Metaphor for learning: an evolutionary algorithm

Jody Lee House
Dept. of Electrical and Computer Engineering
Oregon Graduate Institute
Beaverton, OR 97006
503-748-1254
jhouse@ece.ogi.edu

Alexander Kain
Dept. of Computer Science and Engineering
Oregon Graduate Institute
Beaverton, OR 97006
503-748-1539
kain@cse.ogi.edu

James Hines
Sloan School of Business
Massachusetts Institute of Technology
Cambridge, MA 02139
617-253-9413
jhines@mit.edu

Abstract

The organizational algorithm is examined as a computational approach to representing interpersonal learning. The structure of the algorithm is introduced and described in context to the simple genetic algorithm. A comparison is made of the performance of both algorithms with respect to three different test functions: a simple single-peaked function, the standard Matlab "peaks" function (Mathworks 1998), and a multiple occurrence of a single-optimum Gaussian function. Algorithm performance is discussed relative to traditional optimization figures of merit as well as with respect to a learning analogy.

1 INTRODUCTION

Interpersonal human learning is similar to the simple genetic algorithm (Hines 1997). In this paper we first present the context in which we are interested in such learning, then we give an explicit representation of this learning using the organizational algorithm, and finally we consider the performance of human learning by benchmarking the organizational algorithm to the simple genetic algorithm. We make this comparison using three different test functions: a simple single-peaked function, the standard Matlab "peaks" function, and a multiple occurrence of a single optimum Gaussian function. The algorithm performance is discussed relative to traditional optimization figures of merit as well as with respect to a learning analogy.

Context. For our research learning is important because it is at the heart of organizational evolution – a process (and perhaps the only process) by which companies, schools and other human associations could improve over time. At its heart, improvement is a form of interpersonal learning – a process by which one person passes on ideas to another.

Organizational improvement is difficult because organizations are complex. The complexity of General Motors, Hewlett Packard, AOL, a law firm or even a mom- and pop grocery store exceeds by many orders of magnitude anyone's ability to understand and deterministically control. The complexity is both in the raw number of interacting entities (employees, machines, furniture, orders, customers, roads…) and in the feedback loops that interconnect these entities. Humans – even augmented by computers – are incapable of storing the massive detail (Simon 1976) required for a comprehensive understanding of organizations and are notoriously poor at understanding the implications of intertwined positive and negative feedbacks (Kofman, Sterman et al. 1997).

As a result of our cognitive limitations, organizational improvement is unlikely to come from anyone ″figuring it all out″. Instead, improvement will likely come from an evolutionary process. Evolution is important to organizational improvement precisely because its processes do not depend on anyone figuring out why things work or fail to work. We need to be more precise about what evolves in organizations.

We believe the most important thing that evolves in an organization is a policy. Indeed, we view policies as quite comparable to biological genes. By *policy* we mean an explicit or implicit decision rule in the usual system dynamics sense (Forrester 1961). For example, a manager might set prices by the implicit rule: Raise prices when inventories are low, and lower prices when inventories are high. The policy is *implicit* as long as it remains unspoken or unwritten. Articulating the policy, perhaps by recording it in a policy manual, could make it more explicit. Of course, people might change their approach to pricing even without updating the manual. In this case, the new approach would be a policy, while the old procedure recorded in the manual would no longer be a policy in our use of the term. A policy is a rule or procedure that people actually use to make a series of decisions. In this case, the policy gives rise to a

continuing stream of particular decisions to raise or lower price.

A policy in an organization is comparable to a gene in a cell. A gene is a segment of DNA (or, in some organisms, RNA) that acts as a set of instructions for the ongoing production of a particular protein. The proteins then catalyze reactions in the cell. Indeed, no necessary chemical reaction occurs in a cell without a protein catalyst that is coded by a gene (Steele, Lindley et al. 1998). Genes produce a continuing stream of action in the cell, while policies produce a continuing stream of action in the company.

The creative mechanisms in evolution are mutation and recombination. In genetic mutation, part of a DNA molecule is physically changed, producing a new gene. In our analogy, genetic mutation corresponds to policy change, intentional or unintentional, and (like mutation) produces either favorable or unfavorable results (Meszias and Glynn 1993). The result of such a change, for better or worse, is a new policy. Genetic *recombination* occurs when two DNA molecules mix to form a new DNA molecule. In a company, genetic recombination corresponds to a particular kind of *organizational learning*: Inter-personal learning whereby a person combines a part of someone else's ideas with his or her own (Holland, Holyoak et al. 1986; Miner and Mezias 1996).

In higher animals, sexual reproduction encourages the dissemination and recombination of genetic material. Natural selection is the process by which beneficial recombinants (and mutations) are retained, while deleterious ones are discarded. Sex and selection in the natural world correspond in the corporate world to the various ways in which companies identify certain employees as exemplary and encourage other employees to learn from (or imitate) them (Nei 1987; Smith 1989; Gillespie 1991; Brown 1992; Brown 1995; Leach 1996). We call these processes of identification and encouragement "pointing and pushing mechanisms". For example, pay and organizational position are two ways that a company can *point* to outstanding performers. Pay and position can serve a *pushing* function as well: Employees are motivated ("pushed") to learn via their desire to rise in the pay scale and in the hierarchy.[1] Other pointing and pushing mechanisms are also possible.

1.1 METHODOLOGY

Agent-based models are built around simulated agents – or, in our case, *managers.* Each agent has one or more policies that can control aspects of the organization (e.g. setting prices, or deciding on the number of programmers to hire). The key interaction between agents is learning from another manager, thereby changing the policy and, thus, eventually changing the performance of the organization.

[1] *Pushing* successful people to share policies with others may also be necessary (Constant, Sproull et al. 1996).

In our study of organizational evoluation, we use system dynamics, agent-based modeling and evolution algorithms in the following way: A system dynamics model represents the underlying physics of the organization as well as all policies that are *not* evolving. *Evolving policies* are carried by agents (individual managers) who learn from one another via a process that is similar to a genetic algorithm.

The simulation environment allows us to investigate the conditions and trade-offs that influence the rate at which organizations improve their policies. Conditions include number of teams, team size, frequency of mixing and evaluating, promotion (or other pointing and pushing) policies, number of managers, complexity of evaluation criteria, complexity of policies, number of policies, and many others. An example of a trade-off is between the ability to discriminate between good and bad policies on the one hand and, on the other hand, the speed by which a policy spreads. As the number of teams increase, the rate by which policy changes disseminate will slow, while the ability to discriminate between beneficial and deleterious changes will increase.

2 TECHNICAL OVERVIEW

The simple genetic algorithm and the organizational algorithm are implemented for this presentation in Matlab. In the next few sections, the technical details of the simple genetic algorithm and the organizational algorithm are described. Also, the fitness functions implemented to evaluate the different algorithms are discussed along with the experimental test matrix.

2.1 DATA STRUCTURES

The system dynamics project model used in the application of the organizational algorithm acts on float type variables. For simplicity in the analysis and comparison of the initial organizational algorithm, the learning processes (crossover) and innovation (mutation) in the genetic algorithm and the organizational algorithm act on binary values. Conversion from float notation to binary is achieved by the method discussed in Michalewicz, (Michalewicz 1992).

2.2 SIMPLE GENETIC ALGORITHM

The simple genetic algorithm is designed according to the method outlined in Goldberg (Goldberg 1989). For convenience, the sequence of events followed in the algorithm is listed below.

a. Randomly initialize the population

b. Evaluate the fitness of each individual

c. Compare individual performance and rank the individuals

d. Select by a rank-based roulette process a teacher for each individual

e. Allow crossover between teachers and learners according to the probability of crossover

f. Select at random between the two children, the teacher, and the learner

g. Apply innovation to the individuals who will innovate (according to the probability of mutation)

h. Return to step (b.) for at least *N* generations, where *N* is the population size

The selection process in the simple genetic algorithm will enable it to seek the optimum value in a population regardless of crossover and mutation processes. Trade-offs exist in the parameterization of the simple genetic algorithm: between population size, probability of crossover, and probability of mutation.

2.3 ORGANIZATIONAL ALGORITHM

The organizational algorithm differs from the simple genetic algorithm in four aspects. First, all of the individuals remain in the population throughout the simulation. Second, in each generation every individual makes a decision whether or not to learn (according to some probability of learning). Third, once the individuals are ranked according to their relative performance, an additional promotion (or demotion) is applied to the population. The promotion of the individuals creates positions that are used in the selection process. The promotion process is described in the next section. Finally, the processes of learning and promotion can be applied within a subset of the total population, a team. The team-based learning process is also described in a later section. The steps followed in the organizational algorithm are described below.

a. Randomly initialize the population

b. Evaluate the fitness of each individual

c. Compare individual performance and rank the individuals

d. Promote each individual according to the rank and a promotion base value

e. Select by a position-based roulette process a teacher for each individual who will learn (according to the probability of learning)

f. Allow crossover between teachers and learners

g. Select at random from the two children to replace the learner

h. Apply innovation to the individuals who will innovate (according to the probability of mutation)

i. Return to step (b.) for at least *N* generations, where *N* is the population size

2.3.1 Promotion

Each individual in the population has an assigned position. Because the individuals survive throughout the simulation, the positions become the historical record of each individual. The policies (or phenotypes) of the individuals can change drastically from one generation to the next. However, the positions can only vary according to the promotion base assigned to the simulation. For a promotion base value of 2, the promotion is applied to each individual according to the following equation:

$$\text{Position}_{\text{new}}=\text{Position}_{\text{old}}*2^{K}$$

Where *K* is defined by:

$$K=\frac{2*(rank-1)}{(population\ Size-1)}-1$$

A quick study of *K* shows that at its upper bounds *K*=1, and at its lower bounds *K*= -1. Therefore, the highest ranked individual (the best performing) receives the largest promotion (a doubling of position), whereas the lowest ranked individual receives a demotion (a halving of position).

We have adopted this addition to the genetic algorithm in order to better represent the motivation behind who we choose to learn from. People choose their teachers not just based upon their most recent success, but also due to a whole range of personality characteristics and historical performance.

From one product generation to the next, the CEO of the company (supposedly the fittest manager) is not demoted to the lowest position within the company if he or she enforces a disastrous policy. More realistically, the CEO is either forced to leave the organization or to take a slightly lesser role in management. Similarly, in the organization algorithm, position takes on a multiplicative process – one seen in the K-12 education system. A student performing at an above average level at a young age is put into an honors track for education. The student will be less likely as time progresses to leave this more challenging track, whereby the 'less fit' scenario is true for the child placed into the special needs track from an early age.

2.3.2 Teams

Teams are used in the simulator when we are interested in studying alternatives to an individual-based hierarchy in an evolving organization. A team of managers work together to determine how to manage a project (or to make decisions that impact any dynamic system). Here, a description is given of how teams are implemented in the simulator.

In a population of *N* individuals, generate *T* teams containing an equal number of individuals, N_T.

a. Each individual is defined by a set of policies and a position.

b. The team decides on a single team-based set of policies, *ei*, according to:

$$cumulative\ Position = \sum_{1}^{N} position_i$$

$$Team_{ei} = \frac{1}{cumulativePosition} * \sum_{1}^{N} position_i * ei_i$$

c. The fitness is evaluated for each team using the team's set of policies.

d. Rank each team from 1 to *T*.

e. Establish a promotion factor for each team according to the previously described promotion criteria.

f. Promote every individual according to the team promotion factor.

g. Make new teams. In this paper, the new teams are reassigned by placing the highest *T* individuals in separate teams, and then assigning the next *T* highest individuals in separate teams, and so forth until all of the individuals are placed.

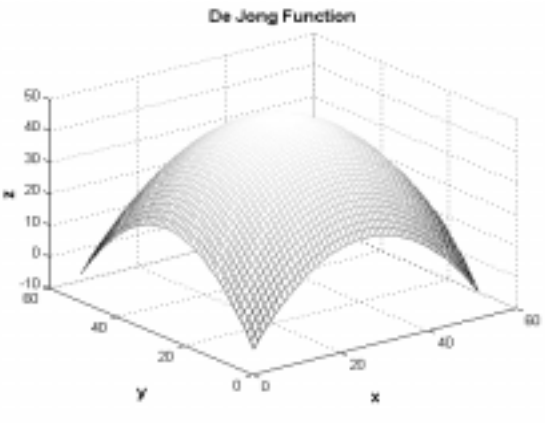
h. Allow teammates to learn from one another by roulette method.

There are two key differences to team-based learning versus individual learning. First, the teams must form a combined set of policies (a combined phenotype) with which to manage. Second, learning only occurs within the bounds of the teams. For example, a population of 50 individuals with 10 teams will have 5 individuals per team. At the beginning of every generation, each individual will have the opportunity to learn from the rest of the team (the other 4 individuals). In the same sized population, a team size of 1 will mean that the probability of crossover is effectiverly zero (the individual can only learn from himself). A team size of 50 will result in a population with random promotion – as there is only a single team for ranking. The result for a single team is learning drift – the individuals select from their teachers with no hierarchical basis.

2.4 FUNCTIONS FOR COMPARISON AND TEST MATRIX

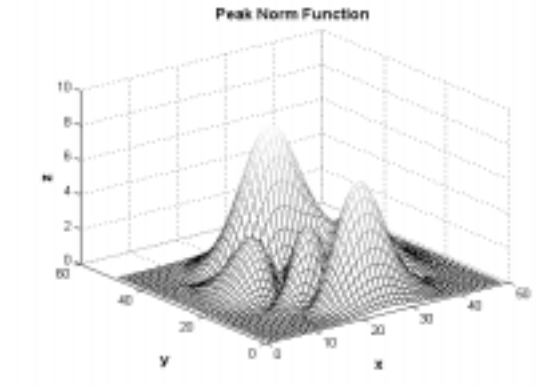
The maximization of three evaluation functions are implemented as fitness criteria for both the simple genetic algorithm and the organizational algorithm. The evaluation functions are chosen to demonstrate the relative ability to optimize by the two algorithms as well as to examine the impact of team-based hierarchies on the optimization. The first function, De Jong (f_1), is similar to a function applied in De Jong's thesis; a simple approach to a single optimum value (Figure 1). The second function, peak-norm (f_2), is a multiple-optima landscape with a unique maximum value (Figure 2). The third function, peak-same function (f_3), possesses multiple equivalent optimums (Figure 3).

Experiments were run on the simple genetic algorithm, the organizational algorithm, and the organizational algorithm with the population divided into teams. The sensitivity of each of these algorithms to the parameters



$$f_1(x,y) = 50 - \sum x^2 + y^2 \qquad -5.12 \le x \le 5.12 \qquad -5.12 \le y \le 5.12$$

Figure 1. De Jong function, f_1, with the corresponding boundary on *x* and *y*.



$$f_3(x,y) = 3(1-x)^2 e^{(-x^2-(y+1)^2)} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{(-x^2-y^2)} - \frac{1}{3}e^{(-(x+1)^2-y^2)} \qquad -3 \le x \le 3 \qquad -3 \le y \le 3$$

Figure 2. Matlab peaks function, f_2, used for the majority of the comparison experiments.

probability of crossover, probability of mutation, and population size were made using the second fitness function, peak-norm.

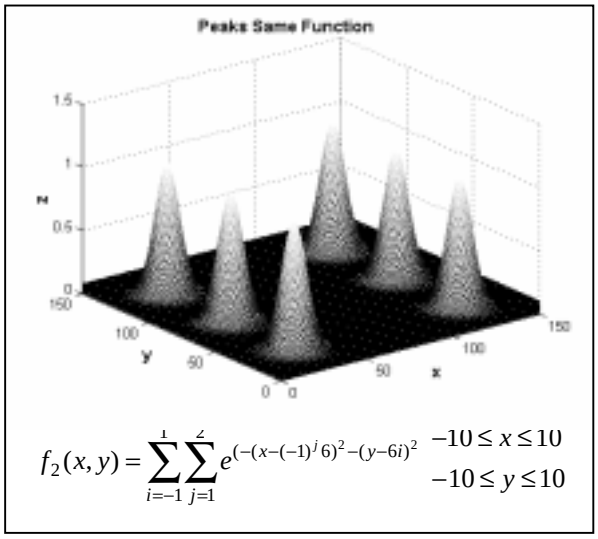


Figure 3. Peaks same function, f_3, six single-optimum Gaussian curves.

The experiment parameters are listed in Table 2. Each experiment was completed five times with the same varying random seeds in the population initialization. The simulations were carried out for 125 generations.

Table 2. Matrix of Experiments (E) on the Simple Genetic Algorithm

E	$P_{Mutation}$	$P_{Crossover}$	N	T
$P_{Mutation}$	0, 0.001, 0.005, 0.01, 0.02, 0.1	0.5	50	1
$P_{Crossover}$	0	0, 0.1, 0.5, 1	50	1
N	0.005	0.5	10, 26, 50, 100, 200	1
T (OA only)	0.001	0.5	50	1, 5, 10, 25, 50

Each of the different algorithms was also examined for performance over the three different fitness functions. The motivation for implementing a function with multiple same-valued optimums (peak-same) was to study the phenomenon of population clustering on multiple solutions.

2.5 PERFORMANCE METRICS

The performance of the simple genetic algorithm and the organizational algorithm was compared for the accuracy of solution upon which the population converges, and the number of generations it takes to find a consensus. For each of the experiments, the mean fitness was calculated across the five simulations characterized by a different random seed for population initialization.

For this presentation, solution convergence is defined as the point at which the mean fitness for a given simulation shows a derivative with respect to generation of less than 10% that endures for at least three generations.

3 RESULTS AND DISCUSSION

3.1 ACCURACY

The genetic algorithm is a powerful search engine – it maintains diversity in its solution space while seeking out what would ultimately be a global maximum. A human population might attempt to seek a global maximum in an organization, but human mental models add inherent biases to this search process. Also, individuals are often part of multiple organizations (family, work, extended family, friends from college, etc.). These other organizations add the potential for innovation (mutation) to feed into an individual's set of policies. Thus, the organizational algorithm is a better representation of human learning as demonstrated in the comparison of the two algorithms with respect to accuracy and convergence time.

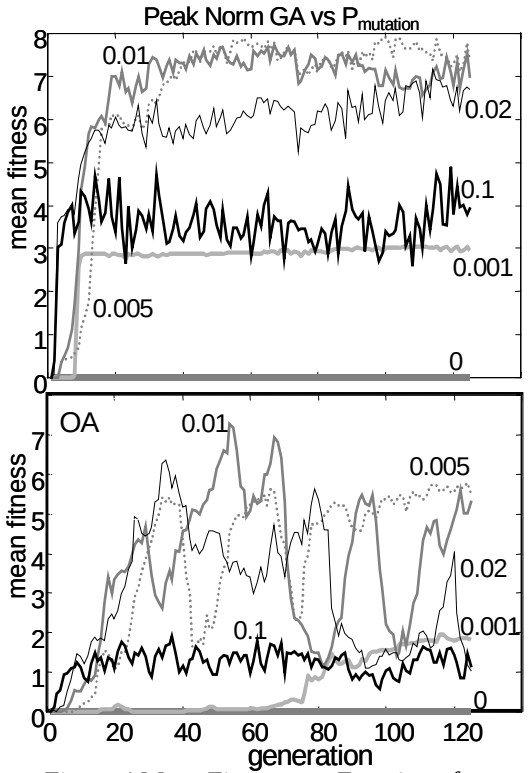


Figure 4 Mean Fitness as a Function of Generation For a Range of Probability of Mutation.

Summaries in the form of the mean fitness of the performance of the two algorithms with respect to the probabilities of crossover and mutation are shown in Figures 4 and 5. In Figure 4. the mutation study was begun on a set of populations that were initialized with every individual having a policy of zero. A summary is also given of the performance of the two algorithms with respect of population size in Figure 6. The labeling convention that is used in these and the remainder of the graphs presented is as follows: OA – organizational algorithm, GA – genetic algorithm, and TM – team implementation in the organizational algorithm. Numerical labels adjacent to plot lines represent corresponding parameter data as labeled in the figure caption. When mean fitness is indicated as the y-axis, this represent the mean fitness over the five runs for each experiement initialized with a varying random seed.

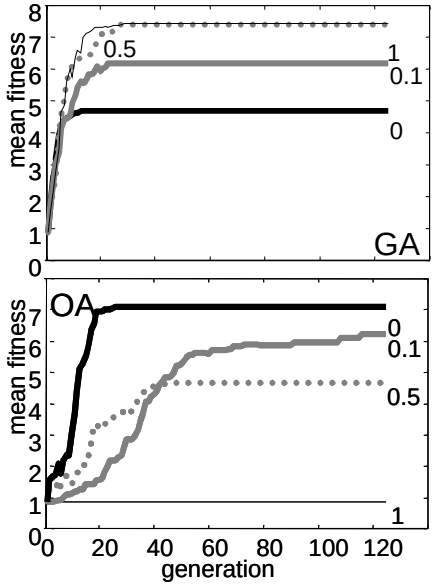


Figure 5. Mean Fitness as a Function of Generation For a Range of Probability of Crossover

Figure 4 indicates that the organizational algorithm is far more susceptible to mutation than the genetic algorithm – particularly in the later generations. The selection process in the organizational algorithm results in a wide spread of positions in the later generations. As a result, the roulette selection method will be far more likely to select an individual of high position even in the face of a debilitating mutation. At worst, the individual will only have a halving of position, still higher than the lowest position by several orders of magnitude. The selection of a mutated individual with a high position will quickly result in a lemming-like behavior, whereby the remainder of the population takes on the less fit policy. In contrast, the genetic algorithm only compares the individuals from one generation to the next – with a linear ranking scheme. A mutation to a less fit policy is immediately seen by the rest of the population and the individual dies.

The comparison of the two algorithms with respect to the probability of crossover now with a truly randomized initial population is shown in Figure 5. In the absence of mutation, both algorithms demonstrate similar behavior at high and low crossover probabilities. For the intermediate cases, the organizational algorithm shows a poor ability to seek an accurate and optimal solution.

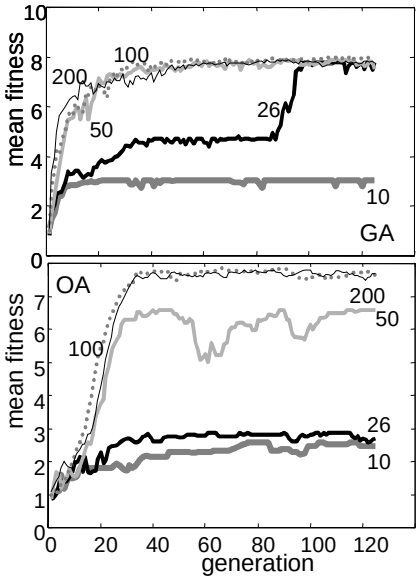


Figure 6. Mean Fitness as a Function of Generation for a Range of Population Size

The differences in the behavior of the two algorithms with respect to the probability of crossover is partially accounted for in the population size sensistivity of the two algorithms (Figure 6). The organizational algorithm needs to operate on a larger population in order to seek a more accurate solution. This is primarily due to the survival of all of the individuals throughout the simulation. A larger population will more likely initially contain the alleles necessary to produce a fit policy. As a result, the larger population will approach a fitter solution prior to the transition from a policy-based selection to one dependent almost entirely on position.

3.2 SPEED OF CONVERSION

When the organizational algorithm is well-behaved, it requires on the order of 20% more generational time to converge (regardless of accuracy) as compared to the genetic algorithm. The inclusion of team-based learning requires an additional 10% generational time for convergence.

The crossover process in the organizational algorithm differs from that of the genetic algorithm in that for each incident of crossover, only one individual learns (as opposed to both parents exchanging information to create two potentially different children). The slower learning added to the increasing selection dependence on position

explain the difference between the algorithms in the conversion time.

3.3 TEAM BASED DYNAMICS

The ability to work in a team-based environment has the advantage of eliminating the personal sting attached to individual demotion. However, as the team has to work together to form policies with which to manage, the potential exists for the individual with the really good policy to never express this policy to the rest of the population. In the simulation of the peak-norm function with different team sizes, Figure 7, it appears that as the number of teams in a fixed population grows the fidelity of information transfer improves. As expected, for a team size of 1 or 50, there is no goal-seeking behavior in the algorithm.

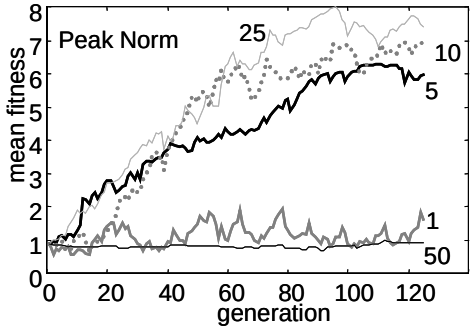


Figure 7. Mean Fitness versus Generation For a Range of Number of Teams

In the comparison of the three different parameterizations: simple genetic algorithm, organizational algorithm, and team-based organizational algorithm (shown in Figure 8) for each of the fitness functions, the team simulations were completed with 10 teams of 5 individuals.

For the relatively simple fitness function, De Jong, all of the different parameterizations perform well, showing similar convergence rates and values. As the complexity of the fitness function increases, to peak-norm, we see degradation of the performance in both of the organizational algorithm parameterizations. Both convergence rate and accuracy are lost. A similar trend is seen with the peak-same function.

From a performance perspective, the organizational algorithm is not as powerful of an optimization tool as that of the genetic algorithm. However, the focus of our research is not to optimize, but to represent and better understand learning within an organization. The organizational algorithm provides a strong metaphor for the impact of hierarchy in an organization on information flow. A delicate balance exists between the use of hierarchy to point to the individuals with the good ideas and the use of an individual's performance immediately based upon those good ideas.

Larger capital producing organizations potentially suffer from the focus on position in learning motivation. For one, employees are not able to immediately see customer response to their new car or dishwasher. Instead, employees see an internal performance record that is generated by someone with a higher position. In consulting companies or law firms, the employees work closer to their end product and immediately receive feedback on their relative performance. This information is often highly visible within the organization (i.e. number of billable hours). The net result is that employees determine who to learn from according to a metric tied more directly to an individual's performance.

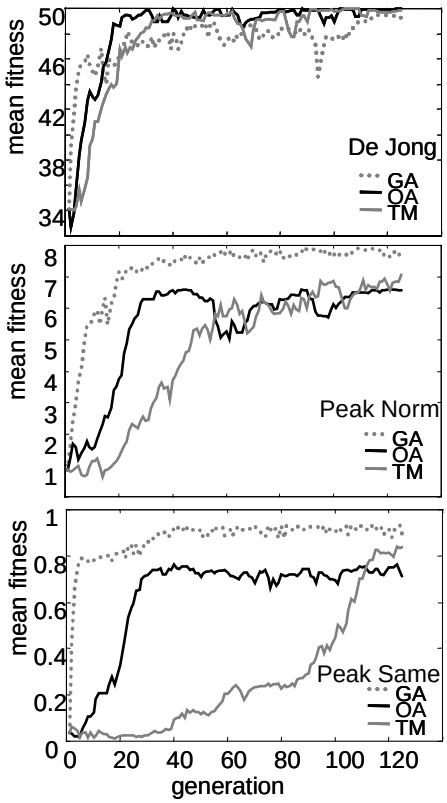


Figure 8. Mean Fitness versus Generation For Each Fitness Function

The reality in all evolutionary algorithms, however, is that hierarchy is a necessary mechanism for goal-seeking behavior. Similarly, in human organizations the presence of hierarchy enables people to seek better solutions for themselves. We naturally compare and contrast our colleagues in our conversations and interactions. These rankings guide us in our learning.

4 CONCLUSIONS

Companies are sub-optimal, and our organizational algorithm metaphorically describes some of the reasons behind why this is. Cultural bias adds inertia that debilitates our ability to accurately identify who to best

seek out for improvement. The organizational algorithm demonstrates how the combination of position and innovation in a population can lower the population to an even lesser performance as individuals follow their leader regardless of the quality of his or her idea.

We have also demonstrated that the parameterization of an evolutionary algorithm can be altered to recreate the dynamics seen in different types of organizations. The simple genetic algorithm better represents a service-related company, while a large capital producing company follows more closely the dynamics of the organizational algorithm.

Finally, team-based learning has been examined as an alternative to individual hierarchy. While team-based learning inhibits the fidelity and rate of solution optimization, the net result is a population with a hierarchy that is decoupled from single individual performance. This result is one step towards using our understanding of organizational evolution to create better organizations.

Acknowledgments

This work is supported by the National Science Foundation, award # 9975942, and our corporate partners; Eastman Chemicals, PriceWaterhouse Coopers, Hewlett Packard, Pugh Roberts Associates, The Lincoln School, and General Motors. The authors would also like to acknowledge the support of Christy Dowding.

References

Brown, T. A. (1992). Genetics: A Molecular Approach. New York, Chapman & Hall.

Brown, T. A. (1995). Gene Cloning: An Introduction. New York, Chapman & Hall.

Constant, D., L. Sproull, et al. (1996). "The kindness of strangers: The usefulness of electronic weak ties for technical advice." Organizational Science **7**(2): 119-135.

Forrester, J. W. (1961). Industrial Dynamics. Portland, OR, Productivity Press.

Gillespie, J. H. (1991). The Causes of Molecular Evolution. New York, Oxford University Press.

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA, Addison Wesley Publishing Company, Inc.

Hines, J. (1997). Organizational Evolution. New England Complex Systems International Conference, Nashua, NH.

Holland, J. H., K. J. Holyoak, et al. (1986). Induction: Processes of Inference, Learning, and Discovery. Cambridge, MA, MIT Press.

Kofman, F., J. D. Sterman, et al. (1997). "Unanticipated Side Effects of Successful Quality Programs: Exploring a Paradox of Organizational Improvement." Management Science **43**(4): 503-521.

Leach, D. R. F. (1996). Genetic Recombination. Cambridge, MA, Blackwell Science, Ltd.

Mathworks, Inc. (1998). Matlab ver 5.3. Natick, Mathworks.

Meszias, S. J. and M. A. Glynn (1993). "The three faces of corporate renewal: Institution, revolution, and evolution." Strategic Management Journal **14**(2): 77-101.

Michalewicz, Z. (1992). Genetic Algorithms + Data Structures = Evolution Programs. New York, Springer-Verlag.

Miner, A. S. and S. J. Mezias (1996). "Ugly Duckling No More: Pasts and Futures of Organizational Learning Research." Organization Science **7**(1): 88-99.

Nei, M. (1987). Molecular Evolutionary Genetics. New York, Columbia University Press.

Simon, H. A. (1976). Adminstrative Behavior. New York, The Free Press.

Smith, J. M. (1989). Evolutionary Genetics. New York, Oxford University Press.

Steele, E. J., R. A. Lindley, et al. (1998). Lamarck's Signature: How Retrogenes are Changing Darwin's Natural Selection Paradigm. Reading, MA, Perseus Books.