Papertype: Paper Track

Conference Topic: Evolutionary Computation, Genetic Algorithms and Neural Networks

Title: New Generic Hybrids Based Upon Genetic Algorithms

Michael Affenzeller

Institute of Systems Science
Systems Theory and Information Technology
Johannes Kepler University
Altenbergerstrasse 69
A-4040 Linz - Austria
email: ma@cast.uni-linz.ac.at
phone: ++43 732 2468 9200

Abstract. In this paper we propose some generic extensions to the general concept of a Genetic Algorithm. These biologically and sociologically inspired interrelated hybrids aim to make the algorithm more open for scalability on the one hand, and to retard premature convergence on the other hand without necessitating the development of new coding standards and operators for certain problems. Furthermore, the corresponding Genetic Algorithm is unrestrictedly included in all of the newly proposed hybrid variants under special parameter settings. The experimental part of the paper discusses the new algorithms for the Traveling Salesman Problem as a well documented instance of a multimodal combinatorial optimization problem achieving results which significantly outperform the results obtained with a conventional Genetic Algorithm using the same coding and operators.

Keywords: soft computing, methodology of evolutionary algorithms, optimization.

New Generic Hybrids Based Upon Genetic Algorithms

Michael Affenzeller

Institute of Systems Science
Systems Theory and Information Technology
Johannes Kepler University
Altenbergerstrasse 69
A-4040 Linz - Austria
ma@cast.uni-linz.ac.at

Abstract. In this paper we propose some generic extensions to the general concept of a Genetic Algorithm. These biologically and sociologically inspired interrelated hybrids aim to make the algorithm more open for scalability on the one hand, and to retard premature convergence on the other hand without necessitating the development of new coding standards and operators for certain problems. Furthermore, the corresponding Genetic Algorithm is unrestrictedly included in all of the newly proposed hybrid variants under special parameter settings. The experimental part of the paper discusses the new algorithms for the Traveling Salesman Problem as a well documented instance of a multimodal combinatorial optimization problem achieving results which significantly outperform the results obtained with a conventional Genetic Algorithm using the same coding and operators.

1 Introduction

Many problems that are treated by Genetic Algorithms belong to the class of NP-complete problems. The advantage of Genetic Algorithms when being applied to such problems lies in the ability to search through the solution space in a broader sense than heuristic methods that are based upon neighborhood search. Nevertheless, also Genetic Algorithms are frequently faced with a problem which, at least in its impact, is quite similar to the problem of stagnating in a local but not global solution. This drawback, called premature convergence in the terminology of Genetic Algorithms, occurs when the population of a Genetic Algorithm reaches such a suboptimal state that the genetic operators can no longer produce offspring that outperform their parents (e.g. [7]).
During the last decades plenty of work has been investigated to introduce new coding standards and operators in order to overcome this essential handicap of Genetic Algorithms. As these coding standards and the belonging operators often are quite problem specific, we try to take a different approach and look upon the concepts of Genetic Algorithms as an artificial self organizing process in a biologically and sociologically inspired generic way in order to improve the global convergence behaviour of Genetic Algorithms independently of the actually employed implementation.
In doing so we have introduced an advanced selection model for Genetic Algorithms that allows adaptive selective pressure handling in a way quite similar to Evolution Strategies. Based upon this enhanced GA-model two further generic extensions are discussed:

(1) The concept of segregation and reunification of subpopulations aims to assure an independent development of building blocks in very different regions of the search

space in order to improve global convergence. The algorithm divides the population into subpopulations. These evolve independently until their fitnesses stagnate. By this approach of width-search, building blocks, which would disappear early in case of standard Genetic Algorithms, are evolved in different regions of the search space at the beginning and during the evolutionary process. In contrast to the Island Models for Genetic Algorithms[19], in our case the single subpopulations grow together again in case of stagnating fitness in order to end up with a final population containing as much essential building blocks as possible.

(2) The second newly introduced concept allows the dynamic usage of multiple crossover operators in parallel in order to somehow imitate the parallel evolution of a variety of species that are struggling for limited resources. This strategy seems very adopted for problems which consider more than one crossover operator - especially if the properties of the considered operators may change as evolution proceeds.

As an important property of all the newly introduced hybrids it has to be pointed out that under special parameter settings the corresponding GA/GAs is/are unrestrictedly included in the new hybrids.
The experimental part discusses the new algorithms for the Traveling Salesman Problem as a very well documented instance of a multimodal combinatorial optimization problem. In contrast to all other evolutionary heuristics known to the author that do not use any additional problem specific information, we obtain solutions close to the best known solution for all considered benchmarks (symmetric as well as asymmetric benchmark problems).

2 The Variable Selective Pressure Model

The handling of selective pressure in the context of Genetic Algorithms mainly depends on the choice of a certain replacement scheme[11]. 'Generational replacement', for example, replaces the entire population by the next one, whereas 'elitism replacement' keeps the best individuals of the last generation and only replaces the rest and therefore usually performs faster. On the other hand, elitism likely causes too homogeneous populations, i.e. little population diversity, and therefore might cause unwanted premature convergence. Anyway, there exists no manageable model for controllable selective pressure handling within the theory of Genetic Algorithms[14]. Therefore, we introduce some kind of intermediate step (a 'virtual population') into selection which provides a handling of selective pressure very similar to that of Evolution Strategies [2], [4]. As we will exemplarily point out, the most common replacement mechanisms can easily be implemented in this intermediate selection step. Furthermore, this Evolution Strategy like variable selective pressure will help us to steer the degree of population diversity on the one hand and, on the other hand, it will act as a basic model for a new hybrid metaheuristics based upon Genetic Algorithms as being proposed in section 3.
Actually, all modifications that are and will be taken into account use exactly the same operators for crossover and mutation as a corresponding Genetic Algorithm. As no further problem specific information is used, the new hybrids should be appliable to a huge number of problems - namely all problems Genetic Algorithms can be applied to.

2.1 Formal Integration of the Variable Selective Pressure Model

Similar to any other conventional Genetic Algorithm (e.g.[11]) we use a population of fixed size that will evolve to a new population of the same size by selection, crossover,

and mutation.
What we additionally have done is to introduce an intermediate step in terms of a so-called virtual population of variable size where the size of the virtual population usually has to be greater than the population size. This virtual population is created by selection, crossover, and mutation in the common sense of Genetic Algorithms. But like in the context of Evolution Strategies, only a certain percentage of this intermediate population will survive.
This handling of selective pressure in our context is mainly motivated by (μ, λ)-Evolution Strategies where μ parents produce λ descendants from which the best μ survive. Within the framework of Evolution Strategies, selective pressure is defined as $s = \frac{\mu}{\lambda}$, where a small value of s indicates high selective pressure and vice versa (for a detailed description of Evolution Strategies see for instance [14]). Even if the interaction between the variable selective pressure within our new model and the notion of temperature within the scope of Simulated Annealing is quite different in detail, we have adopted this notation. Applied to our new Genetic Algorithm, this means that from $|POP|$ (population size) number of parents $|POP| \cdot T$ ((size of virtual population) $> |POP|$, i.e. $T > 1$) descendants are generated by crossover and mutation from which the best $|POP|$ survive as illustrated in Fig. 1. Obviously we define selective

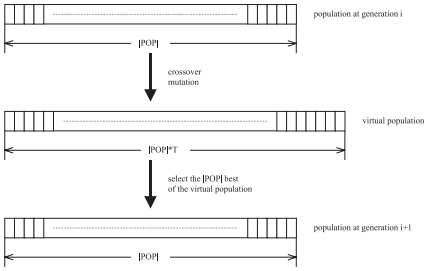

Fig. 1. Evolution of a new population with selective pressure $s = \frac{1}{T}$ for a virtual population built up in the sense of a (μ, λ)-Evolution Strategy.

pressure as $s = \frac{|POP|}{|POP| \cdot T} = \frac{1}{T}$, where a small value of s, i.e. a great value of T, stands for high selective pressure and vice versa. Equipped with this enhanced GA-model it is quite easy to adopt further extensions based upon a controllable selective pressure, i.e. it becomes possible either to reset the temperature up/down to a certain level or simply to cool down the temperature in the sense of Simulated Annealing during the evolutionary process in order to steer the convergence of the algorithm.
Biologically interpreting this (μ, λ)-Evolution Strategy like selective pressure handling, for Genetic Algorithms this means, that some kind of 'infant mortality' has been introduced in the sense that a certain ratio of the population ($|POP| \cdot T - |POP| = |POP| \cdot (T - 1)$) will never become procreative, i.e. this weaker part of a population will not get the possibility of reproduction. Decreasing this 'infant mortality', i.e. reducing the selective pressure during the evolutionary process also makes sense in a biological interpretation because also in nature stronger and higher developed populations suffer less from infant mortality.
From the point of view of optimization, decreasing the temperature during the optimization process means that a greater part of the search space is explored at the beginning of evolution - whereas at a later stage of evolution, when the average fitness is already quite high, a higher selective pressure is quite critical in that sense that

it can easily cause premature convergence. Operating with a temperature converging to zero, this (μ, λ)-Evolution Strategy like selective pressure model for Genetic Algorithms acts like the corresponding Genetic Algorithm with generational replacement. Moreover, implementing the analogue to the $(\mu + \lambda)$-Evolution Strategy denotes the other extreme of immortal individuals. However, also the implementation of this strategy is quite easy to handle with our model by just copying the old population into the virtual population . Other replacement mechanisms, like elitism or the goldcage-model for example, are also easy to implement by just adding the best individuals respectively the best individual of the last generation to the virtual population.

3 Hybrid GA-Concepts Based Upon the Variable Selective Pressure Model

When applying Genetic Algorithms to complex problems, one of the most frequent difficulties is premature convergence. Roughly speaking, premature convergence occurs when the population of a Genetic Algorithm reaches such a suboptimal state that the genetic operators can no longer produce offspring that outperform their parents (e.g. [7]).
Several methods have been proposed to combat premature convergence in the context of Genetic Algorithms (e.g. [5], [6], or [8]). These include the restriction of the selection procedure, the operators and the according probabilities as well as the modification of fitness assignment. However, all these methods are heuristic in nature. Their effects vary with different problems and their implementation strategies need ad hoc modifications with respect to different situations.
A critical problem in studying premature convergence is the identification of its occurrence and the characterization of its extent. Srinivas and Patnaik [16], for example, use the difference between the average and maximum fitness as a standard to measure premature convergence and adaptively vary the crossover and mutation probabilities according to this measurement. As in the present paper, the term 'population diversity' has been used in many papers to study premature convergence (e.g. [15]) where the decrease of population diversity is considered as the primary reason for premature convergence. Therefore, a very homogeneous population, i.e. little population diversity, is considered as the major reason for a Genetic Algorithm to prematurely converge.
The following generic extensions that are built up upon the variable selective pressure model primarily aim to avoid or at least to retard premature convergence in a general way.

3.1 Segregative Genetic Algorithms (SEGA)

In principle, our new SEGA introduces two enhancements to the general concept of Genetic Algorithms. The first is to bring in a variable selective pressure, as described in section 2, in order to control the diversity of the evolving population. The second concept introduces a separation of the population to increase the broadness of the search process and joins the subpopulation after their evolution in order to end up with a population including all genetic information sufficient for locating the region of a global optimum.
The aim of dividing the whole population into a certain number of subpopulations (segregation) that grow together in case of stagnating fitness within those subpopulations (reunification) is to combat premature convergence which is the source of GA-difficulties. This segregation and reunification approach is a new technique to overcome premature convergence [1], [3].

Whereas Island Models for Genetic Algorithms (e.g. in [19]) are mainly driven by the idea of using simultaneous computer systems, SEGA attempts to utilize migration more precisely in order to achieve superior results in terms of global convergence.
The principle idea is to divide the whole population into a certain number of subpopulations at the beginning of the evolutionary process. These subpopulations evolve independently from each other until the fitness increase stagnates because of too similar individuals within the subpopulations. Then a reunification from n to $(n-1)$ subpopulations is done. Roughly spoken this means, that there is a certain number of villages at the beginning of the evolutionary process that are slowly growing together to bigger cities, ending up with one big town containing the whole population at the end of evolution. By this approach of width-search, building blocks in different regions of the search space are evolved at the beginning and during the evolutionary process, which would disappear early in case of standard genetic algorithms and whose genetic information could not be provided at a later date of evolution when the search for global optima is of paramount importance.
Again, like in the context of the variable selective pressure model which is included in SEGA as well, it should be pointed out that a corresponding Genetic Algorithm is unrestrictedly included in the SEGA when the number of subpopulations (villages) and the cooling temperature are both set to 1 at the beginning of the evolutionary process. Moreover, the introduced techniques also do not use any problem specific information.

Segregation and Reunification: Monitoring the behaviour of a Genetic Algorithm when applied to optimization problems shows that the average fitness as well as the fitness of the best member of the actual population often stagnates at a certain point of the evolution process even if the actual fitness is wide off the mark of a potentially best or at least a best-known solution (premature convergence). It appears that Genetic Algorithms prematurely converge to very different regions of the solution space when repeatedly running a Genetic Algorithm. Moreover it is known from GA-theory[11], that extending the population size does not help to avoid premature convergence. In fact, depending on the problem-type and the problem-dimension there is a certain population size, where exceeding this population size doesn't effect any more improvements in the quality of the solution.
Motivated by these observations, we have developed an extended approach to Genetic Algorithms where the total population is split into a certain number of subpopulations or villages, all evolving independently from each other (segregation) until a certain stage of stagnation in the fitness of those subpopulations is reached. Then, in order to bring some new genetic information into each village, the number of villages is reduced by one which causes new overlapping-points of the villages. Fig. 2 shows a schematic diagram of the described process. This process is repeated until all villages are growing together ending up in one town (reunification). The variable selective pressure is of particular importance if the number of subpopulations is reduced by one because this event brings new diversification into the population. In this case a higher selective pressure is reasonable, i.e. if reunifying members of neighboring villages, the temperature is reset to a higher level in order to cool down to 1 as the new system of subpopulations evolves. While the number of villages decreases during evolution, it is recommended to reset the selective pressure to a higher level because the genetic diversity of the emerging greater subpopulations is growing.
For fighting premature convergence, the main advantage of the described strategy is that building-blocks in different regions of the search space are evolved independently from each other at the beginning of the evolutionary process. The aim is that the best building-blocks survive during the recombination phase, yielding in a final population (if the number of villages is 1) containing all essential building-blocks for the detec-

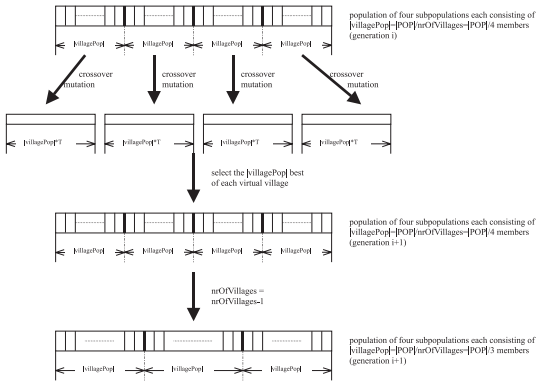

Fig. 2. Evolution of a new population for the instance that four subpopulations are merged to three.

tion of a global optimum. In case of conventional Genetic Algorithms, building blocks that disappear early and which may be important at a later stage of the evolutionary process, when the search for global optima is of paramount importance, can hardly ever be reproduced (premature convergence).

The Algorithm: With all strategies described above, finally the new genetic algorithm is stated as follows:

The segregative genetic algorithm

```
procedure SEGA
  initialize population size|POP|
  initialize number of iterations nrOfIterations ∈ ℕ
  initialize number of villages nrOfVillages ∈ ℕ
  initialize temperature T ∈ [1, ∞ [
  initialize adaptive cooling factor α ∈ ]0, 1]
  generate initial population POP_0 = (I_1, ...., I_|POP|)
  for i:=1 to nrOfIterations do
    if (i = dateOfReunification) then
       nrOfVillages:=nrOfVillages-1
       reset temperature
    end if
    POP_i:=calcNextGeneration(POP_i-1, T, nrOfVillages, |POP|)
    T:=max(T, T * α)
  next i
```

Function "calcNextGeneration" implements the evolution of the next generation of subpopulations.

```
function calcNextGeneration: (POP_i-1, T, nrOfVillages, |POP|)
  villagePopulation=|POP| / nrOfVillages
  for i:=(0 to (nrOfVillages-1)) do
    for j:=(i*villagePopulation) to ((i+1)*villagePopulation) do
      calculate fitness_j of each member of the village population (like in standard GA).
    next j
    |virtualPopulation|=|villagePop|*T
```

```
  for k:=0 to |virtualPopulation| do
    generate individuals of virtual population I_k ∈ S  from the members of the village.
    I_{i*|villagePopulation|} ... I_{(i+1)*|villagePopulation|} ∈ POP_{i-1}
    due to their fitnesses by crossover and mutation
  next k
  select the best |villagePopulation| individuals from the virtual population
  in order to achieve the new village population
  I_{i*|villagePopulation|} ... I_{(i+1)*|villagePopulation|} ∈ POP_i
  of the next generation.
next i
return POP_i
```

SEGA uses a fixed number of iterations for termination. Depending on this total number of iterations and the initial number of subpopulations (villages), the dates of reunification may statically be calculated at the beginning of the evolutionary process as done in the experimental result section. Further improvements, particularly in the sense of running time, are possible, if, in order to determine the dates of reunification, a dynamic criterion for the detection of stagnating genetic diversity within the subpopulations is used.

3.2 Dynamic Habitat Adaptation

Genetic Algorithms as well as its most common variants consider the evolution of a single species, i.e. crossover can be done between all members of the population. This supports the aspect of depth-search but not the aspect of width-search. Considering natural evolution, where a multiplicity of species evolve in parallel, as a role model, we could introduce a number of crossover operators and apply each one to a certain subpopulation. In order to keep that model realistically it is necessary to choose the size of those subpopulations dynamically, i.e. depending on the actual success of a certain species its living space is expanded or restricted. Speaking in the words of Genetic Algorithms, this means that the size of subpopulations (defined by the used crossover and mutation operators) with lower success in the sense of the quality function is restricted in support of those subpopulations that push the process of evolution.
But as no Genetic Algorithm known to the author is able to model jumps in the evolutionary process and no exchange of information between the subpopulations takes place, the proposed strategy would fail in generating results superior to the results obtained when running the Genetic Algorithms with the certain operators one after another. Thus, the achieved profits would 'only' concern the performance of the algorithm. Therefore, it seems reasonable to allow also recombination of individuals that have emerged from different crossover operators, i.e. the total population is taken into account for each crossover operator and the living space (habitat) of each virtual subpopulation is defined by its success during the last iterations as illustrated in Fig. 3.
Exemplarily considering the properties of the OX (order crossover) and the ERX (edge recombination crossover) operators for crossover it is reported (e.g. in [18] or [11]) that the OX-operator significantly outperforms the ERX-operator in terms of speed whereas the ERX-operator surpasses OX in terms of global convergence. Dynamically using multiple crossover operators in parallel utilizes the 'fast' OX-operator for a long evolution period until the performance in terms of solution quality of ERX outperforms OX at a later stage of evolution. Even more, experiments have shown that the described strategy significantly surpasses the results obtained when just using one single operator in terms of solution quality.
Anyway, this dynamic (self-organizing) strategy seems particularly suitable for situations where a couple of crossover operators whose properties are not exactly known are taken into account.

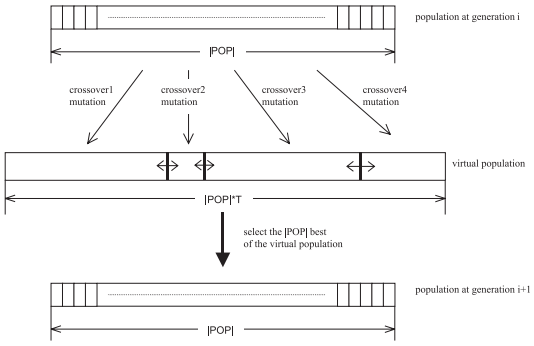

Fig. 3. Evolution of a new population for the instance that four crossover operators are used in parallel.

4 Experimental Results

In our experiment, all computations are performed on a Pentium III PC with 256 megabytes of main memory. The programs are written in the Java programming language.
Even if ongoing experimental research on a variety of problems shows quite similar results it would go beyond the scope of the present paper to present all these tests. So we just give a short summary of the results obtained by SEGA on a selection of symmetric as well as asymmetric TSP benchmark problem instances taken from the TSPLIB [13] using updated results[1] for the best, or at least the best known, solutions. In doing so, we have performed a comparison of SEGA with a conventional GA using exactly the same operators for crossover and mutation and the same parameter settings and with the COSA-algorithm [17] as an established and successful ambassador of a heuristic especially developed for routing problems.
Fig. 4 shows the experimental results for the problem kro124p (100 city problem) as an example of an asymmetric TSP benchmark. For the tests the parameters of COSA are set as suggested by the author in [17]. Both, GA and SEGA use a mutation probability of 0.05 and a combination of OX-crossover and ERX-crossover [11] combined with the golden-cage population model, i.e. the entire population is replaced with the exception that the best member of the old population survives until the new population generates a better one (wild-card strategy). Within SEGA, the described strategies are applied to each subpopulation. The results of a test presented in the present paper start with 32 villages (subpopulations), each consisting of 64 individuals, i.e. the total population size is set to 2048 for SEGA (as well as for COSA and GA).
Table 1 shows the experimental results of SEGA (with dynamic habitat adaptation), COSA, and GA concerning various types of problems in the TSPLIB. For each problem the algorithms were run ten times. The efficiency for each algorithm is quantified in terms of the relative difference of the best's individual fitness after a given number or iterations to the best or best-known solution. In this experiment, the relative difference is defined as relativeDifference $= (\frac{Fitness}{Optimal} - 1) * 100\%$. These examples demonstrate the predominance of the new SEGA (together with an adaptive steering of OX and ERX operators) compared to the standard-GA. The preeminence of

[1] Updates for the best (known) solutions can for example be found on ftp://ftp.zib.de/pub/Packages/mp-testdata/tsp/tsplib/index.html

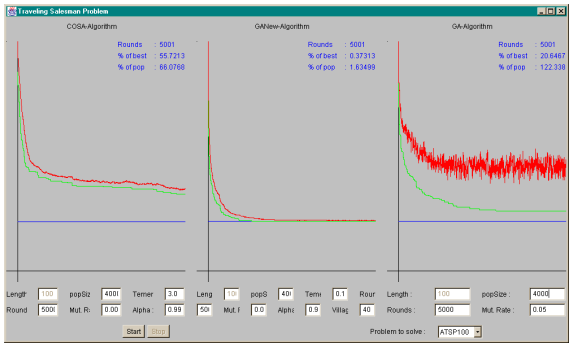

Fig. 4. Comparison of COSA, SEGA, and GA on the basis of the kro124 benchmark problem: For each algorithm the average fitness and the fitness of the best member of the population is diagrammed relatively to the best known solution represented by the horizontal line.

Table 1. Experimental results of COSA, GA (using OX or ERX for crossover) and the new SEGA together with a dynamic combination of OX- and ERX crossover.

Problem	Iter.No.	Average difference(%)			
		COSA	GA_{OX}	GA_{ERX}	GA_{new}
eil76(symm.)	5000	6.36	17.56	7.62	0.32
ch130(symm.)	5000	14.76	84.54	32.44	0.35
kroA150(symm.)	5000	20.91	102.40	71.97	0.74
kroA200(symm.)	10000	48.45	95.69	117.11	1.24
br17(asymm.)	200	0.00	0.00	0.00	0.00
ftv55(asymm.)	5000	44.22	41.34	23.52	0.27
kro124p(asymm.)	10000	26.78	30.61	15.49	0.48
ftv170(asymm.)	15000	187.34	87.12	126.22	1.09

SEGA, especially when being compared to the rather problem specific COSA heuristic, becomes even more evident, if asymmetric benchmark problems are considered.

5 Conclusion

In this paper an enhanced Genetic Algorithm and two upgrades have been presented and exemplarily tested on some TSP benchmarks. The proposed GA-based techniques couple aspects from Evolution Strategies (selective pressure), Simulated Annealing (temperature, cooling) as well as a special segregation and reunification strategy with crossover, mutation, and selection in a general way, so that established crossover and mutation operators for certain problems may be used analogously to the corresponding Genetic Algorithm. The investigations in this paper have mainly focused on the avoidance of premature convergence and on the introduction of methods which make the algorithm more open for scalability in the sense of convergence versus running time.
Concerning the speed of SEGA, it has to be pointed out that the superior performance concerning convergence requires a higher running time, mainly because of the the greater population size $|POP|$ required. This should allow to transfer already developed GA-concepts to increasingly powerful computer systems in order to achieve better results. Using simultaneous computers seems especially suited to increase the

performance of SEGA. Anyway, under special parameter settings the corresponding Genetic Algorithm is fully included within the introduced concepts achieving a performance only marginally worse than the performance of the equivalent Genetic Algorithm. In other words, the introduced models can be interpreted as a superstructure to the GA model or as a technique upwards compatible to Genetic Algorithms. Therefore, an implementation of the new algorithm(s) for a certain problem should be quite easy to do, presumed that the corresponding Genetic Algorithm (coding, operators) is known.
However, the efficiency of a variable selective pressure certainly depends on the genetic diversity of the entire population. Ongoing research indicates that it could be a very fruitful approach to define the actual selective pressure depending on the actual genetic diversity of the population.

References

1. Affenzeller, M.: A New Approach to Evolutionary Computation: Segregative Genetic Algorithms (SEGA). Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, Lecture Notes of Computer Science 2084 (2001) 594–601
2. Affenzeller, M.: Transferring the Concept of Selective Pressure from Evolutionary Strategies to Genetic Algorithms. Proceedings of the 14th International Conference on Systems Science 2 (2001) 346–353
3. Affenzeller, M.: Segregative Genetic Algorithms (SEGA): A Hybrid Superstructure Upwards Compatible to Genetic Algorithms for Retarding Premature Convergence. Internatinal Journal of Computers, Systems and Signals (IJCSS), Vol. 2, Nr. 1 (2001) 18–32
4. Affenzeller, M.: A Generic Evolutionary Computation Approach Based Upon Genetic Algorithms and Evolution Strategies. To be published in: Journal of Systems Science, Vol. 28, Nr. 4 (2002)
5. Cobb, H.J., Grefenstette J.J.: Genetic Algorithms for Tracking Changing Environment. Proceedings of the Fifth International Conference on Genetic Algorithms (1993) 523–530
6. DeJong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, Department of Computer Science, University of Michigan (1975)
7. Fogel, D.B.: An Introduction to Simulated Evolutionary Optimization. IEEE Trans. on Neural Networks 5(1) (1994) 3–14
8. Goldberg, D. E.: Genetic Alogorithms in Search, Optimization and Machine Learning. Addison Wesley Longman (1989)
9. Holland, J. H.: Adaption in Natural and Artificial Systems. 1st MIT Press ed. (1992)
10. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220 (1983) 671–680
11. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
12. Rechenberg, I.: Evolutionsstrategie. Friedrich Frommann Verlag (1973)
13. Reinelt, G.: TSPLIB - A Traveling Salesman Problem Library. ORSA Journal on Computing 3 (1991) 376-384
14. Schneburg, E., Heinzmann, F., Feddersen, S.: Genetische Algorithmen und Evolutionsstrategien. Addison-Wesley (1994)
15. Smith, R.E., Forrest, S., Perelson, A.S.: Population Diversity in an Immune System Model: Implications for Genetic Search. Foundations of Genetic Algorithms 2 (1993) 153–166
16. Srinivas, M., Patnaik, L.: Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms . IEEE Transactions on Systems, Man, and Cybernetics 24(4) (1994) 656–667
17. Wendt, O.: Tourenplanung durch Einsatz naturanaloger Verfahren. Deutscher Universitätsverlag (1995)
18. Whitley, D., Starkweather, T., Fuguay, D.: Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator. Proceedings of the Third International Conference on Genetic Algorithms and their Applications (1989) 133–140
19. Whitley, D.: A Genetic Algorithm Tutorial. Statistics and Computing 4 (1994) 65–85