UPPSALA UNIVERSITET

Software optimization and parallel computing of social network construction in dairy cows

Arvid Forsberg
arvid.forsberg.6529@student.uu.se
Elias Estensen
Elias.estensen.1271@student.uu.se
Alexander Palfelt
alexander.palfelt.7363@student.uu.se

Supervisors:

Keni Ren
keni.ren@slu.se

Freddy Fikse
Freddy.Fikse@vxa.se

Background

This work is part of the international project ***Precision livestock breeding – improving both health and production in dairy cattle*** which uses positional data from real time location systems (RTLS) of cows to map their social interactions. The aim is to investigate how both disease spread, and milk production is affected by social interactions. This is done within two barns, one in Sweden and one in the Netherlands. The goal of this work is to optimize an algorithm in terms of speed and memory usage. The final algorithm should be written in Matlab and Python.

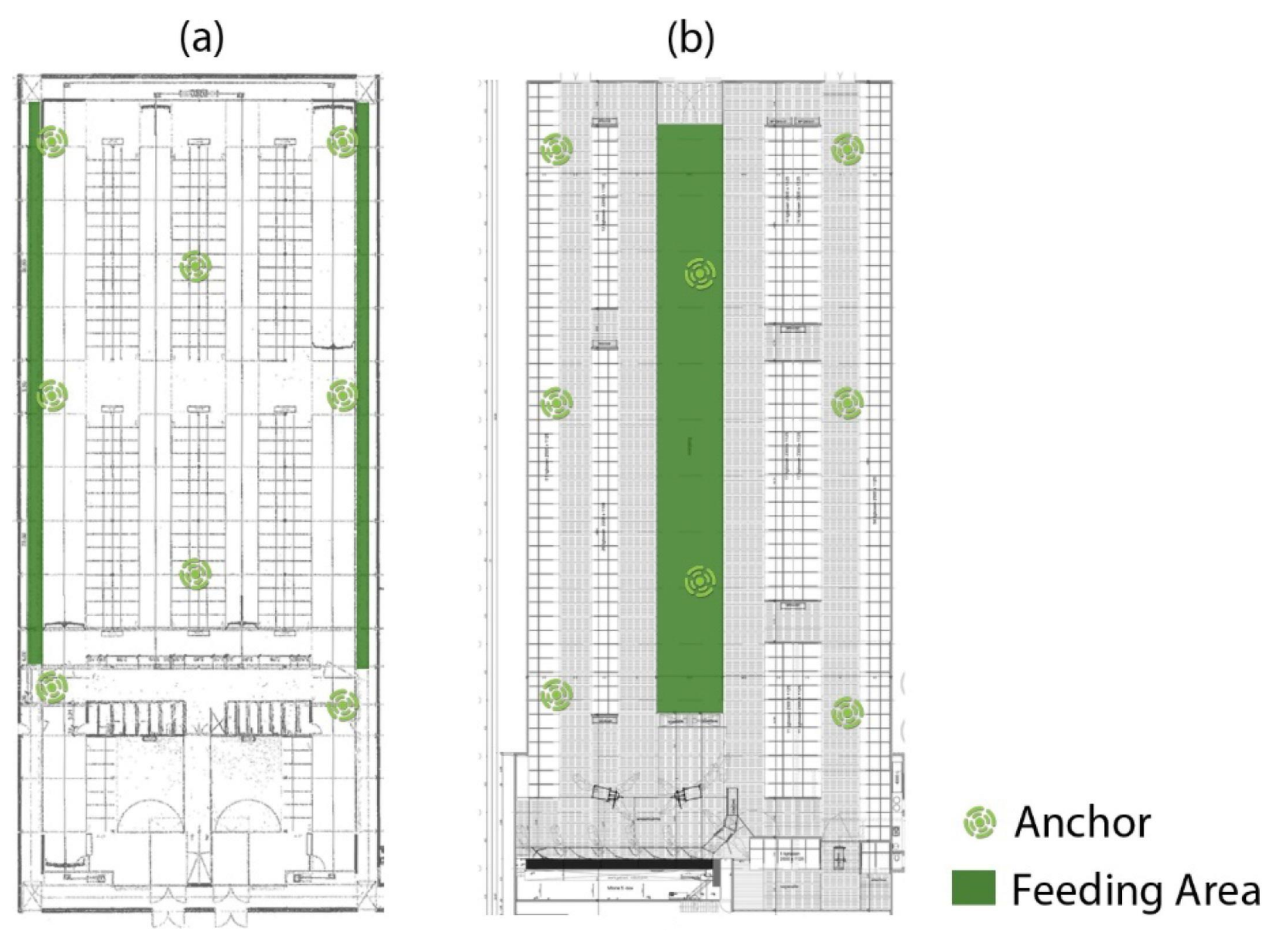

Figure 1. Schematics of the (a) Swedish and (b) Dutch barns with feeding areas and RTLS anchors indicated.

The Data

The data contains positional data for every cow (roughly 200 cows) for every second of the day. The data is unsorted, and some entries are missing.

The Algorithm

The algorithm sorts data from a single day and removes cows that are missing a lot of entries. The remaining data is then interpolated.

Interactions are then computed, where an interaction is defined as a time step where two individuals are closer than a given threshold (2.5 m). The interactions are then saved if they lasted longer than 10 minutes during the given day.

Main issues with the initial algorithm:

- excess use of memory due to storing superfluous information,
- using inefficient data structures,
- saving the distances between every cow pair instead of using the result right away,
- poor structuring,
- use of inappropriate data types.

All of the points above has been optimized, results are reported in Figure 2.

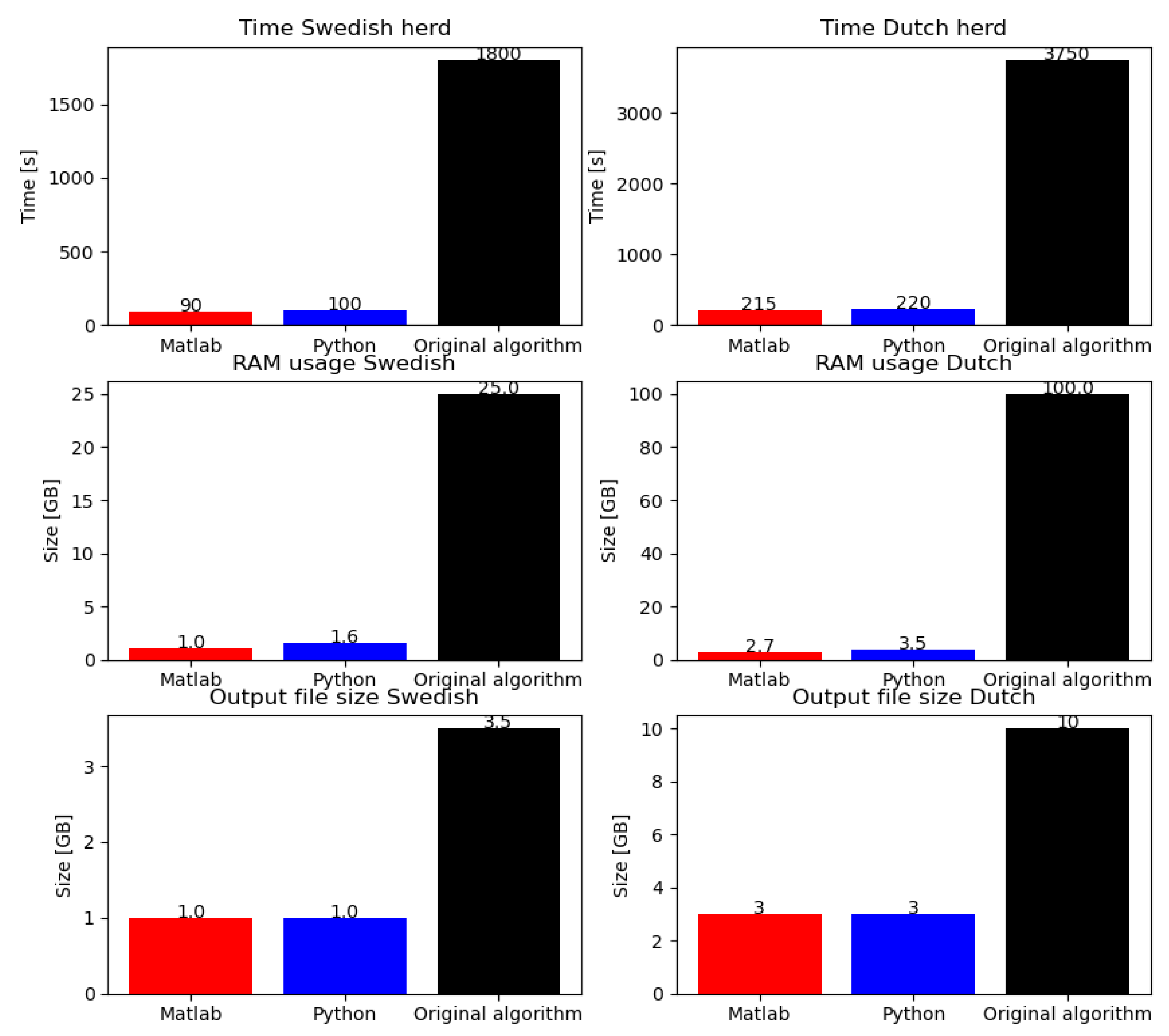

Figure 2. Performance measures of the new algorithms compared to the original. Significant improvement can be seen in both time and memory demands.

Parallelization

By utilizing the multi-core architecture of modern CPUs calculations of several days can be carried out simultaneously. This is done in a shared memory sense in Matlab using the multiprocessing toolbox and in Python using the *concurrent.futures* module. When running the algorithms on the UPPMAX high performance computing cluster, two weeks of data can be processed in roughly two minutes. Since the original code ran using equivalent data in approximately 420 minutes, this corresponds to a speedup of 210.

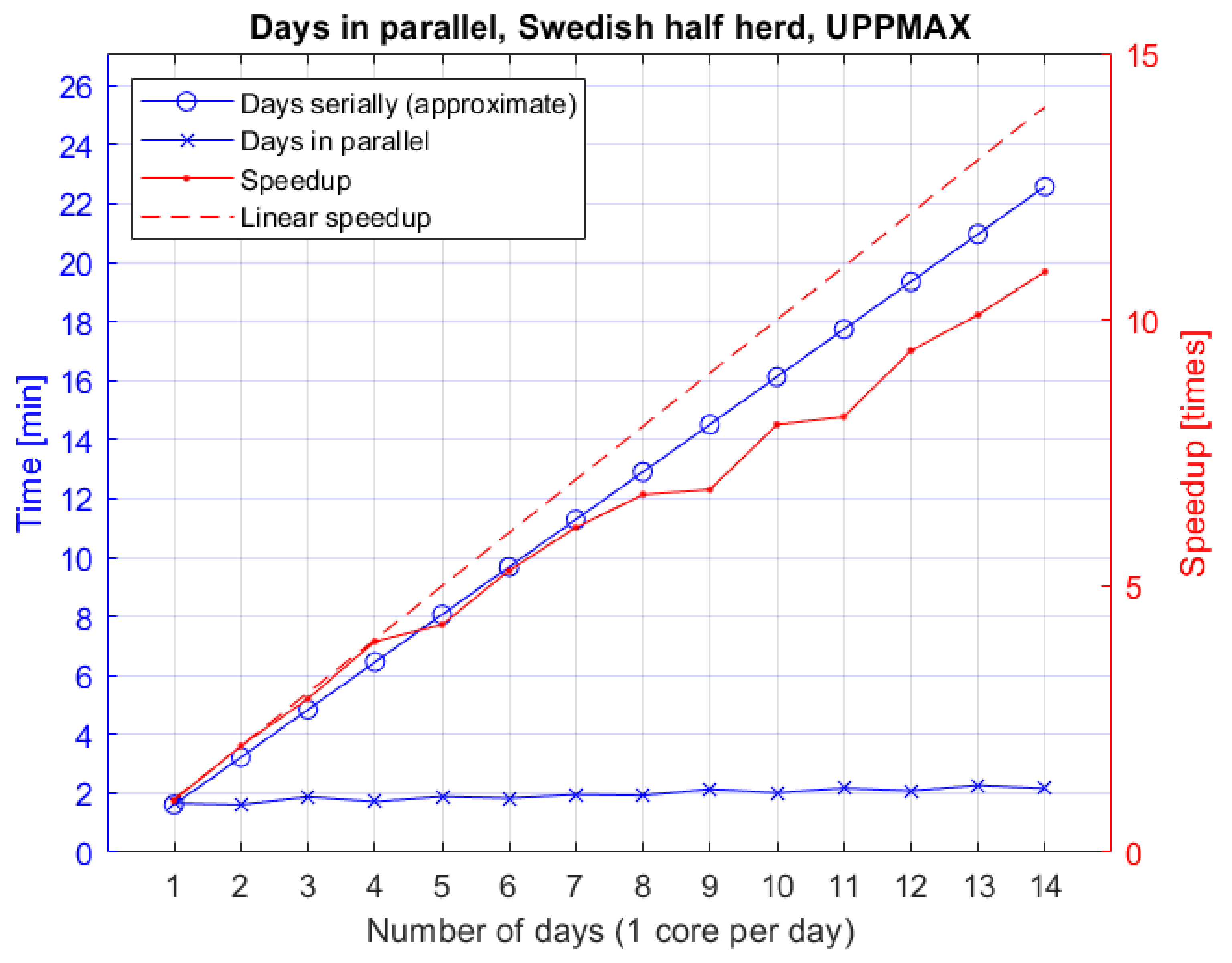

Figure 3. Parallel scaling of the Matlab code running multiple days simultaneously. Similar results were obtained for the Python code