Scientific Paper

Benjamin Vandersmissen[1], Lars Van Roy[2],
Evelien Daems[3], and Frank Jan Fekkes[4]

[1] benjamin.vandersmissen@student.uantwerpen.be
[2] lars.vanroy@student.uantwerpen.be
[3] evelien.daems@student.uantwerpen.be
[4] franciscus.fekkes@student.uantwerpen.be

Abstract. This paper will give an in depth view of the performed adaptations to the Stride project, as well as an overview of the findings that were obtained by comparing these additions to the original stride project.

1 Introduction

The stride project is designed to simulate and evaluate the lifetime of various infectious diseases. In order to properly analyze which factors have an effect on various diseases, the stride project allows multiple factors to be varied so that we can get an idea of which factors are the main influence for the evolution of diseases. This expansion to the original stride project will include 2 major corrections, to make the simulation more realistic, being an improved workplace size distribution and an improved household composition distribution, a major addition to the pools in which diseases can spread, being daycare and preschool pools, a minor addition to make the input more variable by adding two addition input types being HDF5 and JSON and finally a visualizer that will allow users to get a graphical overview of where the diseases are most active, along with a list of parameters to give further information of the evolution of the disease for a given location.

Other than these additions, there will also be a section about the simulation over entire Belgium, compared to isolated within Flanders. other than just the simulated area, there will also be a comparison between the data derived from the Belgian population and the data derived from the Flemish population.

2 Daycare & Preschool

3 Data Formats

3.1 JSON

3.2 HDF5

HDF5 is a scientific dataformat specifically designed to store and organise large amounts of data. In this section, we compare the sizes of the fileformats already

available in stride (protobuf and JSON) with the size of an HDF5 file and we also cover the write speed. Both these metrics are based on the configuration file *run_generate_default.xml* with only the output file changed.
As shown in table 1, HDF5 is memory-wise between protobuf and JSON. This is because protobuf is a heavily compressed format and HDF5 not, but HDF5 isn't a plain text format like JSON either. In terms of write speed, HDF5 is way worse than Protobuf and JSON. A simple explanation for this is the fact that HDF5 doesn't support streams, while JSON and Protobuf do. This means that HDF5 has to write everything with it's own procedures which isn't as efficient. HDF5 doesn't support streams, because it was originally a C library, when streams didn't exist yet and the C++ implementation is just a wrapper around the original C library.

FileFormat	**Size (Kb)**	**Average write Time (s)**
Protobuf	13.713	1.74
JSON	317.050	5.25
HDF5	125.066	30.45

Table 1. Performance of different data formats based on 10 runs of run_generate_default.xml

4 Data Visualization

5 Workplace Size Distribution

5.1 Introduction

The original stride project made use of a set workplace size distribution, being an average size of 20, that would be randomly populated. So it might be that there would be smaller and larger workplaces, but the average would be around 20, where this is not accurate. The majority of the workplaces is smaller than 20 and there are workplaces that are way larger than 20. It would therefore be more accurate to add an input file that would give a distribution of which workplace size occurs with which chance.

5.2 Implementation

The input is given in a CSV file. The data itself consists of the chance for a workplace size being within the given range, along with the bounds for that range. The tag is given within the configuration under the tag workplace_file.

The generation will consist of a major loop that will iterate over the different locations. For each location we will calculate the maximal number of workers

that is presumed to be working at this location, this will not be the exact value however, as many of the people who could be working, might not be, due to studies, or just general unemployment. After the number of workers is calculated there will be another loop, where workplaces will be randomly generated (using the provided distribution when available) until every possible worker is capable of working.

The population of the generated workplaces will start with a loop, iterating over the possible locations. For each location we will loop over the population of that location in a secondary loop. For each person we will check if the person is of age, and that the person is not a student. If this is ok, we will perform a coin flip, to determine whether or not the person is employed. If the person turns out to be employed, we will do one final coin flip, to determine whether or not the person commutes. When this is all done, we will select a random workplace that still has free space (if there are any workplaces left with free space), or a random workplace in case there were no spaces left.

5.3 Impact

To proof whether or not this addition had a major impact, and to show the size of its impact, we ran two kinds of simulations.

The first one is a simulation where either extinction of the disease occurred or the entire part of the population that was not immune would get infected. This proofs whether or not this addition has an impact on the likeliness of a disease spreading.

The second one is a simulation where there were very few people who were immune, to proof whether or not this addition has an impact on the speed with which a disease spreads.

6 Household Composition Distribution

6.1 Introduction

One more flaw of the original stride project, is that it had no regard for the differences in household size within smaller regions. It normally uses one big input file, that would give the general household configuration over the entire simulation area (in this case Flanders) but there are significant differences between households over the different provinces. We therefore wanted to be able to specify household information for each province.

6.2 Implementation

The input is given in multiple CSV files, each called after the referred province. Each file is given separately within the config file, and there are no requirements

on how many configuration files have to be given. In case there would only be a need to get specific data for Antwerp, only the Antwerp household file needs to be given. There will also be an option to specify which cities are considered major cities along with a separate household file for their configuration. If there is no need or one prefers not to use the new addition for the Households, it is always possible to just fill in the old Household tag (which was required, and still is). This will use the old implementation. This is all done using the tags:

- antwerp_household_file
- flemisch_brababant_household_file
- gen.west_flanders_household_file
- east_flanders_household_file
- limburg_household_file
- major_cities
- major_cities_household_file

7 Belgian Simulations