Rob Fitzgerald
CSCI 7551
2016-08-29

Massively concurrent agent-based evolutionary computing
by D. Krzywicki, W. Turek, A. Byrski, M. Kisiel-Dorohinicki

Much of the area of study in developing evolutionary algorithms has been developed without consideration of asynchronous implementations across distributed systems. These explorations began in an era of software design ruled by object-oriented, monolithic and synchronous services. Stages of algorithmic "evolution" for each agent would occur at a global time interval. All data would be stored in memory. These techniques would eventually become limitations in the area of distributed evolutionary computing. D. Krzywicki et. al. propose an algorithm design which allows agents to engage in the procedures associated with evolutionary computing (reproduction, death, and combining or fighting) in such a way that allows agent evolution to occur based on decisions at the agent level and within a distributed context. They explore these algorithms and compare their performance when implemented in Erlang and in Scala's Akka framework. Their findings illustrate how their algorithms allow for increased agent reproduction which, when applied to a benchmark test, result in monotonic improvements to the fitness per reproductions of the result as concurrency is increased. They found greater optimization using Scala-Akka, which they attributed to Scala's memory management and message passing.

Reference

D. Krzywick et. al., "Massively concurrent agent-based evolutionary computing" in *Journal of Computational Science* vol 11 (2015), pp. 153-162.

URL

http://ac.els-cdn.com/S1877750315300041/1-s2.0-S1877750315300041-main.pdf?_tid=79eef28c-6e12-11e6-8fb3-00000aacb361&acdnat=1472493811_ca6c7d3e0aac58d39b9746e1454bbc7b