Traplas, a Transport Planning Simulator

Jonne Zutt and **Willem Drost** and **Herbert de Vos**
Delft University of Technology
J.Zutt@tudelft.nl, drost2@zonnet.nl, grasman@zandbak.org

Introduction

This demonstration is about two jointly developed tools for pickup and delivery transportation problems. The first is a transport planning simulator, called *Traplas*. Traplas is a multi-agent transport simulator, which is responsible for the planning and scheduling of agents in order to successfully execute transportation requests by customers. The second is a 3D graphics visualisation tool for transportation problems, called *TraplasViz*, which is especially developed for Traplas, but with a general set-up and well-documented interface so that it can be very useful to other projects as well.

Traplas has been used for the experiments described in another paper at this conference, see (ter Mors, Zutt, & Witteveen 2007). Traplas and TraplasViz are both open-source projects available at `http://traplas.sf.net` and `http://traplasviz.sf.net` respectively. Development and testing of the projects is done on Linux platforms.

Traplas

Traplas is an agent-based transport planning simulator for online pickup and delivery transportation tasks with constraints. Customers can, at any time, request freight to be transported. Besides a source and destination location for this freight, time-windows are specified within which the loading and unloading is desired to take place. A reward function defines the reward that the responsible agent achieves for executing the transportation task (for instance, the customer might penalize the executing agent for being too late using this function). The transport network is modeled using resources that have a distance, maximum allowed speed and capacity. The vehicles also have maximum driving speeds and loading capacity. Traplas is highly parameterized, so that the end-user can choose whether overtaking is allowed, what default policy is used when multiple agents want to enter the same location at the same time (e.g. first-come-first-served).

Several (re)planning and scheduling methods are implemented, resulting in plans that can execute the transportation tasks, such as illustrated by Figure 1 and 2. Different

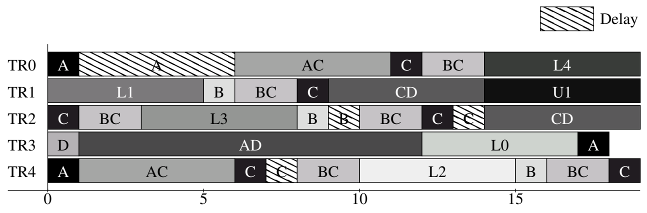

Figure 1: Example plan (cut off after 20 time units). For five agents, TR0 to TR4, a sequence of locations they will visit is displayed along the horizontal time axis. L1 and U1 is the loading and unloading of package 1 respectively.

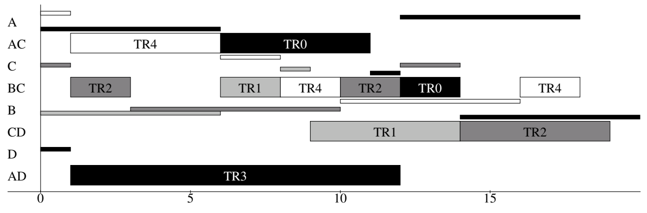

Figure 2: The same plan from a resource-based orientation, for each location a sequence of agents is displayed along the horizontal time axis. The height of a box denotes the fraction of capacity used by the agent.

planning methods can be compared due to maintaining predefined performance indicators. Furthermore, an incident model can be used to model uncertainty in the environment (disturbances). This can be done by specifying a resource failure probability, by specifying an extensive list of incidents, by specifying temporary road blocks or, for instance, by specifying temporary communication failure of an agent. Regardless of the chosen planning method, the execution framework is the same. This ensures a correct execution even when the planning method produced incorrect results, or when there was not enough time for replanning due to incidents.

Traplas is a discrete event-based simulator built upon the Pamela Run-Time Library (van Gemund 1994). This library provides a concurrent, general-purpose simulation interface, based on the procedure-oriented (”P/V-style”) paradigm (Andrews & Schneider 1983). There are two important data types: processes and semaphores. Processes

Algorithm 1 Travelling by claiming location resources.

```
1: procedure DRIVE($r_1, r_2$)
2:     pam_P(cap($r_2$))                    ▷ Obtaining a claim.
3:     arbitrated_pam_V(cap($r_1$))
4:     pam_delay($drive\_cost$)

5: procedure ARBITRATED_PAM_V($sema$)
6:     pam_V($sema$)                        ▷ Releasing a claim.
7:     Reschedule processes blocked for $sema$
```

The cap(r) refers to the capacity semaphore of location resource r, which initial credit equals the capacity of the location resource. Functions pam_P(s) and pam_V(s) are part of the Pamela run-time library and correspond to claiming and releasing semaphore s respectively. The pam_P(s) operation will block if semaphore s has zero credit; when there is the credit is decreased by one and the process can continue. The pam_V(s) operation never blocks and increments the credit of semaphore s by one.

are light-weight threads with their own local time stamp, in which the global simulation time is stored either at which it has been suspended (in the past) or at which it is resumed (in the future). At any time, only one process can be running. All other processes are either runnable, i.e. scheduled to run at their local time stamp, or they are blocked, i.e. waiting on a semaphore until another process lifts this block on this semaphore. Pamela processes are scheduled non-preemptively. This means that execution of a process continues until it voluntarily releases control. In general, the agents are represented by Pamela processes. Communication between the agents is modeled using message queues protected with Pamela semaphores. For traversing the transport network, the transport resources have to claim a semaphore that corresponds to the location (see Algorithm 1).

The author has set-up a benchmark to compare the different planning methods to each other, both with and without incidents. An important aspect here is the robustness of the planning methods in case there are incidents.

The output of a simulation run is an extensive set of statistics (easily configurable) together with the full history of actions performed by all entities.

TraplasViz

The TraplasViz visualization application is a visualization for the Traplas transport planning simulator. Although TraplasViz was especially designed for Traplas, one of the goals has always been to create a general-purpose visualization tool for pickup and delivery transportation. It should be fairly easy to adapt the visualization for other similar purposes.

OpenSceneGraph is used to visualize the execution of a simulation run performed by Traplas. The OpenSceneGraph (`http://www.openscenegraph.com`) is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modeling. It is written entirely in Standard C++ and OpenGL.

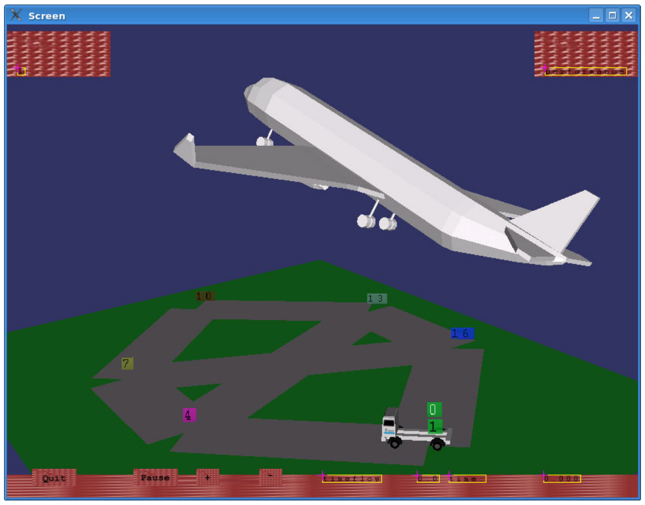

Figure 3: TraplasViz screenshot, where two types of transport resources are visible.

The nodes are represented as grey squares and the edges as grey connections between these nodes. Alternatively, it is possible to add nodes and edges without actually drawing these; this can be useful, for example, to model routes in the air. In the future, new functionality will be added to allow the visualization to load a custom made world. This world can be created using OSGEdit (`http://osgedit.sourceforge.net/`) or some other OSG editor. We can then use the invisible infrastructure and the custom world to present a high quality visualization suitable for demonstration purposes.

Interface

There are two modes of communication between Traplas and TraplasViz. First, Traplas can create an intermediate file containing a simulation. This file is then read by TraplasViz which visualizes the recorded simulation. Second, Traplas and TraplasViz can communicate using TCP/IP sockets. This communication is bi-directional allowing feedback from the visualization to be send to back the simulator. Feedback can consist of incidents (a truck breaks down, a road is blocked) or the creation of additional transport orders.

The well-documented interface specifies messages to construct a transport network containing of nodes and edges. Then, transport entities (such as trucks) can be created, these transport entities can move through the transport network by issuing drive commands. Also, transportable entities (such as packages or cargo containers) can be created. The transport entities can then load and unload these transportable entities and move these through the network.

There is also a performance indicator message which can be used to display all kinds of real-time performance information on the current task execution by the agents.

The most challenging part of coupling Traplas and TraplasViz has been the clock synchronization. Since Traplas is a discrete event-based simulator, where time progresses in sometimes big steps, and TraplasViz maintains a clock that is incremented with small time steps, the following strategy is adopted. Just before one side increments its clock, it sends a message to the other side with the intended new clock value. It then waits for a message from the other side that the intended clock value is reached, after which it continues. While the application waits, it can still receive and process other incoming messages.

For now, it is assumed that Traplas runs faster, with respect to simulation time, than TraplasViz and, hence, this synchronization is only implemented in one way.

Acknowledgments

This research has been supported by the TNO-TRAIL project (16) on *Fault detection and recovery in multi-modal transportation networks with autonomous mobile actors* and by the Dutch Ministry of Economic affairs under the SENTER TSIT program on *Cybernetic Incident Management* (TSIT2021).

References

Andrews, G. R., and Schneider, F. B. 1983. Concepts and notations for concurrent programming. *ACM Computing Surveys* 15(1):3–43.

ter Mors, A. W.; Zutt, J.; and Witteveen, C. 2007. Context-aware logistic routing and scheduling. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (to appear)*.

van Gemund, A. J. C. 1994. The pamela run-time library. Technical Report 1-68340-44(1994)03, Delft university of technology. Version 1.0.