

LAMPSSys – An experiment-platform for the optimization of processes and agent-behaviors

Christoph Mies, Benjamin Wolters, Timo Steffens

Real-world processes and modern multi-agent systems typically have high-dimensional characteristics that need to be configured and tuned for optimal performance. Optimizing processes and agent behaviors is an active field of research and relevant methods exist. This paper introduces a flexible and scalable framework that allows to plug in a variety of optimization methods. Two case studies demonstrate the flexibility of the framework.

1 Introduction

Learning and optimizing processes is an active field of research. Typically, such methods are evaluated and tuned on specific applications and implementations. However, typical applications for commercial use do not focus on the development of new methods, but apply existing methods in a broader application. The optimization method is only part of a larger suite which also provides functionalities to model, simulate, analyze and visualize processes (e.g. [4]). In such settings, it is necessary to easily integrate, configure and evaluate diverse optimization techniques.

This paper introduces a platform called LAMPSSys, which is a general-purpose agent-based simulation system. It has been designed to satisfy requirements for operations research and process optimization applications. It applies the paradigm that agent-behaviors are processes that can be represented and executed in the same way as business processes. In this view, learning agent-behaviors and optimizing processes are equivalent. LAMPSSys provides facilities to plug-in and evaluate process optimization techniques.

In order to illustrate the flexibility of LAMPSSys, two case studies are presented, in which very different optimization techniques were integrated into the system.

The remainder of the paper is organized as follows. Section 2 outlines the simulation system, its architecture and the process modeling language that is used to specify the agent-behaviors. In section 3 two case-studies are described. Finally, the last section concludes and outlines future work.

2 LAMPSSys

LAMPSSys applies the Petri-net-based language LAMPS to represent processes and agent behaviors. LAMPS (Language for Agent-based Modeling of Processes and Scenarios). It extends the concept of colored Petri nets and hierarchical Petri nets [6]. The former introduced the concept of typed tokens, the latter structured so-called places (states) and actions hierarchically. The most important extension is the concept of agents that correspond to conditions in the Petri net model. Please refer to [4, 7] for more details.

LAMPSSys is a simulation system that is able to execute LAMPS graphs. It enhances the Flip-Tick-Architecture (FTA) [5]. FTA is a design paradigm for scalable distributed systems that exhibit priorly unknown dynamic characteristics as well as disturbances and inaccuracies which are difficult, if not impossible, to model in a closed-form mathematical approach.

LAMPSSys is based on the concept of a society of agents. The whole environment is modeled in terms of agents and the whole interaction is based on effects of actions executed by agents. Additionally, agents can form so-called *clans*, which are headed by one agent acting as *clan-chief*. A clan can take certain functions for its clan members, e.g. scheduling or communication.

LAMPSSys comprises three classes of entities: agents, clans, and tags. The society of all agents A is formed by a set of individual agents a_j : $A = \{a_1, \dots, a_j, \dots, a_k\}$. Each agent a_j is composed of a set of typed attributes U_j , whose value assignment determines the agent's state, together with an action function f_j , which entails all operations that can be performed by the agent $a_j = (U_j = (u_{1,j}, \dots, u_{m,j}), f_j)$.

The principle of autonomy of agents forbids the direct manipulation of internal data structures and behaviors of other agents. Consequently, all interactions between agents are handled via messages. Formally, a message N_v is comprised of a number of attributes: $N_v = \{u_{1,v}, \dots, u_{l,v}\}$. Upon receiving a request, the agent is able to analyze its content and to decide whether it wants to comply.

The basic unit of execution is called a cycle. During one cycle, the agent reads its messages and triggers the appropriate actions, which might consist of writing messages to other agent. This is called the *tick* of an agent.

During each tick a certain amount of time dt is simulated. Action executions that lasts longer than dt are split and partially executed several times in consecutive ticks. A scheduler-clan is a set of agents that have the same dt and, thus, simulate the world with the same pace. Time steps can be adapted from cycle to cycle and are determined by the scheduler-clan-chief. Thus, the time resolution of the simulation can vary from cycle to cycle. This is particularly valuable for increasing the time resolution in the computation of dynamic equations for fast moving objects. Note that different scheduler-clans can have different time resolutions. With this approach, fully synchronized (all agents form one