Adaptive Tactical Decisions in Pedestrian Simulation: A Hybrid Agent Approach

Luca Crociani, Andrea Piazzoni, Giuseppe Vizzari(✉), and Stefania Bandini

CSAI - Complex Systems & Artificial Intelligence Research Center, University of Milano-Bicocca, Viale Sarca 336/14, 20126 Milano, Italy
{luca.crociani,giuseppe.vizzari,stefania.bandini}@disco.unimib.it, andrea.piazzoni@campus.unimib.it

Abstract. Tactical level decisions in pedestrian simulation are related to the choice of a route to follow in an environment comprising several rooms connected by gateways. Agents are supposed to be aware of the environmental structure, but they should also be aware of the level of congestion, at least for the gateways that are immediately in sight. This paper presents the tactical level component of a hybrid agent architecture in which these decisions are enacted at the operational level by mean of a floor-field based model, in a discrete simulation approach. The described model allows the agent taking decisions based on a static a-priori knowledge of the environment and dynamic perceivable information on the current level of crowdedness of visible path alternatives.

Keywords: Pedestrian simulation · Tactical level · Hybrid agents

1 Introduction

Simulation is one of the most successful areas of application of agent-based approaches: models and techniques employed by researchers in different disciplines are not necessarily in line with the most current results in the computer science and engineering (see, e.g., [2]), and yet the area still presents interesting opportunities for agent research and computer science in general. Pedestrians and crowds simulation is an example of this situation: both the automated analysis and the synthesis of pedestrian and crowd behavior, as well as attempts to integrate these complementary and activities [13], present open challenges and potential developments in a smart environment perspective [11].

Modeling human decision making activities and actions is an extremely challenging goal, even if we only consider choices about walking behavior: different types of decisions are taken at different levels of abstraction: [12][1] provides a well-known scheme to model the pedestrian dynamics, describing 3 levels of behavior: (i) **Strategic level**, managing abstract plans and final objectives motivating the overall decision to move (e.g. "I am going to the University today to follow my

[1] A similar classification can be found in vehicular traffic modeling from [10].

M. Gavanelli et al. (Eds.): AI*IA 2015, LNAI 9336, pp. 58–71, 2015.
DOI: 10.1007/978-3-319-24309-2_5

courses and meet my friend Paul"); (ii) **Tactical level**, constructing sequences of activities to achieve the defined objectives (e.g. "I'll take the 7:15 AM train from station X, get off at Y and then walk to the Department, then ..."); (iii) **Operational level**, physically executing the defined plans (i.e. creating a precise walking trajectory, such as a sequence of occupied cells and related simulation turn in a discrete simulation).

Most of the literature has been focused on the reproduction of the physics of the system, so on the lowest level: this is partly due to the fact that data on the fundamental diagram achieved with different set of experiments and in different environment settings (see, e.g[15]) supports a robust validation of the models. Relevant recent works, such as [7] and [14], start exploring the implications of tactical level decisions during evacuation. In particular, [7] modifies the floor-field Cellular Automata approach for considering pedestrian choices not based on the shortest distance criterion but considering the impact of congestion on travel time. [14] explores the implications of four strategies for the route choice management, given by the combination of applying the shortest or quickest path, with a local (i.e., minimize time to vacate the room) or global (i.e., minimize overall travel time) strategy. The global shortest path is calculated with the well-known Floyd-Warshall algorithm, implying computational times that can become an issue by having a large number of nodes or by considering special features in the simulated population (i.e. portion of the path where the cost differs from an agent to another). The work in this paper will propose an alternative and efficient approach to find a global path, where each agent will be able to consider additional costs in sub-paths without adding particular weight to the computation.

We must emphasize the fact that the measure of success and validity of a model is not the *optimality* with respect to some cost function, as in robotics, but the *plausibility*, the similarity of results to data acquired by means of observations or experiments. Putting together *tactical* and *operational* level decisions in a comprehensive framework, preserving and extending the validity that, thanks to recent extensive observations and analyses (see, e.g., [4]), can be achieved at the operational level, represents an urgent and significant open challenge.

The following Sect. will present the adaptive tactical level part of the model whereas its experimental application in benchmark scenarios showing the adequacy in providing adaptiveness to the contextual situation will be given in Section 3. Conclusions and future developments will end the paper.

2 A Model for Tactical Level of Pedestrians

The model described in this paper provides an approach to deal with tactical choices of agents in pedestrian simulation systems. For sake of space, the description of the operational level components of the model is omitted and it can be found in [1].

2.1 A Cognitive Representation of the Environment for Static Tactical Choices

The framework that enables agents performing choices on their plans implies a graph-like, topological, representation of the walkable space, whose construction is defined in [6] and only briefly reported in this section. This model allows agents to perform a static path planning, since dynamical information such as congestion is not considered in the graph. These additional elements will be considered in the extension that is presented in the next section and represent the innovative part of this paper.

The environment abstraction identifies *regions* (e.g. a room) as node of the labeled graph and *openings* (e.g. a door) as edges. This form of *cognitive map* is computed starting from the information of the simulation scenario, provided by the user and necessarily containing: (i) the description of the walkable space, that is, the size of the simulated environment and the positions of obstacles and walls; (ii) the position of final destinations (i.e. exits) and intermediate targets (e.g. a ticket machine); (iii) borders of the logical regions of the environment that, together with the obstacles, will define them. Approaches to automatically configure a graph representation of the space, without any additional information by the user, have been already proposed in the literature (e.g. [9]), but they are not leading to a cognitively logical description, i.e., a *topological* map.

The cognitive map is defined as a graph $\mathcal{CM} = (\mathcal{V}, \mathcal{E})$ generated with a procedure included to the floor field diffusion, starting from the statements that each user-defined opening generates a floor field from its cells and spread only in the regions that it connects, and that each region has a flag indicating its properties among its cells. The floor fields diffusion procedure iteratively adds to $\mathcal{CM}$ the couple of nodes found in the diffusion (preventing duplicates) and respectively labeled with the region and edge identifiers. Each *final destination*, different from the normal openings since it is located in only one region, will compose an edge linking the region to a special node describing the external *universe*. Intermediate targets will be mapped as attributes of their region's node.

To allow the calculation of the *paths tree*, that will be described in the following section, functions $Op(\rho)$ and $Dist(\omega_1, \omega_2)$ are introduced describing respectively: the set of openings accessible from the region ρ and the distance between two openings linking the same arbitrary region. Since, in general, an opening is associated to a set of cells associated, the value of the floor field in the center cell of ω_1, ω_2 will be used for the computation of the distance among them.

2.2 Modeling Adaptive Tactical Decisions with A Paths Tree

To enhance the route choice and enable dynamical, adaptive, decisions of the agents in a efficient way, a new data structure has been introduced, containing information about the cost of *plausible* paths towards the exit from each region of the scenario.

The well-known Floyd-Warshall algorithm, in fact, can solve the problem but it introduces issues in computational time: the introduction of dynamical elements in the paths cost computation (i.e. congested paths) implies a re-computation of the cost matrix underlying the algorithm every step. More in details, the penalty of a congested path is a subjective element for the agents, since they are walking with different desired velocities, thus the calculation cost increases also with the number of agents.

The approach proposed here implies an off-line calculation of the data-structure that we called *paths tree*, but is computationally efficient during the simulation and provides to the agents direct information about the travel times describing each path.

The Paths Tree. We define the *Paths Tree* as a tree data-structure containing the set of plausible paths towards a destination, that will be its root.

A path is defined as a finite sequence of openings $X \rightarrow Y \rightarrow \ldots \rightarrow Z$ where the last element represents the final destination. It is easy to understand that not every sequence of openings represents a path that is walkable by an agent.

First, a walkable path must be a sequence of consecutive oriented openings in the physical space: an opening E connects two regions R_1 and R_2, can be formally defined as $E = R_1, R_2$; (R_1, E, R_2) and (R_2, E, R_1) are the oriented representations of E. Consecutive openings E_1 and E_2 are such that (R_i, E_1, R_j) and (R_j, E_2, R_k).

In addition to this constraint, a valid walkable path must lead to a *universe* region (i.e. towards a final target). In particular, an agent will consider only valid paths towards its goal, starting from the region where the agent is located.

An important element in the definition of the adopted approach is the expected travel time associated to given path p:

Definition 1. *Let p a path, T(p) is the function which return the expected travel time from the first opening to the destination.*

$$T(p) = \sum_{i \in [1, |p|-1]} \frac{Dist(opening_i, opening_{i+1})}{speed} \tag{1}$$

We consider that a plausible path must be loop-free: by assuming the aim to minimize the time to reach the destination, a plan passing through a certain opening more than once would be not plausible. This will not imply that an agent cannot go through a certain opening more than once during the simulation, and that this could actually happen only with a change of the agent plan, due for instance to an unexpected congestion perceived in a point of the planned path.

Should only convex regions be present in the simulated space, we could easily achieve the set of plausible paths by extending previous constraint and consider not plausible a path passing twice in the same region. However, since the definition of region describes also rooms, concave regions must be considered. Some paths may, thus, imply to pass through another region and then return to the first one to reduce the length of the path.

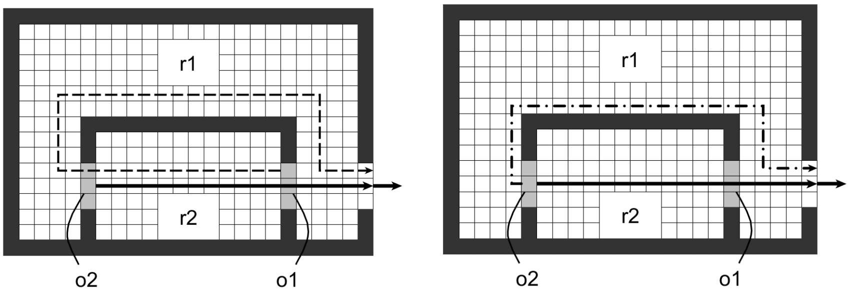


Fig. 1. In the left, a concave region can imply the plausibility of a path crossing it twice, but its identification is not elementary: only the path represented by the continuous line is plausible. All the correct paths for this environment are shown on the right. Inside $r2$ the choice between the two openings is determined by the level of congestion on o1.

As we can see by the Figure 1, on the left, both paths start from r_1, go through r_2, and then return to r_1. However, only the path represented by the continuous line is plausible. To support the definition of the constraint that identifies the correct paths, the concept of *sub-path* and a minimality rule must be defined.

Definition 2 (Sub-path). *Let p a path, a sub-path p' of p is a sub-sequence of oriented openings denoted as $p' \subset P$ which respects the order of appearance for the openings in p, but the orientation of openings in p' can differ from the orientation in p. p' must be a valid path.*

The reason of the orientation change can be explained with the example in Fig. 1 in the right: given the path $p = (r1, o2, r2) \rightarrow (r2, o1, r1) \rightarrow \mathit{end}$, the path $p' = (r2, o2, r1) \rightarrow \mathit{end}$ is a valid path and is considered as a sub-path of p, with a different orientation of $o2$. In addition, given the path $p_1 = (r_2, o_2, r_1) \rightarrow \mathit{end}$, the path $p_2 = (r_1, o_2, r_2) \rightarrow (r_2, o_1, r_1) \rightarrow \mathit{end}$ is as well a minimal path if and only if the travel time of p_2 is less than p_1. It is easy to understand that this situation can emerge only if r_1 is concave. As we can see, the starting region of the two paths is different, but the key element of the rule is the position of the opening o_2. If this rule is verified in the center position of the opening o_2, this path will be a considerable path by the agents surrounding $o2$ in $r1$.

In Figure 1, on the right, the correct paths for this example environment are shown. An agent located in $r2$ can reach $r1$ and then the destination D using both of the opening considering the congestion in the environment at the time of planning. An agent located in $r1$ can go directly to the exit or chose the path $o2 \rightarrow o1 \rightarrow D$, according to its starting position.

Definition 3 (Minimal Path). *p is a minimal path if and only if it is a valid path and $\forall p' \subset p : S(p') = S(p) \wedge D(p') = D(p) \implies T(p') > T(p)$*

The verification of this rule is a sufficient condition for the opening loop constraint and it solves the problem on the region loop constraint independently from the configuration of the environment (i.e. convex or concave regions).

Given this constraint on path minimality, which we consider an indication of plausibility, the complete set of minimal paths towards a destination can be built. It must be noted that an arbitrary path, through the notion of sub-paths, represents a set of paths itself: an agent, in fact, could select a sub-path of a larger minimal path. So a minimal representation of the set is a tree-like structure defined as:

Definition 4 (Paths Tree). *Given a set of minimal paths towards a destination, the Paths-Tree is a tree where the root represents the final destination and a branch from every node to the root describes a minimal path, crossing a set of openings (other nodes) and region (edges). Each node is associated to an attribute describing the expected travel time to the destination.*

An Algorithm to Compute the Paths Tree. The proposed algorithm constructs the Paths Tree recursively, starting from a path containing only the destination and adding nodes if and only if the generated path respects the definition of minimality.

Formally the Paths Tree is defined as $PT = (N, E)$ where N is the set of nodes and E the set of edges. Each node $n \in N$ is defined as a triple (id, o, τ) where $id \in \mathbb{N}$ is the id of the node, $o \in \mathcal{O}$ is the name of the opening and $\tau \in \mathbb{R}^+$ is the expected travel time for the path described by the branch. Each edge $e \in E$ is defined as a triple (p, c, r) where $p \in \mathcal{O}$ is the id of the parent, $c \in \mathcal{O}$ is the id of the child and $r \in \mathcal{R}$ is the region connecting the child node to its parent. To allow a fast access to the nodes describing a path that can be undertaken from a certain region, we added a structure called M that maps each region r in the list of edges $e : (p, c, r) \in E$ (for every c).

Given a destination $D = (r_x, \text{universe})$, the paths tree computation is defined with the following procedures.

Algorithm 1. Paths tree computation

```
1: add $(0, D, 0)$ to $N$
2: add 0 to $M[r_x]$
3: $\forall s \in \mathcal{O}$ ShortestPath[s] $\leftarrow \infty$
4: ExpandRegion$(0, D, 0, R_x, ShortestPath)$
```

With the first line, the set N of nodes is initialized with the destination of all paths in the tree, marking it with the id 0 and expected travel time 0. In the third row the set of *ShortestPath* is initialized. This will be used to track, for each branch, the expected travel time for the shortest sub-path, given a start opening s. *ExpandRegion* is the core element of the algorithm, describing the

Algorithm 2. ExpandRegion

```
Require: input parameters ($parentId$, $parentName$,
    $parentTime$, $RegionToExpand$, $ShortestPath$)
 1: $expandList \leftarrow \emptyset$
 2: $opList = Op(RegionToExpand) \setminus parentName$
 3: for $o \in opList$ do
 4:     $\tau = parentTime + \frac{D(o, parentName)}{speed}$
 5:     if $CheckMinimality(ShortestPath, o, \tau)$ == True then
 6:         add $(id, o, \tau)$ to $N$
 7:         add $(parentId, id, r)$ to $E$
 8:         $ShortestPath[o] \leftarrow \tau$
 9:         $nextRegion = o \setminus r$
10:         add $id$ to $M[nextRegion]$
11:         add $(id, o, \tau, nextRegion)$ to $expandList$
12:     end if
13: end for
14: for $el \in expandList$ do
15:     $ExpandRegion(el, ShortestPath)$
16: end for
```

recursive function which adds new nodes and verifies the condition of minimality. The procedure is described by Alg. 2.

In line 2 a list of openings candidates is computed, containing possible extensions of the path represented by $parentId$. Selecting all the openings present in this region (except for the one labeled as $parentName$) will ensure that all paths eventually created respect the validity constraint.

At this point, the minimality constraint 3 has to be verified for each candidate, by means of the function $CheckMinimality$ explained by Alg. 3. Since this test requires the expected travel time of the new path, this has to be computed before. A failure in this test means that the examined path – created by adding a child to the node $parentId$ – will not be minimal. Otherwise, the opening can be added and the extension procedure can recursively continue.

In line 6, id is a new and unique value to identify the node, which represents a path starting from the opening o and with expected travel time τ; line 7 is the creation of the edge from the parent to the new node. In line 8, $ShortestPath[o]$ is updated with the new discovered value τ. in line 9 the opening is examined as a couple of region, selecting the one not considered now. In fact, the element $nextRegion$ represents the region where is possible to undertake the new path. In line 10 the id of the starting opening is added to $M[nextRegion]$, i.e., the list of the paths which can be undertaken from $nextRegion$. In line 11 the node with his parameter is added to the list of the next expansions, which take place in line 13-14. This passage has to be done to ensure the correct update of $ShortestPath$.

To understand how the constraint of minimality is verified, two basic concepts of the procedure need to be clarified. Firstly, the tree describes a set of paths towards a unique destination, therefore given an arbitrary node n, the path described by the parent of n is a subpath with a different starting node and

Algorithm 3. CheckMinimality

```
Require: input parameter ($ShortestPath, o, \tau$)
1: if $ShortestPath[o] > \tau$ then
2:     return True
3: else
4:     return False
5: end if
```

leading to the same destination. Furthermore, the expansion procedure implies that once reached a node of depth l, all the nodes of its path having depth $l - k$, $k > 0$ have been already expanded with all child nodes generating other minimal paths.

Note that the variable $ShortestPath$ is particularly important since, given p the current path in evaluation, it describes the minimum expected travel time to reach the destination (i.e. the root of the tree) from each opening already evaluated in previous expansions of the branch. Thus, if τ is less than $ShortestPath[o]$, the minimality constraint 3 is respected.

Congestion Evaluation. The explained approach of the paths tree provides information on travel times implied by each path towards a destination. By only using this information, the choice of the agents would be still static, essentially describing the shortest path. This could lead to an increase of the experienced travel times, since congestion may emerge without being considered.

For the evaluation of congestion, we provide an approach that estimates, for each agent, the additional time deriving by passing through a jam. The calculation considers two main aspects: the size of the possibly arisen congestion around an opening; the average speed of the agents inside the congested area. Since the measurement of the average speed depends on the underlying model that describes the physical space and movement of the agents, we will just clarify that the speed is estimated through the adoption of an additional grid counting the recent *blocks* (i.e. when agents maintain positions at the end of the step although desired to move) in the surrounding area of each opening. The average number of blocks influences the probability to move into the area per step, thus the speed of the agents. For the size of the area, our approach is to define a minimum radius of the area and (i) increase it when the average speed becomes too low or (ii) reduce it when it returns normal.

As we can see in Figure 2, the presence of an obstacle in the room is well managed by using floor field while defining the area for a given radius. If a many agents try to go through the same opening at the same time, a congestion will arise, reducing the average speed and increasing the size of the monitored area until included agents are no more involved in blocks.

During this measurement the average speed value for each radius is stored. Values for sizes smaller than the size of the area will be used by the agents inside it, as will be explained in the next section. Two function are introduced

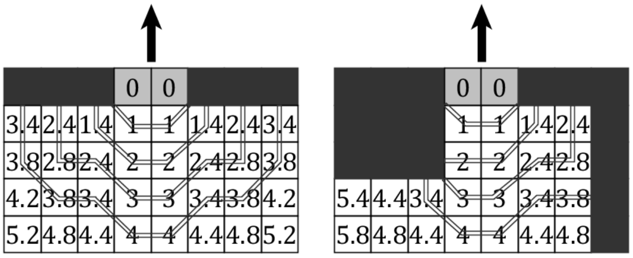


Fig. 2. Examples of surroundings of different sizes, for two configurations of the environment.

for the calculation: $size(o)$: return the size of the congestion around the opening, $averageSpeed(o, s)$: return the average speed estimated in the area of size s around the opening o.

Agents Dynamic Path Choice. At this point we have defined which information an agent will use to make its decision: (i) the Paths Tree, computed before the simulation, will be used as a list of possible path choice; (ii) the position of the agent, will be used to adjust the expected travel time considering the distance between the agent and the first opening of a path: $d(a, o)$; (iii) the information about congestion around each opening, computed during the simulation, will be used to estimate the delay introduced by each opening in the path.

The agent, who knows in which region R_x he is located, can access the Paths Tree using the structure $M[R_x]$. The structure returns a list of nodes, each representing the starting opening for each path. At this point the agent can compute the expected travel time to reach each starting opening and add it to the travel time τ of the path.

To consider congestion, the agent has to estimate the delay introduced by each opening in a path, by firstly obtaining the size of the jammed area.

$$size_a(o) = \begin{cases} size(o) & \text{if } d(a, o) \geq i(x) \\ d(a, o) & \text{otherwise} \end{cases} \tag{2}$$

At this point, the agent can suppose that for the length of the area it will travel at the average speed around the opening.

$$delay(o) = max(size_a(o)(\frac{1}{averageSpeed(o)} - \frac{1}{speed_a}), 0) \tag{3}$$

If the agent is slower than the average speed around an opening, the delay will be lower than 0. In this case it is assumed that the delay is 0, implying that the congestion will not increase his speed.

At this point the agent can estimate the delay introduced by all openings.

$$pathDelay(p) = \sum_{o \in p} delay(o) \tag{4}$$

We can consider that agents only have access to delay information about openings that are present in the area it is located into, whereas the delay is considered zero (in an optimistic hypothesis) in openings that are far from its perception.

$$Time(p) = \tau_p + \frac{d(a, S(p))}{speed_a} + pathDelay_a(p) \tag{5}$$

Where:

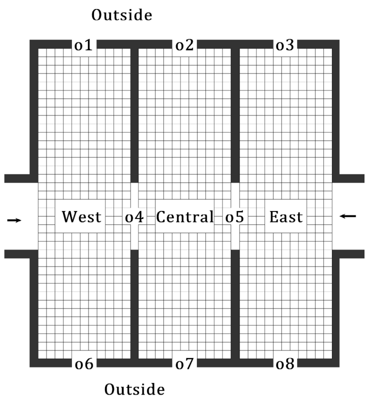
- τ_p : the expected travel time of the path p
- $\frac{d(a,S(p)}{speed_a}$: the expected time to reach $S(p)$ from the position of the agent
- $pathDelay_a(p)$: the estimation of the delay introduced by each opening in the path, based on the memory of the agent (which may or may not be updated for each opening).

3 Applications with an Experimental Scenario

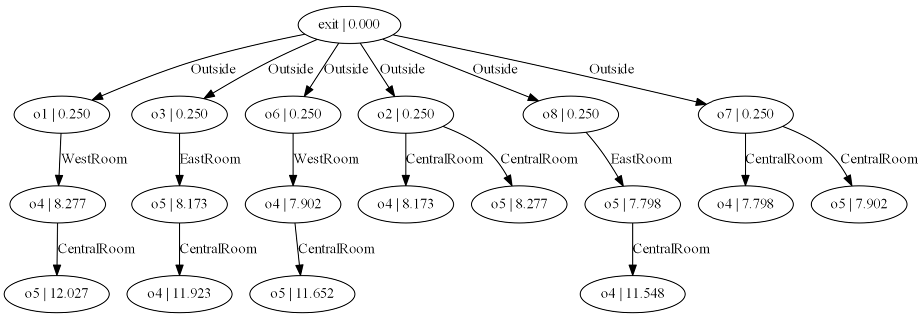
In order to show the potential and the possibility to fine tune the proposed approach, the evacuation in a hypothetical scenario has been simulated with a consistent incoming flow of people. A graphical representation of the environment and flow configuration is depicted in Fig. 3(a): it illustrates a sample situation in which two flows of pedestrians enter an area with six exits, distributed among 3 equal rooms, at a rate of 10 pedestrians per second. An important peculiarity is the slightly asymmetrical configuration of the environment, that causes shorter distances towards the three southern exits. This is reflected by the illustrated paths tree in Fig. 3(b) where, to give an example, the paths starting from $o4$ and $o5$ and leading out through $o2$ take a little more time than the ones going out by using $o7$. This variation significantly affected the results of the simulations, here shown with cumulative mean density maps [5][2] in Fig. 4.

In particular, the results of two simulations in which different approaches have been implemented for the dynamic estimation of the path traveling times by the agents are shown. In the first approach, shown in the top row, all the agents perceived the same *congestion time* for the openings that they can detect during the simulation (i.e. the travel time corrected considering the path delay discussed in the previous section). In the second approach, instead, a random error of $\pm 10\%$ has been added to the overall calculation of the traveling time

[2] These heat maps describe the mean local density value in each cell. It is calculated in a time window of 50 steps where, at each step, only values of occupied cells are collected.



(a)

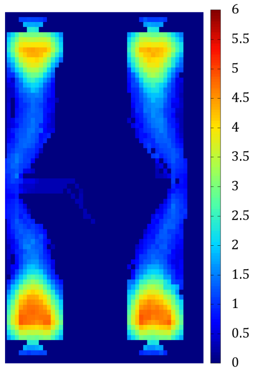


(b)

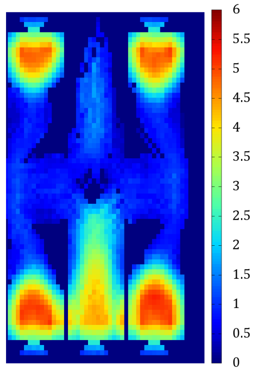
Fig. 3. The experimental scenario (a) and the associated paths tree (b).

$Time(p)$ in order to consider the fact that pedestrians do not have an *exact* estimation of distances and delays caused by perceived congestion, in a more commonsense spatial reasoning framework [3].

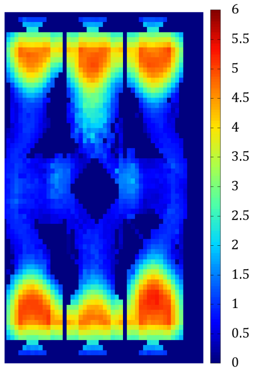
By comparing the results it is possible to notice that, counter–intuitively, the insertion of the random perturbation caused an optimization of the flows in this overcrowded scenario. In the firsts 100 steps of the simulations, the dynamics for the two approaches is similar and described by the missed usage of the central room, since the distance between the northern and southern exits is quite small. The less precise calculation causes the agents to start using the central room and associated exits earlier than in the precise delay estimation case, in particular, around 130th step vs 150th step in the first scenario, generating lower level of densities and, thus, higher outgoing flow rates. Moreover, this error balances the attractiveness of middle southern and northern exits that are more evenly adopted than in the precise calculation approach (as shown in Fig. 4 (b) and (e)), leading not only to a more efficient but especially more plausible space utilization.



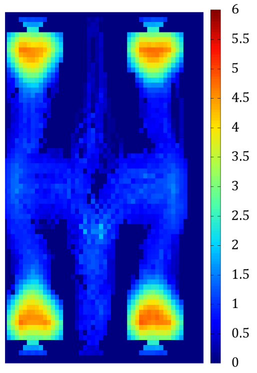
(a) step 150-200, w/o RE



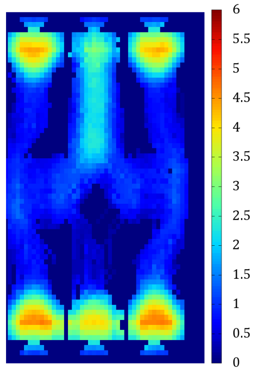
(b) step 300-350, w/o RE



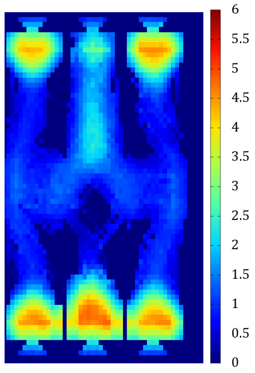
(c) step 450-500, w/o RE



(d) step 150-200, with RE



(e) step 300-350, with RE



(f) step 450-500, with RE

Fig. 4. The test scenario respectively without and with a random perturbation of the agent estimated travel time.

4 Conclusions

The paper has presented a hybrid agent architecture for modeling tactical level decisions in pedestrian simulations. The agents make decisions based on a static a-priori knowledge of the environment and dynamic perceivable information on the current level of congestion of visible path alternatives. The model was experimented in a sample scenario showing the adequacy in providing adaptiveness to the contextual situation while preserving a plausible overall pedestrian dynamic: congestion is detected and, when possible, longer trajectories are adopted granting overall shorter travel times. The actual validity of this approach must still be proven, both in evacuations and other kinds of situations: this represents an open challenge, since there are no comprehensive data sets on human tactical level decisions and automatic acquisition of this kind of data from video cameras is still a challenging task [8].

Acknowledgments. This work was partly supported by the ALIAS project ("Higher education and internationalization for the Ageing Society"), funded by Fondazione CARIPLO.

References

1. Bandini, S., Crociani, L., Vizzari, G.: Heterogeneous speed profiles in discrete models for pedestrian simulation. In: Proceedings of the 93rd Transportation Research Board Annual Meeting (2014). http://arxiv.org/abs/1401.8132
2. Bandini, S., Manzoni, S., Vizzari, G.: Agent based modeling and simulation: An informatics perspective. Journal of Artificial Societies and Social Simulation **12**(4), 4 (2009)
3. Bandini, S., Mosca, A., Palmonari, M.: Common-sense spatial reasoning for information correlation in pervasive computing. Applied Artificial Intelligence **21**(4&5), 405–425 (2007). doi:10.1080/08839510701252676
4. Boltes, M., Seyfried, A.: Collecting pedestrian trajectories. Neurocomputing **100**, 127–133 (2013). http://www.sciencedirect.com/science/article/pii/S0925231212003189
5. Castle, C.J.E., Waterson, N.P., Pellissier, E., Bail, S.: A comparison of grid-based and continuous space pedestrian modelling software: analysis of two uk train stations. In: Peacock, R.D., Kuligowski, E.D., Averill, J.D. (eds.) Pedestrian and Evacuation Dynamics, pp. 433–446. Springer, US (2011). doi:10.1007/978-1-4419-9725-8_39
6. Crociani, L., Invernizzi, A., Vizzari, G.: A hybrid agent architecture for enabling tactical level decisions in floor field approaches. Transportation Research Procedia **2**, 618–623 (2014)
7. Guo, R.Y., Huang, H.J.: Route choice in pedestrian evacuation: formulated using a potential field. Journal of Statistical Mechanics: Theory and Experiment **2011**(04), P04018 (2011)
8. Khan, S.D., Vizzari, G., Bandini, S.: Identifying sources and sinks and detecting dominant motion patterns in crowds. Transportation Research Procedia **2**, 195–200 (2014). http://www.sciencedirect.com/science/article/pii/S2352146514000660, the Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014), 22-24 October 2014, Delft, The Netherlands
9. Kretz, T., Bönisch, C., Vortisch, P.: Comparison of various methods for the calculation of the distance potential field. In: Klingsch, W.W.F., Rogsch, C., Schadschneider, A., Schreckenberg, M. (eds.) Pedestrian and Evacuation Dynamics 2008, pp. 335–346. Springer, Heidelberg (2010). doi:10.1007/978-3-642-04504-2_29
10. Michon, J.A.: A critical view of driver behavior models: what do we know, what should we do? In: Evans, L., Schwing, R.C. (eds.) Human Behavior and Traffic Safety, pp. 485–524. Springer, US (1985). doi:10.1007/978-1-4613-2173-6_19
11. Pianini, D., Viroli, M., Zambonelli, F., Ferscha, A.: HPC from a self-organisation perspective: the case of crowd steering at the urban scale. In: International Conference on High Performance Computing and Simulation, HPCS 2014, Bologna, Italy, 21–25 July, 2014, pp. 460–467. IEEE (2014). doi:10.1109/HPCSim..6903721

12. Schadschneider, A., Klingsch, W., Klüpfel, H., Kretz, T., Rogsch, C., Seyfried, A.: Evacuation dynamics: empirical results, modeling and applications. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and Systems Science, pp. 3142–3176. Springer (2009)
13. Vizzari, G., Bandini, S.: Studying pedestrian and crowd dynamics through integrated analysis and synthesis. IEEE Intelligent Systems **28**(5), 56–60 (2013). doi:10.1109/MIS.2013.135
14. Wagoum, A.U.K., Seyfried, A., Holl, S.: Modelling dynamic route choice of pedestrians to assess the criticality of building evacuation. Advances in Complex Systems **15**(07), 15 (2012)
15. Zhang, J., Klingsch, W., Schadschneider, A., Seyfried, A.: Transitions in pedestrian fundamental diagrams of straight corridors and t-junctions. Journal of Statistical Mechanics: Theory and Experiment **2011**(06), P06004 (2011). http://stacks.iop.org/1742-5468/2011/i=06/a=P06004