Markov Aggregation for Speeding Up Agent-Based Movement Simulations

Bernhard C. Geiger
Know-Center GmbH
Graz, Austria
geiger@ieee.org

Alireza Jahani
Brunel University London
Uxbridge, UK

Hussain Hussain
Institute for Interactive Systems and Data Science, Graz University of Technology
Graz, Austria

Derek Groen
Brunel University London
Uxbridge, UK

ABSTRACT

In this work, we investigate Markov aggregation for agent-based models (ABMs). Specifically, if the ABM models agent movements on a graph, if its ruleset satisfies certain assumptions, and if the aim is to simulate aggregate statistics such as vertex populations, then the ABM can be replaced by a Markov chain on a comparably small state space. This equivalence between a function of the ABM and a smaller Markov chain allows to reduce the computational complexity of the agent-based simulation from being linear in the number of agents, to being constant in the number of agents and polynomial in the number of locations.

We instantiate our theory for a recent ABM for forced migration (*Flee*). We show that, even though the rulesets of *Flee* violate some of our necessary assumptions, the aggregated Markov chain-based model, *MarkovFlee*, achieves comparable accuracy at substantially reduced computational cost. Thus, *MarkovFlee* can help NGOs and policy makers forecast forced migration in certain conflict scenarios in a cost-effective manner, contributing to fast and efficient delivery of humanitarian relief.

KEYWORDS

Agent-Based Model, Markov Chains, Model Reduction, Social Simulation

ACM Reference Format:
Bernhard C. Geiger, Alireza Jahani, Hussain Hussain, and Derek Groen. 2023. Markov Aggregation for Speeding Up Agent-Based Movement Simulations. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

To model the dynamics of complex systems of autonomous and interacting agents, agent-based models (ABMs) are an effective approach. It enables the observation of the collective effects of agent behaviour and interaction [22], and is well-suited for modelling sociological and psychological behaviour and the interactions of humans with each other and their environment [28]. Therefore, ABMs can be applied across a wide variety of domains and disciplines [14]. Capable of eliciting an understanding of the behaviour of moving people, the ABMs have been used to develop and implement movement simulations [1, 4, 21, 28]. A major drawback, however, is the computational complexity of ABMs, which typically increases with the number of agents and the complexity of agent interactions, and limits their applicability in certain large-scale application scenarios.

To remedy this shortcoming, several approaches have been proposed to reduce the computational complexity of simulating ABMs, including parallelization [6], distributed computation [26], and surrogate models, which replace the ABM by machine learning models that predict aggregate statistics of ABM simulation results, e.g., [2]. For settings in which agent interactions are determined by (static or dynamic) social graphs (such as in epidemic simulations), distributed computation of ABM simulations can further be sped up by allocating agents to processes according to their community membership [26]. That ABMs are Markov chains (MCs) on large state spaces has been utilized in [3, 20], where the authors proposed to *aggregate* this MC to a chain on a much smaller state space, and discussed in which situations the thus obtained computationally efficient model yields results that are equivalent to the ABM.

While the works of [3, 20] focus on voter models and similar ABMs, Markov aggregation has not been applied to agent-based movement models, to the best of our knowledge: While [10] reduced the ABM to a MC on a small state space (see Section 2), their resulting model can only be used to simulate average movement behavior, i.e., is deterministic and therefore does not allow for uncertainty quantification.

We complement the existing literature by applying Markov aggregation to agent-based movement models. We investigate the setting in which N agents move independently from each other on a graph with L vertices (Section 3). We show that, if movement probabilities do not depend on agent identities but only on properties of locations, the sequence of vertex populations derived from the ABM is a MC (Section 4). This MC allows for computationally more efficient simulation than the original ABM, while yielding a mathematically equivalent model for the vertex populations.

Using these insights, in Section 5 we instantiate our theory for the ABM for forced migration (*Flee*) proposed in [30, 31]. By deviating only slightly from the ruleset of *Flee*, we show in Sections 6 and 7 that Markov aggregation (*MarkovFlee*) can provide a substantial speed-up with little or no cost of accuracy in both synthetic and realistic settings. Such a speed-up is particularly attractive for

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

NGOs, which rely on accurate migration forecast simulations to provide humanitarian relief. We believe that this work is thus not only interesting theoretically, but that it also contributes to coping with real-world conflict scenarios.

2 RELATED WORK

ABMs are used to simulate different types of movement, including route configuration [9, 12, 27] and route choice [5, 7, 13, 24, 25] in pedestrian movement, traffic flows [23], bicycle traffic [15], and forced displacement [4, 19, 21, 29]. Gilbert et al. [8] and Suleimenova et al. [32] studied effects of policy decisions in human movement simulations to provide insights for governments, stakeholders, and policymakers. Searle et al. [28] designed an ABM to simulate conflicts and decisions behind the movement of refugees from the affected areas. Jahani et. al. [11] proposed and analyzed a weather-coupled multi-scale ABM for South Sudanese refugee movement.

A work related to ours is the paper of Huang and Unwin [10]. There, the authors replaced the ABM of [31] by a MC on a smaller state space. Indeed, the authors obtain a probabilistic model conceptually similar to ours (for which they also recursively compute journey probabilities, cf. Section 5 and [10]). For N agents migrating on a graph with L locations during T days and with a maximum daily distance of $d_{\max}$, the authors find that their approach has a runtime complexity of $O(L^3 + L^4 d_{\max} + L^2 T)$, compared to $O(LNT)$ of the original ABM. However, while we aim at simulating this reduced MC to sample vertex population trajectories, the authors instead used the probabilistic model to obtain the sequence of expected vertex populations. Thus, their approach results in a deterministic model and does not allow uncertainty quantification.

Another branch of the literature connected to our work is [3, 20]. There, the authors investigated opinion dynamics on graphs, e.g., voter models, and investigated settings in which the corresponding ABMs can be simplified. Specifically, Banisch showed that there exist conditions such that the *opinion frequencies*, i.e., the number of agents with a given opinion, form a MC; these conditions are rare [3, Sec. 5.7]. Thus, the authors propose aggregations to MCs on a state space with finer granularity than the opinion frequencies would require, and quantify discrepancies with information-theoretic quantities. Similarly, Lamarche-Perrin et al. [20] investigate information-theoretic formulations to find the optimal aggregate statistics of an ABM at time t to best predict a desired aggregate statistic at time $t + T$. They show that, if agent behavior is homogeneous, then aggregate statistics derived from the ABM have similar predictive performance for a future aggregate state, while having lower complexity [20, Fig. 8 & 9].

3 SETTING AND NOTATION

We consider an ABM with N agents moving on a graph $\mathcal{G} = (V, E)$, where $V = \{1, \dots, L\}$ is the set of vertices and E is the set of (undirected or directed) edges. The *state* of agent i at time t is the vertex at which it resides at time t and shall be denoted by the random variable (RV) $X_{i,t}$. Specifically, if agent i is at vertex ℓ at time t, then we write $X_{i,t} = \ell$. At each time t, agents select their next vertex according to a (potentially probabilistic) ruleset. We call the collection $X_t = (X_{1,t}, \dots, X_{N,t})$ of agent states the *world state*.[1] The world state evolves according to a MC $\{X_t,\ t \in \mathbb{N}_0\}$, with transition probabilities determined by the agent ruleset and (potentially) by vertex metadata. Since the transition probabilities thus may be different for each t, the MC is non-homogeneous. The state space $\mathcal{X} = V^N$ of the world has cardinality L^N. We denote a realization of states by lower-case letters, e.g., $X_t = x := (x_1, \dots, x_N)$.

Depending on the scenario under investigation, the resulting ABM is used to determine some statistics derived from the world state, e.g., the numbers (but not the identities) of agents at a given vertex or the number of agents in some induced subgraph of $\mathcal{G}$. The quantity of interest at time t, denoted by Y_t, is then a function of X_t. Specifically, let $f\colon \mathcal{X} \to \mathcal{Y}$ be a function from the state space of the world to a smaller set and let the process $\{Y_t,\ t \in \mathbb{N}_0\}$ be defined by $Y_t = f(X_t)$ for all t. A function of a MC need not possess the Markov property in general; the situation in which it does is known under the concept of *lumpability* [17, §6.3]. Indeed, a sufficient condition for $\{Y_t,\ t \in \mathbb{N}_0\}$ to be a non-homogeneous MC is that [16]

$$\mathbb{P}(Y_{t+1} = y | X_t = x) = \mathbb{P}(Y_{t+1} = y | Y_t = f(x)) \tag{1}$$

holds for all $y \in \mathcal{Y}$, $x \in \mathcal{X}$, and all $t \in \mathbb{N}_0$. In this situation, the MC $\{Y_t,\ t \in \mathbb{N}_0\}$ is *probabilistically equivalent* to the function of the MC $\{X_t,\ t \in \mathbb{N}_0\}$. In other words, rather than simulating the ABM, it suffices to create a trajectory of the MC $\{Y_t,\ t \in \mathbb{N}_0\}$, which typically lives on a much smaller state space.

4 VERTEX POPULATIONS ARE MARKOV

We now investigate a specific statistic derived from the world state, namely, the sequence of *vertex populations*, i.e., the number of agents that reside at each vertex. Specifically, let $Y_{\ell,t}$ be the number of agents in location ℓ at time t, i.e.,

$$Y_{\ell,t} = f_\ell(X_t) := \sum_{i=1}^{N_t} \mathbb{I}(X_{i,t} = \ell) \tag{2}$$

where $\mathbb{I}(A) = 1$ if and only if A is true. The process $\{Y_t,\ t \in \mathbb{N}_0\}$ defined by $Y_t = (Y_{1,t}, \dots, Y_{L,t})$ is a function of a MC. For the sake of brevity, we define the function $f := (f_1, \dots, f_L)$, i.e., $Y_t = f(X_t)$.

We now show that the sequence of vertex populations is Markov given that the following two assumptions hold.

Assumption 1. *Agents move independently from one another, i.e.,*

$$\forall x, x' \in \mathcal{X}\colon \qquad \mathbb{P}(X_{t+1} = x | X_t = x') = \prod_{n=1}^{N_{t+1}} p_n(x_n | x'). \tag{3}$$

Assumption 2. *The movement probabilities of all agents are identical and influenced only by vertex populations, i.e.,*

$$\forall x, x' \in \mathcal{X}, \forall n\colon \qquad p_n(x_n | x') = p(x_n | x'_n, f(x')). \tag{4}$$

Both assumptions are common in ABMs While Assumption 1 allows different movement probabilities of different agents, Assumption 2 requires that all agents have the same movement model. Thus, while Assumption 1 excludes models in which agent behavior is coupled (e.g., via an underlying social network), Assumption 2

[1] Simulations with changing agent numbers, i.e., where $N = N_t$, can be accomplished by creating a separate location that acts as a *reservoir* form which agents can be spawned or to which agents can resign.

excludes agent rule sets that depend on internal agent states (but see also Section 8 for potential generalizations). With these assumptions, we are ready to present the following theorem.

Theorem 1. *Under Assumptions 1 and 2 and with f defined in* (2)*, the sequence of vertex populations $\{Y_t\}$ satisfies the Markov property.*

The proof relies on [3, Th. 3.2] and the fact that, given the two assumptions, neither $\mathbb{P}(X_{t+1} = x|X_t = x')$ nor the vertex populations change under permutations of agent identities. To keep the paper self-contained, we carry out the proof in detail.

Proof. It suffices to prove that (1) holds. Let $f^{-1}(y) := \{x \in \mathcal{X}\colon \forall \ell \in V\colon \sum_{i=1}^{N} \mathbb{I}(x_i = \ell) = y_\ell\}$ denote the preimage of y under f and suppose that at time t the world configuration is x'. Then, with (3) and (4) we obtain

$$\begin{aligned}
&\mathbb{P}(Y_{t+1} = y|X_t = x') \\
&= \sum_{x \in f^{-1}(y)} \prod_{n=1}^{N} p(x_n|x'_n, f(x')) && (5)\\
&= \sum_{x \in f^{-1}(y)} \prod_{\ell'=1}^{L} \prod_{\ell=1}^{L} \prod_{n\colon x'_n=\ell', x_n=\ell} p(x_n|x'_n, f(x')) && (6)\\
&= \sum_{x \in f^{-1}(y)} \prod_{\ell'=1}^{L} \prod_{\ell=1}^{L} p(\ell|\ell', f(x'))^{|\{n\colon x'_n=\ell', x_n=\ell\}|}. && (7)
\end{aligned}$$

The first sum in the last line contains the element $x^\bullet$ such that the first y_1 elements of $x^\bullet$ are equal to 1, the next y_2 elements of $x^\bullet$ are equal to 2, etc, i.e.,

$$x^\bullet = (1, 1, \ldots, 1, 2, 2, \ldots, L, L, \ldots, L). \quad (8)$$

All other elements in this sum are *multiset permutations* of $x^\bullet$. Letting $\Pi(y)$ denote the set of multiset permutations, we have $f((x^\bullet_{\pi(1)}, \ldots, x^\bullet_{\pi(N)})) = y$ for every $\pi \in \Pi(y)$. Thus,

$$\begin{aligned}
&\mathbb{P}(Y_{t+1} = y|X_t = x') \\
&\quad = \sum_{\pi \in \Pi(y)} \prod_{\ell'=1}^{L} \prod_{\ell=1}^{L} p(\ell|\ell', f(x'))^{|\{n\colon x'_n=\ell', x^\bullet_{\pi(n)}=\ell\}|}. && (9)
\end{aligned}$$

Suppose further that $y' = f(x')$ and that x'' is an arbitrary world state such that $y' = f(x'')$, i.e., x'' and x' are world states resulting in the same vertex population. Evidently, x'' is obtained by permuting the agent indices of x'; let $\pi_\circ$ denote the corresponding permutation, i.e., $x''_n = x'_{\pi_\circ(n)}$. Then,

$$\begin{aligned}
&\mathbb{P}(Y_{t+1} = y|X_t = x'') \\
&= \sum_{\pi \in \Pi(y)} \prod_{\ell'=1}^{L} \prod_{\ell=1}^{L} p(\ell|\ell', f(x'))^{|\{n\colon x'_{\pi_\circ(n)}=\ell', x^\bullet_{\pi(n)}=\ell\}|} && (10)\\
&= \sum_{\pi \in \Pi(y)} \prod_{\ell'=1}^{L} \prod_{\ell=1}^{L} p(\ell|\ell', f(x'))^{|\{n\colon x'_n=\ell', x^\bullet_{\pi_\circ^{-1}(\pi(n))}=\ell\}|}. && (11)
\end{aligned}$$

Since $\pi_\circ$ is a permutation, so is $\pi_\circ^{-1}$. Further, the set of all multiset permutations is closed under permutations (any unique assignment of the labels in (8) must be achievable by applying some $\pi \in \Pi(y)$), thus $\pi_\circ^{-1}\Pi(y) = \Pi(y)$. Using this in (11) and comparing the result to (7) yields $\mathbb{P}(Y_{t+1} = y|X_t = x') = \mathbb{P}(Y_{t+1} = y|X_t = x'')$. Since this holds for every pair x', x'' of elements from the preimage $f^{-1}(y')$, we have $\mathbb{P}(Y_{t+1} = y|X_t = x) = \mathbb{P}(Y_{t+1} = y|Y_t = f(x))$, i.e., we have (1). This completes the proof. □

Given Theorem 1, one can thus simulate how vertex populations change by simulating $\{Y_t\}$ directly instead of simulating the ABM. The result of this direct simulation is probabilistically equivalent, i.e., the distribution of several random rollouts of $\{Y_t\}$ coincides with the distribution of the vertex populations from several runs of the ABM — the ABM does not provide any additional useful information for the computation of the vertex populations than what a direct simulation of $\{Y_t\}$ does.

We close this section by providing a closed form for $\mathbb{P}(Y_{t+1} = y|Y_t = y')$. To this end, suppose that at iteration t there are $y'_{\ell'}$ agents at vertex ℓ'. Since these agents all decide for their next vertex independently (Assumption 1) and homogeneously (Assumption 2), the number of agents that move from vertex ℓ' to any other vertex is given by a multinomial distribution with $y'_{\ell'}$ attempts and with probabilities given by $p(\ell|\ell', y')$. Specifically, letting $n_{\ell'\to\ell}$ denote the number of agents that move from vertex ℓ' to vertex ℓ, we have that the probability for the vector $\underline{n}_{\ell'\to} = (n_{\ell'\to 1}, \ldots, n_{\ell'\to L})$ is given by

$$p(\underline{n}_{\ell'\to}) = \binom{y'_{\ell'}}{\underline{n}_{\ell'\to}} \prod_{\ell=1}^{L} p(\ell|\ell', y')^{n_{\ell'\to\ell}} \quad (12)$$

if $\sum_{\ell=1}^{L} n_{\ell'\to\ell} = y'_{\ell'}$ and zero otherwise, and where

$$\binom{y'_{\ell'}}{\underline{n}_{\ell'\to}} = \frac{y'_{\ell'}!}{n_{\ell'\to 1}! \cdot \cdots \cdot n_{\ell'\to L}!} \quad (13)$$

is the multinomial coefficient. From this follows that the MC of vertex populations has a transition probability matrix defined by

$$\mathbb{P}(Y_{t+1} = y|Y_t = y') = \sum_{\underline{n}_{\ell'\to}} \prod_{\ell'=1}^{L} \binom{y'_{\ell'}}{\underline{n}_{\ell'\to}} \prod_{\ell=1}^{L} p(\ell|\ell', y')^{n_{\ell'\to\ell}} \quad (14)$$

where the sum runs over all $\underline{n}_{\ell'\to}$ such that $\sum_{\ell=1}^{L} n_{\ell'\to\ell} = y'_{\ell'}$ and $\sum_{\ell'=1}^{L} n_{\ell'\to\ell} = y_\ell$ hold.

5 INSTANTIATION FOR MIGRATION SIMULATION

We now instantiate the previous results for the agent-based refugee movement model *Flee* proposed in [30, 31].

5.1 Description of *Flee*

Flee is an agent-based simulation toolkit tailored for simulating the movement of individuals across geographical locations. In *Flee*, agents move on a weighted graph $\mathcal{G} = (V, E, W)$, where each vertex in V corresponds to a geographic location of interest (city, settlement, conflict region, humanitarian camp, etc.), and each edge $(\ell, k) \in E$ has a weight $w_{\ell,k} \in W$ corresponding to the distance between locations ℓ and k. The total number of agents changes over time as conflicts evolve, i.e., $N = N_t$. Each agent represents a forcibly displaced person that, in each time step, decides whether to stay at its current or move to a neighbouring location, attempting to reach a safety zone (i.e., camps).

Specifically, let $\rho(\ell, t)$ denote the probability that an agent leaves location ℓ at iteration t, and let the probability of moving to location

$$\mathbb{P}(X_{i,t+1} = \ell | X_t = x')$$
$$= \sum_{(\ell_0,\ell_1,\ldots,\ell_n)\in\overline{\mathcal{R}}_{\ell',\ell}} \prod_{m=1}^{n-1} \frac{\rho(\ell_m,t)}{Z_{\ell_m}} \frac{\phi(\ell_{m+1},t)}{w_{\ell_m,\ell_{m+1}}} \kappa(\ell_{m+1}, f_{\ell_{m+1}}(x')) + \sum_{(\ell_0,\ell_1,\ldots,\ell_n)\in\underline{\mathcal{R}}_{\ell',\ell}} (1-\rho(\ell_n,t)) \prod_{m=1}^{n-1} \frac{\rho(\ell_m,t)}{Z_{\ell_m}} \frac{\phi(\ell_{m+1},t)}{w_{\ell_m,\ell_{m+1}}} \kappa(\ell_{m+1}, f_{\ell_{m+1}}(x')) \tag{15}$$

ℓ from one of its neighbors be proportional to its attractivity score $\phi(\ell, t)$. Further, let $\kappa(\ell, y_\ell)$ be a scaling factor that depends on the vertex population y_ℓ of location ℓ – concretely, if ℓ is a humanitarian camp, then $\kappa(\ell, y_\ell)$ is 1 if its vertex population is below 90% of its capacity, 0 if it is above capacity, and decreases smoothly in between. At the beginning of iteration t, each agent has a travel budget of $B = d_{\max}$. Each agent i then performs the following steps in sequence (see also Algorithm 1): i) it decides whether to change its current location $x_{i,t}$ depending on $\rho(x_{i,t}, t)$ (line 11); if it decides not to move, then $x_{i,t+1} = x_{i,t}$; ii) it picks its next destination ℓ, which is a neighbor of $x_{i,t}$ in $\mathcal{G}$, with a probability proportional to $\frac{\phi(\ell,t)}{w_{x_{n,t},\ell}}\kappa(\ell, f_\ell(x_t))$ (line 15); iii) if $w_{x_{i,t},\ell} > B$, then the agent stays on the route to ℓ and can only complete its journey in one of the next iterations; otherwise the agent starts again at step i) with $x_{i,t} = \ell$ and $B = B - w_{x_{i,t},\ell}$ (line 20). The simulation runs for T time steps.

In the second version of the *Flee* ruleset (*FleeV2*), some rules have been revised, enabling travelling on foot in case of blocked or waylaid roads and lack of resources. Further, the movement speed of agents has been changed, a new mechanism has been developed to prevent agents pausing at the start of new trip, and the weights of camp and conflict locations have been revised.

Computational Complexity Analysis. To evaluate the computational complexity of one iteration of *Flee*, let $h := d_{\max}/\min_{\ell,k} w_{\ell,k}$ denote the maximum number of edges an agent can traverse before its travel budget is exhausted. Therefore, while the outer loop in Algorithm 1 runs over N agents, the inner loop runs over at most h iterations. In practice, the number of iterations of the inner loop will be much smaller, because the agents may choose to stay at a location without exhausting their budget. Therefore, the worst-case runtime complexity of a single iteration is $O(Nh)$.

Algorithm 1 Flee [31] (abbridged)

```
1:  Initialize agent ecosystem
2:  t ← 0
3:  while t < T do
4:      t ← t + 1
5:      for all agents n do
6:          Travel budget B ← d_max
7:          if Agent n is on a route then
8:              Move agent towards destination ℓ and adjust B
9:          end if
10:         while B > 0 do
11:             Choose whether to move based on ρ(x_{n,t}, t)
12:             if chose to stay then
13:                 break
14:             end if
15:             Pick destination ℓ based on (φ(ℓ,t)/w_{x_{n,t},ℓ}) κ(ℓ, f_ℓ(x_t))
16:             B ← B − w_{x_{n,t},ℓ}
17:             if B < 0 then
18:                 Assign agent to route
19:             else
20:                 x_{n,t} ← ℓ
21:             end if
22:         end while
23:     end for
24: end while
```

5.2 Markov Aggregation of *Flee*: *MarkovFlee*

To convert the agent ruleset of *Flee* to a probabilistic model that is compatible with Assumption 2, we assume that agents do not reside on routes, but always terminate their journey at a location in the graph $\mathcal{G}$. Thus, we either have to allow that some agents exceed their daily movement budget in order to arrive at their next destination, or that agents cannot fully exploit their movement budget. Both options exhibit similar performance if the distances in $\mathcal{G}$ are small in comparison to $d_{\max}$. In this work, we choose the former option.

Specifically, let $P(\ell', \ell) \subset V^*$ denote all sequences $(\ell_0, \ell_1, \ldots, \ell_n)$ of vertices with initial vertex $\ell_0 = \ell'$ and last vertex $\ell_n = \ell$. Then, we have

$$\underline{\mathcal{R}}_{\ell',\ell} := \left\{ (\ell_0,\ell_1,\ldots,\ell_n) \in P(\ell',\ell) \colon \sum_{m=1}^{n} w_{\ell_{m-1},\ell_m} < d_{\max} \right\} \tag{16}$$

and

$$\overline{\mathcal{R}}_{\ell',\ell} := \left\{ (\ell_0,\ell_1,\ldots,\ell_n) \in P(\ell',\ell) \colon \sum_{m=1}^{n-1} w_{\ell_{m-1},\ell_m} < d_{\max}, \sum_{m=1}^{n} w_{\ell_{m-1},\ell_m} \geq d_{\max} \right\} \tag{17}$$

for the set of routes starting at ℓ' and terminating at ℓ, having a total length less than $d_{\max}$ and exceeding $d_{\max}$ only at the last leg of the route, respectively. For example, for the toy graph in Fig. 1 and $d_{\max} = 250$, we have $\underline{\mathcal{R}}_{A,C} = \{(A,C),(A,B,C)\}$ and $\overline{\mathcal{R}}_{A,C} = \{(A,B,A,C),(A,B,A,B,C)\}$.

Now suppose that the world state at iteration t is x' and that agent i is at location ℓ', i.e., $X_{i,t} = \ell'$. Then, the probability that agent i moves to location ℓ at iteration $t+1$ is given by (15) at the top of the previous page, where Z_{ℓ_m} is a normalization constant

that ensures that the term $\frac{\phi(\ell_{m+1},t)}{w_{\ell_m,\ell_{m+1}}}\kappa(\ell_{m+1}, y'_{\ell_{m+1}})$ sums to one over the neighbors of ℓ_m in $\mathcal{G}$.

In our code, we use a recursion to compute the probabilities in (15) for all pairs of vertices once at the beginning of each simulation, and update them whenever the change in camp utilization y_ℓ between iteration t and $t+1$ induces a change in the scaling factor $\kappa(\ell, y_\ell)$. Repeatedly performing these updates slows down *MarkovFlee*, as we show in the experiments in Sections 6.2. To improve the computational efficiency of these updates, we only update the probabilities for pairs of vertices for which at least one path passes through a camp with changed scaling factor. To further improve the efficiency of the recursion, we use dynamic programming, i.e., we store the results of each recursive state once computed and reuse them when we encounter the same state later. This avoids computing probabilities that are already computed, which may happen in the recursive approach due to cycles in the graph $\mathcal{G}$.

As it is evident from (15), the dependence of the movement of agent i on the past world state simplifies as

$$\begin{aligned}&\mathbb{P}(X_{i,t+1} = \ell | X_t = x') \\ &\quad = \mathbb{P}(X_{i,t+1} = \ell | X_{i,t} = \ell', Y_t = f(x')) =: p(\ell|\ell', f(x')) \quad (18)\end{aligned}$$

i.e., Assumption 2 is fulfilled. Together with Assumption 1, Theorem 1, (12), and (15) we can thus formulate our MC-based simulation in Algorithm 2. We henceforth refer to this algorithm as *MarkovFlee*.

Algorithm 2 *MarkovFlee* (this work)

1: Initialize simulation ecosystem
2: Compute travel probabilities $p(\ell|\ell', f(x'))$ by (15)
3: $t \leftarrow 0$
4: **while** $t < T$ **do**
5: $t \leftarrow t + 1$
6: **for** all locations ℓ' **do**
7: Sample agent reassignments $\underline{n}_{\ell' \rightarrow}$ according to (12)
8: **end for**
9: **for** all locations ℓ **do**
10: update population by $y_\ell \leftarrow \sum_{\ell'} \underline{n}_{\ell' \rightarrow \ell}$
11: **if** Population y_ℓ induces a change in $\kappa(\ell, y_\ell)$ **then**
12: Update travel probabilities for all affected routes
13: **end if**
14: **end for**
15: **end while**

Remark 1. *The main difference between Algorithm 1 and Algorithm 2 is that the former iterates over all agents, while the latter iterates over all locations. Thus, in* MarkovFlee*, agents move jointly, while in Flee they move individually. In Flee, as individual agents enter a camp ℓ, the scaling factor $\kappa(\ell, y_\ell)$ decreases, which makes the camp less attractive for agents that are iterated subsequently. In* MarkovFlee*, in contrast, the fact that agents move jointly may lead to camp overutilization, i.e., more agents arrive at the camp location ℓ than the camp can handle.*

Computational Complexity Analysis. We now evaluate the computational complexity of a single iteration of *MarkovFlee*. Sampling agent reassignments (line 7) and updating vertex populations (line 10) have a complexity of $O(L|\mathcal{N}_{\max}|)$, where $|\mathcal{N}_{\max}| \leq L$ is maximum number of locations that can be reached from an arbitrary location with a budget of $d_{\max}$. We remain to analyze the complexity of computing travel probabilities (15). For each location $\ell \in V$, the recursive implementation explores travels to locations in the neighborhood $\mathcal{N}(\ell)$ of ℓ in $\mathcal{G}$. With $|\mathcal{N}(\ell)| \leq L$ and h being the the maximum number of edges an agent can traverse with a budget of $d_{\max}$, the worst-case runtime complexity of *MarkovFlee* is $O(L^h)$. Since h is a very crude upper bound, the worst-case runtime complexity will be smaller for most realistic graphs.

The dynamic programming implementation of one iteration of *MarkovFlee* has the same worst-case runtime complexity of $O(L^h)$. However, this worst-case scenario is only attainable when no two paths have the same total distance, which is impossible in practice due to cycles. In particular, if we have a path with distance d that has c non-identical (possibly nested) cycles, the same budget $B = d_{\max} - d$ is obtained regardless of the direction in which the cycles are traversed. This effectively reduces the number of unique location/budget tuples that the dynamic programming implementation has to consider.

Remark 2. *Note that h and $|\mathcal{N}_{\max}|$ depends on both the agent ruleset (via $d_{\max}$) and the graph (via $w_{\ell,k}$). While for large geographic regions these numbers may increase if the number L of locations is increased, in practice increasing L will at some point not lead to a decrease of $w_{\ell,k}$, as routes below a certain minimum distance will not lead to more accurate simulation results. Therefore, h and $|\mathcal{N}_{\max}|$ can be considered constant, or at most $o(L)$.*

6 SYNTHETIC EXPERIMENTS

We performed of experiments on two synthetic toy examples (Sections 6.1 and 6.2) to illustrate and validate our approach. The code for our experiments is fully backwards compatible with Flee and is available online[2]. All synthetic experiments were run on a virtual machine with 250 GB RAM and 28 Intel® Xeon® Gold 6248 CPUs with 2.50 GHz.

6.1 Toy Example on a Small Graph

The first example considers movements on a small graph with four vertices, as depicted in Figure 1. Initial agent populations are set to $Y_0 = (500\gamma, 0, 0, 0)$, and in each iteration $\gamma \cdot 5500/365$ agents are spawned at vertex A, where $\gamma \in \{1, 2, 5, 10, 20, 50, 100, 200, 1000\}$. We set $d_{\max} = 250$. The two camps C and D have unlimited capacities, i.e., we have $\kappa(C, y_C) = \kappa(D, y_D) \equiv 1$. The probabilities that agents leave a location are static, i.e., $\rho(\ell, t) = \rho(\ell)$ for all ℓ. We ran both *Flee* and *MarkovFlee* for $T = 100$ iterations and recorded the vertex populations for all four locations. To account for random effects, 50 independent simulations were run. Figure 1 displays the means and standard deviations of vertex populations and shows good agreement between *Flee* and *MarkovFlee*. Differences seen especially early during simulation (small t) can be attributed to the fact that agents may reside on routes for *Flee*, but must be assigned to vertices during each iteration for *MarkovFlee*. This difference continues to affect vertex populations throughout the simulation, giving the appearance that the numbers of *Flee* "lag behind" those of *MarkovFlee*.

[2] https://github.com/bcgeiger/MarkovFlee

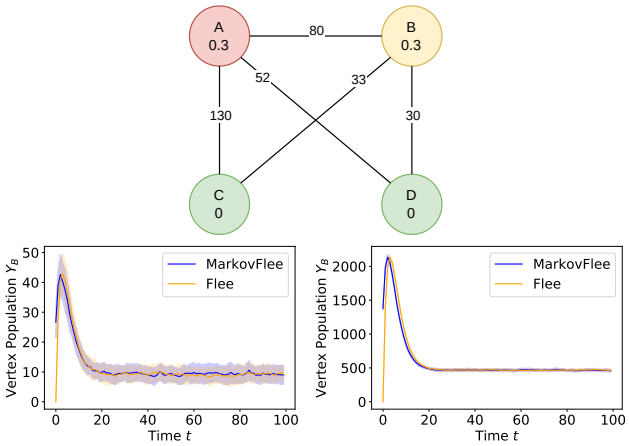

Figure 1: Top: Toy example with a single conflict site (A) and two camps (C,D) with unlimited capacity. Edge labels correspond to route distances $w_{\ell,\ell'}$, values inside vertices denote the probability that agents leave the vertex $\rho(\ell, t)$. Bottom: Population (means and standard deviations) at vertex B computed by *Flee* and *MarkovFlee* for $\gamma = 2$ (left) and $\gamma = 100$ (right).

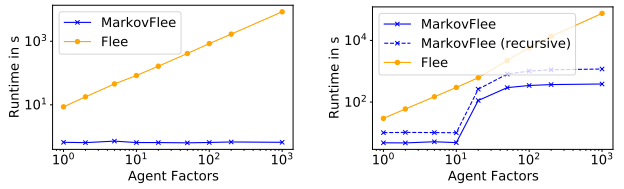

Figure 2: Runtime in seconds for *Flee* and *MarkovFlee* in the toy example (left) and the synthetic conflict scenario (right).

We present the runtime complexity of *Flee* and *MarkovFlee* in Figure 2. While, as anticipated, the runtime of *Flee* increases linearly with the number of agents (or, equivalently, with the scaling factor γ), the runtime of *MarkovFlee* appears to be independent of the number of agents. While we believe that drawing samples from (12) should depend mildly on the vertex populations $Y_{\ell,t}$, apparently this is not a bottleneck for the current implementation.

6.2 Synthetic Conflict Scenario with Limited Camp Capacities

The second example is a synthetic conflict scenario with a single conflict site (at which agents are spawned with the same rule as in Section 6.1) and two camps. The camps have limited capacity, i.e., $\kappa(E, y_E)$ and $\kappa(F, y_F)$ depend on y_E and y_F as indicated in Section 5. As a consequence, the travel probabilities in (15) must be updated whenever the camp utilization approaches capacity. We still have $\rho(\ell, t) = \rho(\ell)$ for all ℓ and $d_{\max} = 250$. We again ran *Flee* and *MarkovFlee* for $T = 100$ iterations, with 25 restarts to account for random effects. The results in Figure 3 indicate a good agreement between *Flee* and *MarkovFlee*. For larger values of γ, we see larger deviations between *Flee* and *MarkovFlee* for the camp populations. Specifically, we see that the camp populations regularly overshoot capacity, followed by a phase during which agents leave the camps (with a probability of $\rho(E) = \rho(F) = 0.01$).

This behavior is related to our observations in Remark 1 and to the fact that in *MarkovFlee* agents move jointly, while in *Flee* they move individually. At some iteration t, let $Y_{D,t} = 100000$ and $Y_{E,t} = Y_{F,t} = 0$. Since camps can be considered more attractive than other locations, i.e., $\phi(E, t), \phi(F, t) > \phi(B, t) > \phi(A, t)$, we can assume that *MarkovFlee* will distribute the 100000 agents at D over the camps E and F, exceeding their capacities. Since then $\kappa(E, y_E) = \kappa(F, y_F) = 0$, no more agents will travel to these camps, until capacity has been freed by agents leaving the camps. *Flee*, in contrast, will assign agents individually and recomputes κ after each agent movement. Thus, as soon as camp capacities are reached, agents will stop moving from D to E or F, but will rather move to location B instead.

The fact that camps have finite capacity and that, thus, the factors $\kappa(E, y_E)$ and $\kappa(F, y_F)$ need to be recomputed regularly also has effects on the runtime of *MarkovFlee*. While Flee still scales linearly in the number of agents, we now see that the repeated computation of (15) increases the runtime of *MarkovFlee* as γ increases from 10 to 100 (see Figure 2). After γ has reached 100, the travel probabilities must be updated after each iteration. Since these updates do not depend on the number of agents, the overhead remains constant and thus compares favorably w.r.t. *Flee*. Indeed, for $\gamma = 1000$, *MarkovFlee* is again two orders of magnitude faster than *Flee*. In addition, Figure 2 shows that dynamic programming further reduces the runtime complexity compared to the recursive computation of travel probabilities.

7 SIMULATION OF REALISTIC CONFLICT SCENARIOS

We finally investigate realistic conflict scenarios, in South Sudan, Central African Republic (CAR), Burundi, and Mali. The South Sudan conflict between forces of the government and opposition forces started in December 2013. Since then, more than four million people have been forced to displace and become refugees. The simulation period, in which we covered the most intense conflicts, covers 604 days between 15th December 2013 and 10th August 2015. In the CAR, which has been in turmoil since a violent takeover of power in 2013, more than 640,000 people fled the country in search of safety, and an additional 630,000 were internally displaced according to UNHCR. The majority of those sought refuge in neighboring countries such as Cameroon, Chad, the Democratic Republic of the Congo (DRC) and the Republic of the Congo, with smaller numbers in Sudan and South Sudan. The crisis in Burundi started in April 2015 after President Pierre Nkurunziza announced he was running for a third term, in violation of the country's constitution. By November of the same year, around 2% of Burundi's population had fled to neighbouring countries, Tanzania, Rwanda, DRC and Uganda. Since 2015, more than 400,000 refugees and asylum-seekers have fled the country. For the Mali conflict scenario, since the uprising started in 2012, the security situation has gone from bad to worse in much of the region and attacks by armed groups have increased since then. According to UNHCR and UNICEF, since

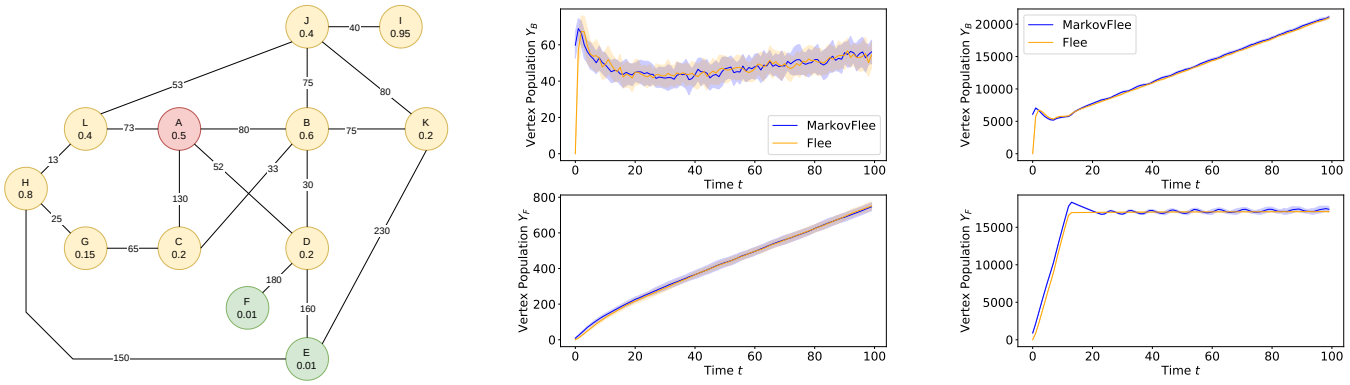

Figure 3: Left: Synthetic conflict scenario with a single conflict site (A) and two camps (E,F) with capacities of 18000 and 17000, respectively. Vertex populations on locations B and F computed by *Flee* and *MarkovFlee* for $\gamma = 2$ (middle) and $\gamma = 200$ (right) in the synthetic conflict scenario.

	T	N_T	L	$\lvert E\rvert$	$\lvert C\rvert$	$\bar{h}$
South Sudan	604	433k	51	60	10	2.2
CAR	820	424k	63	86	14	2.4
Burundi	396	205k	30	41	5	4.9
Mali	300	90k	23	35	7	1.5

Table 1: Summary statistics of the conflict scenarios, depicting the simulation time T, the number of agents at the end of the simulation N_T, number of locations L and camps $|C|$, the number of routes $|E|$, and $\bar{h}$, which is the ratio between the travel budget $d_{\max}$ and the average route distance.

January 2012, nearly 375,000 Malians have fled the conflict in the north of their country. Some 145,000, the majority of them women and children, have crossed into Burkina Faso, Mauritania and Niger. Our simulation starts on 1st March 2012 and ends in December 2012, for a total of 300 days. Table 1 provides an overview of the numbers characterizing the scenarios.

We ran simulations with *MarkovFlee*, *Flee* [31], and *FleeV2* [30], with the default parameter settings. The updated ruleset of *FleeV2* prevents agents from residing at a location if they have not exhausted their movement budget in the previous time steps, which effectively increases the probabilities ρ that agents leave their current locations. To emulate the effects of the corresponding rule, we ran *MarkovFlee* with an increased value of ρ for locations that are neither camps nor conflict zones (i.e., we increased $\rho(\ell)$ from 0.3 to 0.5). We executed all simulations on a Notebook with 12 GB RAM and an Intel® Core™ i5-8250U CPU with 1.60 GHz

We compare *Flee*, *FleeV2*, and *MarkovFlee* in terms of runtime and the differences between camp populations $y_{\ell,t}$ predicted by the simulation and interpolated from UNHCR data, respectively. Specifically, if $C \subset V$ is the set of camps and if $y^{\bullet}_{\ell,t}$ is the true population of camp $\ell \in C$ at time t, then the mean absolute percentage error (MAPE) at time t is given as

$$e(t) = \frac{1}{\sum_{\ell' \in C} y^{\bullet}_{\ell,t}} \sum_{\ell \in C} \left| y_{\ell,t} - y^{\bullet}_{\ell,t} \right|. \tag{19}$$

The error reported in Table 2 is the MAPE averaged over the entire simulation duration, i.e., $e = 1/T \sum_{t=1}^{T} e(t)$.

Our results in Table 2 show that, in general, *MarkovFlee* achieves similar accuracy as the *Flee*, and that *FleeV2* consistently outperforms competing methods, which is in line with the arguments in [30]. Indeed, *MarkovFlee* with default settings performs close to *Flee* in the South Sudan and CAR scenarios, while it performs worse for the Burundi and Mali conflicts. Adapting the simulation settings of *MarkovFlee* by increasing the movement probability ρ from 0.3 to 0.5 for locations that are neither conflict sites nor camps substantially improved the performance of *MarkovFlee*. Thus, even though *MarkovFlee* is based on the reduced ruleset of *Flee*, adapting its parameters may recover parts of the thus reduced accuracy.

Our runtime analyses in Table 2 show that *MarkovFlee* is approximately one order of magnitude faster than *Flee* and *FleeV2* in three out of four scenarios. Indeed, *Flee* and *FleeV2* suffer from long runtimes especially when the number of refugees N is large. In contrast, we see that in the Burundi conflict scenario, *MarkovFlee* is slower than *Flee* and *FleeV2*. This conflict is characterized by relatively short distances between locations, which means that agents may visit multiple locations at each iteration. This, in turn increases the cost of recomputing the probabilities in (15).

We finally investigate exemplary camp populations in Figure 4. It can again be seen that *MarkovFlee* performs similarly to its predecessors *Flee* and *FleeV2*, and that the general agreement between simulations and ground truth is satisfactory for many locations. The results for the Fassala-Mbera camp in the Mali conflict further shows that *FleeV2* is capable of initializing camp populations correctly, which we believe is an important factor contributing to its superior accuracy. Future work shall incorporate such prior information about camp utilization also in *MarkovFlee*. Finally, we see that in the Mentao camp in the Mali conflict, also *Flee* and, to a lesser extent, *FleeV2* initially exceed camp populations. Indeed, Mentao's distance to its neighbors in $\mathcal{G}$ exceeds $d_{\max}$. Thus, multiple agents at neighboring locations can start travelling to Mentao without immediately affecting the camp capacity and thus κ (which would prevent agents from selecting the corresponding route). When the

	South Sudan		CAR		Burundi		Mali	
	MAPE	runtime (s)	MAPE	runtime (s)	MAPE	runtime (s)	MAPE	runtime (s)
Flee [31]	0.461	1954	0.387	3189	0.646	351	0.412	310
FleeV2 [30]	0.48	1979	0.333	2993	0.528	657	0.358	204
MarkovFlee ($\rho = 0.3$)	0.48	51	0.398	516	0.711	776	0.48	10
MarkovFlee ($\rho = 0.5$)	0.46	55	0.39	538	0.698	897	0.404	10

Table 2: Mean absolute percentage error and runtime (in seconds) of *MarkovFlee* and competing methods for four realistic conflict scenarios. See text for details.

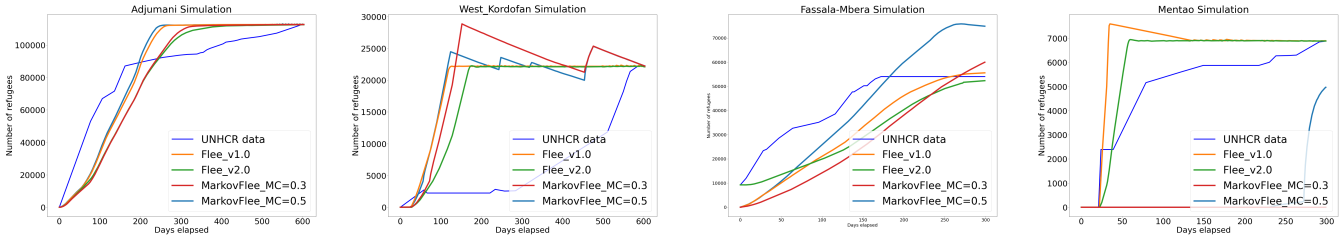

Figure 4: Exemplary camp populations for the camps in (from left to right): Adjumani and West Kordofan (South Sudan conflict) and Fassala-Mbera and Mentao (Mali conflict scenario). In general, all ABM approaches perform similarily, which is in line with the results from Table 2.

agents arrive at the next iteration, camp capacity is exceeded. The corresponding behavior of *MarkovFlee* is explained by Remark 1.

8 DISCUSSION AND CONCLUSION

In this work, we discussed ABMs for agent movements on a graph and showed that under certain assumptions the sequence of vertex populations is a MC (Theorem 1). We then instantiated this insight for a recent ABM for forced migration (*Flee*) and found that the resulting MC-based approach (*MarkovFlee*) can lead to a substantial speed-up with little or no loss of accuracy, both in synthetic and realistic conflict scenarios. Specifically, our analysis in Section 5 shows that the runtime complexity is reduced from $O(NhT)$ to $O(L^hT)$. Under the reasonable assumption that h is constant with L and N, this means that the runtime complexity of *MarkovFlee* is polynomial in the number of locations and constant in the number of agents, while that of *Flee* is linear in the number of agents. Indeed, runtime complexity was substantially reduced in conflicts with a large number N of refugees and in settings where routes between locations are long (small h). Future work shall investigate whether this computational speed-up can be maintained for large graphs with 100–10,000 locations.

We believe that results similar to Theorem 1 can be derived for other ABMs. For example, for ABMs with finitely many agent types that are characterized by different rulesets (e.g., travelling by foot and by car), we believe that the sequence of type-specific populations is still Markov if Assumption 1 holds. The authors of [3, 20] showed that aggregation can be useful for the voter model. Another example is epidemic spreading: Assuming a well-mixed population, epidemic spreading can be simulated by agent movement on a graph with three vertices (susceptible, infected, recovered). Given that Assumption 2 holds, i.e., that the infection probability depends on the total number of infected agents, Markov aggregation leads to the SIR model, cf. [18]. Finally, continuous-time ABMs may benefit from the concept of lumpability for continuous-time MCs.

While the equivalence between the ABM and a MC on vertex populations shown in Section 4 holds under quite general conditions, seemingly small changes to the agent rulesets can cause this equivalence to break down. We illustrated this at the hand of *Flee* and *MarkovFlee*. First, in *Flee* one has the option of computing the camp capacity factor κ either after each agent movement, or at the end of each iteration, with the authors of *Flee* having chosen the former option. In *MarkovFlee*, this choice is not available, as the concept of individual agent movements does not apply. Second, the fact that agents reside on links or that agents decide whether they leave a location depends on the total distance they have traveled within the last two iterations requires that agents have an internal state: the position on the route towards the target destination (see line 8 in Algorithm 1) or a log of traveled distances. Such an internal state prevents us from treating agents equivalently, as the agent identity then affects travel probabilities (via the extended agent state). Effectively, the probability that an agent i leaves location ℓ at iteration t depends also on i directly, i.e., $\rho(\ell, t) = \rho(\ell, t, i)$. As a consequence, Assumption 2 is violated and Theorem 1 does not apply. Nevertheless, in the case of *FleeV2* or *Flee*, we showed that comparable accuracy can be achieved by appropriately adjusting other ruleset parameters of *MarkovFlee*, effectively enabling us to reap the computational speed-up implied by Theorem 1. Hence, even if lumpability in the sense of Theorem 1 does not hold, Markov aggregation may prove to be useful if the sequence of aggregate statistics is at least approximately Markov. Future work shall investigate Markov aggregation to different ABMs and characterize the resulting trade-off between accuracy and runtime.

ACKNOWLEDGMENTS

This work has been supported by the HiDALGO, ITFLOWS, SEAVEA ExCALIBUR, and BrAIN projects. The projects HiDALGO (Grant No. 824115) and ITFLOWS (Grant No. 882986) have been funded by the European Commission's H2020 Programme. The project SEAVEA ExCALIBUR (Grant No. EP/W007711/1) has received funding from EPSRC. The project BrAIN – Brownfield Artificial Intelligence Network for Forging of High Quality Aerospace Components (Grant No. 881039) is funded in the framework of the program "TAKE OFF", which is a research and technology program of the Austrian Federal Ministry of Transport, Innovation and Technology.

The Know-Center is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Federal Ministry of Climate Action, Environment, Energy, Mobility, Innovation and Technology, the Austrian Federal Ministry of Digital and Economic Affairs, and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

REFERENCES

[1] James Anderson, Alok Chaturvedi, and Mike Cibulskis. 2007. Simulation tools for developing policies for complex systems: Modeling the health and safety of refugee communities. *Health care management science* 10, 4 (2007), 331–339.

[2] Claudio Angione, Eric Silverman, and Elisabeth Yaneske. 2022. Using machine learning as a surrogate model for agent-based simulations. *PLOS ONE* 17, 2 (02 2022), 1–24. https://doi.org/10.1371/journal.pone.0263150

[3] Sven Banisch. 2016. *Markov Chain Aggregation for Agent-Based Models*. Springer, Cham et al.

[4] Andrew T Crooks and Sarah Wise. 2013. GIS and agent-based models for humanitarian assistance. *Computers, Environment and Urban Systems* 41 (2013), 100–111.

[5] Dario Esposito, Stefania Santoro, and Domenico Camarda. 2020. Agent-based analysis of urban spaces using space syntax and spatial cognition approaches: A case study in Bari, Italy. *Sustainability* 12, 11 (2020), 4625.

[6] Nuno Fachada, Vitor V. Lopes, Rui C. Martins, and Agostinho C. Rosa. 2017. Parallelization Strategies for Spatial Agent-Based Models. *International Journal of Parallel Programming* 45 (2017), 449–481.

[7] Gabriele Filomena and Judith A Verstegen. 2021. Modelling the effect of landmarks on pedestrian dynamics in urban environments. *Computers, Environment and Urban Systems* 86 (2021), 101573.

[8] Nigel Gilbert, Petra Ahrweiler, Pete Barbrook-Johnson, Kavin Preethi Narasimhan, and Helen Wilkinson. 2018. Computational modelling of public policy: Reflections on practice. *Journal of Artificial Societies and Social Simulation* 21, 1 (2018), 14.

[9] Mordechai Haklay, David O'Sullivan, Mark Thurstain-Goodwin, and Thorsten Schelhorn. 2001. "So go downtown": simulating pedestrian movement in town centres. *Environment and Planning B: Planning and Design* 28, 3 (2001), 343–359.

[10] Vincent Huang and James Unwin. 2020. Markov chain models of refugee migration data. *IMA Journal of Applied Mathematics* 85, 6 (09 2020), 892–912. https://doi.org/10.1093/imamat/hxaa032

[11] Alireza Jahani, Hamid Arabnejad, Diana Suleimanova, Milana Vuckovic, Imran Mahmood, and Derek Groen. 2021. Towards a Coupled Migration and Weather Simulation: South Sudan Conflict. In *Proc. Int. Conf. on Computational Science (ICCS)*. Springer, Krakow, Poland, 502–515.

[12] Bin Jiang. 1999. SimPed: simulating pedestrian flows in a virtual urban environment. *Journal of geographic information and decision analysis* 3, 1 (1999), 21–30.

[13] Bin Jiang and Tao Jia. 2011. Agent-based simulation of human movement shaped by the underlying street structure. *International Journal of Geographical Information Science* 25, 1 (2011), 51–64.

[14] Rachel T Johnson, Thorsten A Lampe, and Stephan Seichter. 2009. Calibration of an agent-based simulation model depicting a refugee camp scenario. In *Proc. Winter Simulation Conf. (WSC)*. IEEE, Austin, TX, USA, 1778–1786.

[15] Dana Kaziyeva, Martin Loidl, and Gudrun Wallentin. 2021. Simulating spatio-temporal patterns of bicycle flows with an agent-based model. *ISPRS International Journal of Geo-Information* 10, 2 (2021), 88.

[16] F. P. Kelly. 1982. Markovian Functions of a Markov Chain. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)* 44, 3 (1982), 372–379.

[17] John G. Kemeny and James Laurie Snell. 1976. *Finite Markov Chains* (2 ed.). Springer, New York et al.

[18] Wasiur R. KhudaBukhsh, Boseung Choi, Eben Kenah, and Grzegorz A. Rempała. 2020. Survival dynamical systems: individual-level survival analysis from population-level epidemic models. *Interface Focus* 10 (2020), 20190048. https://doi.org/10.1098/rsfs.2019.0048

[19] Anna Klabunde and Frans Willekens. 2016. Decision-making in agent-based models of migration: state of the art and challenges. *European Journal of Population* 32, 1 (2016), 73–97.

[20] Robin Lamarche-Perrin, Sven Banisch, and Eckehard Olbrich. 2016. The Information Bottleneck Method for Optimal Prediction of Multilevel Agent-based Systems. *Adv. Complex Syst.* 19, 01n02 (2016), 1650002.

[21] Carlos Lemos, Helder Coelho, Rui J Lopes, et al. 2013. Agent-based Modeling of Social Conflict, Civil Violence and Revolution: State-of-the-art-review and Further Prospects. In *Proc. European Workshop on Multi-Agent Systems (EUMAS)*. EURAMAS, Toulouse, 124–138.

[22] Charles M Macal and Michael J North. 2005. Tutorial on agent-based modeling and simulation. In *Proc. Winter Simulation Conf. (WSC)*. IEEE, Orlando, FL, USA, 14–pp.

[23] Johannes Nguyen, Simon T Powers, Neil Urquhart, Thomas Farrenkopf, and Michael Guckert. 2021. An overview of agent-based traffic simulators. *Transportation Research Interdisciplinary Perspectives* 12 (2021), 100486.

[24] Itzhak Omer and Nir Kaplan. 2017. Using space syntax and agent-based approaches for modeling pedestrian volume at the urban scale. *Computers, Environment and Urban Systems* 64 (2017), 57–67.

[25] Alan Penn and Alasdair Turner. 2002. Space syntax based agent simulation. In *Pedestrian and Evacuation Dynamics*, Michael Schreckenberg and Som Deo Sharma (Eds.). Springer-Verlag, Berlin, Germany, 99–114.

[26] Antoniya Petkova, Charles Hughes, Narsingh Deo, and Martin Dimitrov. 2016. Accelerating the distributed simulations of agent-based models using community detection. In *Proc. IEEE Int. Conf. on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*. IEEE, Hanoi, Vietnam, 25–30. https://doi.org/10.1109/RIVF.2016.7800264

[27] Thorsten Schelhorn, David O'Sullivan, Mordechay Haklay, and Mark Thurstain-Goodwin. 1999. STREETS: An agent-based pedestrian model. In *Proc. Int. Conf. Computers in urban planning and urban management on the edge of the millenium (CUPUM)*. FrancoAngeli, Venice, Italy, 13.

[28] Christa Searle and Jan Harm van Vuuren. 2021. Modelling forced migration: A framework for conflict-induced forced migration modelling according to an agent-based approach. *Computers, Environment and Urban Systems* 85 (2021), 101568.

[29] Miranda Simon, Cassilde Schwartz, David Hudson, and Shane D Johnson. 2018. A data-driven computational model on the effects of immigration policies. *Proceedings of the National Academy of Sciences* 115, 34 (2018), E7914–E7923.

[30] Diana Suleimenova, Hamid Arabnejad, Wouter N. Edeling, and Derek Groen. 2021. Sensitivity-driven simulation development: a case study in forced migration. *Philosophical Transactions of the Royal Society A* 379, 2197 (2021), 20200077. https://doi.org/10.1098/rsta.2020.0077

[31] Diana Suleimenova, David Bell, and Derek Groen. 2017. A generalized simulation development approach for predicting refugee destinations. *Scientific Reports* 7, 1 (2017), 13377.

[32] Diana Suleimenova and Derek Groen. 2020. How policy decisions affect refugee journeys in South Sudan: a study using automated ensemble simulations. *Journal of Artificial Societies and Social Simulation* 23, 1 (2020), 2.