A-Evac: the evacuation simulator for stochastic environment

Adam Krasuski · Karol Krenski

Received: 24.11.2017 / Accepted: date

Abstract We introduce an open-source software Aamks for fire risk assessment. This article focuses on a component of Aamks – an evacuation simulator named a-evac. A-evac models evacuation of humans in the fire environment produced by CFAST fire simulator. In the article we discuss the probabilistic evacuation approach, automatic planning of exit routes, the interactions amongst the moving evacuees and the impact of smoke on the humans. The results consist of risk values based on FED, F-N curves and evacuation animations.

Keywords Evacuation Modeling · Stochastic Simulations · Risk Management · Monte Carlo

1 Introduction

These days, there's not much you can understand about what is going on around you if you do not understand the uncertainty attached to pretty much every phenomenon.

— J. N. Tsitsiklis

There is a continuous progress in understanding the fire phenomenon, its impact on the structure and on the reaction of humans and safety systems.
There are scientific methods and models of the fire and the emergency scene, there are computer implementations, but the complexity of the domain impedes the more widespread use of these tools.

Currently, the most typical approach for assessing the safety of the building is a precise choosing of the input parameters for a small number of lengthy, detailed simulations. This procedure is managed by a practitioner, based on his experience. However, based on heuristics and biases \cite{31,22} we have concerns that human judgement surpasses statistical calculations. The alternative is to let computer randomly choose the parameters and run thousands of simulations. The resulting collection allows us, after further processing, to judge on the safety of the building.

2 Aamks, the multisimulations platform

We created Aamks – the platform for running simulations of fires and then running the evacuation simulations, but thousands of them for a single project. This is the Monte-Carlo approach. We use CFAST, which is a rough, but a fast fire simulator. This allows us to explore the space of possible scenarios and assess the probability of them. The second component of risk – consequences – is taken from an evacuation simulator capable to model evacuation in the fire environment. We use a-evac as the evacuation simulator which we have built from scratch. The multisimulation is a handy name for what we are doing. Aamks tries to assess the risk of failure of humans evacuation from a building under fire. We applied methodology proposed in \cite{16,8,1} – stochastic simulations based on the Simple Monte-Carlo approach \cite{7}. Our primary goal was to develop an easy to use engineering tool rather than a scientific tool, which resulted in: AutoCAD plugin for creating geometries, web based interface, predefined setups of materials and distributions of various aspects of buildings features, etc. The workflow is as follows: The user draws a building layout or exports an existing one. Next, the user defines a few parameters including the type of the building, the safety systems in the building, etc. Finally, they launch a defined number of stochastic simulations. As a result they obtain the distributions of the safety parameters, namely: available safe egress time (ASET), required safe egress time (RSET), fractional effective dose (FED), hot layer height and temperature and F-N curves as well as the event tree and risk matrix.

Fortran is a popular language for coding simulations of physical systems. CFAST and FDS+Evac are coded in fortran. Since we don’t create a fire simulator we code Aamks in python which is more comfortable due to extremely rich collection of libraries. We decided that borrowing Evac from FDS and integrating it with Aamks would be harder for us than to code our own evacuation simulator, hence a-evac was born. There’s also a higher chance of attracting new python developers than fortran developers for our project.

Aamks consists of the following modules:
3 A-evac, the evacuation simulator

In the following subsections we describe the internals of a-evac, sometimes with the necessary Aamks context.

3.1 Geometry of the environment

The Aamks workflow starts with a 3D geometry where fires and evacuations will be simulated. We need to represent the building, which contains one or more floors. Each floor can consist of compartments and openings in them, named respectively COMPAS and VENTS in CFAST. Our considerations are narrowed to rectangular geometries. There are two basic ways for representing architecture geometries: a) cuboids can define the insides of the rooms (type-a-geometry) or b) cuboids can define the walls / obstacles (type-b-geometry). CFAST uses the type-a-geometry. We create CFAST geometries from the input files of the following format (there are more entities than presented here):

```json
{
  "FLOOR 1": {
    "ROOM": [
      [ [ 3.0 , 4.8 , 0.0 ] , [ 4.8 , 6.5 , 3.0 ] ],
      [ [ 3.0 , 6.5 , 0.0 ] , [ 6.8 , 7.4 , 3.0 ] ]
    ],
    "COR": [
      [ [ 6.2 , 0.2 , 0.0 ] , [ 7.6 , 4.8 , 3.0 ] ]
    ],
    "D": [
      [ [ 3.9 , 3.4 , 0.0 ] , [ 4.8 , 3.4 , 2.0 ] ]
    ],
    "W": [
      [ [ 1.2 , 3.4 , 1.0 ] , [ 2.2 , 3.4 , 2.0 ] ]
    ],
    "HOLE": [ }
```
ROOM and COR(RIDOR) belong to COMPAS. D(OOR), W(INDOW) and HOLE belong to VENTS. HOLE is a result of CFAST restrictions – it is an artificial entity which serves to merge two compartments into a single compartment as shown on Figure 1.

![Fig. 1](image1) The concept of a HOLE: a) the room in reality, b) the room representation in CFAST: two rectangles for separated calculations, but open to each other via HOLE.

All the entities in the example belong to the same FLOOR 1. The triplets are \((x_0, y_0, z_0)\) and \((x_1, y_1, z_1)\) encoding the beginning and the end of each entity in 3D space. In practice we obtain these input files from AutoCAD, thanks to our plugin which extracts data from AutoCAD drawing. There’s also an Inkscape svg importer – useful, but without some features. Adding basic support for another graphics tools is not much work.

In later sections we will introduce problems of guiding evacuees throughout the building. Those modules require the type-b-geometry. We convert from a type-a-geometry to a type-b-geometry by duplicating the geometry, translating the geometry and applying some logical operations. Figure 2 shows the idea.

![Fig. 2](image2) The conversion from type-a-geometry to type-b-geometry

There are three aspects of movement when it comes to evacuation modeling [9]: (a) path-finding – for the rough route out of the building, (b) local movement – evacuees interactions with other evacuees, with obstacles and with environment, and (c) locomotion – for "internal" movement of the agent (e.g. body sway). A-evac models only (a) and (b).
3.2 Path-finding (roadmap)

The simulated evacuees need to be guided out of the building. The type-b-geometry provides the input for path-finding. Each of the cuboids in type-b-geometry – representing obstacles – is defined by the coordinates. These coordinates represent corners of the shapes. Since we model each of the floors of a building separately, we flatten 3D geometry into 2D and represent obstacles as rectangles. Therefore type-b-geometry in path-finding module is represented as set of 4-tuple coordinates \((x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)\).

The set of 4-tuple elements is then flatten to the set of coordinates – bag-of-coordinates. Due to the fact that majority of the obstacles share the coordinates, we remove duplicates from the set (for the sake of performance). Then, this bag-of-coordinates is the input for triangulation. We apply Delaunay triangulation \([10]\), that represents space as a set of triangles. Figure 3 depicts the idea of triangulation.

The triangles are used as navigation meshes for the agents. The navigation meshes define which areas of an environment are traversable by agents.

After triangulation of bag-of-coordinates, some of the triangles are located inside the obstacles – those (by definition) are not traversable so we remove them. What is left is a traversable-friendly space.

We create then the graph of spatial transitions for the agents, based on the adjacency of triangles obtained from the triangulation. Spatial transition means that an agent can move from one triangle to another.

An agent on an edge of a triangle can always reach the other two edges. For triangles which share edges it allows an agent to travel from one triangle to another.

The pairs of all neighbouring edges are collected. We use python networkx module \([4]\) which creates a graph made of the above pairs. For further processing we add agents positions to the graph, by pairing them with the neighbouring edges.

The graph represents all possible routes from any node to any other node in the graph. We can query the graph for the route from the current agent’s position to the closest exit. It means that agent will walk through the consecutive nodes and will finally reach the exit door. We instruct networkx that we need the shortest distances in our routes (default is the least hops on the

---

**Fig. 3** The idea of triangulation. a) original geometry, b) bag-of-coordinates, c) triangulation.
graph) and we obtain a set of edges the agent should traverse in order to reach the exit. Figure 4 depicts the set of edges returned by the graph for an example query.

![Image of graph](image)

**Fig. 4** The roadmap defined by the graph for an example query. The red line crosses centers of edges that an agent needs to travel to reach the exit.

The set of edges returned by the graph cannot be used directly for path-finding. Neither the vertices of the edges nor the centers of them, do define the optimal path that would be naturally chosen by evacuees during real evacuation. Therefore an extra algorithm should be used to smooth the path. For this purpose we apply funnel algorithm defined in [6]. The funnel is a simple algorithm finding straight lines along the edges.

The input for the funnel consists of a set of ordered edges (named portals) from the agent origin to the destination. The funnel always consist of 3 entities: the origin (apex) and the two vectors from apex to vertices on edges – the left leg and the right leg.

The apex is first set to the origin of the agent and the legs are set to the vertices of the first edge. We advance the left and right legs to the consecutive edges in the set and observe the angle between the legs. When the angle gets smaller we accept the new vertex for the leg. Otherwise the leg stays at the given vertex. After some iteration one of the legs should cross the other leg defining the new position of the apex. The apex is moved and we restart the procedure.

As a result the path is smoothened and defined only by the points where the changes in velocity vector are needed. Moreover, we used an improved version of the funnel algorithm that allows for defining points keeping a distance from the corners reflecting the size of the evacuee. This allows for modeling the
A-Evac: the evacuation simulator for stochastic environment

impaired evacuees on wheeled chairs or beds in the hospitals. Figure 6 depicts
the smoothened path by funnel algorithm.

3.3 Local movement

Local movement focuses on the interaction with (a) other agents (b) static
obstacles (walls) and (c) environmental conditions. A-evac handles (a) and (b)
via RVO2, which is an implementation of the Optimal Reciprocal Collision
Avoidance (ORCA) algorithm proposed in [3,32]. Later in this section we
describe how we are picking the local targets which is an aspect of (b). (c) is
basically altering agent’s state such as speed.

[1] http://gamma.cs.unc.edu/RVO2/
RVO2 aims at avoiding the velocity obstacle [11]. The velocity obstacle is the set of all velocities of an agent that will result in a collision with another agent or an obstacle. Otherwise, the velocity is collision-avoiding. RVO2 aims at asserting that none of the agents collides with other agents within time $\tau$.

The overall approach is as follows: each of the agents is aware of other agents parameters: their position, velocity and radius (agent’s observable universe). Besides, the agents have their private parameters: maximum speed and a preferred velocity which they can auto-adjust granted there is no other agent or an obstacle colliding. With each loop iteration, each agent responds to what he finds in his surroundings, i.e. his own and other agents radiuses, positions and velocities. The agent updates his velocity if it is the velocity obstacle with another agent. For each pair of colliding agents the set of collision-avoiding velocities is calculated. RVO2 finds the smallest change required to avert collision within time $\tau$ and that is how an agent gets his new velocity. The agent alters up to half of his velocity while the other colliding agent is required to take care of his half. Figure 8 a-b depicts the idea of velocity collision avoidance.

The algorithm remains the same for avoiding static obstacles. However, the value of $\tau$ is smaller with respect to obstacles as agents should be more ‘brave’ to move towards an obstacle if this is necessary to avoid other agents.

It turned out problematic how to pick the local target from the roadmap. Local targets need to be updated (usually advanced, but not always) near points defined by funnel algorithm during path-fining phase – the disks on Figure 7 after they become visible to the agent. However, the disks can be crowded and agents can be driven away from the correct courses by other agents. We carefully inspected all possible states that agents can find themselves in. In order to have a clearer insight and control over the agents inside the disks we use the Finite State Machine[2] instead of just plain algorithm block in our code. The state of the agent is defined by 4 binary features: (a) is agent inside the disk? (b) are where agent is walking to and what agent is looking at the same target? (c) can agent see what he is looking at (or are there obstacles in-between)? (d) has agent reached the final node?

Within each iteration of the main loop we check the states of the agents. The states can be changed by agents themselves – e.g. agent has crossed the border of the disk, or by our commands – e.g. agent is ordered to walk to another target. Consider these circumstances: the agent has managed to see his next target and now he walks towards this next target – he is in state S1. But now he loses the eye contact with this new target and finds himself in state S2. The program logic reacts to such a state by transiting to the state S3: start looking at the previous target and walk towards this previous target. Based on what happens next we can order the transition to another state or just wait for the agent to change the state himself. By careful examination of all possible circumstances we can make sure that our states and their transitions can handle all possible scenarios.

[2] https://en.wikipedia.org/wiki/Finite-state_machine
On Figure 8c) we show how agents are passing through a HOLE. Due to our concept of the disks (where searching for new targets takes place) and due to the internals of RVO2 we gain the desired effect of agents not crossing the very center of the disk. Instead, the agents can walk in parallel and advance to another target which looks natural and doesn’t create an unnecessary queue of agents eager to cross the very center of the HOLE.

3.4 Evacuation under fire and smoke

Each a-evac simulation is preceded with the simulation of the fire. We have only tested a-evac with CFAST [24][21]. CFAST writes its output to csv files.
We need to query these CFAST results quite a bit, therefore we transform and store these results in a fast in-memory relational database\footnote{https://www.sqlite.org/}. For each frame of time we are repeatedly asking the same questions: (a) given the agent’s coordinates, which room is he in? (b) what are the current conditions in this room?

When it comes to (b), the environment effects on the agent can be: (b.1) limited visibility (eyes), (b.2) poisonous gases (nose) and (b.3) temperature in the room (body). Both (b.1) and (b.2) are read from the default (but configurable) height of 1.8 m. There are always two zones in CFAST, which are separated at a known height, so we need to read the conditions from the correct zone, based on where our 1.8 m belongs.

The value of visibility ($OD$ – optical density) affects agent’s speed. We use the relation proposed in \cite{12} following the FDS+Evac \cite{23}:

$$v_{n}^{pref}(K_s) = \max \left\{ v_{n,\min}, v_{n}^{pref} \left( 1 + \frac{\beta}{\alpha} \cdot K_s \right) \right\}$$

(1)

where: $K_s$ is the extinction coefficient ($[K_s] = m^{-1}$) calculated as $OD/\log_{10}e$ according to \cite{19,20}, $v_{n,\min}$ is the minimum speed of the agent $A_n$ and equals $0.1 \cdot v_{n}^{pref}$ (agent’s preferable velocity), and $\alpha$, $\beta$ are the coefficients defined in \cite{12}.

Setting the minimal value of speed means that the agent does not stop in thick smoke. They continue moving until the value of incapacitated Fractional Effective Dose (FED) is exceeded, which is fatal to the agent. FED is calculated from CFAST-provided amounts of the following species in the agent environment: carbon monoxide (CO), hydrogen cyanide (HCN), hydrogen chloride (HCl), carbon dioxide ($CO_2$) and oxygen ($O_2$) by the equation \cite{26,23}:

$$FED_{total} = (FED_{CO} + FED_{HCN} + FED_{HCl}) \times HV_{CO_2} + FED_{O_2}$$

(2)

where $HV_{CO_2}$ is the hyperventilation induced by the concentration of $CO_2$. Following are the formulas for the terms in the above equation. FEDs are given in ppm and time $t$ in minutes. $C$ stands for concentration of the species in %:

$$FED_{CO} = \int_{0}^{t} 2.764 \times 10^{-5}(C_{CO}(t))^{1.036} dt$$

(3)

$$FED_{HCN} = \int_{0}^{t} \frac{e^{ GC_{HCN}(t)} 220}{48} - 0.0045 dt$$

(4)

Based on \cite{17}

In contrast to the model applied in Evac, CFAST does not allow for proactive correction of effect of nitrogen dioxide – $C_{CN} = C_{HCN} - C_{NO_2}$. Therefore this effect is not included in the calculations.
\[ FED_{HCl} = \int_0^t \frac{C_{HCl}(t)}{1900} \, dt \]  

(5)

Based on [28,17]

\[ FED_{O_2} = \int_0^t \frac{dt}{60 \cdot \exp[8.13 - 0.54(20.9 - C_{O_2}(t))]} \]  

(6)

\[ HV_{CO_2} = \frac{\exp(0.1903 \cdot C_{CO_2}(t) + 2.0004)}{7.1} \]  

(7)

There are few quantitative data from controlled experiments concerning the sublethal effect of the smoke on people. In works [28,5,26,14,29] sublethal effect in a form of incapacitation (IC\textsubscript{50}), escape ability (EC\textsubscript{50}), lingering health problems and minor effects were reported. Incapacitation was inferred from lethality data, to be about one-third to one-half of those required for lethality. The mean value of the ratios of the IC\textsubscript{50} to the LC\textsubscript{50} was 0.50 and the standard deviation 0.21, respectively. In [14] a scale for effects based on FED was introduced. The three ranges were proposed: 1 FED indicating lethality, 0.3 FED indicating incapacitation and 0.01 FED indicating no significant sublethal effects should occur. We propose based on this data a scale for sublethal effects of smoke for evacuees as presented in Table 1.

\textit{FED\textsubscript{total}} affects the agent’s movement in the smoke. For \( FED_{\text{total}} > 0.3 \), the smoke inhalation leads to sublethal effects [5] – the agent is not able to find safety from the fire and just stays where he is. For \( FED_{\text{total}} > 1 \) we model lethal effects. We later use these effects in the final risk assessment.

Table 1 summarizes the FED effects on human health, what is our original proposition for evaluation of sublethal effect of smoke. These ranges are incorporated in Aamks. It is based on the following works: [28,5,26,14,29].

| FED   | Effect on human health                  |
|-------|----------------------------------------|
| < 0.01| Minor or negligible                    |
| [0.01 - 0.3] | Low – short period of hospitalization |
| [0.3 – 1] | Heavy – lingering health problems or permanent disability |
| ≥ 1   | Lethal                                |

3.5 Probabilistic evacuation modeling

This section presents the internals of our probabilistic evacuation model, which we find distinct across the available, similar software.

Table 2 presents the distributions of the input parameters used in Aamks. Each of the thousands of simulations in a single project is initialized with some
random input setup according to these distributions. Aamks has a library of the default parameters values for important building categories (schools, offices, malls etc.). The Aamks users should find it convenient to have all the distributions in a library, but they may choose to alter these values, which is possible.

Most of the data in table 2 come from the standards and from other models, mostly FDS/Evac. Following are some comments on table 2.

Aamks puts much attention to the pre-evacuation time [9], which models how people lag before evacuating after the alarm has sounded. Positions 7. and 8. are separated, because the behaviour of humans in the room of fire origin is distinct. We compile two regulations C/VM2 Verification Method: Framework for Fire Safety Design [30] and British Standard PD 7974-6:2004 [2] in order to get the most realistic, probability-based pre-evacuation in the room of fire origin and in the rest of the rooms.

The Horizontal/Vertical speed (unimpeded, walking speed of an agent) is based on [18,13,25,15]. Speed in the smoke is modeled by formula [1].

Table 2 Parameters of the distributions for the exemplary scenario.

| Parameter | Distribution | μ/ min | σ/ max |
|-----------|--------------|--------|--------|
| 1. Density in rooms \( m^2/\text{humans} \) | normal | 5 | 2 |
| 2. Density on corridors \( m^2/\text{humans} \) | normal | 20 | 3 |
| 3. Density in stairways \( m^2/\text{humans} \) | normal | 50 | 3 |
| 4. Human location in the compartment \( x \) | uniform 0 | room width |
| 5. Human location in the compartment \( y \) | uniform 0 | room depth |
| 6. Time of the alarm | log-normal 0.7 | 0.2 |
| 7. Pre-evacuation time in the room of fire origin | uniform 0 | 30 |
| 8. Pre-evacuation time in other compartments | log-normal 3.04 | 0.142 |
| 9. Horizontal speed | normal 1.2 | 0.2 |
| 10. Vertical speed | normal 0.7 | 0.2 |
| 11. \( \alpha \) for speed in the smoke | normal 0.706 | 0.069 |
| 12. \( \beta \) for speed in the smoke | normal -0.057 | 0.015 |
| 13. Humans taking an alternative evacuation route | binomial 0.03 | 0.97 |

The randomness of the simulations comes from the random number generator’s seed. We save the seed for each simulation so that we can repeat the very same simulation, which is useful for debugging and visualisation.

We register all the random input setups and the corresponding results in the database. We are expecting to research at some point the relationships in these data with data mining or sensitivity analysis.
The final result of Aamks is the compilation of multiple simulations as a set of distributions i.e. F-N curves. The F-N curves were created as in [12]. Figure 9 depicts the exemplary results.

![Fig. 9 The results of evacuation modeling as F-N curves.](image)

3.6 Visualization

In Aamks we use a 2D visualization for supervising the potential user’s faults in his CAD work (e.g. rooms with no doors (Figure 10)), for the final results, and for our internal developing needs. We use a web based technology which allows for displaying both static images and the animations of evacuees.

![Fig. 10 2D visualization: animation of evacuees](image)
We also have a web based 3D visualization made with WebGL Three.js. This subsystem displays realistic animations of humans during their evacuation under fire and smoke. (Figure 11).

**Fig. 11** 3D visualization

### 4 Quality and the performance of a-evac

Below we evaluate the quality of a-evac as described in [9,27] as well as it’s computer performance.

#### 4.1 Verification of a-evac

Verification and validation deals with how close the results of the simulations to the reality are. We took care to be compliant with the general development recommendations [9] by: (1) obeying good programming practices, (2) verifying intermediate simulation outputs, (3) comparing simulation outputs against the analytical results, and (4) creating debugging animations.

The are three types of errors that can be generated by our software: a) error in deterministic modeling of single scenario, b) error of Monte Carlo approximation, c) statistical error – disturbance.

For the first type of error we applied the methods proposed in [27]. The proposed tests are organized in five core components: (1) pre-evacuation time, (2) movement and navigation, (3) exit usage, (4) route availability, and (5) flow conditions/constraints. For each category there are detailed tests for the geometry, the scenario and the expected results. The results are in table 3.

---

*The method is not straightforward*
Table 3 The results of Aamks tests

| Id. | Name of the tests                        | Test code | Results |
|-----|------------------------------------------|-----------|---------|
| 1.  | Pre-evacuation time distributions        | Verif.1.1 | OK      |
| 2.  | Speed in a corridor                      | Verif.2.1 | OK      |
| 3.  | Speed on stairs                          | Verif.2.2 | OK      |
| 4.  | Movement around the corner               | Verif.2.3 | OK      |
| 5.  | Assigned occupant demographics            | Verif.2.4 | OK      |
| 6.  | Reduced visibility vs walking speed      | Verif.2.5 | OK      |
| 7.  | Occupant incapacitation                   | Verif.2.6 | OK      |
| 8.  | Elevator usage                           | Verif.2.7 | –       |
| 9.  | Horizontal counter-flows (rooms)         | Verif.2.8 | OK      |
| 10. | Group behaviours                         | Verif.2.9 | –       |
| 11. | People with movement disabilities        | Verif.2.10| –       |
| 12. | Exit route allocation                     | Verif.3.1 | OK      |
| 13. | Social influence                         | Verif.3.2 | –       |
| 14. | Affiliation                               | Verif.3.3 | –       |
| 15. | Dynamic availability of exits            | Verif.4.1 | OK      |
| 16. | Congestion                               | Verif.5.1 | OK      |
| 17. | Maximum flow rates                       | Verif.5.2 | OK      |

RVO2, the core library of a-evac which drives the local movement, was also evaluated in [33]. The conclusions are that RVO2 is of the quality comparable with the lattice gas and social force models. The social force model is commonly used in a number of evacuation software.

The above is the evaluation of a single, deterministic simulation. However, the final result is the compilation of the whole collection of such single simulations – this is how we get the big picture of the safety of the inquired building. The picture is meant to present risk. Probability of risk is calculated as a share of simulations resulted in fatalities, in the total number of simulations.

The accuracy of this evaluation depends on the method applied – stochastic simulations. The error is proportional to the square root of number simulations. Namely for the discrete Bernoulli probability distributions used for example for evaluation of probability of scenario with fatalities, the error is calculated as follows:

\[
\hat{\sigma}_n = 1.96 \sqrt{\frac{\hat{p}_n (1 - \hat{p}_n)}{n}}; \quad (8)
\]

where: \(\hat{p}_n\) is the probability of fatalities obtained as a number of simulations resulted in fatalities to the total number of simulations, \(n\) is the number of simulations.

We are aware of the third type of error which may be generated by the application. The input for Aamks is a set of various probability distributions what may occasionally generate unreal scenario. For example an evacuee who moves very slowly on corridors and very fast on stairs. In most cases these errors are related to the other parts of Aamks i.e. probabilistic fire modeling.
However, fire environment impacts the evacuation. This error can be evaluated by comparison of data generated by Aamks with real statistics. So far we do not have idea how to tackle this problem efficiently. We consider to evaluate this error by launching simulations for the building stock and check whether we reconstruct historical data. This method is very laborious and not justified at that moment, because our application still lacks some models i.e. fire service intervention – what has significant impact on fire.

4.2 Performance of the Model

The main loop of Aamks processes all agents in the time iteration. Table 4 summarizes how costly the specific calculations for a single agent within a single time iteration are. The tests were performed on computer with Intel Core i5-2500K CPU at 3.30 GHz with 8 GB of RAM.

| Activity        | Time       | Total share |
|-----------------|------------|-------------|
| Position update | $8.12 \times 10^{-6}$ s | 3.97 %      |
| Velocity update | $1.07 \times 10^{-5}$ s | 5.26 %      |
| Speed update    | $8.50 \times 10^{-5}$ s | 41.63 %     |
| State update    | $7.75 \times 10^{-8}$ s | 0.03 %      |
| Goals update    | $9.25 \times 10^{-8}$ s | 0.04 %      |
| FED update      | $1.01 \times 10^{-4}$ s | 48.98 %     |
| Time update     | $1.22 \times 10^{-7}$ s | 0.06 %      |

The total time of a single step of the simulation for one agent is $2 \times 10^{-4}$ s and it grows linearly with the number of agents. The speed and FED calculations are most costly, because they both make database queries against the fire conditions in the compartment. The time step for a-evac iteration is 0.05 s. There is no significant change in fire conditions within this time frame. Therefore for performance optimization we update speed and FED every 20-th step of the simulation.

5 Discussion

Vertical evacuation is troublesome and not implemented. There is RVO2 3D, but it is for aviation where agents can pass above each other – clearly not for our needs. Besides, we think things look actually better in 2D. We like the idea that vertical evacuation can be still considered 2D, just rotated, and we plan to move in this direction.
A-evac does not model the social or group behaviours. However, it is difficult to evaluate, how much the lack of such functionalities impacts the resulting probability distributions.

In the workflow we run a CFAST simulation first. Then a-evac simulation runs on top of CFAST results. This sequential procedure has its drawbacks, e.g. we don’t know how long the CFAST simulation should last to produce enough data for a-evac, so we run ”too much” CFAST for safety. Also, evacuees cannot trigger any events such as opening the door, etc. We considered a closer a-evac-CFAST integration.

There seems to be lot’s of space for improvement in Aamks. We work with the practitioners and know the reality of fire engineering. We know the limitations of our current implementations and most of them can be addressed – there are models and approaches that we can implement and the major obstacle are the limits of our team resources. Therefore we invite everyone interested to join our project at [http://github.com/aamks](http://github.com/aamks).

6 Conclusion

Aamks is actively developed since 2016 and we are truly engaged in making it better. The software, though not really ready for end-users has already served as a support for commercial projects and fire engineers and scientists regard Aamsk as having potential. The stochastic based workflow of Aamks is not a new concept. There are opinions in the community that this approach is how fire engineering should be done. Since no wide-used implementation has been created so far, this is an additional motivation that drives our project.

Acknowledgements Aamks is hosted at github at: [http://github.com/aamks](http://github.com/aamks) This work was supported in part by ConsultRisk LTD and F&K Consulting Engineers LTD under research grants CR-2016 SiMo (Simulation Modules).

References

1. Association of Building Engineers: Part 7: Probabilistic risk assessment. In: PD 7974-6:2004, The application of fire safety engineering principles to fire safety design of buildings. British Standard (2003)
2. Association of Building Engineers: The application of fire safety engineering principles to fire safety design of buildings – Part 6: Human factors: Life safety strategies Occupant evacuation, behaviour and condition (Sub-system 6) (2004). PD 7974-6:2004
3. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pp. 1928–1935. IEEE (2008)
4. Brandes, U., Erlebach, T.: Network analysis: methodological foundations, vol. 3418. Springer Science & Business Media (2005)
5. Bryner, R.G.G.N.P.: Combustion products and their effects on life safety. Fire Protection Handbook pp. 6–11 (2008)
6. Chazelle, B.: A theorem on polygon cutting with applications. In: Foundations of Computer Science, 1982. SFCS’82. 23rd Annual Symposium on, pp. 339–349. IEEE (1982)
7. Christian, R.P., Casella, G.: Monte Carlo statistical methods. New York: Springer (2007)
8. Colaborative work: FRAME Fire Risk Assessment for Engineering. FRAME – Validation. Tech. rep. (2014)
9. Cuesta, A., Abreu, O., Alvear, D.: Evacuation Modeling Trends. Springer (2015)
10. Delaunay, B.: Sur la sphere vide. Bulletin of Academy of Sciences of the USSR pp. 793–800 (1934)
11. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research 17(7), 760–772 (1998)
12. Franzich, H., Nilsson, D.: Uttrymning genom tät rök: beteende och förflyttning. Univ. (2003)
13. Fruin, J.J.: Pedestrian planning and design. Tech. rep. (1971)
14. Gann, Richard G and Averill, Jason D and Butler, KM and Jones, WW and Mulholland, GW and Neviase, JL and Ohlemiller, TJ and Peacock, RD and Reneke, PA and Hall Jr, JR: Sublethal effects of smoke on survival and health. In: Human Behaviour in Fire, Proceedings of the 2nd Int. Symposium, MIT, Cambridge, USA, pp. 285–296 (2001)
15. Helbing, D., Farkas, I.J., Molnar, P., Vicsek, T.: Simulation of pedestrian crowds in normal and evacuation situations. Pedestrian and evacuation dynamics 21(2), 21–58 (2002)
16. Hostikka, S., Keski-Rahkonen, O.: Probabilistic fire simulator – Monte Carlo simulation tool for fire scenarios (2002)
17. Hull, T.R., Stec, A.A., Paul, K.: Hydrogen chloride in fires. Fire Safety Science 9, 665–676 (2008)
18. Hurley, M.J., Gottuk, D.T., Hall Jr, J.R., Harada, K., Kuligowski, E.D., Puchovsky, M., Watts Jr, J.M., Wieczorek, C.J., et al.: SFPE handbook of fire protection engineering. Springer (2015)
19. Jin, T.: Visibility through fire smoke in main reports on production, movement and control of smoke in buildings. Japanese Association of Fire Science and Engineering pp. 100–153 (1974)
20. Jin, T.: Visibility and human behavior in fire smoke. The SFPE Handbook of Fire Protection Engineering, ed. PJ DiNenno, D. Drysdale, CL Beyler, WD Walton, RLP Custer, JR Hall, Jr. and JM Watts, Jr., 3rd edition, Society of Fire Protection Engineers/National Fire Protection Association, Quincy, Massachusetts (2016)
21. Jones, W.W., Forney, G.P., Peacock, R.D., Reneke, P.A.: A technical reference for cfast: an engineering tool for estimating fire and smoke transport. NIST TN 1431 (2017). URL https://github.com/firemodels/cfast/releases
22. Kahneman, D., Klein, G.: Conditions for intuitive expertise: a failure to disagree. American Psychologist 64(6), 515 (2009)
23. Korhonen, T.: FDS + Evac – Technical reference + users guide. VTT Technical Research Centre of Finland (2016). Evac 2.5.2
24. Peacock, R.D., Jones, W., Reneke, P., Forney, G.: CFAST-consolidated model of fire growth and smoke transport (version 7) user’s guide. Citeseer (2017). URL https://github.com/firemodels/cfast/releases
25. Predtetschenski, W., Milinski, A.: Planning for foot traffic in buildings (1978)
26. Purser, D.A.: Toxicity assessment of combustion products. SFPE handbook of fire protection engineering 3, 2–6 (2002)
27. Ronchi, E., Kuligowski, E.D., Reneke, P.A., Peacock, R.D., Nilsson, D.: The process of verification and validation of building fire evacuation models. NIST Technical Note 1822 (2013)
28. Speitel, L.C.: Toxicity assessment of combustion gases and development of a survival model. Tech. rep., DTIC Document (1995)
29. Stec, A.A., Hull, T.R.: Fire toxicity. Elsevier (2010)
30. The Ministry of Business, Innovation and Employment: C/VM2 Verification Method: Framework for Fire Safety Design (2013). New Zealand
31. Tversky, A., Kahneman, D.: Judgment under uncertainty: Heuristics and biases. Science 185(4157), 1124–1131 (1974)
32. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: In: Robotics research, pp. 3–19. Springer (2011)
33. Viswanathan, V., Lee, C.E., Lees, M.H., Cheong, S.A., Sloot, P.: Quantitative comparison between crowd models for evacuation planning and evaluation. arXiv preprint arXiv:1401.0366 (2014)