Software for agent based social simulation with raster inputs in distributed HPC environments

Sergiy Gogolenko

Oct 11, WSSP’17
GSS
scientific evidence to support policy-making · public action · civic society

pilots: grean growth health habits urban planning

LESSONS

Deliver a stack of **HPC compliant software** for GSS users

- creating realistic synthetic populations
- network reconstruction (creating synthetic social networks, etc)
- parallel and distributed **multi-agent systems simulation** (for GSS only)
- analytics: basic data and graph analytics, sensitivity analysis, parameter sweep, etc
Agent-based modeling and simulation

Annual Workshops on HPC Forums

- 2012 – EuroPar: Workshop on Parallel and Distributed Agent-Based Simulations (PADABS)
- 2016 – IPDPS: IEEE Workshop on Parallel and Distributed Processing for Computational Social Systems (ParSocial)
Agent-based modeling and simulation

Annual Workshops on HPC Forums

- 2012 – EuroPar: Workshop on Parallel and Distributed Agent-Based Simulations (PADABS)
- 2016 – IPDPS: IEEE Workshop on Parallel and Distributed Processing for Computational Social Systems (ParSocial)

ABMS in a Nutshell

- **bottom-up approach**
  - individual-based simulation → tendency
- **components**
  - agents: discrete autonomous entity
  - environment: GIS, relationships, etc
  - actions and interactions

Diagram:

- Agent\(_1\) attributes
- Agent\(_i\) attributes
- Agent\(_n\) attributes

Environment

3/26 :: Sergiy Gogolenko :: Software for agent based social simulation with raster inputs in distributed HPC environments :: Oct 11, WSSP’17 ::
MAS in GSS: Lesson 1. Agents

"molecular dynamics society"

The Simpsons: Treehouse of Horror XIII (Season 27) [from fx.com]
MAS in GSS: Lesson 1. Agents

“molecular dynamics society”

real society is heterogeneous
MAS in GSS: Lesson 1. Agents

- heterogeneous
MAS in GSS: Lesson 1. Agents

- heterogeneous
- mobile
MAS in GSS : Lesson 1. Agents

- heterogeneous
- serializable
  (introspection)
  - I/O
  - data exchange
  - checkpointing
- mobile
MAS in GSS : Lesson 2. Environment

Border fence Tijuana-San Diego. Photo from ©wikimedia.org
MAS in GSS: Lesson 2. Environment

obstacles imposed by nature

Border fence Tijuana-San Diego. Photo from ©wikimedia.org
MAS in GSS: Lesson 2. Environment

urbanization

obstacles imposed by nature

Border fence Tijuana-San Diego. Photo from ©wikimedia.org
MAS in GSS: Lesson 2. Environment

- Urbanization
- Obstacles imposed by nature
- Other concerns

Border fence Tijuana-San Diego. Photo from © Wikimedia.org
MAS in GSS: Lesson 2. Environment

Earth at Night. Photo by NASA ©wikimedia.org
MAS in GSS: Lesson 2. Environment

Earth at Night. Photo by NASA ©wikimedia.org

sparse 2D
MAS in GSS: Lesson 3. Interactions

▶ **short distance** (spatial proximity)

[Image: © The Walt Disney Company [img from youtube.com]]
MAS in GSS: Lesson 3. Interactions

- **short distance** (spatial proximity)
  - metric: *halo* (area of interest – AOI)
  - *halo* size: agent type independent
  - BC: von Neumann, Moore, periodic

![Diagram of short distance interactions with halo concept](https://via.placeholder.com/150)

- **long distance** (social relationships)
  - undirected graphs
  - dynamic (evolving)
  - might be weighted
  - might need several graphs
MAS in GSS: Lesson 3. Interactions

- **short distance** (spatial proximity)
  - metric: *halo* (area of interest – AOI)
  - *halo* size: agent type independent
  - BC: von Neumann, Moore, periodic

- **long distance** (social relationships)
  - undirected graphs
    - dynamic (evolving)
    - might be weighted
  - might need several graphs
scientific collaboration network (© Olivier H. Beauchesne)
MAS in GSS: Lesson 3. Interactions

Social relationships may dominate the scientific collaboration network (© Olivier H. Beauchesne)
MAS in GSS : Lesson 4. Users

**Technical traits background of average GSS modeler**

- Python and/or R
- minimum boilerplates
- spend time on mastering low-level parallel APIs (MPI, OpenMP, CUDA,...)
Technical traits background of average GSS modeler

- Python and/or R
- minimum boilerplates
- spend time on mastering low-level parallel APIs (MPI, OpenMP, CUDA,...)

Embedded DSLs

- linear algebra
  - Blaze
  - Eigen
  - $M \times L$
  - ...

- CFD
  - OpenFOAM®
  - FEniCS
  - ...

MAS in GSS: Lesson 4. Users

10/26 :: Sergiy Gogolenko :: Software for agent-based social simulation with raster inputs in distributed HPC environments :: Oct 11, WSSP'17 ::
Model: Formalization

rasters convey GIS env. input

Collection of sites:
map: position $\rightarrow$ struct raster

sites (loci of interaction) are pixel positions

never settled agents (with social nets) settle sites

social graphs (nets)
Model: Formalization

rasters convey GIS env. input

sites (loc of interaction) are pixel positions

Collection of sites: map

v₀ v₁ v₂ v₃ v₄ v₅

settle sites v₀ v₁ v₂ v₃ v₄ v₅

social graphs (nets)

agents (with soc. nets) never settled

11/26 :: Sergiy Gogolenko :: Software for agent based social simulation with raster inputs in distributed HPC environments :: Oct 11, WSSP’17 ::
Model: Formalization

agents
(with soc. nets)
settle sites

sites
(loc of interaction)
are pixel positions

rasters
convey
GIS env. input

never settled

social graphs (nets)
**Model: Formalization**

- **Agents** (with social nets) settle sites.

- **Sites** (locus of interaction) are pixel positions.

- **Rasters** convey GIS env. input.

- **Social graphs (nets)** are pixel positions.

- **Spatial graphs** map position to structure.

- **Collection of sites:** raster inputs in distributed HPC environments.

---

11/26 :: Sergiy Gogolenko :: Software for agent-based social simulation with raster inputs in distributed HPC environments :: Oct 11, WSSP'17 ::
Model: Formalization

Sites (loci of interaction) are pixel positions.

Social graphs (nets) are used to represent agent interactions.

Agents (with social nets) settle sites.

Collection of sites: map: position → struct

Rasters convey GIS env. input.

Rasters (individuals) never settled.
Model: Formalization

**Agents** (with social nets) settle sites

Sites (loc of interaction) are pixel positions

Rasters convey GIS env. input

Collection of sites: map: position → struct

Social graphs (nets)

Spatial graph

never settled

v₀ v₁ v₂ v₃ v₄ v₅

agents

settle sites

sites

(loc of interaction)

are pixel positions

rasters

convey

GIS env. input

11/26 :: Sergiy Gogolenko :: Software for agent based social simulation with raster inputs in distributed HPC environments :: Oct 11, WSSP’17 ::
Model: Basic domain decomposition

Computational graph

- **Vertex weight:** $w_i = \#\text{agents}_i$
- **SDC-edge weight:** $w_{ij} = w_i + w_j$
- **LDC-edge weight:** $w_{ij} = \#\text{links}_{ij}$
Model: Basic domain decomposition

Computational graph

1. vertex weight: $w_i = \#agents_i$
Model: Basic domain decomposition

Computational graph

1. vertex weight: $w_i = \# \text{agents}_i$
2. SDC-edge weight: $w_{ij} = w_i + w_j$
Model: Basic domain decomposition

Computational graph

1. vertex weight: $w_i = \#\text{agents}_i$
2. SDC-edge weight: $w_{ij} = w_i + w_j$
3. LDC-edge weight: $w_{ij} = \#\text{links}_{ij}$
Model: Basic domain decomposition

Computational graph

1. vertex weight: $w_i = \#\text{agents}_i$
2. SDC-edge weight: $w_{ij} = w_i + w_j$
3. LDC-edge weight: $w_{ij} = \#\text{links}_{ij}$
Model: Workaround

But what if rasters have very high resolution?
Model: Workaround

But what if rasters have very high resolution?

Issue with mapping of sites to processes

- need it for agent move operation

![Inverse of Spain at Night from Wikimedia](https://commons.wikimedia.org/wiki/File:Inverse_of_Spain_at_Night.png)
Model: Workaround

But what if rasters have very high resolution?

**Issue** with mapping of sites to processes

▶ need it for *agent move* operation

---

**Computational graph-coarsening**

▶ uniform mesh

▶ bound min chunk width: \( w \geq 2 \cdot \text{halo} \)

▶ disadvantages: disbalanced weights, highly skewed degree distribution
Model: Workaround

But what if rasters have very high resolution?

Issue with mapping of sites to processes
- need it for agent move operation

Computational graph-coarsening
- **uniform mesh**
  - bound min chunk width: \( w \geq 2 \cdot \text{halo} \)
  - disadvantages: disbalanced weights, highly skewed degree distribution

- **multi-scale mesh**
  - quad-tree
    (like in Barnes-Hut algorithm)
Model: Workaround

But what if rasters have very high resolution?

Issue with mapping of sites to processes

- need it for agent move operation

Computational graph-coarsening

- **uniform mesh**
  - bound min chunk width: \( w \geq 2 \cdot \text{halo} \)
  - disadvantages: disbalanced weights, highly skewed degree distribution

- **multi-scale mesh**
  - quad-tree
    - (like in Barnes-Hut algorithm)

- **vector GIS information**
  - (local) administrative units \( \equiv \) unstr. tree
Software: ABMS Frameworks

- 24 popular frameworks
- 3 have parallel facilities (only RepastHPC in C++)

... rather pessimistic paper
(seems like RepastHPC is the only choice)
Software: Distributed ABMS Frameworks for HPC

- **Java**: 40%
- **C/C++**: 60%

- **10 open source** ABMS frameworks for HPC
- **some in Java**
  - D-MASON, Jade, MACE3J, SWAGES
- **6 out of 10 in C/C++** (Charm++)
  - recent: RepastHPC, FlameHPC, Pandora
  - oldies: EcoLab, PDES-MAS, ABM++
Software: Distributed ABMS Frameworks for HPC

Java 40%

C/C++ 60%

▸ 10 open source ABMS frameworks for HPC
▸ some in Java
  ▸ D-MASON, Jade, MACE3J, SWAGES
▸ 6 out of 10 in C/C++ (Charm++)
  ▸ recent: RepastHPC, FlameHPC, Pandora
  ▸ oldies: EcoLab, PDES-MAS, ABM++
▸ beyond the paper scope: EpiSemdemics, FluTE, ...
Software: Distributed ABMS Frameworks for HPC

- **Java**: 40%
- **C/C++**: 60%

**10 open source** ABMS frameworks for HPC
- some in Java
  - D-MASON, Jade, MACE3J, SWAGES
- **6 out of 10 in C/C++ (Charm++)**
  - recent: RepastHPC, FlameHPC, Pandora
  - oldies: EcoLab, PDES-MAS, ABM++
- beyond the paper scope: EpiSemdemics, FluTE, ...

... we need another tool though!
What?
Yet another ABMS framework?

Maybe we can reuse something from existing...

Munch, Edvard. *The Scream* © wikimedia.org
Software: Building blocks (Primitives)
Software: Building blocks (Primitives)

- agents (& synthetic population)
  - type introspection
  - heterogeneous container
Software: Building blocks (Primitives)

- agents (& synthetic population)
  - type introspection
  - heterogeneous container

- social relationships
  - evolving undirected graphs
  - with and without edge weights
Software: Building blocks (Primitives)

- agents (& synthetic population)
  - type introspection
  - heterogeneous container
    - fast heterogeneous hash table
- social relationships
  - evolving undirected graphs
  - with and without edge weights

Social graphs (nets)

Spatial graph

Processes
Software: Building blocks (Primitives)

- agents (& synthetic population)
  - type introspection
  - heterogeneous container
    (fast heterogeneous hash table)
- social relationships
  - evolving undirected graphs
  - with and without edge weights
- spatial information
  - map: position → pixel info
Software: Building blocks (Primitives)

- agents (& synthetic population)
  - type introspection
  - heterogeneous container (fast heterogeneous hash table)
- social relationships
  - evolving undirected graphs
  - with and without edge weights
- spatial information
  - map: position → pixel info
- distribution (domain decomposition)
  - graph coarsening support
    - map: position → coarse piece
  - graph partitioner
  - communication layer
Software: Building blocks (Primitives)

- agents (& synthetic population)
  - type introspection
  - heterogeneous container
    (fast heterogeneous hash table)
- social relationships
  - evolving undirected graphs
  - with and without edge weights
- spatial information
  - map: position → pixel info
- distribution (domain decomposition)
  - graph coarsening support
    - map: position → coarse piece
  - graph partitioner
  - communication layer
- I/O and checkpointing
  - serialization in HPC compliant format
    (HDF5, NetCDF)
  - GIS support
Software: Pandora & RepastHPC

Pandora

- version: 1
- commits: 630; Jun 17, 2016
- since 2013

RepastHPC

- version: 2.2.0
- commits: 249; last Sep 30, 2016
- since 2008

AGENT-BASED Modeling and Simulation for EXASCALE Computing
## Software: Environment Representation

<table>
<thead>
<tr>
<th></th>
<th>Pandora</th>
<th>RepastHPC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Social relationships representation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>evolving social graphs</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>edge weights</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>multiplicity</td>
<td>+</td>
<td>native (ptrs)</td>
</tr>
<tr>
<td>implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Spatial component representation</strong></td>
<td>std::map of dense matrices</td>
<td>grid projector + vector of dense matrices (val layers)</td>
</tr>
<tr>
<td>format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boundary conditions</td>
<td>von Neumann</td>
<td>von Neumann, Moore</td>
</tr>
<tr>
<td>GIS support</td>
<td>rasters with GDAL</td>
<td>-</td>
</tr>
</tbody>
</table>
Software : Domain Decomposition

“naïve” uniform rectangular mesh

Proc 00  Proc 01  Proc 11  Proc 21
Proc 09  Proc 10  Proc 20

\textsuperscript{a}RepastHPC allows to own partitioners written from scratch if one uses network projectors only (e.g., see chiSIM project by N.Collier et al.)
Software: Container for Agents

\[ \text{std::map (world)} \]

\begin{align*}
\text{id} & \quad \text{agent} \rightarrow \text{links} \\
\text{id} & \quad \text{agent} \rightarrow \text{links} \\
\text{id} & \quad \text{agent} \rightarrow \text{links} \\
\emptyset &
\end{align*}

\[ \text{std::unordered_map (shared context)} \]

Problems

- \[ \text{std::map: } O(\ln n) \]
- \[ \text{std::unordered_map?} \]

Can we do better?
Software: Container for Agents

std::map (world)

Problems
- std::map: $O(\ln n)$, non-contiguous
- std::unordered_map: non-contiguous

Can we do better?

std::unordered_map (shared context)
Software: Container for Agents

std::map (world)

Problems
- std::map: $O(\ln n)$, non-contiguous
- std::unordered_map: non-contiguous

Can we do better?
- Sure! (e.g., google::dense_hash_map)
  see, e.g., tessil benchmark of major hash maps

std::unordered_map (shared context)
## Compile-time introspection substitutes

<table>
<thead>
<tr>
<th>Pandora</th>
<th>RepastHPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>- code generation (+ some boilerplates)</td>
<td>- boilerplates</td>
</tr>
<tr>
<td>- error-prone pseudo-parsing with Python</td>
<td>- Package pattern</td>
</tr>
</tbody>
</table>

Do some work at run-time

**Why?** Can we do better?

- Compile-time introspection was a headache in C++ up to C++14 standard
- Boost::HANA (by Louis Dionne) → improved Boost::Fusion
- Apache Thrift (and its Facebook branch) since C++17 we can convert POD-struct → std::tuple non-intrusively without macro and external tooling (A. Polukhin. CppCon, 2016)
- since C++20 reflection might be part of standard
Software: Agent Types and Introspection

Compile-time introspection substitutes

<table>
<thead>
<tr>
<th>Pandora</th>
<th>RepastHPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>code generation (+ some boilerplates)</td>
<td>boilerplates</td>
</tr>
<tr>
<td>error-prone pseudo-parsing with Python</td>
<td>Package pattern</td>
</tr>
</tbody>
</table>

Do some work at run-time

Why? Can we do better?

- compile-time introspection was a headache in C++ up to C++14 standard
Software: Agent Types and Introspection

### Compile-time introspection substitutes

<table>
<thead>
<tr>
<th>Pandora</th>
<th>RepastHPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>- code generation (+ some boilerplates)</td>
<td>- boilerplates</td>
</tr>
<tr>
<td>- error-prone pseudo-parsing with Python</td>
<td>- Package pattern</td>
</tr>
</tbody>
</table>

Do some work at run-time

### Why? Can we do better?

- compile-time introspection was a headache in C++ up to C++14 standard
  - Boost::HANA (by Louis Dionne) → improved Boost::Fusion
  - Apache Thrift (and its Facebook branch)
- since C++17 we can convert POD-struct → std::tuple non-intrusively without macro and external tooling (A. Polukhin. *CppCon*, 2016)
- since C++20 reflection might be part of standard
No more destruction!

Brullov, Karl. *The Last Day of Pompeii*. © wikimedia.org
Let's construct...

What do we have?

► RepastHPC is a good tool
Let’s construct...

What do we have?

- RepastHPC is a good tool for years 2008-2013
- ... though we can do a way better with modern C++ (since C++14) and third-parties
  - compile-time introspection, lambdas + std::function, std::tuple, ...

... prototype with run-time heterogeneity mechanisms and introspection
- use SNAP graph library for dealing with social relationships
- graph coarsening with multi-scale meshes
- HDF5 I/O format and its implementation (Rafal Lichwala, PSNC)

What do we plan and work on?

- shift to extensive use of compile-time: templates, introspection, tuples, CRT, etc
- more third parties: hash tables, graph libraries
- graph coarsening with multi-scale meshes
Let’s construct...

What do we have?

- **RepastHPC** is a good tool for years 2008-2013
- ... though we **can do a way better** with modern C++ (since C++14) and third-parties
  - compile-time introspection, lambdas + `std::function`, `std::tuple`, ...
- recently we started to work on a **new framework**
  - prepare “state-of-the-art” HPC compliant primitives for GSS users
  - strive both user-friendliness (EDSL like API) and performance
Let’s construct...

What do we have?

- **RepastHPC is a good** tool for years 2008-2013
- ... though we **can do a way better** with modern C++ (since C++14) and third-parties
  - compile-time introspection, lambdas + `std::function`, `std::tuple`, ...
- recently we started to work on a **new framework**
  - prepare “state-of-the-art” HPC compliant primitives for GSS users
  - strive both user-friendliness (EDSL like API) and performance
- advances in this development... modest so far
  - **prototype** with run-time heterogeneity mechanics and introspection
  - use SNAP graph library for dealing with social relationships
  - graph coarsening with uniform meshes
  - HDF5 I/O format and its implementation (Rafal Lichwala, PSNC)
Let’s construct...

What do we have?

- **RepastHPC is a good** tool for years 2008-2013
- ... though we **can do a way better** with modern C++ (since C++14) and third-parties
  - compile-time introspection, lambdas + std::function, std::tuple, ...
- recently we started to work on a **new framework**
  - prepare “state-of-the-art” HPC compliant primitives for GSS users
  - strive both user-friendliness (EDSL like API) and performance
- advances in this development... modest so far
  - **prototype** with run-time heterogeneity mechanics and introspection
  - use **SNAP** graph library for dealing with social relationships
  - graph coarsening with uniform meshes
  - HDF5 I/O format and its implementation (Rafal Lichwala, PSNC)

What do we plan and work on?

- **shift to extensive use of compile-time**: templates, introspection, tuples, CRT, etc
- **more third parties**: hash tables, graph libraries
- **graph coarsening with multi-scale meshes**
Thank you for your patience!
Don’t blame me only! Don’t cheer me only!