Simulation of a simplified Ecosystem to study the Influence of Environmental factors on the Bee Population

Henrik Jonathan Seeliger, Georg Eckardt

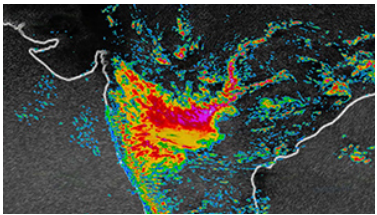
10.07.2023

Table of Contents

Motivation

- Bees are essential part of the agriculture and many ecosystems
- In recent years the bee population receded rapidly
- Many theories exist trying to find a correlation between this observation and many factors

Idea



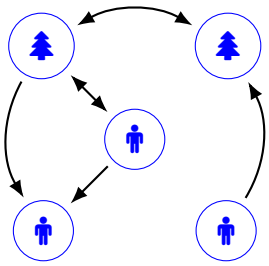
- Simulating bee ecosystems to study factors affecting bee populations
- Individial calculation of each entity i.e. biotical and abiotical factors for more simulation accuracy
- Extensible and customizable simulation base for easy extension by more entites

Relevance

- Running a (ecosystem) simulation system requires a high computational workload
  - Every entity has specific behaviours and effects to the world
  - Every time unit, calculations for every entity are necessary
- Simulation size may scale with computational resources
  - More resources $\rightarrow$ larger simulation $\rightarrow$ more entities, larger world, ...

  $\rightarrow$ More possibilities and a more accurate simulation
- Good problem for high-performance computing

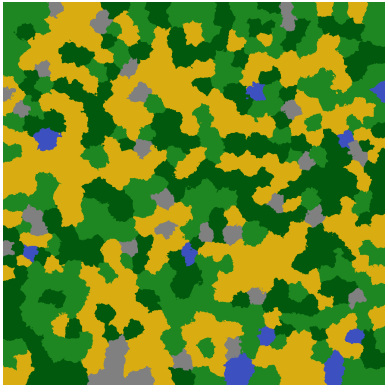
General Approach



- Multi-agent system
- All biotical (and abiotical) factors are *agents*
- Behaviours and effects to the world are calculated per agent

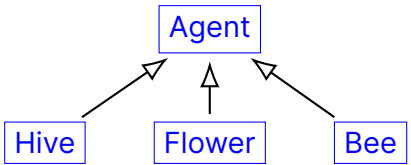
- C++ for C's speed advantages and MPI support, as well as modern language features
  → Enabling a performant and extensible simulation

World Map



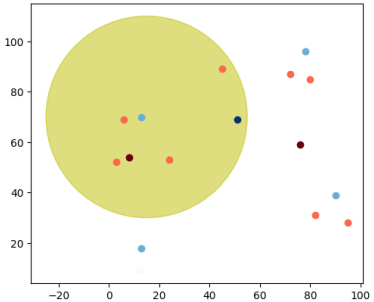
- Randomly generated world map (by seed)
- Based on Voronoi diagrams and simplex noise
- Assigns each world block unit ($1m^2$) a biome
- Enables specific environmental factors to be considered

Agent Hierarchy



- Every agent is a sub class of the *agent* class
- Agents have to implement specific methods: *update* and *move*
- Have a reference to the current world state for committing effects
- Access to other agents via world state
- How implement interactions without encountering circular dependencies

Agent Organization



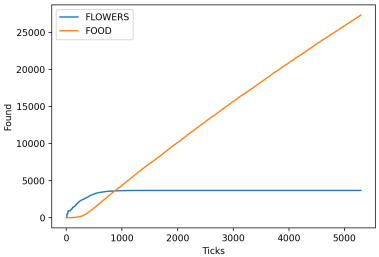
- Current world data and agents are organized in a *world state*
- Efficient way needed for indexing agents
  - → Complexity of indexing agents may be one of the biggest bottlenecks
  - → Spatial, multi-dimensional index
  - → Range queries, nearest neighbour, . . .
- Solution: **k-d tree** i.e. 2-d tree

Simulation Process

- Tick-based system
- Every tick represents one fixed time value, e.g. one second
- Every tick execution of fixed phases:
  - Update phase
  - Move phase

→ Execution of every agent's methods in respective phases

Sequential Approach

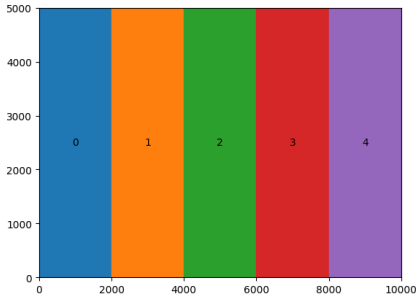


- The only process existing generates the world map and calculates every tick for every agent
- Usually high number of agents (≈ 10.000-40.000 bees per hive)
  → High computational intensity, very slow

Parallelization Approaches I

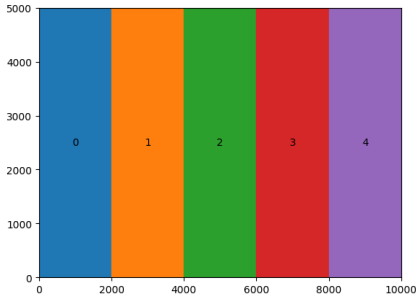
- Easy to parallelize: world generation
  - Algorithmic procedure involving large matrix calculations (e.g. simplex noise calculations for every world block)
  - → Parallelized using OpenMP
  - But: Only executed in the beginning and small part of the total runtime
- Non-trivial: Parallelization of the tick system and the simulation procedure
- First idea: scattering agents to calculate on among all processes
  - How to synchronize the world state?
  - → Large communication effort necessary

Parallelization Approaches II



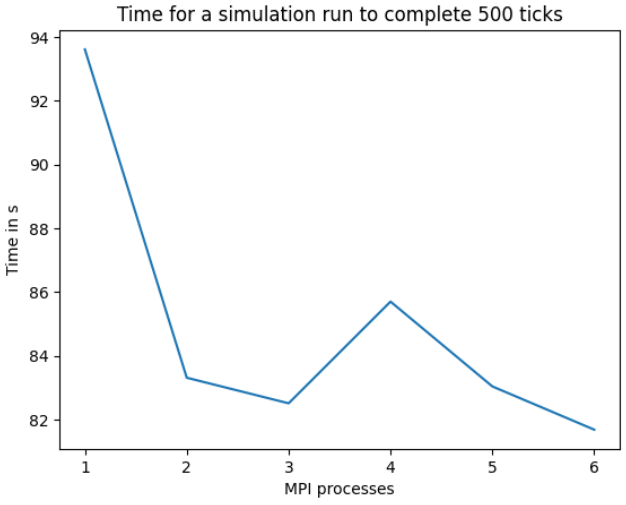
- Better idea: Chunking
  - World map is split into *n* chunks for *n* (MPI) processes
  - Each chunk calculates the tick phases for its agents
  - Agents operate and change the world state individually and only in their direct environment
  - If agents want to move to another chunk, they are sent using a MPI gather operation

Parallelization Approaches III



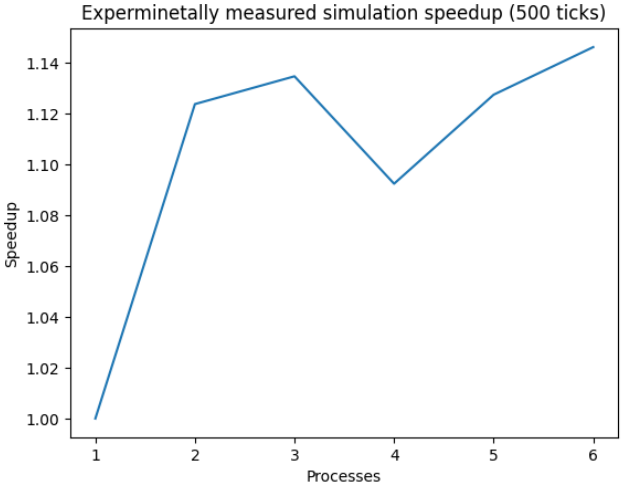
- Advantages:
  - Computational effort is higher than communication and network effort
  - Processes calculate more than moving data
- Disadvantages:
  - Unequal distribution of computational intensity is possible
  - Sending agents using MPI is complex

Performance Analysis: Single Benchmark



CPU	Intel(R) Co-re(TM) i7-8750H CPU @ 2.20GHz
RAM	DDR4 16 Gb

Performance Analysis: Speedup (Strong Scaling)



CPU	Intel(R) Co-re(TM) i7-8750H CPU @ 2.20GHz
RAM	DDR4 16 Gb

Technical Challenges

- Circular dependencies
- Memory management
- MPI with C++ (classes, C-style arrays, ...)

Current Results

- Base simulation is finished including basic agents
- Enables an extensible simulation, offering a platform to build on for the future, e.g. adding more agent types

Open Points

- Cleaning up code
- Externalizing simulation configuration parameters
- Further and more precise benchmarks
- Testing on the GWDG cluster