INTELLIGENT TILES: PUTTING SITUATED MULTI-AGENTS MODELS IN REAL WORLD

Nicolas Pépin, Olivier Simonin, François Charpillet
Project-Team INRIA-MAIA, LORIA, 615 rue du Jardin Botanique, 54600 Villers-lès-Nancy, France
Nicolas.Pepin@inria.fr, Olivier.Simonin@loria.fr, Francois.Charpillet@loria.fr

Keywords: Situated multi-agent models, Intelligent tiles, Perception, Communication, Mobile robots.

Abstract: In this paper we propose to pave indoor floors with “communicating” tiles in order to extend perception and communication of mobile agents and more generally to implement environment-based multi-agent models. Each tile supports a real-time process which ensures communication with its neighbours and any agent laid on it. We details algorithms required for tiles to interact with mobile agents and to carry out distributed processes. Then we apply our approach to a behavior-based model, by splitting the model into the tiles and a simple agent. We show this new version is equivalent to the original one and so discuss its advantages.

1 INTRODUCTION

Future factories, homes and public places will be inhabited by people and autonomous mobile robots working or helping them, see e.g. (Guizzo, 2008). This population of independent individuals will need to efficiently navigate (avoid collisions, block) and communicate with distant systems or persons. Today such facilities could be achieved using wireless communications and vision-centred navigation. However, ensuring a lot of wireless communications without interferences or faults remains hard (Zhou et al., 2004). Robots' navigation is also limited when using camera vision and global positioning systems, due to the presence of obstacles, walls and individuals. Moreover, global positioning is difficult to perform inside buildings, even when it is connected to wireless networks. In this paper we tackle such challenges by considering another way for communication and navigation of agents. As it is currently used by living systems, we suggest to exploit the physical environment as a medium for communication and cooperation between agents (Beckers et al., 1994). This approach is well known in social insects self-organisation, where e.g. the environment is marked with pheromones (Parunak, 1997).

In order to extend agent abilities and to implement environment-based algorithms, we propose to pave indoor floors with communicating tiles. Each tile is defined to ensure communication with its adjacent neighbours, and to store simple information that can be read/write by an agent laid on the tile. As a consequence tiles can be exploited to extend agents' perceptions and communications (of human or robot), and to simplify agent model by computing parts of the algorithms. Finally, this distributed support aims at physically implement bio-inspired algorithms.

The paper is organised as follows. Section 2 discusses the requirements for an intelligent floor, and the consequences on tile definition. Then in Section 3 we define a generic tile model. In Section 4 we show how a model dedicated to multi-robot navigation, the Satisfaction-Altruism model (Simonin and Ferber, 2000), can be partially translated to the tiles. Finally Section 5 concludes the paper.

2 INTELLIGENT ENVIRONMENTS FOR SITUATED AGENTS

Transmitting information through the environment is a common approach in reactive multi-agents systems. We can quote digital pheromone techniques (Parunak, 1997), potential field computation and cellular automata based environment. In some deliberative models, a discrete environment representation may be also used, e.g. for path planning or for Markov Decision Process models. In both reactive and deliberative approaches, implementing such algorithms in the real world remains a challenge. In the first case, robots/agents cannot directly read/write information on the floor. In the second case, deliberative agents have to build a map of the environment, locate themselves accurately and share their map with others agents.

So, augmenting the environment with a concrete grid can provide a flexible and smart way to deploy or extend these algorithms. We now examine which kind of technology/network could be envisaged.

Computing and communicating through a grid has become a common idea, but such projects as the GRID one (Foster et al., 2001) consider large networks of computers. The scale of such networks is not adapted to our problem, in particular we just need to connect simple mote-like nodes (Polastre et al., 2004). In the domain of surveillance, we assist to the development of sensor networks. They consist in a myriad of simple sensors or nodes deployed in the environment. Such an approach seems more adapted to define a fine-grained network for providing communication and perceptions to agents. For instance (Mamei and Zambonelli, 2007) proposed to use sensor network and RF-ID based infrastructures to perform physical object tracking. We can note this approach relies on wireless technology, having drawbacks pointed in the introduction. Moreover nodes remain fragile to the traffic of the robots and people.

This analysis led us to imagine a particular network adapted to indoor environments. It consists in physically connecting a set of mote-like nodes, which have to be positioned regularly as a grid and integrated inside the floor. In practice, we envisage to design these nodes as tiles to pave the floor, that naturally ensures a regular organisation. This should ease the interaction between agents and the nodes, which can be physical (sensors, contact) or wireless (radio, light, etc.).

As in the majority of discrete models, we consider that the tiles are identical with a squared topology. Their size must be adapted to support just one person or one robot at once. Figure 1a simply illustrates such a paved environment.

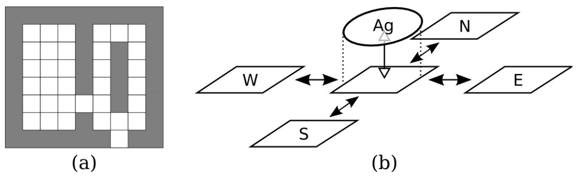


Figure 1: (a) Example of a tiled environment. (b) Representation of communications between tiles and agent.

3 DEFINITION OF THE TILES

3.1 General Features

We define a tile as an autonomous, reactive and communicating entity. It may exchange messages with a carried agent, and with the neighbouring connected tiles (Figure 1b). In the model's adaptation we have in sight, we choose to limit the neighbourhood to 4 tiles.

Features of a Tile:

- a (limited) memory to store information,
- communication links with the neighbouring tiles,
- ability to support one and only one agent and to communicate with it (wireless or contact link),
- a main process answering to requests from the agent and the neighbouring tiles,
- an internal clock allowing to date operations,
- optionally other processes which can send messages to the main process.

Hypotheses on Tiles:

- tiles are independent processes (no scheduling is supposed),
- tiles processes do not require to be synchronised,
- tile's main process routines do not use blocking operations.

We now details in the following subsections algorithms and structures needed to enable in a tile the features just above-mentioned. In particular we define tile to tile communication, interaction between mobile agents and tiles, and mechanisms allowing tiles to perceive and spread information.

3.2 Tile's Process, Data and Communication

Tile main process. Our approach do not define tiles as agents because they have no proper objective (they are not proactive). Each tile is reactive and based on a main process which consists in an infinite loop treating incoming messages. The main process' form is as follow:

```
Algorithm 1: General tile definition
while true
  if request R in queue then
    switch descriptor of R
    case descriptor_1:
      instructions
    case descriptor_2:
      // Example:
      for i in {N,S,E,W}
        send message to Tile(i)
    ...
    end switch
  end if
end while
```

The tiles are designed to store received messages in a FIFO queue and treat them as they come. The messages' treatment is organised as a set of different cases, and each type of message is treated according to a message descriptor (see *Message formalism* below). These treatments must not contain any blocking operation, as we want the tiles to be able to answer agent's requests in real time. We envisage optional processes, that can communicate with the main one, in order to add pro-activity to the tiles. These processes can manage time based or blocking operations without interfering with the main process. For instance, such a process could allow a pheromone diffusion at a fixed frequency.

Stored data. To limit energy consumption we reduce stored data as much as possible. In particular, information about other tiles is limited to one small set of data per adjacent direction. In the Section 4 application, each set contains a description of the tile's direct neighbourhood.

Connexions between tiles. We consider that neighbouring tiles are only known and addressed through their relative direction to the current tile: $\{N,S,E,W\} = \mathcal{C}$ (in a 4-connexity model).

Information on the neighbourhood is stored in a tile as a 4 elements vector named `D`. We note `D(dir)` the stored information in the `dir` direction.

Message formalism. We define all messages as $descriptor(dir, arg2, \ldots)$. *descriptor* reports the nature of the transmitted information, in order to identify the answer to carry out. When `dir` is precised, it

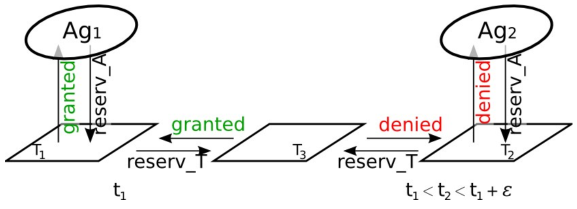


Figure 2: Resolution of the concurrent access problem.

represents the direction the message describes, i.e. an information concerning an adjacent tile (`dir` $\in \mathcal{C}$). Other values are the information itself. We consider the direction and the information values as optional, some messages only needing a descriptor to inform the receptor.

In the rest of the article we add to the descriptors' names _A or _T to clearly differentiate messages between a mobile agent and a tile and messages between two tiles. The following example shows an answer to an agent's perception request (from section 4): *description_A(N,agent), description_A(S,nothing), description_A(E,empty Tile), description_A(W,agent), max signal_A(N, -59).*

3.3 Interactions with a mobile agent

Managing concurrent access. From the tiles point of view, robots displacements are always discrete, whether agents have discrete moves or not. To ensure that two agents don't move onto the same tile at the same time, it is necessary to set up a local reservation mechanism to grant a tile access to only one agent. Moreover, such mechanism must be simple and must not require a "global" supervisory control.

Before moving to a target tile the agent ask to the one supporting it the permission to move. This tile then transfers the request to the target tile. If the latter is free of agent and of other running reservation request, it grants the access, saves reservation time and then does not accept anymore reservation until the agent arrives (and notice it) or a timeout on the reservation expires.

Figure 2 illustrates the *Concurrent access* algorithm presented below. Both agents Ag_1 and Ag_2 want to go on tile T_3 at the same time t. They both send a reservation request (`reserv_T`), through their own tile to T_3. As T_3 is empty, one request can be accepted. As unprocessed messages are stored in a single queue, one of the two reservation is processed first and the request is accepted, while the second is discarded.
When a request is accepted (`granted`), the tile stores a time stamp (`rsv_stamp`) corresponding to the ac-

ceptation time. If the agent does not arrive, the next reservation will be possible after the timeout delay.

```
Algorithm 2: Concurrent access
case reserv_A(dir):
  send reserv_T to Tile(dir)
case reserv_T(dir):
  if tile is occupied
     or (time - rsv_stamp < timeout)
  then
    send access_T("denied") to Tile(dir)
  else
    send access_T("granted") to Tile(dir)
    rsv_stamp <- time
  end if
case access_T(message,dir):
  send access_A(message) to the Agent
```

In algorithms, dir always represents a direction from the current tile's point of view.

As we consider that this process occurs each time a displacement is performed, it will not be presented again in the following algorithms.

Updating Tile Information. From the tiles' point of view, it is necessary they know whether a mobile agent arrives or leaves it. We employ "active" identification through agent's messages. Thus we define two specific message's descriptors: arriving_A and leaving_A. Their implementation is detailed in the next subsection. Another practical solution could be tiles that detect automatically movements using sensors. The former solution is more reliable because it makes sure that nothing but agents can be detected.

Agent's local perception. When an agent needs to know the state of the neighbouring tiles it sends a perception_A message to its supporting tile. The tile answers with a series of messages description_A describing the tiles in each direction dir. Tile's algorithm 1 is completed with:

```
case perception_A:
  for dir in {N,S,E,W}
    if sense(dir,isTile)
       and D(dir) = "no tile"
      D(dir) <- "tile"
    else if sense(dir,isNotTile)
      D(dir) <- "no tile"
    end if
    send description_A(dir,D(dir)) to agent
  end for
```

The sense function tests the existence of a tile in a given direction by checking the physical connexion. It is used to update D, the representation of the local neighbourhood.

3.4 Tile's Perception

Collecting vs. diffusing. There are two ways for the tile to perform a perception process.

1. If we stay centred on the agent, the perception process is started when the agent requests it. Then the tile asks its neighbours information using a bidirectional communication process.

2. We can also imagine that the process of information could be done when an event occurs, e.g. if an agent arrives on a tile, the latter notify its neighbours. We refer to this process as *diffusion*, because it doesn't require an answer from the informed tile.

We adopt this second solution as it is not centred on the requesting agent. Moreover this one has just to read *D* when needing the information.

The mechanism consists, first, to diffuse the events arriving_A and leaving_A to neighbouring tiles with the descriptive messages description_T:

```
case arriving_A:
  for dir in {N,S,E,W}
    send description_T("agent") to Tile(dir)
  end for
case leaving_A:
  for dir in {N,S,E,W}
    send description_T("empty") to Tile(dir)
  end for
```

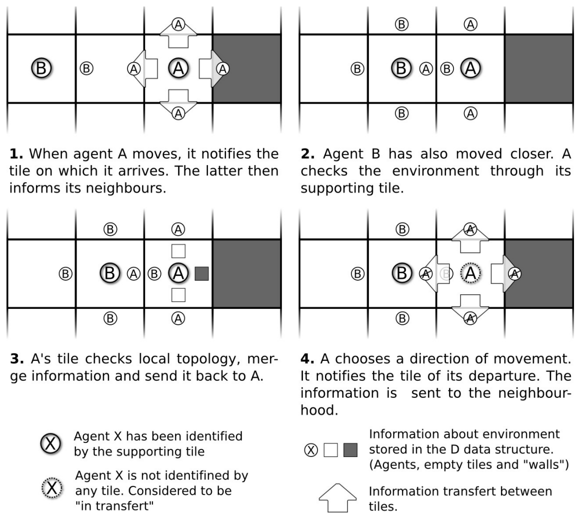
Second, the neighbouring tiles store the transmitted information in their local representation:

```
case description_T(dir,info):
  D(dir) <- info
```

Example. Figure 3 illustrates a diffusion process, where an agent (A) moves onto a tile, has its presence diffused to the neighbourhood, requests a perception and then moves again. The diffusion process occurs on steps 1 and 4 when the agent notifies a change in its state to the tile, e.g. when it arrives or leaves it (the large arrows represent the description_T messages)

It is necessary to differentiate the diffusion process (steps 1 & 4) from the neighbourhood sensing (step 3) that identifies changes in the local environment. The latter can be achieved by testing the physical link between two tiles.

Note that all communication processes cannot be implemented following the diffusion approach. It is the case for the concurrent access mechanism proposed in Section 3.3.


1. When agent A moves, it notifies the tile on which it arrives. The latter then informs its neighbours.

2. Agent B has also moved closer. A checks the environment through its supporting tile.

3. A's tile checks local topology, merge information and send it back to A.

4. A chooses a direction of movement. It notifies the tile of its departure. The information is sent to the neighbourhood.

Ⓧ Agent X has been identified by the supporting tile

Ⓧ Agent X is not identified by any tile. Considered to be "in transfert"

Ⓧ ☐ ■ Information about environment stored in the D data structure. (Agents, empty tiles and "walls")

⇧ Information transfert between tiles.

Figure 3: Diffusion mechanism of information using communications between tiles

3.5 Spreading Information

The diffusion model is natural to use when tiles have to spread information in the environment. Indeed, a point-to-point communication would be less robust to possible failures.

Reactive models make often use of a diffusion mechanism, e.g. numerical gradient, signals, etc. The tile model we propose allows this feature.

The principle of the diffusion process is defined in the following algorithm:

```
case descriptor_T(dir, nb_hops):
  (...)
  if nb_hops > 0 then
    for d in {N,S,E,W} - {dir}
      send descriptor_T(nb_hops -1) to d
    end for
  end if
```

When a tile receives a descriptive message from a neighbouring tile, it has the possibility to diffuse it to its other neighbours. `nb_hops` is initialised to a value corresponding to the desired diffusion radius when the propagation starts.

Such a mechanism can however be extended, if we need the message to be communicated only once to each tile. In this case, the message must be identified uniquely.

4 APPLICATION TO THE SATISFACTION-ALTRUISM MODEL

The *Satisfaction-Altruism* model (Simonin and Ferber, 2000), noted Sat-Alt, is designed for cooperation between situated agents. In this paper, we focus on spatial conflicts resolution in unknown and strained environments.

4.1 The Sat-Alt model

We present the Sat-Alt model for grid environments, where agents move, communicate and perceive on their four neighbouring cells.

To achieve its current task, each agent can switch among two states: the normal state and the altruist one. In the former, it follows its own goals while in the latter it tries to satisfy another agent. To switch between states, agents are able to compute a level of satisfaction $P(t)$. This value is peculiar to each agent and vary depending on the level of constraint sensed by the agent:

- $P(t+1) = P(t) + v_p$
- $v_P = \begin{cases} -(coef_{wall}.nb\_walls + coef_{ag}.nb\_agents) \\ \quad \text{if unable to move} \\ cste > 0 \ \text{otherwise} \end{cases}$

If the agent is blocked in its progression, the level of satisfaction decreases according to the number of walls and other agents perceived in the local area. On the contrary, if it can move, this level increases at a constant rate. The overall level is limited in $]-P_{max}, P_{max}[$ and when $P(t)$ is negative, the agent is considered unsatisfied.

Agents can emit their satisfaction values as signals noted I. In reception they can select the one with the lowest value, noted I_{ext}. Each time an agent updates its perceptions it performs the test of altruism $I_{ext} < (0, P, I)$ which compares the sensed signal carrying the lowest satisfaction to its own satisfaction and its own signal. If it is lower than all of them, the agent switches to the altruist state. In this state, it tries to flee the agent emitting the most unsatisfied signal, i.e. to move to any free area.

The principle used to solve a conflict consists in propagating dissatisfaction signals (negative ones) from the most dissatisfied agents to their neighbours. This principle is defined in the following algorithm.

```
Algorithm 3: Sat-Alt Agent
perception of neighbourhood
perception of obstructing agents
I_ext <- max(ext signals)
P <- P + vp
```

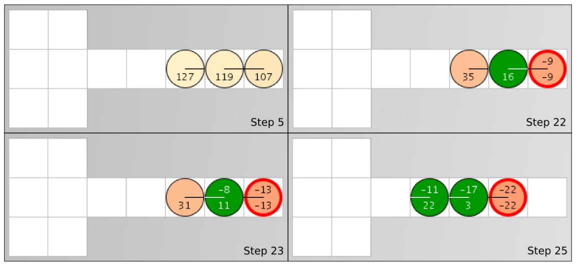


Figure 4: Illustration of the dead-end problem's resolution by the *Satisfaction-Altruism* model. The bottom value is the personal satisfaction, the top one an emitted signal and the line shows the current direction. Agents in the altruist state are drawn in dark-grey (green) colour. Otherwise, the higher is personal satisfaction, the lighter is the colour.

```
if I_ext < (0,I,P) then
  altruist <- true
  compute fleeing_dir
else compute goal direction
if not altruist and P < 0 then I <- P
else if altruist and I_ext<I then I<-I_ext
else if no obstructing agents then I <- 0
emits I
if agent can move then move()
```

To always emit and relay the highest dissatisfaction, signal values are updated as soon as $P(t)$ or I_{ext} go down. This allows to unlock situations where several blocked agents emit their dissatisfaction. Finally, an agent continues to emit a signal while it perceives an obstructing agent.

Figure 4 illustrates this principle on a corridor exploration. Initially at step 1, three agents enter in a narrow corridor. Their personal satisfaction is maximal (127). Blocked by the dead-end (step 5), the first arrived agent tries to escape. Then its direction is opposed to the two others. All agents satisfaction has fallen at step 22 and first agent starts to emit a dissatisfaction signal. Then its neighbour switches to the altruist state. At the next step the latter agent has changed its direction and start to emit the last dissatisfaction he has received. As a consequence the third agent also switches to the altruist state. Then all agents are oriented in the same direction and can escape the dead-end.

4.2 Splitting Sat-Alt into Agents and Tiles

What is changing when splitting the *Satisfaction-Altruism* model into agents and tiles? The core of the model – computation of the satisfaction, decision taking and movement – is still under control of the mobile agents. But effective perception and emission modules, and the responsibility to diffuse signals and collect data is now supported by the tiles.

Sat-Alt Tile model. Tiles have to carry out perceptions and communications required by the model.

Concerning communications, we extended the `arriving_A` message for the tile to be informed also of the agent signal, and renamed the descriptor as `state_A(I)`. To reach other agents, tiles diffuse `description_T("agent",I)` messages to their neighbours. Each tile stores presence information in `D` plus signals information about their neighbourhood in a similar vector `S`.

Tile's perception is now in charge of the computation of I_{ext} value, which is communicated with a `maxsignal_A(dir_ext,I_ext)` message.

```
Algorithm 4: Sat-Alt Tiles
case state_A(I):
  for dir in {N,S,E,W}
   send description_T("agent",I) to Tile(dir)
case leaving_A:
  (...)
case description_T(dir,type,signal):
  D(dir) <- type
  if type = "agent" then S(dir) <- signal
  else S(dir) <- null
case perception_A:
  (...) cf. Section 3.3
  (I_ext,dir_ext) <- max(S)
  send maxsignal_A(dir_ext,I_ext) to agent
  send endoftransmission to agent
```

New agent model. From the previous tiles definition, we can define the new Sat-Alt agent algorithm. One can see that perceptions and emissions of signals are now implemented using only a few instructions.

```
Algorithm 5: New Sat-Alt Agent
send perception_A to Tile
waiting for {
    description_A(dir1,desc1), ...,
    description_A(dir4,desc4),
    maxsignal_A(dir,I_ext),
    endoftransmission
  }
P <- P + vp
if I_ext < (0,I,P) then (...)
if agent can move then
  send leaving_A to Tile
  move()
end if
send state_A(I) to Tile
```

4.3 Simulation and Validation

In order to compare the original Sat-Alt model and the new one using tiles, we have developed simulators based on the Turtlekit framework[1] (Michel et al., 2005), which is dedicated to reactive multi-agent systems simulation.

Tiles and mobile agents have been implemented in independent processes. As tiles implement only not blocking short processes they answer in real-time compared to the mobile agent actions. The average frequency of the Turtlekit scheduler is very low compared to the tile's one.

Our objective was to validate the rewriting of the original Sat-Alt in the tiles-based model. Our approach consists in comparing executions with both models starting from the same initial state (same agents and environment). To allow the comparison, we biased random computation by always using the same seed as the Turtlekit agents scheduling is fixed. Indeed, some displacements may use random choices, for instance when two fleeing directions are equivalent. Then we checked that agents have the same evolution in both simulations using the different models.

We performed simulations with 11 agents exploring a 9x9 map composed of rooms and corridors. We compared logs of both models execution, i.e. comparing agents position, orientation, satisfaction value and signal value, to detect differences. Up to thousand steps, we did not see any difference between the two models. This result was obtained by setting the TurtleKit time step at $0.5s$.

This first empirical validation showed the equivalence of both models. However the tiles based model introduces interesting properties as it separates agents' decision from perception and communication mechanisms. It first allows to have a high frequency to perform perceptions and distributed processes through the environment. It also provides to agents new way of communication, as the diffusion approach used in our case study.

5 CONCLUSION

In order to extend robots' perception and communication we proposed to pave indoor floors with communicating tiles, each one being able to communicate only with its neighbouring tiles and a mobile agent. We shown that diffusing and collecting information can then be managed by the tiles through local and recursive mechanisms. We defined each tile as a real-time autonomous process and as simple as possible to limit time computation and energy consumption. We illustrated the interest of the approach by splitting the Satisfaction-Altruism model into tiles and a simple agent behaviour. Experiments shown the equivalence of both models, where tiles have the advantage to manage communication and perception independently of the agent activity.

Concerning future work, we started to study the electronic implementation of the tiles, by considering Mote technology, to carry out some experiments. We also plan to continue the study of the model, by evaluating algorithm complexity and robustness.

[1] http://www.madkit.net

REFERENCES

Beckers, R., Holland, O., and Deneubourg, J.-L. (1994). From local actions to global tasks: stigmergy and collective robotics. In *Artificial Life IV: Proc. of the 4th Int. Workshop on the synthesis and the simulation of living systems, third edition, MIT Press*.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the Grid: enabling scalable virtual organizations. *Supercomputer Applications*, 15(3):200–222.

Guizzo, E. (2008). Three Engineers, Hundreds of Robots, One Warehouse. In *IEEE Spectrum online, http://www.spectrum.ieee.org/jul08/6380*.

Mamei, M. and Zambonelli, F. (2007). Pervasive pheromone-based interaction with RFID tags. *ACM Trans. Auton. Adapt. Syst.*, 2(2):4.

Michel, F., Beurier, G., and Ferber, J. (2005). The TurtleKit Simulation Platform: Application to Complex Systems. In *Proceedings of the Workshops Sessions at the 1st International Conference on Signal & Image Technology and Internet-Based Systems (IEEE SITIS05)*, pages 122–128.

Parunak, H. V. D. (1997). Go to the Ant: Engineering Principles from Natural Agent Systems. *Annals of Operations Research*.

Polastre, J., Szewczyk, R., Sharp, C., and Culler, D. (2004). The Mote Revolution: Low Power Wireless Sensor Network Devices. In *Hot Chips 2004*.

Simonin, O. and Ferber, J. (2000). Modeling self satisfaction and altruism to handle action selection and reactive cooperation. In *In The Sixth International Conference on the Simulation of Adaptative Behavior*.

Zhou, G., He, T., Krishnamurthy, S., and Stankovic, J. (2004). Impact of Radio Irregularity on Wireless Sensor Networks. In *The 2nd Int. conf. on Mobile Systems, Applications, and Services MobiSYS' 04*, pages 125–138.