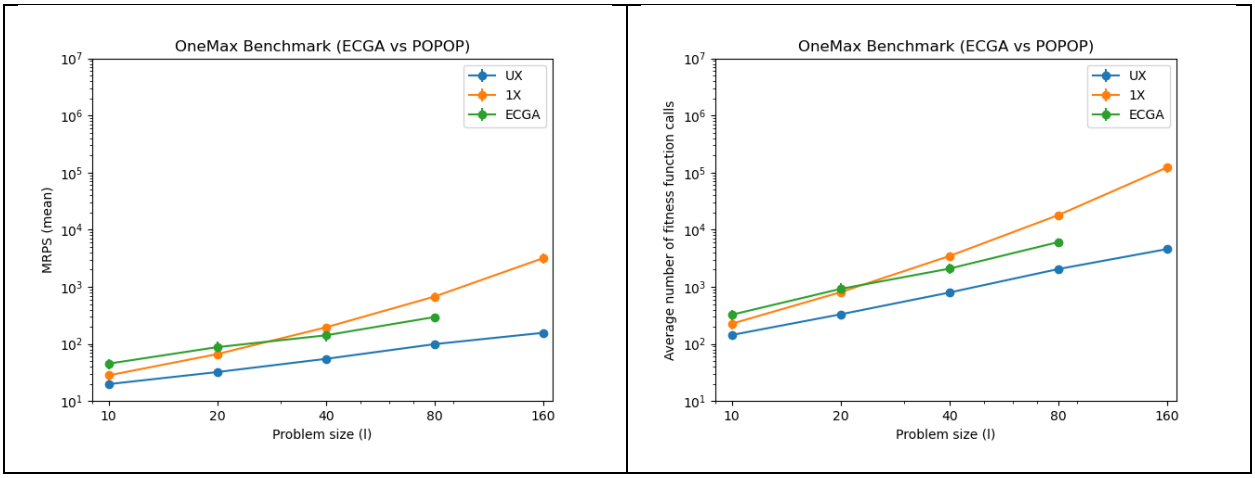
Bài Báo Cáo

Môn: **Mạng Nơ Ron và Thuật Giải Di Truyền** | CS410.K22.CNCL – GV: *Lương Ngọc Hoàng*
Đồ án: **Báo cáo thống kê hiệu suất thuật toán ECGA**

➢ One Max

Table 1 Kết quả thực nghiệm trên hàm One Max

	ECGA	
Problem size	MRPS	# Evaluations
10	45.2 (9.13)	325 (64.75)
20	88 (23.25)	921.68 (224.54)
40	141.6 (29.73)	2083.76 (421.12)
80	296 (40.63)	6067.84 (788.28)
160	-	-



Dựa vào bảng kết quả và đồ thị trên, ta rút ra được:

- Bản cài đặt với thuật toán ECGA có thể tìm được lời giải tối ưu cho hàm OneMax với mọi kích thước vấn đề (problem size). Trong quá trình chạy thực nghiệm thuật toán, vì chi phí về thời gian quá cao khi thực hiện bisection, do đó người viết chưa thể tìm được kết quả nghiệm thu đủ 10 lần chạy trung bình / std cho bài toán với kích thước vấn đề 160. Do đó, bảng kết quả và biểu đồ không có kết quả.
- Hiệu suất của ECGA (MRPS, số lần gọi hàm) là trung bình, yêu cầu quần thể tối thiểu và hội tụ nhanh hơn so với s-GA với phép lai đồng nhất (UX). Tuy nhiên hiệu suất của ECGA vẫn thua s-GA với phép lai đồng nhất.

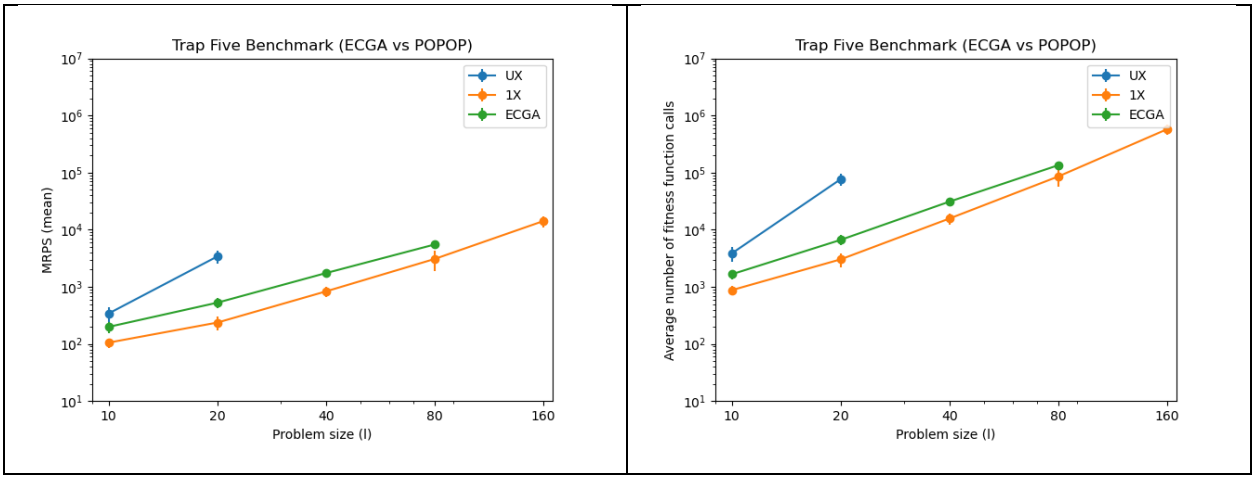
- Ở kích thước vấn đề = 20, ECGA cần kích thước quần thể lớn hơn so với s-GA sử dụng 1X. Tuy nhiên, chênh lệch là không đáng kể. Điều đó cũng tương tự đối với số lần gọi hàm của ECGA ở kích thước vấn đề 20, số lần gọi hàm của EGCA xấp xỉ bằng s-GA với 1X.
- Nhìn vào biểu đồ, ta thấy, khi kích thước vấn đề gia tăng, quần thể tối thiểu để giải quyết bài toán cũng tăng, song chênh lệch là không cao. Ta thấy, với problem size = 20, MRPS = 88; với problem size = 40 (tăng gấp đôi) , MRPS = 141.6 (tăng xấp xỉ 1.6 lần).

Kết luận:

- ECGA cho hiệu suất giải quyết quyết bài toán OneMax là trung bình (phức tạp không gian và thời gian).
- Đối với bài toán đơn giản (OneMax) không cần phải tìm mô hình, hiệu suất của ECGA là tương đối tốt.

➢ Trap Five

ECGA		
Problem size	MRPS	# Evaluations
10	199.2 (43.3)	1662.4 (326.5)
20	528 (109.22)	6642.56 (1330.56)
40	1740.8 (130.53)	31047.68 (2276.32)
80	5504 (221.7)	134668.8 (3967.65)
160	-	-



Dựa vào bảng kết quả và đồ thị trên, ta rút ra được:

- Bản cài đặt với thuật toán ECGA có thể tìm được lời giải tối ưu cho hàm Trap Five với mọi kích thước vấn đề (problem size). Trong quá trình chạy thực nghiệm thuật toán, vì chi phí về thời gian quá cao khi thực hiện bisection, do đó người viết chưa thể tìm được kết quả nghiệm thu đủ 10 lần chạy trung bình / std cho bài toán với kích thước vấn đề 160. Do đó, bảng kết quả và biểu đồ không có kết quả.

- Hiệu suất của ECGA (MRPS, số lần gọi hàm) là tốt hơn so với s-GA sử dụng UX, nhưng ECGA vẫn kém hơn so với s-GA sử dụng 1X.
- Trong quá trình thực nghiệm, tốc độ ECGA là tương đối kém vì phải phát sinh mô hình và đánh giá để chọn ra mô hình tốt nhất cho thế hệ sau. Dù số lần gọi hàm của ECGA là trung bình, có nghĩa ECGA hội tụ được kết quả cũng tương đối nhanh, tuy nhiên thời gian tính MC (model complexity) là khá cao.
- Dù được thiết kế để giải quyết bài toán khó, cần có được model để giải; song với hàm Trap Five, ECGA vẫn không mang lại hiệu suất tốt hơn s-GA sử dụng 1X.
- Tuy nhiên, có thể do tính chất của phép lai 1X mà với bài toán Trap Five, sGA có hiệu suất tốt hơn.

Kết luận:

- Với hàm Trap Five, ECGA cho hiệu suất tìm ra vấn đề là tương đối tốt.
- Tuy nhiên, chi phí phát sinh và đánh giá mô hình là khá cao.

Tổng kết:

- ECGA có thể giải quyết được bài toán đơn giản cũng như bài toán có tri thức về mô hình.
- Trong quá trình thực nghiệm, ECGA cần kích thước quần thể lớn đề có thể tìm ra được mô hình đúng của bài toán.
- Với kích thước quần thể nhỏ, ECGA tìm được lời giải tối ưu với một số kích thước vấn đề, dù vậy ECGA vẫn chưa tìm ra đúng mô hình của bài toán.
- Một điểm yếu nữa của ECGA là chi phí tính toán MDL (Minimum Description Length), cụ thể là MC (Model Complexity) là rất cao, khi kích thước vấn đề tăng.
- Ưu điểm của ECGA là không yêu cầu kích thước quần thể phải quá lớn để có thể giải quyết được bài toán phức tạp.