An Investigation into how concepts of modularity affect the evolution of complex morphologies

Liam Ranil Fernando

School of Electronics and Computer Science
University of Southampton

January 12, 2010

Abstract

There are many different ways in which complex morphologies can be represented. While a simple string representation could be sufficient, often the most impressive artificial life simulations utilise Context Free Grammars (1994, Karl Sims) or Recursive Tree Structures. When modelling a complex morphology using these encodings, it is possible to harness the creatures complex modularity to create more sensible and fit individuals. This article aims to compare and contrast the varying affects of evolutionary algorithms which utilise or disregard the organisms modularity.

1 Introduction

The affects of modularity in the evolution of complex structures are very closely related to the success and role of cross-over in sexual populations. 'The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance' [3] famously investigated this issue in 1992, proposing a 'Royal Road' function that theoretically should allow genetic algorithms (algorithms that utilise cross-over) to produce superior results. The 'Royal Road', rewarded the successful recombination of lower level segments into larger adjoining segments. It was initially thought that a GA with cross-over would solve this problem efficiently. However, the results showed that instead of quickly assembling lower level segments to produce an individual of higher fitness, there existed a high latency between the discovery of lower level segments, and their adjoining counterparts.

The results of this research [3] would imply that there is little point utilising a GA to create artificial life, yet people continued to do so, with impressive results. In 1994, Karl Sims revolutionised artificial life when he evolved complex individuals to perform tasks in a physical environment [6]. In his experiments, he harnessed the power of context free grammars to represent creatures of a modular nature. Specifically, he used tree grafting techniques to exploit the modularity in his system. While his experiments produced interesting results, he did not prove that the modular structure he implemented provided any fitness benefits to the creatures it produced. Several years later, 'A building-block royal road where crossover is provably essential' [7] revisited this problem, showing that the Royal Road failed as it was an easy stylised problem to solve. It then demonstrated that a GA could be exploited to solve problems that required semi-independent search, using trap functions. This is not decimilar to the fitness landscape of movement by a creature in a physical world - where several combinations of body structure and muscle contractions could lead to sub-optimal fitness peaks.

The papers researched so far have investigated the strengths of a GA using encoded bit-strings. However, most artificial life experiments prefer to use context free grammars or recursive tree structures. Mitchell, Forrest and Holland stated that the 'success of the GA on a particular function is certainly related to how the function is encoded' [3], implying that these more modular encodings would exhibit better results with a GA. Through the creation of a Sims-like investigation, a comparison may be made between the fitness of creatures evolved using varying naive and modular-aware algorithms.

2 Background Research and Existing Work

As this paper will investigate whether the modularity of a creatures morphology affects evolution, it is vital that an advanced representation of a creature be developed. In an earlier paper [1], a basic framework for evolving creatures was established. Written in Java, it utilised the Java Monkey Engine with ODE for Java Physics Engine. While the framework successfully provided a tree-based notation for defining Individuals with muscles and sensors, the resulting system was slow and unable to evolve interesting creatures as a result of an inaccurate physics engine.

Learning from these mistakes, a new solution was developed in C#, instead using a wrapper for nVidia's PhysX physics engine [4] - a hardware accelerated physics engine through CUDA.

The new system differed from the older representation in several ways. The original system placed each Individual in an incubation environment where they could train their muscles using an evolutionary strategy (ES). While in theory the use of an ES mimicked 'real-life' skill learning, there seemed to be no real advantage to this system when implemented, and it added complexity to the system, requiring a number of isolated physics engines per Individual. As the new system was using PhysX, it was only possible to create one physics engine at any time. This made the use of a muscle training ES unfeasible, therefore resulting in the abolishment of this feature.

The resulting system was also far cleaner in design as re-writing the framework allowed for refactoring, removing un-necessary bloat. To obtain better than real-time performance, further optimisations were made by segmenting visualisation away from the main framework. Instead of a live visual output of creatures in the simulation, the dimensions, positions and rotations of each body part were logged into a file at each stage of the simulation. This would allow if required, a visualisation tool to be created to analyse the creatures, without the need to calculate any complicated physics. However, for the purposes of this investigation, it was not necessary to create such a visualisation tool, as creatures movements could easily be deciphered from the logs in enough detail to show correct operation.

Despite the changes, the underlying concept remained true. The system contained one static Environment, to which a Mode and EA (evolutionary algorithm) could be attached. The Environment could then generate a specified number of random Individuals to fill the population. Each Individual comprised of one PhenoTree, which itself could contain one Phenotype and six more PhenoTree's (one parent, and five children), attached to their parent by a Joint. The notion of a Phenotype is akin to Karl Sims Segments [6], each Phenotype with a variable dimension, position and rotation. In the system, Phenotypes were restricted to cube-shapes for simplicity and performance, while Joints were restricted to Rigid, and Vertical or Horizontal Hinges. A Ball and Socket Joint was also developed, however this was disabled in the experiments after it was suspected the Joint was operating incorrectly.

Complete class diagrams of the original and revised systems can be seen in Appendix A.

3 Experiment

To test the hypothesis that modularity affects the evolution of complex morphologies, this paper will implement several generational and steady-state algorithms on a fixed size population. In this first experiment, the algorithms will compare mutation and sexual based evolution strategies. Through the analysis of the minimum, maximum and average fitness improvements over each generation, it will be possible to contrast and compare the results.

3.1 Mutation Evolution Strategies

In this experiment mutation will occur at a phenotypic level. Phenotype's will mutate their dimension, position and muscle angle with a mutation probability of 0.2, before correcting to compensate for the changes. Three different selection strategies implemented phenotypic mutation as the sole method for altering an Individual. The mutation selection operators used in this experiment were Fitness Proportional Selection and Rank Order Selection. Fitness Proportional Selection sorted the population in descending order of fitness (highest first), and then allowed cloned and mutated the population according to the percentage of total fitness they provided. Rank Order Selection also ordered the population in descending order of fitness - each Individual being given a rank R. The reproductive probability of an Individual was calculated as 1/R.

3.2 Sexual Evolution Strategies

Tournament Selection was used in the sexual population. This selection operator took a random 10% of the population, selecting the most fit Individual to become a parent. The process was then repeated so that two fit Individuals had been selected to reproduce sexually.

The sexual evolutionary strategies implemented in this investigation were devised to exploit the modular nature of a creatures morphology. As an Individual is defined as a recursive PhenoTree, it was easy to visualise the structure of an Individual as a graph. Each node of the graph represented a PhenoTree containing one Phenotype, and the arcs between the nodes represented the pairing of a Joint and PhenoTree. After applying edge recombination, the resulting Individual retained the common qualities of its parent's. As both parents had survived environmental selection pressures, it was likely that the common feature had allowed the parent's to be superior in the population.

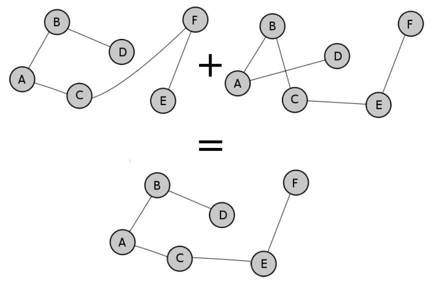


Figure 1: Edge Recombination

In addition to this, the sexual population experienced mutation at a phenotypical level (as described earlier) and through tree grafting. Tree grafting is where nodes of the Individual are switched according to a mutation probability (set at 0.2 in the experiment). This allows the population to explore a larger area of the fitness landscape, than using phenotypical mutation alone.

Both the mutation and sexual strategies described above were implemented as generational and steady-state models. If after a series of simulations, the sexually reproducing models gained superior results, then it would help show that modularity in the evolution of complex creatures is important.

Each model was simulated 20 times with a population size of 100, over 20 generations.

4 Results

During the simulations, data was logged to a CSV file, detailing metrics about the simulation after each generation. A selection of this data can be found in appendix B.

4.1 Generational Simulations

The results from table 2 show that the algorithm with cross-over that utilises the creatures modular morphology performs far better than any of the mutation based algorithms. As the simulation was generational, the mutation-based algorithms were unable to retain as much of the previous generations genetic makeup. This is as the new generation of Individuals was entirely comprised of mutations to the existing generation. In contrast, the sexual population were able to pass on their genes through edge recombination. When the two parents shared the same gene, the child inherited it, explaining the dramatic performance enhances shown by the cross-over algorithm.

Some interesting traits of the mutation-based algorithms were visable too. Initially, the fitness proportional algorithm declined in average fitness, despite one Individual comprising of a large share of the populations fitness. It is likely that this Individual was on a spike in the fitness landscape. As the Individual was much more fit than the rest of the population, the population was overtaken by creatures of a similar genome. When these creatures mutated, the average fitness of the population plumeted, as shown in figure 2. When a good solution was found, a rapid increase in average fitness was seen as the Individual fixed on the population. However, this time the solution was in an area of the fitness landscape with a more shallow gradient, allowing the population to improve.

The Rank algorithm performed similarly to the fitness proportional algorithm. As it is harder for an Individual to fix on the population, changes in the average fitness were slower, preventing the initial plumet in fitness, but leading to a slow decline later on.

4.2 Steady State Simulations

Under steady state conditions, the mutation algorithms were able to perform better than they did in the generational tests. This is as when no better solution was found, the algorithm would retain the previous version of the Individual. However, it was the cross-over algorithm that conclusively outperformed the other algorithms. Now able to retain older Individuals, the algorithm became even more affective, by generation 18 obtaining a 1828% fitness improvement over the starting population.

5 Further Investigation

It was clear from the results of the first experiment (figure 2, tables 1, 2, 3 and 4), that the individuals evolved through cross-over showed a far greater fitness improvement than those evolved through mutation alone. This would back up the hypothesis that cross-over provides benefits as both parents have already succumbed to selection pressures of the environment, and that added variation can be introduced - reducing the risk that a population fix on a sub-optimal solution.

While these results back-up the hypothesis, they do not prove it. It may be that these benefits were provided as a result of cross-over and not the exploited modularity of this representation. To prove that this is not the case, it would be required to show that the results obtained that exploit

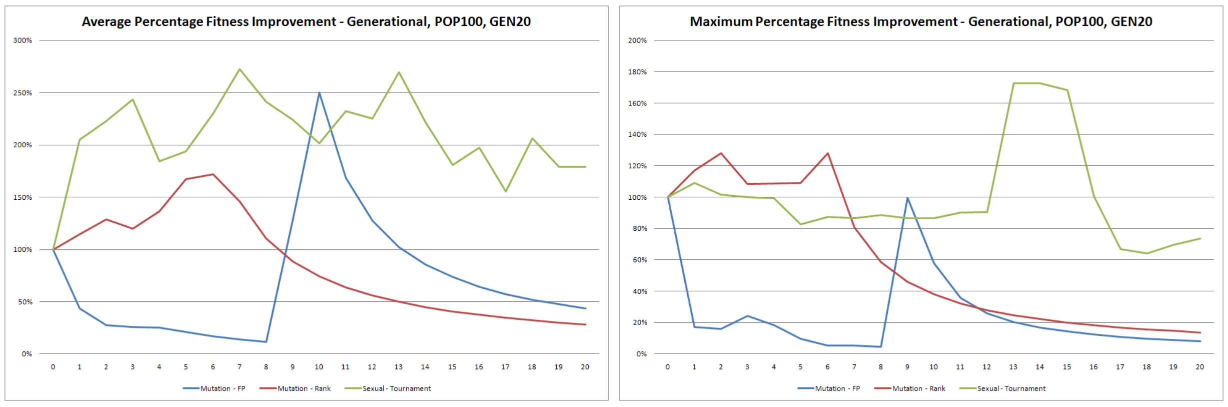


FIGURE 2: Comparison of the average and maximum percentage fitness increase in a generational model with a population size of 100, over 20 generations

modularity are more fit than those in a system which do not exploit modularity. Rather develop a different cross-over algorithm that didn't utilise the creatures modularity, Jones argued that by using one randomly generated parent, this would approximate such a system [2]. Furthermore, by using the same algorithm a fair test would be created. If the original cross-over algorithm outperformed the revised algorithm using one randomly generated parent, this would further support the hypothesis.

6 Analysis

After running simulations, the results obtained trended towards better performance when exploiting modularity. While the average fitness gap showed that for generational evolution, cross-over which utilises modularity has a clear advantage, the gap was much closer when simulating a steady state. This is likely because the steady state algorithm allows good solutions to be retained, encouraging new Individuals towards that solution.

7 Conclusions

While the results of these experiments show strong signs that a GA will exploit modularity in complex creatures better than mutation alone, the results are far from conclusive. There can be no doubt that the mutation only algorithms would have benefited from Joint and PhenoTree mutation, allowing for a far wider fitness landscape to be explored. This was further compounded by an inability to develop a reliable muscle driven Ball and Socket Joint, which ultimately restricted creatures to crawling and tumbling actions in Run Mode. However, the impressive performance of the GA should not be completely disregaurded. While the framework developed was not as complex as the system developed by Sims [6], it would have been interesting to see how the expanded fitness landscape and co-evolutionary environment's he developed [5] would have affected performance of the GA.

From the results of this experiment, it would be forseable that a wider fitness landscape would require a larger population size for successful edge recombination to occur. Here, the GA's performance would be dependent on the distribution of Individuals on the fitness landscape. If there was a wide range of initial approaches to movement in the population, then it would follow that the GA would be more likely to avoid traps in the fitness landscape and find a more optimal solution. However, further research would be required to prove this.

Acknowledgements

Thanks to Dr Richard Watson for his input into this research, and Rob Shorter, who kindly lent me his programmable GPU after the machine I was using ceased to turn on. His generosity allowed me to continue my investigation and produce results, for which I am very grateful.

References

[1] Liam Fernando. Evolution of complexity. 11 2009.

[2] Terry Jones. One operator, one landscape. Working Papers 95-02-025, Santa Fe Institute, February 1995.

[3] Melanie Mitchell, Stephanie Forrest, and John H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In *Proceedings of the First European Conference on Artificial Life*, pages 245–254. MIT Press, 1991.

[4] nVidia. nvidia physx for developers. http://developer.nvidia.com/object/physx.html.

[5] Karl Sims. Evolving 3D morphology and behavior by competition. *Artif. Life*, 1(4):353–372, 1994.

[6] Karl Sims. Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 1994. ACM.

[7] Richard A. Watson and Thomas Jansen. A building-block royal road where crossover is provably essential. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1452–1459, New York, NY, USA, 2007. ACM.

A Class Diagram of Evolution Framework

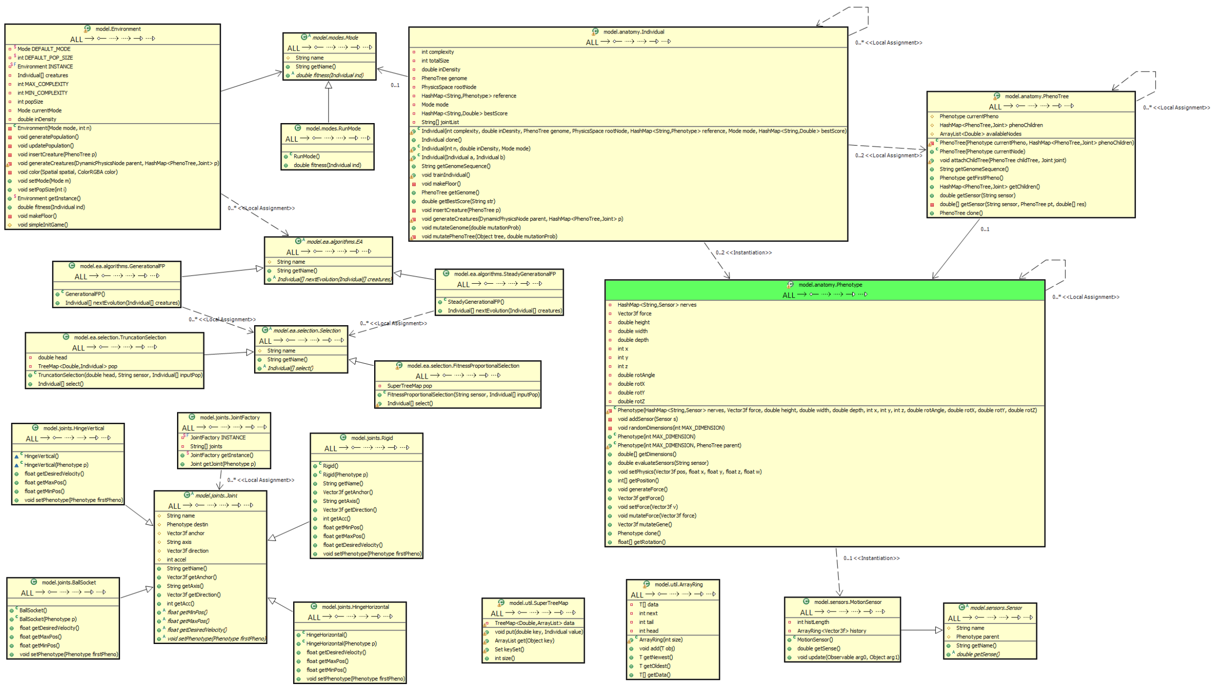


FIGURE 3: Class Diagram : Original Framework using the Java Monkey Engine and ODE Physics.

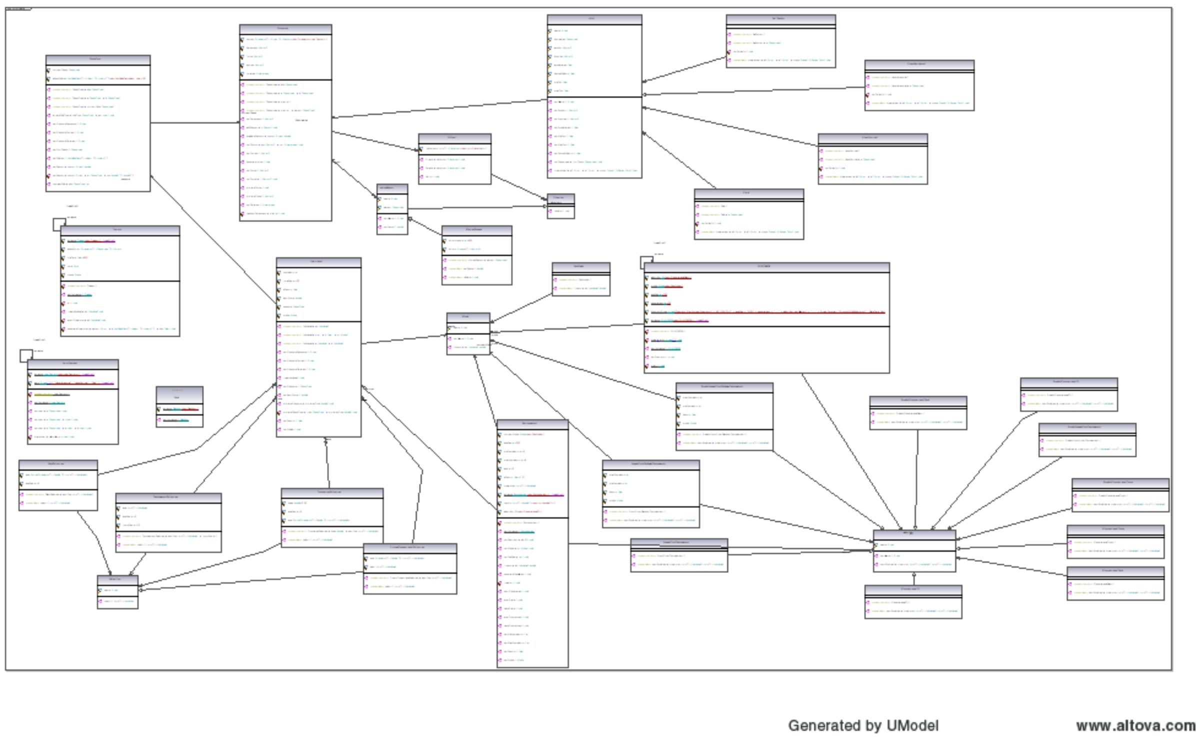
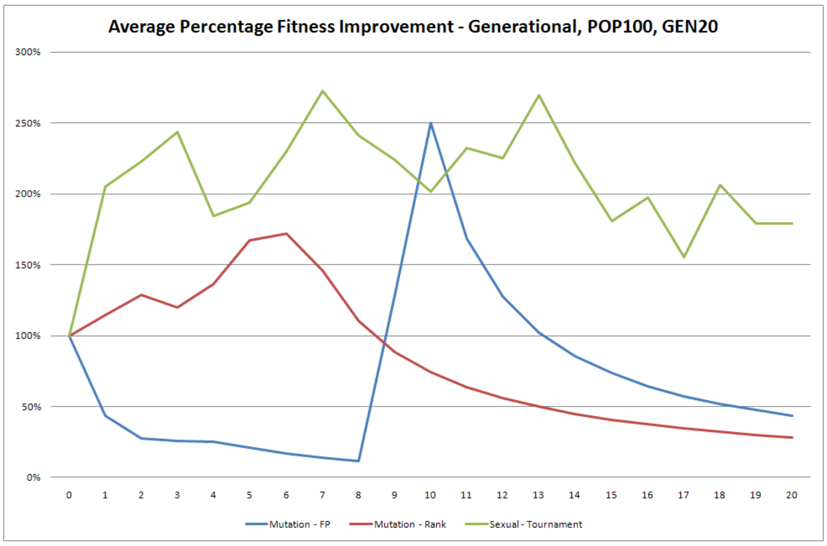


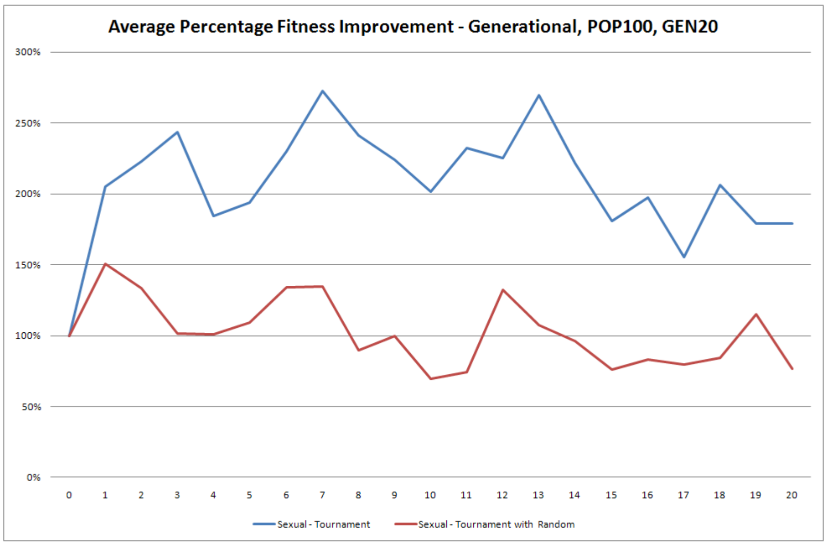
FIGURE 4: Class Diagram : Improved Framework using C# and StillDesign's PhysX.Net Wrapper.

B Evolved Creatures - Data



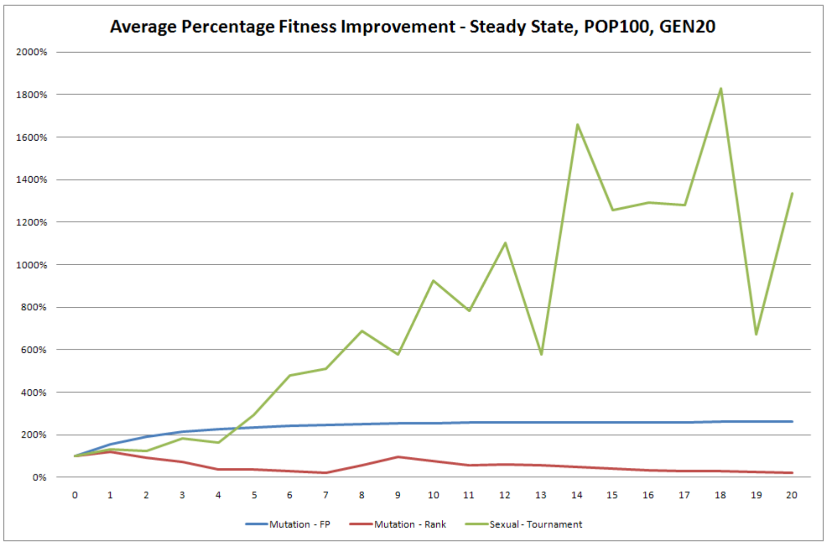
Average	Generational - Mutation			
Gen	FP - Fit	FP - %	Rank - Fit	Rank - %
0	11.75	100%	51.98	100%
1	5.12	44%	59.62	115%
2	3.26	28%	66.80	128%
3	3.01	26%	62.36	120%
4	2.94	25%	70.88	136%
5	2.45	21%	86.97	167%
6	1.96	17%	89.47	172%
7	1.67	14%	75.83	146%
8	1.33	11%	57.33	110%
9	15.08	128%	46.11	89%
10	29.41	250%	38.56	74%
11	19.82	169%	33.13	64%
12	14.97	127%	29.05	56%
13	12.03	102%	25.87	50%
14	10.06	86%	23.31	45%
15	8.64	74%	21.22	41%
16	7.57	64%	19.47	37%
17	6.72	57%	17.98	35%
18	6.09	52%	16.71	32%
19	5.58	47%	15.61	30%
20	5.13	44%	14.64	28%

TABLE 1: An average comparison of mutation evolution using Fitness Proportional (FP) and Rank Selection over 20 generations with a population of 100.



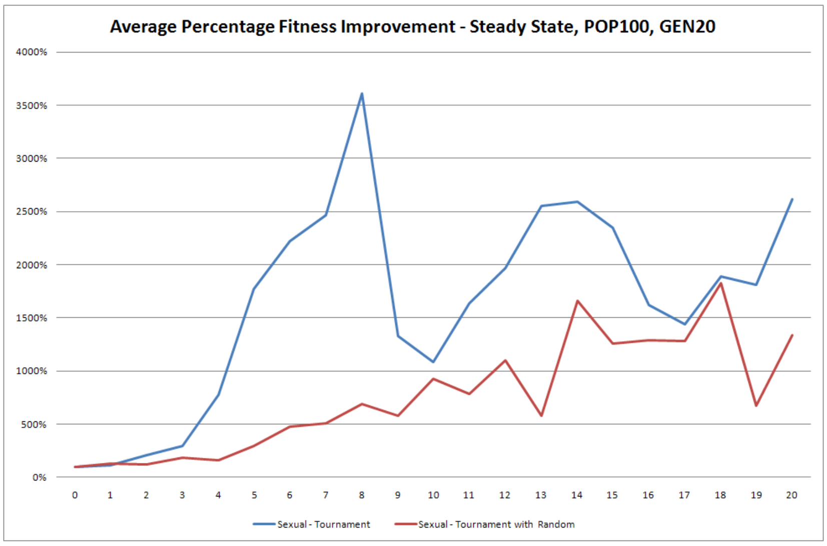
Average	Generational - Sexual			
Gen	T - Fit	T - %	T/R - Fit	T/R - %
0	25.88	100%	109.79	100%
1	53.14	205%	165.35	151%
2	57.74	223%	146.22	133%
3	63.02	243%	111.47	102%
4	47.66	184%	110.54	101%
5	50.16	194%	119.92	109%
6	59.59	230%	147.17	134%
7	70.60	273%	147.53	134%
8	62.41	241%	98.65	90%
9	58.00	224%	109.49	100%
10	52.24	202%	76.41	70%
11	60.06	232%	81.47	74%
12	58.30	225%	144.90	132%
13	69.83	270%	118.10	108%
14	57.35	222%	105.48	96%
15	46.82	181%	83.60	76%
16	51.06	197%	91.07	83%
17	40.18	155%	87.30	80%
18	53.33	206%	92.56	84%
19	46.31	179%	126.17	115%
20	46.38	179%	84.23	77%

Table 2: An average comparison of sexual evolution utilising (T) and not utilising modularity (T/R) over 20 generations with a population of 100.



Average	Steady State - Mutation			
Gen	FP - Fit	FP - %	Rank - Fit	Rank - %
0	11.24	100%	4.35	100%
1	17.53	156%	5.25	121%
2	21.54	192%	3.98	92%
3	23.99	213%	3.09	71%
4	25.48	227%	1.68	39%
5	26.48	236%	1.58	36%
6	27.18	242%	1.29	30%
7	27.70	246%	1.02	23%
8	28.11	250%	2.56	59%
9	28.43	253%	4.25	98%
10	28.69	255%	3.35	77%
11	28.89	257%	2.44	56%
12	28.99	258%	2.61	60%
13	29.06	259%	2.54	58%
14	29.12	259%	2.15	49%
15	29.17	260%	1.76	40%
16	29.22	260%	1.48	34%
17	29.26	260%	1.32	30%
18	29.30	261%	1.21	28%
19	29.33	261%	1.09	25%
20	29.36	261%	0.97	22%

TABLE 3: An average comparison of mutation evolution using Fitness Proportional (FP) and Rank Selection over 20 generations with a population of 100.



Average	Steady State - Sexual			
Gen	T - Fit	T - %	T/R - Fit	T/R - %
0	16.22	100%	10.96	100%
1	18.71	115%	14.45	132%
2	34.38	212%	13.82	126%
3	48.06	296%	20.23	185%
4	125.52	774%	18.00	164%
5	286.98	1769%	32.39	295%
6	360.75	2224%	52.56	479%
7	400.08	2466%	56.07	512%
8	585.34	3608%	75.34	687%
9	215.71	1330%	63.19	576%
10	175.87	1084%	101.47	926%
11	266.23	1641%	85.84	783%
12	319.16	1967%	120.84	1102%
13	413.73	2550%	63.45	579%
14	420.87	2594%	181.95	1660%
15	380.47	2345%	137.71	1256%
16	263.60	1625%	141.59	1292%
17	233.75	1441%	140.26	1280%
18	305.99	1886%	200.35	1828%
19	293.86	1811%	73.73	673%
20	424.91	2619%	146.55	1337%

TABLE 4: An average comparison of sexual evolution utilising (T) and not utilising modularity (T/R) over 20 generations with a population of 100.