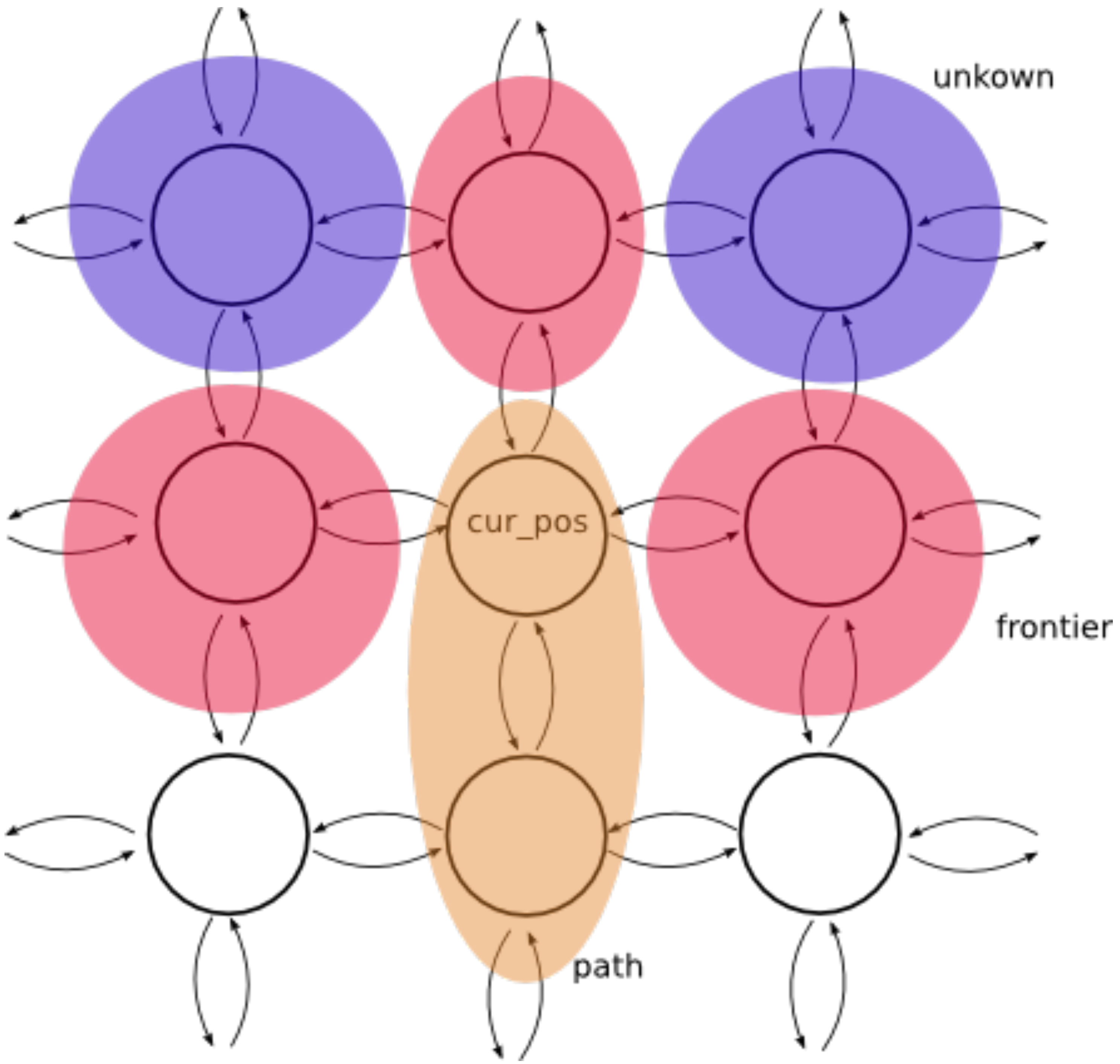
Collaborative Algorithm

- Used it when the optimal solution researching is a very long process.
- We dispose a set of very simple entities that solves part of a complex problem based on some knowledge.
- This knowledge involves in time.
- Goal: converge to a global solution of the problem, *i.e.* after a certain among of time, all entities find the same solution and this solution is the optimal solution of the problem.

Path finding

- Suppose a regular graph (the map),
- Define
  - a neighbouring system (frontier),
  - the cost to move from a position to its neighbouring,
- Find the best limited depth path in graph,
- Goal: visit minimum node as possible before finding the optimal depth limited path.





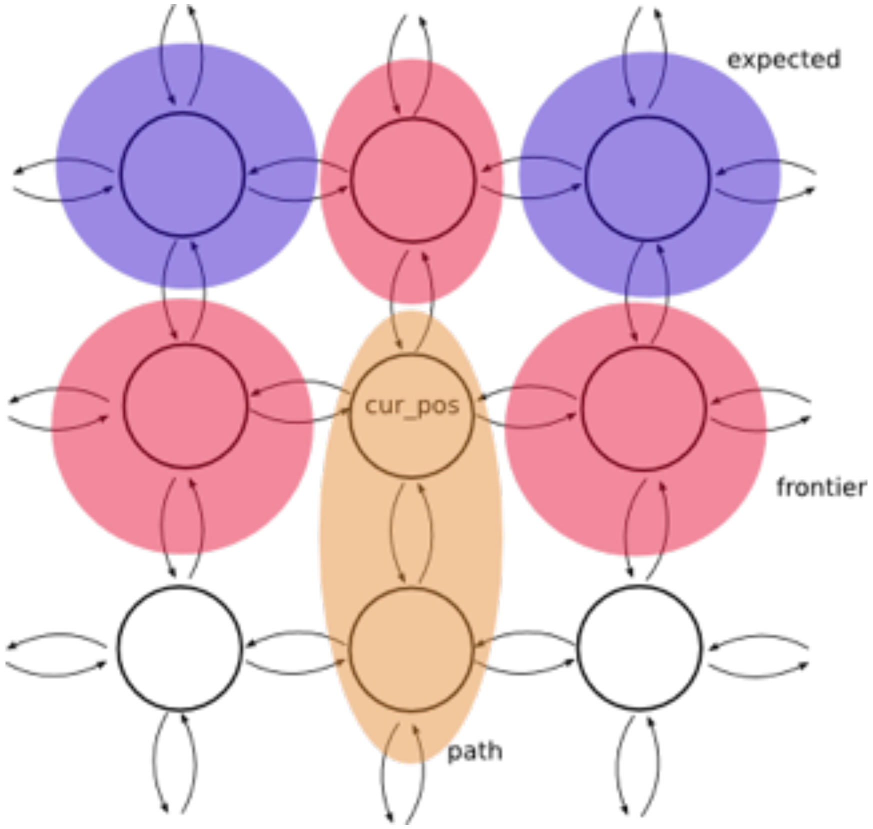
Some Definitions:
The current position, the frontier, unknown and the path

Path finding and prior knowledge

- Define a prior model of the map,
  - exemple: locally smooth.
- Use the prior knowledge to build an expected map.
- Use this expected map to estimate the path.
- While the expected map doesn't fail, move along that path.

Path finding and collaborative algorithm

```
@ init: cur_pos, knowledge, frontier, path, exp_map, est_path
while( length(frontier) != 0 &&  length(path) != MAX_PATH )
    // make a move
    cur_pos = est_path.pop();
    frontier = closet_neig( cur_pos, map );
    knowledge = union( knowledge, cur_pos, frontier );
    // did the expected map fail ?
    if( exist( map(frontier) != exp_map(frontier) ) )
        exp_map = build_expected_map( knowledge );
        est_path = build_estimated_path( exp_map );
    endif
endwhile
```



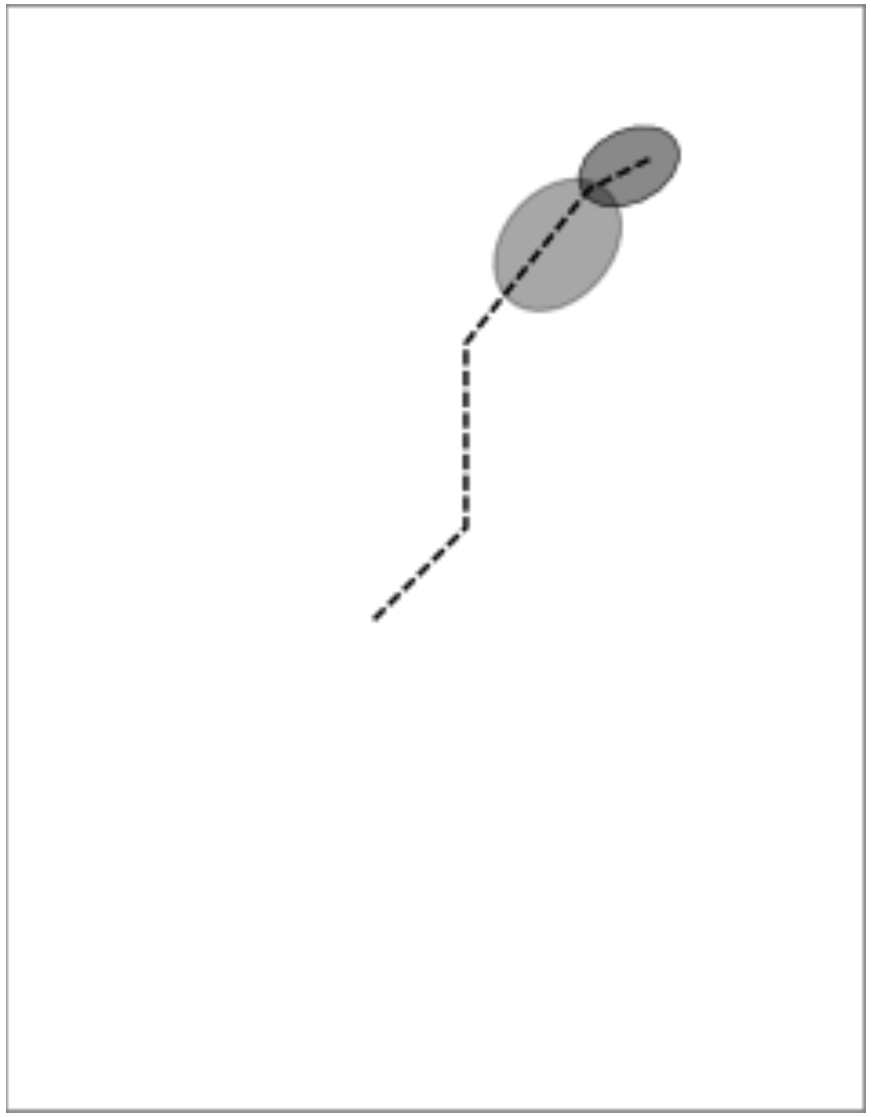
Path finding and collaborative algorithm

```
@ init: cur_pos, knowledge, frontier, path, exp_map, est_path
while( length(frontier) != 0 &&  length(path) != MAX_PATH )
    // make a move
    cur_pos = est_path.pop();
    frontier = closet_neig( cur_pos, map );
    knowledge = union( knowledge, cur_pos, frontier );
    // did the expected map fail ?
    if( exist( map(frontier) != exp_map(frontier) ) )
        exp_map = build_expected_map( knowledge );
        est_path = build_estimated_path( exp_map );
    endif
endwhile
```

- And knowledge ?
    - if knowledge is empty:
        - expected map fails frequently.
        - compute a lot of path.
    - if knowledge is the complete map:
        - build directly the true minimal path.

Build expected map

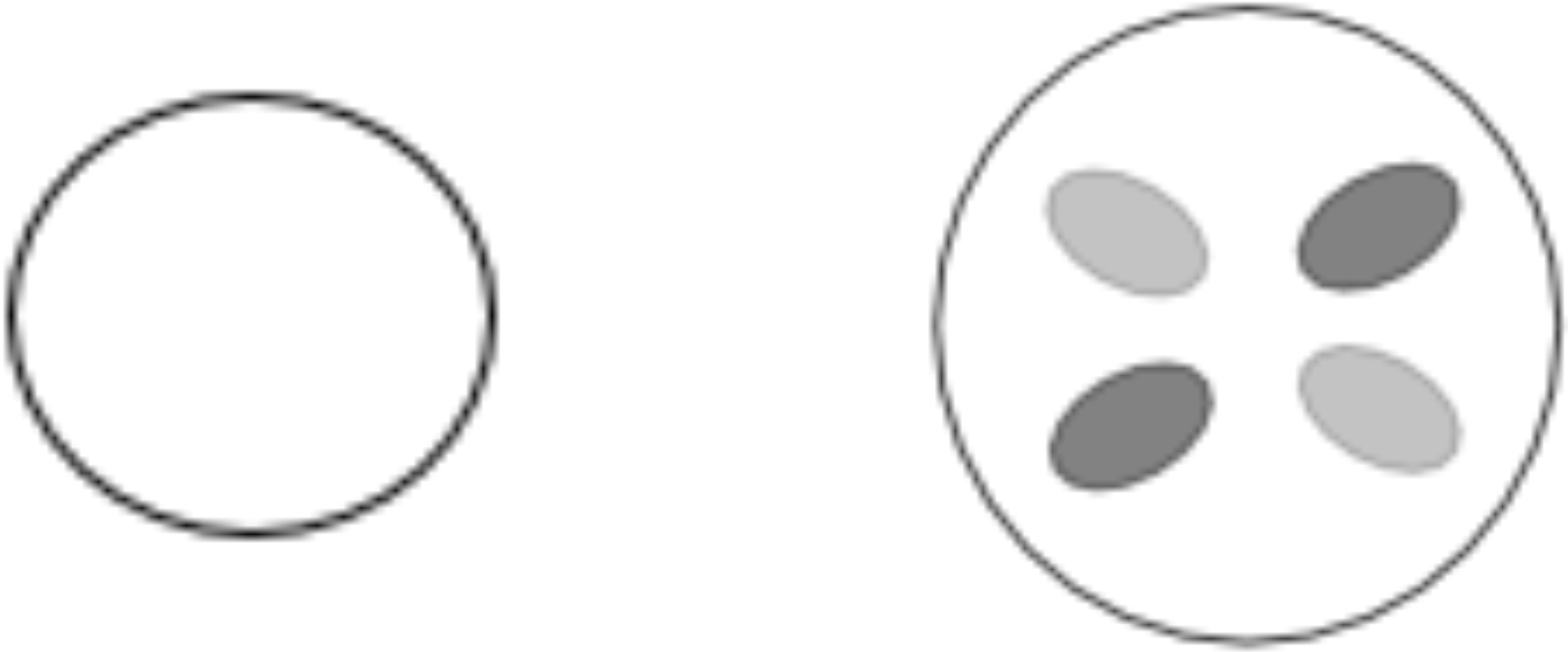
- Prior model : the map is locally smooth.
    - local Gaussian curves with parameters estimated based on knowledge.
- Pessimist agent
    - all positions that can not be estimate by model are assumed to be costy.
        - P(X=0) is high
    - Huge limitation ! (see later)



Path Finding

- Path is defined between the current position and **a goal**.
- We need to define the objective of our path based on the knowledge:
  - Simple agent:
    - pos_max = positions of the first N max(exp_map),
    - goal: max(exp_map(pos_max) / dist(cur_pos,pos_max))
  - Critical point ! (see later)
- Plenty of algorithms may be investigated to actually compute this path: Dijkstra, A*, etc.

Experiments



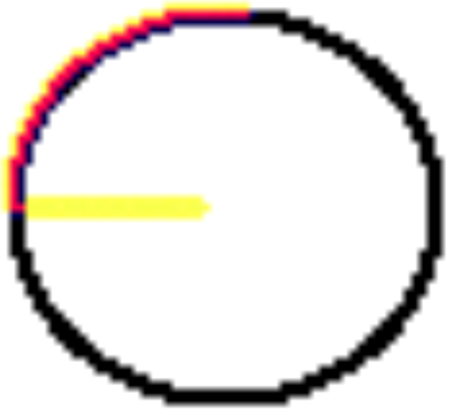
Inputs maps

Initial position is the center of the image

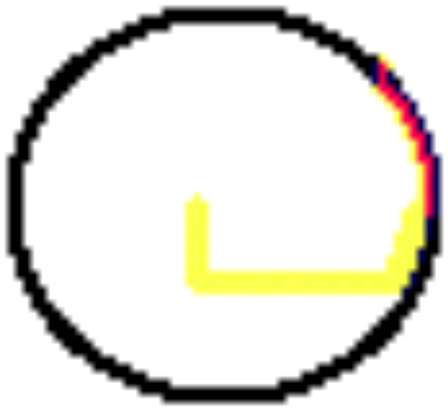
Rich pollen area are black

Initial knowledge is empty.

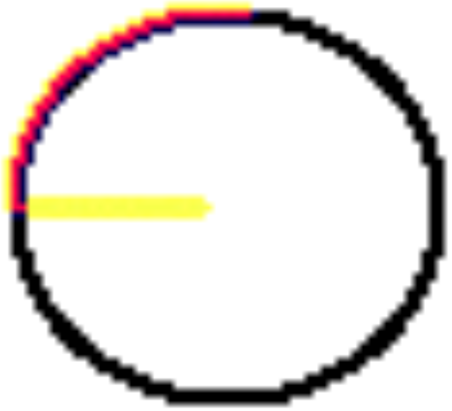
Simple Circle
Without Randomness



entity 1, iter 1



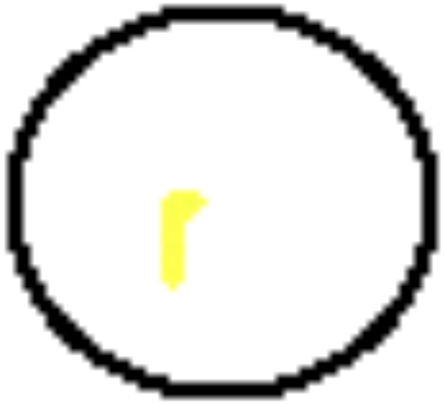
entity 2, iter 1

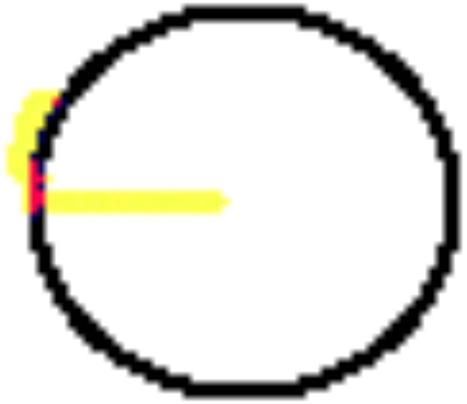


both, iter 20

Simple circle:
Without Randomness

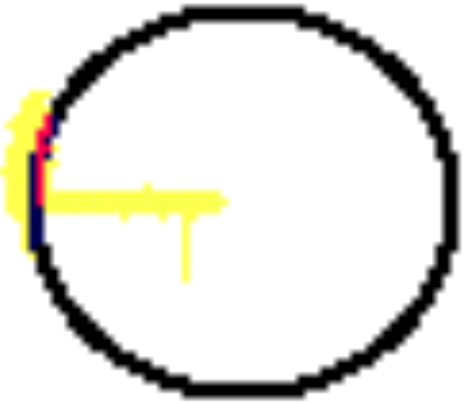
entity 1, iter 1

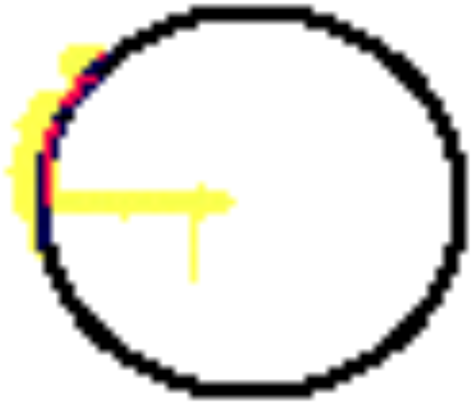




entity 2, iter 1

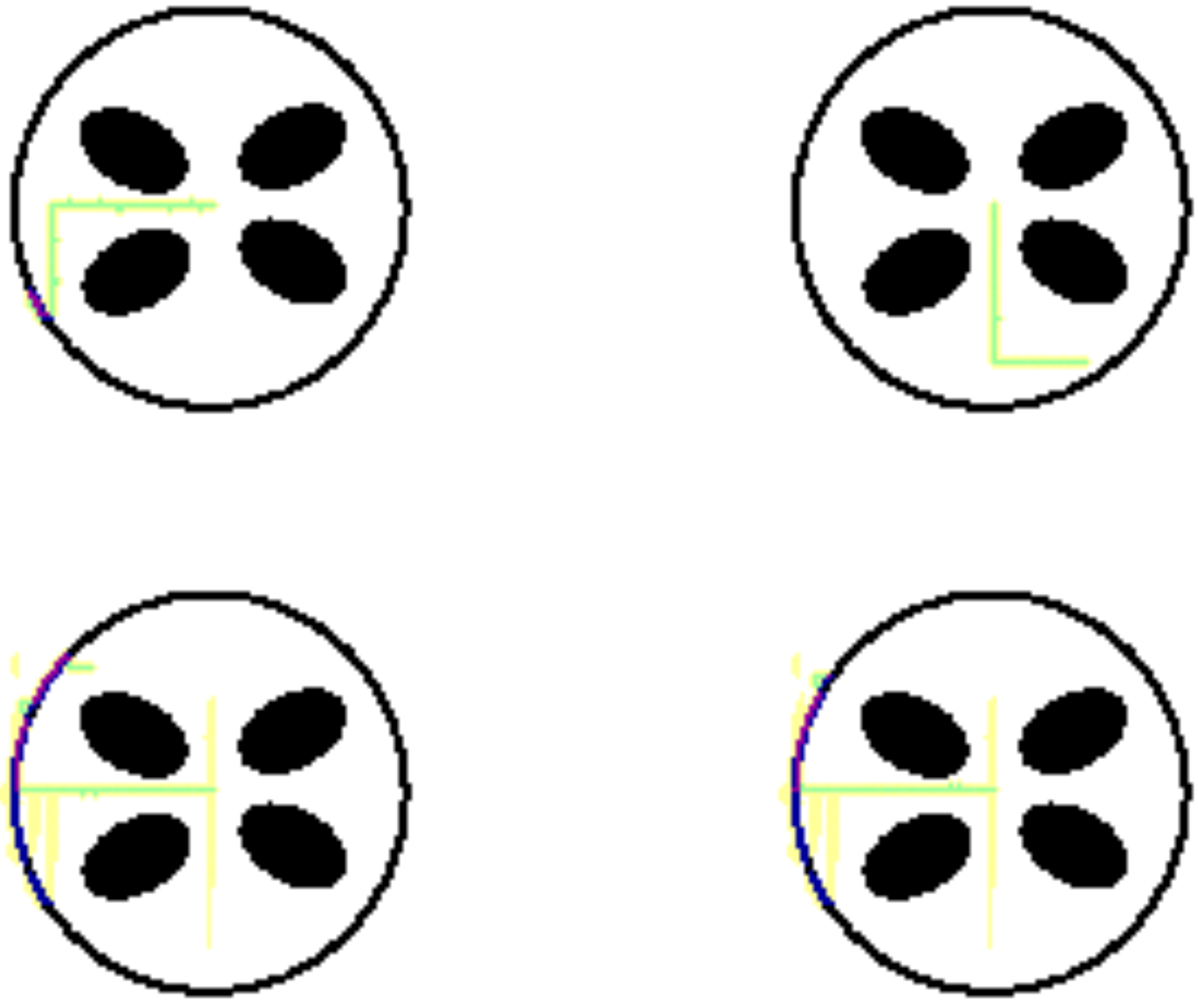
entity 1, iter 10





entity 2, iter 10

Complex circle



entity1, iter 1

entity 2, iter 1

iter 10

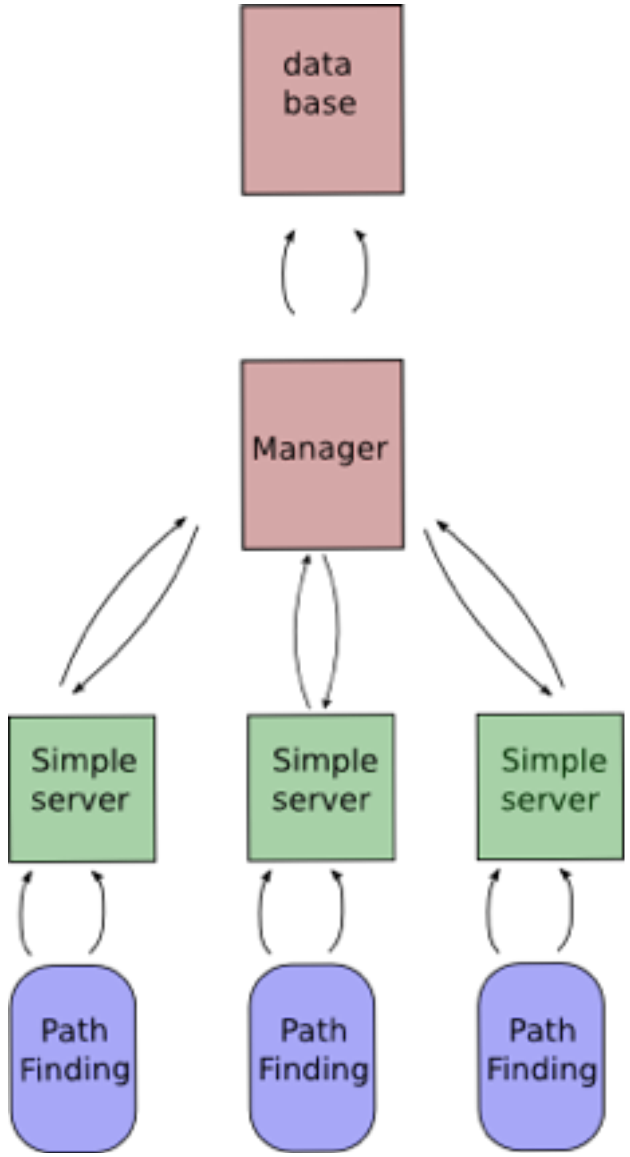
iter 50

Perspectives

- Add some randomness:
  - With a certain probability, explore an unknown position in close neighbourhood,
  - With a certain probability, define the goal as unknown position.

- Define a different goal:
  - explore the map
  - 2 steps goals:
    - Exploration
    - Gathering

Few words about the full open source implementation



- World manager
  - NodeJS server: 200 lines script
- Data base
  - CouchDB: 0 lines of codes
- Server entities
  - NodeJS server: 50 lines script
- Path finding
  - Octave code: 300 lines script

Conclusion

- An interesting way to concretely enter in the Artificial Intelligence field.
- Highlight the most critical points:
    - What is Prior knowledge of unknown ?
        - What is the ration between exploring the map and gather pollen ?
    - What is the goal of an agent ?
        - minimal path with the maximal benefit.
- Very easy to code!