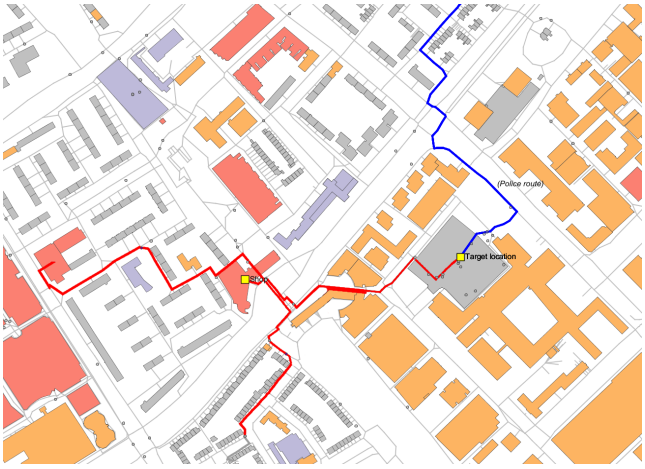

(a)

(b)

Figure 1: Examples of run-time injected incidents: (a) an arrest of two shoplifters (red routes) by a police officer (blue route) occurs at the specified location, (b) an improvised protest march (green circles represent participants), which attracts some passers-by (gray circles) using an area trigger.

Movement between actions. Once a plan has been created that resolves all role preconditions, the planner enables agents to travel between locations. For this, the planner introduces movement actions, inserted before each action in the plan. The algorithm for inserting suitable moves takes into account two aspects. First, agents may have acquired a certain role as a result of actions they performed earlier (e.g., Police Officer or Shoplifter). Therefore, we need to select a type of movement that fits this role. The second aspect considered when inserting suitable move actions is the target of the action we move towards, e.g. a police officer should choose Pursuit over Flee or Walk for movement towards an offender. Once the most fitting move action is chosen, it is inserted in the plan.

Assigning agents and locations. The result of the planning algorithm described above is a sequence of actions (i.e., a plan), with correctly linked variables that can be instantiated by assigning agents and locations that are present in the environment. An algorithm based on sampling and iterative improvement now finds such agents and locations, and ensures that the plan can be executed within the requested time constraints (i.e., the enabling action of the plan happens exactly when the user wants it to).

Intuitively, the algorithm works by first sampling a random valid assignment for the locations and agent variables in the plan. A valid assignment takes into account any constraints on the variables. For example, a shoplifting needs to take place in a shop, as represented in a constraint on the location/time variable of the Shoplift action.

Then, while the plan still takes too long to execute, a number of improvement steps are performed, in which (1) the critical path in the plan is found, i.e. the sequence of actions that takes most time, and (2) one variable for a chosen action on the critical path will be assigned differently, in such a way that the plan is expected to take less time. Whenever there is an action preceding the chosen action on the critical path, the current location of this preceding action is apparently too far away from the location of the chosen action. Therefore, the value of the location/time variable of this preceding action needs to be adjusted such that it refers to a valid location closer to the chosen action. If there is no preceding action, the chosen action involves one or more agents that are too far away from the location the chosen action will take place at, and we need to find valid agents that are closer to this location.

If a fixed number of improvement iterations does not yield a plan that fits the time constraints, the algorithm is restarted with a new random valid assignment of the variables in the plan. After a number of failed restarts, the algorithm concludes that planning the incident within the required time is not feasible. The user is then presented with the best plan found so far, and can choose whether s/he wants to have the incident happen at the corresponding time instead.

3. DISCUSSION AND FUTURE WORK

Our contribution. We present an approach for injecting incidents into an urban scenario at run-time, interrupting daily activities and temporarily promoting selected background agents to play a specific role in the sequence of events. Examples are shown in Figure 1. Because incidents are planned on-the-fly based on a library of available actions and associated pre- and postconditions, new incidents can build upon content that is already present.

In addition to run-time incidents, the prototype system also implements a simulation of daily patterns-of-life, based on real-world census data. Individuals in the simulation are given daily activities (such as work, leisure, shopping, having dinner) that match their properties (such as age, gender, and household). We designed the prototype system such that it can be easily extended to include more daily activities, behavior models and different modalities.

Future work. At the moment, the prototype system supports population models at a fixed level of detail. When, for example, trainee interaction with an individual population member is a scenario requirement, more detailed models are needed. Due to the computational complexity of such models, it is not feasible to have this level of detail for the entire population all the time. Instead, the system must be able to dynamically adapt to the level of detail required.

Other work could focus e.g. on including other traffic modalities than pedestrians, or on allowing users to specify population behavior on the level of intent as well, e.g. by drawing a morning rush hour on a specific street, which influences population attributes such as residence and place of work to create the desired dense traffic.

Online repository. The current prototype system is available as open source [1].We encourage others to experiment with the framework and to extend it with novel models and algorithms.

References

1. Population Modelling with Run-Time Incidents. URL `https://github.com/TNOCS/idsa`.

2. M. O. Riedl and R. M. Young. An intent-driven planner for multi-agent story generation. In *AAMAS 2004: The Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

3. G. Zacharias, J. MacMillan, and S. Van Hemel, editors. *Behavioral modeling and simulation: from individuals to societies*. National Academies Press, 2008.