Co-operating populations

Improving the performance of Parallel Genetic Algorithms with Co-operating populations with different evolution behaviours

Panagiotis Adamidis

Automation & Robotics Lab.,
Department of Electrical and Computer Engineering,
Aristotle University of Thessaloniki,
Greece

adamidis@egnatia.ee.auth.gr
adamidis@uom.gr

Abstract

A framework for unifying Simple and Parallel Genetic Algorithm implementations as Co-operating Populations is presented. Using this framework, a method called *Co-operating Populations with Different Evolution Behaviours (CoPDEB)*, for generalizing and improving the performance of Parallel Genetic Algorithms (PGAs) is also presented. The main idea of CoPDEB is to maintain a number of populations exhibiting different evolution behaviours. CoPDEB was tested on three problems (the optimization of a real function, the TSP problem and the problem of training a Recurrent Artificial Neural Network), and appears to significantly increase the problem-solving capabilities over PGAs with the same evolution behaviour on each population. This paper also studies the effect of the migration rate (*Epoch*) and the population size on the performance of both PGAs and CoPDEB.

Keywords: Parallel genetic algorithms, co-operating populations, different evolution behaviour, epoch.

1. Introduction

GAs are stochastic search techniques that explore combinatorial search spaces using simulated evolution (Holland, 1975). The effectiveness of Genetic Algorithms (GAs), is limited by their ability to balance the need for a diverse set of sampling points, with the desire to quickly focus search upon potential solutions. Exploration is achieved through the recombination of data structures (which represent candidate solutions). The data structures constitute the population of the GA, which is evolved towards the optimum, concentrating search in those areas of higher fitness (exploitation). Recombination and mutation perturb the individuals of the population providing general heuristics for exploration.

The objective of this paper is twofold. First, we introduce a framework for unifying Simple and Parallel Genetic Algorithm implementations. A recursive model at different levels is defined (*Co-operating populations*) offering an engine to implement any type of GA in any kind of hardware layout, including simple GAs, fine-grained and coarse-grained PGAs (cgPGAs). The level one of Co-operating Populations can represent any implementation of cgPGAs.

Following the definition, one has a range of various representation methods, population structures, selection methods, parameters, operators and mechanisms to choose from. Theoretical analysis cannot provide a tool that can help the appropriate choice of these, leaving the selection of the appropriate operators, parameters and their rates, to depend on the demands of the problem and on the user's experience and aesthetic preferences (Davis, 1989; Davis, 1991; Schaffer, Caruana, Eshelman & Das, 1989; Srinivas & Patnaik, 1994; Bäck, 1994).

Although separating individuals into independent populations is a natural way to achieve a better balance of exploration and exploitation for a particular problem, the above problem of choice still remains. Common PGA paradigms are described throughout the literature and use the same evolution behaviour on each population. One still has to optimize the evolutionary algorithm running on each population trying to select the most appropriate operators, parameters, mechanisms and communication methods. See references (Adamidis, 1994; Belding, 1995; Bianchini & Brown, 1993; Cantú-Paz, 1995; Cohoon, Martin & Richards, 1990; Cohoon, Martin & Richards, 1991; Dorigo & Maniezzo, 1993; Fogarty & Huang, 1990; Gordon & Whitley, 1993; Gorges-Schleuter, 1989; Hauser & Manner, 1994; Levine, 1994; Lin, Punch & Goodman, 1994; Macfarlane & East, 1989; Marin, Trelles-Salazar & Sandoval, 1994; McClurkin, Geary & Durrani 1990; Mühlenbein, 1991; Mühlenbein, Schomisch & Born, 1991; Norman, 1988; Pettey & Leuze, 1989; Seredynski, 1994; Shonkwiler, 1993; Tamaki & Nishikawa, 1992; Tanese, 1987; Tanese, 1989; Voigt, Santibanez-Koref & Born, 1992) to mention but a few.

The second objective of this paper is to propose a method for overcoming the problem of choice, using a variety of selection mechanisms, operators, communication methods, and parameters. The optimization method is called *Co-operating Populations with Different Evolution Behaviours (CoPDEB)*, and uses different evolution behaviour on each population to overcome the problem of operator selection and improve the performance of cgPGAs. In CoPDEB a number of populations co-evolve, usually, on different processors. Thus, we can integrate a variety of tools, in the same evolutionary engine, which match well the properties of different parts of the search space, improving the exploration and exploitation of the search space.

A function proposed by Rastrigin (Törn & Zilinskas, 1989) was used to study some of the characteristics of the proposed method. We have also tested CoPDEB in training a fully connected Recurrent Artificial Neural Network as a pattern generator, and solving the TSP problem. Our results from these three problems show that CoPDEB significantly outperforms cgPGAs where each population evolves using the same operators, parameters, and mechanisms.

2. Definition of "Co-operating Populations"

Genetic evolution takes place according to a set of rules regarding selection, recombination and mutation.

We define a model (CoP) to formalize a type of complex GA-like system, with the following characteristics

1. The system is composed of a finite number of items at each of several levels
2. The structure of the items at each level depends on the structure of the items of all the previous levels.
3. The structure of the items at the same level is similar.
4. Items on the same level can exchange some information, which might be the result of some processing, while maintaining their structure.
5. Each item functions as a global random search method emulating natural evolution.

CoP can be informally described as follows. Populations of individuals live on islands that are able to communicate with each other. At a second level, populations of islands can form a larger entity with each island regarded as an individual. At a next level, these larger entities can also form a population with each one regarded as an individual. The procedure can be continued to the required level. New offsprings are created by genetic operators within a population of individuals.

An individual can be a string, an island or an entity at a higher level. Each population can evolve globally, or individuals can select from, crossover with, and replace individuals in a bounded region (neighborhood/deme). Different operators and parameters can be used by each population. Every k generations, some of the best individuals of a population are sent to the other populations. Genetic evolution takes place according to a set of rules regarding selection, recombination and mutation.

Definition: CoP^k is defined recursively over k as a four-tuple which denotes level-*k* of the model

$$CoP^k = \langle\ CoP^{k-1},\ t,\ Reproduction,\ Structure\ \rangle$$

with initial level defined for k=0 as:

$$CoP^0 = \langle\ P^0,\ t,\ Reproduction,\ Structure\ \rangle$$

where:

$\boldsymbol{CoP^{k-1}}$ is the level *k-1* of the model

$\boldsymbol{P^0}$ *is the initial population of chromosomes*

t denotes the termination criterion

Reproduction is the scheme used to advance one generation at level k, and can be described as a five-tuple

$$Reproduction = \langle\ epoch,\ selection,\ operators,\ mechanisms,\ parameters\ \rangle$$

epoch denotes the number of generations at level k-1, to start reproduction of new generation at level k. At initial level 0 (CoP^0) *epoch* is equal to 1.

selection denotes selection methods used, such as proportionate selection, tournament selection, ranking selection, steady state selection, Boltzmann selection, and sigma truncation

operators is a tuple of genetic operators used in the transition from population at time t, to the next population at time t+1. These operators include crossover and mutation, along with their variations, inversion, hill-climbing, or other problem or implementation specific devised operators

mechanisms denotes general mechanisms such as elitism, replacement method, fitness scaling, fitness windowing, fitness ranking, evolution of operators, migration strategy used to communicate individuals between demes or between populations

parameters denotes selection methods parameters, such as selection probabilities, tournament size, binary tournament probability, number of reproductive trials, and operator parameters such as mutation rate, crossover probability, crossover points (number and position), length of the string of an individual (length may vary between different populations or from one generation to another), frequencies of probability adaptation of the parameters, number of individuals to migrate between populations or demes, etc.

Structure of the CoP can be described as a six-tuple

$$Structure = \langle\, popsize, demesize, demestruct, \lambda, topology, popalloc \,\rangle$$

popsize denotes the number of individuals of a population

demesize denotes the size of each deme, that is the number of individuals in a bounded region when population is not "panmictic"

demestruct denotes the structure of the demes. Demes may be overlapping. In this case genetic material flows through demes and no direct exchange of individuals is needed.

λ denotes the number of available nodes.

topology denotes the physical interconnection between nodes and it affects the performance. It may be static (when the connections between nodes are established at the beginning of the run and are not modified, keeping the network topology static during execution) or dynamic (when connections between nodes change over time).

popalloc denotes the specific configuration used for mapping population(s) to node(s)

When $k \geq 0$, reproduction usually reduces to mere exchange of individuals (migration), and epoch corresponds to migration rate. Depending on the level this could mean migration of strings, islands, or larger entities. Evolution of individuals before migration is also a possibility. Operators like crossover and mutation can be defined for any level of the model.

Since the model is hierarchically organized and based on its scalability, it can be implemented on different levels of parallelism. The previous definition is a unifying framework for all kinds of PGAs both coarse and fine-grained.

The initial level of CoP (CoP^0) can describe both massively parallel (fine-grained or cellular) GAs and simple GAs. When *demesize* is equal to *popsize*, there are no neighborhoods and the population is "panmictic", that is each individual can mate with any other individual in the population. When only crossover and mutation are used as genetic operators, the model reduces to the simple genetic algorithm.

3. Co-operating Populations at level 1 (*CoP^1*)

The CoP model is hierarchically organized. It is composed of several subsystems, which are composed of even smaller subsystems, and so on. At level 1, the model is composed of several islands. Each island evolves running an evolutionary algorithm on its population. The population of each island is composed of the initial population of chromosomes, that is a population of candidate solutions to a given problem, after being properly encoded.

The main representative of CoP^1 used in the research so far is the cgPGA and specifically the island model.

3.1. Coarse-Grained Parallel Genetic Algorithms

It is the more well studied instance of CoP^1. In a coarse-grained PGA (cgPGA) the population is divided into several equal subpopulations, each of which runs, independently and in parallel, the same GA configuration on its population.

The individuals of a population are relatively isolated from individuals in another population. Isolated islands help maintain genetic diversity. Each island is potentially able to explore a different part of the search space. Occasionally, fit individuals migrate from one population to another. Migrant individuals may move to geographically nearby populations (stepping stone model), or to any arbitrary population (island model). The term "distributed GAs" is used sometimes when referring to cgPGAs, since they are usually implemented on distributed MIMD computers. CgPGAs have been used successfully in solving a broad range of difficult problems. In most cases the parallel GA with migration outperformed a serial GA. Overcoming the limits set from present availability of computing power, is another reason for the popularity of cgPGAs.

According to the definition of our model, the *Reproduction* tuple of level 1 used in the bibliography is as follows:

epoch is the number of generations that each population evolves before migration occur, i.e., how often processors exchange individuals (also called migration interval or migration frequency)

selection denotes the method used to choose individuals to migrate

No *operators* are used. Eventhough a number of operators can be applied to migrating individuals, no processing of migrants is used to the implementations through the literature

parameters denotes the number of individuals to migrate, i.e., the number of individuals that populations exchange with each other (migration rate)

mechanisms denotes the migration strategy i.e. which other islands an island exchanges individuals with; the replacement method that migrants will use to replace individuals in the heading island

The components of the S*tructure* tuple are as follows:

popsize is equal to the total number of available islands *demesize* is equal to number of islands that communicate with each other

demestruct denotes the structure of each deme

λ is equal to the number of available processors at level 1

topology denotes the actual physical connections between the available processors at level 1. It is usually static.

popalloc denotes which islands are allocated to each processor. One or more islands may be allocated to each processor

Since the model is recursively defined, the model at level 0 must also be defined before any implementation of cgPGA is possible. Eventhough the model at level 0 can be any evolutionary algorithm (it can even be any optimization method such as simulated annealing) it is usually a simple genetic algorithm. The implementations presented in the bibliography run the same identical evolutionary algorithm on each island (level 0 of the model).

Few exceptions on this rule are:

- the different encodings to the same problem using different individual sizes on different populations (Punch, & Goodman, 1994),
- the competitive evolution, where two different crossover operators and different population sizes are used, allowing only the best populations to evolve further (Pham, 1994),
- the different mutation rates used on three populations of the GAMAS system (Potts et al. 1994),
- the co-operating populations where populations are allowed to evolve using a number of different operators and parameters (Adamidis & Petridis, 1996).

Since cgPGAs are the most popular model, a great deal of research has been carried out, both empirically and analytically. Despite the attempts to provide some theoretical foundations (Goldberg, 1989b; Goldberg, Kargupta, Horn, & Cantu-Paz, 1995; Lin, Punch & Goodman, 1994; Mühlenbein, 1991; Pettey & Leuze, 1989; Shonkwiler, 1993; Thierens & Goldberg, 1993), the setting of the parameters is still implemented using intuition rather than analysis.

In most published papers, migration occurs at predetermined constant intervals (synchronous migration). In some of them, migration occurs after the populations converge (Braun, 1990; Munetomo, Takai & Sato, 1993), or asynchronously based on a single event, that does not relate to the state of evolution across all of the system's subpopulations (Lin, Punch & Goodman, 1994; Norman, 1988). Generally, the correct building blocks in the migrants influence the search in the right direction; thus migration must not occur too early.

In our implementation of cgPGA, migration occurs at predetermined intervals (*epoch* value). Before the first migration, we let the populations to evolve for a number of generations. So, each population is allowed to search a different part of the solution space, before enriched with information from the other populations, and correct building blocks have the chance to develop in each migrant individual.

The number of individuals exchanged is directly connected to the *epoch* value. Small epoch value and too few migrating individuals, or very large epoch value and too many migrating individuals degrade the performance of the algorithm (Tanese, 1989).

Another important issue is the topology of the implementation. Eventhough most implementations use the native topology of the available computer, there is a significant number of implementations with different topologies, and some comparisons between different topologies. Generally, if the topology has a dense connectivity (or a short diameter, or both), good solutions will spread fast to all the demes, and may quickly take over the population. On the other hand, if the topology is sparsely connected (or has a long diameter), solutions will spread slower and the populations will be more isolated from each other, permitting the appearance of different solutions. A more recent empirical study showed that topologies with higher arity find the global solution faster than the more sparsely connected ones (Cantú-Paz, Mejía-Olvera 1994).

Different migration strategies (schemes) and replacement policies have been used and tested by several researchers. The insertion of migrants with some replacement method that considers the fitness of the individual to be replaced and/or the improvement of the populations, usually improves performance (Maresky, 1994)

3.2. Co-operating Populations with Different Evolution Behaviour

The main idea of CoPDEB is to maintain a number of populations exhibiting different evolution behaviours, as opposed to cgPGAs presented through the bibliography where each population evolves using the same operators, parameters and mechanisms. The exchange of information between the populations allows them to co-operate. Thus, they can exploit promising areas of the search space found by the other populations, and reintroduce in the population previously lost genetic material.

The definitions made in section 3.1 for the components of the *Reproduction* and *Topology* tuples of cgPGAs are still valid in the case of CoPDEB.

The difference in evolution behavior results by varying the implementation of the model at level 0 for each island. Besides the choices that we have to make in the case of cgPGAs, here, we have also to choose the evolution behavior of each population, that is, to choose the appropriate operators, parameters, and mechanisms for each island. In this direction we have to find a configuration for each island so that a variety of tools will be integrated in the same evolutionary engine. These tools should match well the properties of different parts of the search space, improving the exploration and exploitation of the search space.

The choice of operators and parameter values for each population is important for the performance of CoPDEB.

CoPDEB Advantages: CoPDEB have the following advantages over cgPGAs with identical evolution behavior on each population

1. Relieve the problem of selection of operators, parameters and other mechanisms. Eventhough many rules have been proposed and suggestions have been made, setting of parameters and configuration of the evolutionary algorithm is not a science. It still depends on the demands of the problem and on the user's experience and aesthetic preferences. In CoPDEB the use of a variety of settings makes it easier to select the configuration for each population, since it is not unique; other populations can use different configurations.
2. Longer building blocks have more chances to survive. The existence of an operator, or the lack of it, may have a detrimental effect, which is not known in advance. Some operators may be

oriented to handle longer schemata, having crossover to occur out of the defining length of the schema. Thus, the use of different types of operators and parameters on different populations, make it possible for a longer schema to survive. At the same time, it is possible to use operators oriented at disrupting long length schemata. When the average fitness of the strings that represent a schema is above average, it will propagate to all populations, and it will be handled by all operators. Thus the detrimental effect of an operator, or the lack of it, will not influence all populations, allowing longer building blocks to survive in some islands and propagate to the others.

3. Formation of niche and species is encouraged. Each population uses different operators, creating a different species, which can be seen as a class of organism with common characteristics. In this way, we can observe stable subpopulations of strings (species) serving different subdomains of a function's domain (niches). In CoPDEB, the different evolution behaviour of each population, encourages the formation of niche and species
4. Increase the speed of convergence having to speculate fewer combinations. A GA discovers new solutions by speculating on many combinations of the best partial solutions contained within the current population. Having longer building blocks, the GA has to speculate fewer combinations.
5. Different block sizes can be used by each population to encode the problem, resulting in different levels of resolution between different populations. Using lower resolution, the search space is proportionally smaller. This results in finding better solutions more quickly. These solutions are refined, when migrated to other populations with higher resolution.

4. Implementation Issues of Coarse-grained PGAs and CoPDEB

Both cgPGAs and CoPDEB are completely specified by defining all the components of the CoP^1 four-tuple introduced earlier in section 2. The following is a description of the components of the tuple that we adopted in the specific implementations used in this paper.

The *Reproduction* tuple of both models, defined for CoP^1 in section 3.1., is implemented as follows:

epoch values used in our experiments vary between 1 and 500 generations.

Best individual from each population is *selected* to migrate

No *operators* are used to process individuals, prior to migration.

One individual from each population is migrated (*parameters*).

Migration strategy: Each island sends its best individual to every population, and receives the best individual from all the other islands. Incoming individuals replace random individuals of receiving island. Best individual of receiving island is not allowed to be replaced. (*mechanisms*)

The components of the S*tructure* tuple are as follows:

Eight islands were used (*popsize*)

Since *demesize* is equal to *popsize*, no demes are defined

Number of available processors (*ë*) is eight (8).

topology is static. All available processors are directly or indirectly connected to each other.

Both algorithms were implemented on the Quintek Fast9 transputer board. Since these algorithms are independent of the specific machine architecture, they can be implemented on any parallel (and even serial) machine or network. Each transputer is directly connected to four other transputers, and is able to communicate with the rest of them, through those that it is directly connected. Each island is allocated on a different processor. Since all processors are able to communicate with each other, any population can be allocated to any processor (*popalloc*).

Termination criterion (*t*) is satisfied when either a number of generations has been completed, or the optimum has been found, or the best solution found does not improve for a number of generations.

The implementation of cgPGAs and CoPDEB would be the same if the model at level 0 (CoP^0) was the same. Since the model is recursively defined, CoP^0 must be defined before any implementation is possible.

With these implementations, we are trying to establish that the performance improves, when populations use different evolution behaviors. We tried to choose the best configuration for the CoP^0 model in the case of cgPGAs, where each island evolves under the same evolution behaviour. This makes the comparison between cgPGAs and CoPDEB, general, adequate, objective, and not biased to the specific implementations. In the case of CoPDEB, each CoP^0 model evolves using a different configuration.

The algorithm run by each population is as follows:

```
{
   Initialization
   Repeat  {
      Repeat  {
         Elitism (Best survives always)
         Reproduction of m-1 offsprings
      }  until  an_epoch_is_completed
      Send best individual to all other populations
      Receive the best individual from each one of the other
      populations
   }  until termination_criterion
}
```

P^0, the initial population of each island is randomly created, after the encoding for each problem is determined. We used fixed-length, binary encodings for both cgPGAs and CoPDEB for all three problems. Binary encodings are the most common encodings and GA practice has tended to follow this lead (Goldberg, 1989a, Mitchell, 1996). Since CoPDEB focuses on the effect of different evolution behaviours, we did not choose to use different encodings on different populations.

The implementation of the *Structure* tuple of CoP^0 is as follows and it is the same for both cgPGAs and CoPDEB.

popsize is equal to the number of individuals per population. Values used in our experiments vary from 50 to 200 individuals. As population of each island is “panmictic” *demesize* is equal to *popsize,* and no demes are defined. Since the whole population is allocated to one processor (ë, and *popalloc*), *topology* is not defined.

As a first step in the investigation of CoPDEB, we chose to focus on the effect of the *Reproduction* tuple, which is instrumental to the generation of different evolution behaviours.

4.1 Reproduction Tuple of Coarse-grained Parallel Genetic Algorithms

This tuple is what makes evolution behaviour to emerge. In order to have an objective and adequate comparison between cgPGAs and CoPDEB, we tried to make optimal choices for its components.

By definition the *epoch* value is equal to 1, since every generation, a new population is produced.

The selection component is oriented to emphasize the fitter individuals of the population, in hopes that, their offspring will in turn have even higher fitness. Selection has to be balanced with variation from crossover and mutation. One cannot provide rigorous guidelines to choose the most appropriate method for each problem. Selective pressure and genotypic diversity must be balanced properly. We have chosen to use fitness proportionate selection with “roulette wheel”, in combination with linear dynamic scaling method. Proportionate selection is used in order to maintain a large genotypic diversity of the population during search. Scaling is used, because proportionate selection without scaling was found to be less effective in keeping a steady pressure towards convergence (Bäck, 1994b; Bäck & Hoffmeister, 1991; de la Maza & Tidor, 1993, Farrell, Kieronska & Schulze, 1994; Goldberg & Deb, 1991).

There are many papers in the GA literature, which quantify aspects of various crossover operators, but these do not give definitive guidance on choosing the appropriate type of crossover. There are also many papers, which empirically compare the usefulness of different types of crossover. Our initial results, with different types of crossover on the first two problems mentioned in section 5, has lead us to choose adaptive multi-point crossover (Spears & De Jong, 1991; De Jong & Spears, 1992). Number of crossover points is set equal to the number of genes.

Some researchers, mainly in the GA community, regard mutation as much less significant than crossover, serving merely as a background operator. Other judgements of mutation’s importance are higher; in some evolutionary computation methods, mutation is the only source of variation.. The balance among crossover, mutation and selection, is considered to be more important than the choice between crossover and mutation. These parameters cannot be optimized one at a time. There is a great deal of discussion of parameter setting and approaches to parameter adaptation in the evolutionary computation literature (Bäck, 1993; Bramlette, 1991; Davis, 1989; Fogarty, 1989; Mühlenbein, 1992; Schaffer, Caruana, Eshelman & Das, 1989; Spears, 1993; Starkweather, Whitley, & Mathias, 1990;).

On line adaptation of crossover and mutation parameters, and fitness scaling are implemented as follows. Premature convergence or excessive diversity are monitored by extracting statistical information from the population. When such a situation is speculated, the operator probabilities and the fitness scaling function are adjusted to keep the population distribution in a reasonable state. When premature convergence is observed, the crossover probability is lowered while mutation probability is increased. The opposite happens when excessive diversity occurs (Bakirtzis, Petridis & Kazarlis, 1994; Kazarlis, Bakirtzis & Petridis, 1994; Petridis & Kazarlis, 1994; Petridis, Kazarlis & Papaikonomou, 1993).

Though most GA applications use only crossover and mutation, many other operators and strategies for applying them have been explored in the literature. Local hill-climbing was successfully used for combinatorial and parametric optimization problems (Mühlenbein, 1989; Mühlenbein, 1992; Mühlenbein, Schomisch & Born, 1991). In order to accelerate the search we used a hill-climbing like operator called "phenotype mutation" (Petridis, Kazarlis & Papaikonomou, 1993). The operator is applied only to the best individual of every generation. It performs a local direct search in the neighborhood of the best individual, in an effort to find a better point in the search space. This operator increases to a great extend the speed of convergence of the GA, especially in case of search spaces without many discontinuities.

In the case of the TSP problem (search space with many discontinuities) this operator rarely finds a valid point and it is not used. Since we do not want to solve the TSP problem, but rather focus on the better optimization and searching capabilities of CoPDEB over cgPGAs, we did not alter substantially the original behaviour of the algorithm used in the previous two problems. In this context we did not use any operators, codings or mechanisms oriented at solving this particular problem such as edge recombination, order crossover, PMX, cycle crossover, position based crossover, matrix crossover, locus based encoding, redundant genetic coding, etc. (Bui & Moon, 1994; Cui & Fogarty, 1992; Davis, 1991; Dzubera & Whitley, 1994; Homaifar, Guan & Liepins, 1993; Jog, Suh & Van Gucht, 1989; Tamaki et al., 1994; Mathias & Whitley, 1992; Mühlenbein, 1991; Mühlenbein, 1992; Mühlenbein, Gorges-Schleuter & Kramer, 1988; Peterson, 1990; Tang & Leung, 1994; Valenzuela & Jones, 1993; Whitley, Starkweather & Fuquay, 1989).

Elitism is also used, in order to eliminate the possibility of destruction of the best individual by a genetic operator. We retain only the best individual at each generation.

4.2 Reproduction Tuple of CoPDEB

Our objective is to provide some experimental evidence that it is not only useful but it is preferable to evolve each population using a different *Reproduction* tuple. When each population exhibits different evolution behavior, exploration and exploitation of the search space are improved. In essence different evolution behaviour is the result of different operators, parameters and mechanisms for each population.

The configuration chosen for each population of CoPDEB, will produce results comparable with the configuration of cgPGA described earlier. The different parameter settings for each population of CoPDEB are not the best, and are not tuned to the specific test problems, in order to test the usefulness and effectiveness of the CoPDEB idea. Under these restrictions the specific implementation of CoPDEB is **not optimal**. Thus the better searching capabilities and the improved performance of CoPDEB cannot be attributed to the better configuration of its populations but on the combination of the different evolution behaviours.

Since *popsize* of CoP^1 is eight, we have to define eight different *Reproduction* tuples.

By definition the *epoch* value is equal to 1, and it is the same with cgPGAs.

In order to keep differences from cgPGAs to a minimum, and choose configurations that are not better from the cgPGAs configuration, we have chosen to use on each population:

- fitness proportionate selection with "roulette wheel" in combination with linear dynamic scaling method.
- hill-climbing operator "phenotype mutation".
- elitism, retaining only the best individual at each generation.
- adaptive crossover probability and mutation rate (except when stated otherwise).

Their different characteristics are described below. For convenience of reference, the populations are numbered from 1 to 8:

Population 1: Its configuration is identical to configuration of cgPGA populations.

Population 2: Instead of multi-point crossover we used an operator that handles(crossover) genes and not whole chromosomes. It uses one point crossover for each gene separating it into two parts. It creates the offspring taking the most significant part from the better parent and the least significant part from the worse. This technique produces new chromosomes with small perturbations of their values in the proximity of the chromosome value of the better parent (MSP_LSP operator).

Population 3: Instead of multi-point crossover we used a crossover operator that functions like MSP_LSP, but it creates the offspring with the least significant part of the best parent and the most significant part of the worse. This allows for small perturbations of the chromosome values, in the proximity of the chromosome value of the worse parent (LSP_MSP operator).

Population 4: It uses two point crossover instead of multi-point crossover.

Population 5: Number of crossover points is set equal to half the number of genes.

Population 6: Uniform crossover (Syswerda, 1989) is used, instead of multi-point crossover

Population 7: Like population 1, but with adaptive parameter probabilities in favour of mutation. That is in the case of Population 1, premature convergence is determined when the 50% or more of the population has fitness near to the best fitness. Here the limit is reduced to the 8% of the population. Divergence, in case of Population 1, is determined when less than 10% of the population has fitness near the best fitness. Here this limit is reduced to 3%. Thus in this population we can maintain a higher value of mutation rate, in order to increase exploration of the search space, and mitigate (globally and in co-ordination with the other populations) the problem of premature convergence.

Population 8: Configuration of Population 1, without adaptation of crossover probability and mutation rate. Constant smaller crossover probability (0.2) and larger mutation rate. This configuration has the same objective of the previous Population 7.

Constructing each population we have chosen some configurations that are not optimal on their own, in order to test the usefulness of the CoPDEB idea.

5. Optimization Problems

The following problems were used to test the performance of CoPDEB over cgPGAs.

In the first problem, CoPDEB and cgPGAs were used to train a fully connected Recurrent Artificial Neural Network (RANN) to generate two limit cycles, in the form of two sinusoidal functions of different amplitudes and frequencies. The RANN has had 5 neurons (35 trainable weights), two of which were the output units. Since every weight is encoded in a 16 bit string, the genotype strings are 35x16=560 bits long, resulting in a search space of 2^{560}=3.77x10^{168} different values. Different input levels (0.2 and 0.8) had to lead the two output units in different oscillations. The desired outputs were: y1=0.2∗sin(2ðt/T) and y2=0.8∗sin(ðt/T) respectively. The model of the fully connected RANN has been presented in Petridis, Kazarlis & Papaikonomou, 1993. The overall network error at time t is:

$$J(t) = \frac{1}{2}\sum_{k}^{M}[d_k(t) - y_k(t)]^2 \tag{1}$$

where M is the number of units for which there exists a desired target value $d_k(t)$ for every time step, and $y_k(t)$ is the output of the k-th unit at the same time.
The target is to adjust the weight values so that the total network error

$$J_T = \sum_{t=1}^{T} J(t) \tag{2}$$

becomes less than a predefined quantity. T is the largest period of the M desired output signals (period of the desired output signal or the presentation time of a temporal sequence).

The RANN was trained until its total network error reached a negligible error of 0.003. It is a difficult problem demanding a lot of time and computational resources. More on RANN as pattern generators can be found in (Petridis & Papaikonomou, 1994).

The second problem is the optimization function proposed by Rastrigin (Törn & Zilinskas, 1989)

$$f(x) = nA + \sum_{1}^{n}(x_i^2 - A\cos(\omega x_i)) \tag{3}$$

n=20; A=10; ù=2ð; $-5.12 \le x_i \le 5.12$;

The surface of the function is determined by the external variables A and ω, which control the amplitude and frequency of the modulation respectively. With A=10 the selected domain is dominated by the modulation. The function is highly multimodal. The local minima are located at a rectangular grid with size 1. The global minimum is at x_i=0, i=1...,n, giving $f(x)$=0. Grid points with x_i=0 except one co-ordinate, where x_i=1, give f=1, the second best minimum. With increasing distance to the global minimum the fitness values of local minima become larger.

The third problem is the well-known combinatorial Travelling Salesperson problem, which involves choosing the optimal tour among a set of cities, visiting each city exactly once and then returning to the start city, where the object is typically to minimize the total distance overall. Data is a set of routes (population); a route might implicitly contain one or more best sub-route. The problem is to find an optimum complete route. We have taken a symmetric TSP problem from the TSPLIB, a library of travelling salesperson and related problem instances. We chose a moderate size problem with 29 cities in Bavaria, in order to conduct many experiments for statistical reasons.

6. Experimental Results

Before comparing the results of CoPDEB method with cgPGA, we tested the performance of the eight configurations on the first two problems (RANN, and Rastrigin function). For both problems we run a set of 100 experiments for each configuration. Population size was set to 200 individuals. The results show that the configuration of population 1 used in cgPGAs has better or comparable performance with the other configurations (Table 1).

Table 1. Comparison results from different population configurations

	Rastrigin Function		RANN	
Configuration of population	Error of best individual	Mean Error	Success Rate	Mean Error
1	0.994010	6.7559	52%	0.0105
2	1.990128	7.1242	41%	0.0114
3	2.984909	6.8653	40%	0.0140
4	0.994995	6.7535	43%	0.0117
5	2.984970	7.0645	20%	0.0221
6	1.989944	6.8852	46%	0.0117
7	2.984924	6.8311	35%	0.0126
8	1.990173	8.1290	50%	0.0102

We compared the results of CoPDEB method with the results of a cgPGA, with populations evolved using identical evolution behaviours, on the same problems.

We have performed two sets of experiments for comparison purposes. The first set concerns CoPDEB, and the second concerns a cgPGA with the evolution characteristics of population 1.

In the case of RANN, each set comprises 90 experiments, that is 15 experiments for six different epoch values. The epoch values used were: 1, 2, 5, 10, 20, and 50 generations. Population size for each island was set to 200 individuals. The experiments showed an improved performance of CoPDEB over cgPGAs. The success rate of CoPDEB was, in the worst case, doubled (Figure 1). That is, for epoch value equal to 2, the success percentage for CoPDEB was 53.3%, which means that the number of successful experiments was 0.533x15=8, whereas for the cgPGA it was 26.7%, which means that the number of successful experiments was 0.267x15=4.

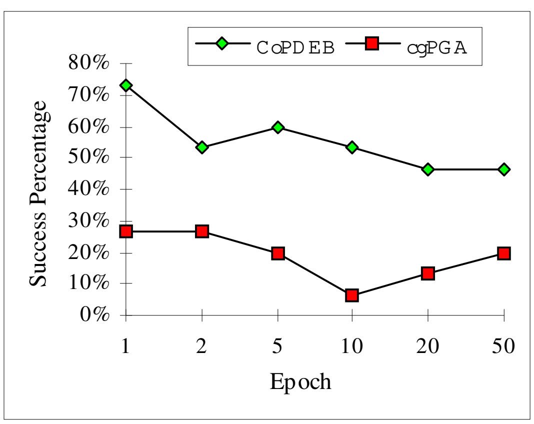

Figure 1. Successful experiments (RANN)

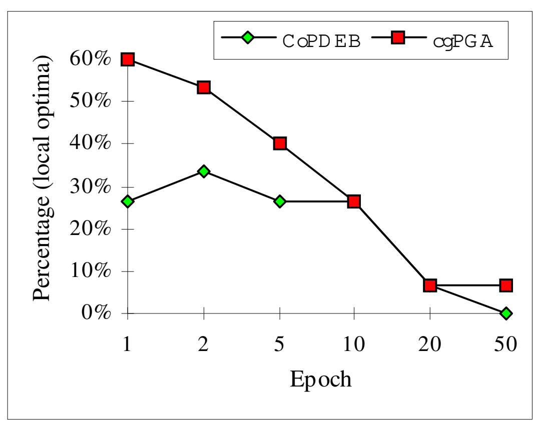

Figure 2. Experiments stuck at local optima (RANN)

Apart from the improvement in the success rate, we have also observed that, when using CoPDEB, the percentage of the experiments that were stuck at local optima was decreased, and it kept decreasing as the Epoch value was increasing. The failure of the most of the experiments using CoPDEB, was because the generation limit (10000 generations) was reached. In the case of the cgPGA, most of the failures were due to being stuck at local optima (Figures 2 and 3). Thus we can assume that CoPDEB improves the exploration of the search space.

The number of generations that were required for the RANN to be trained in the case of CoPDEB was also smaller (Figure 4).

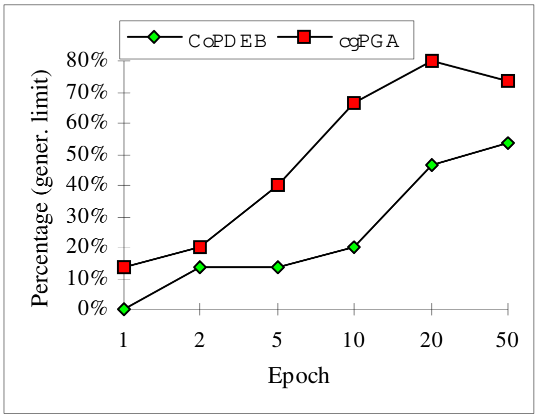

Figure 3. Experiments that were stopped by the generation limit (RANN).

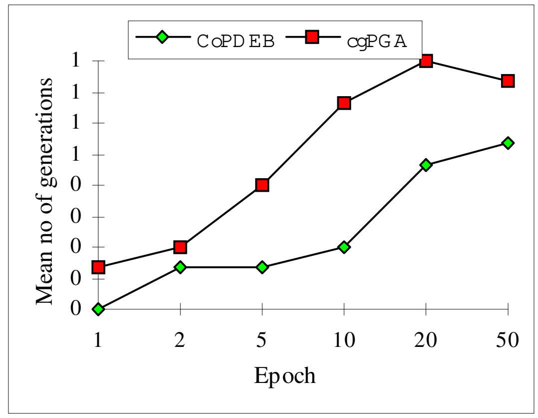

Figure 4. Average number of generations for successful experiments (RANN).

The second problem (real function optimization), apart from the performance comparison of CoPDEB and cgPGAs, was also used to test extensively the effect of different epoch values on the performance of CoPDEB and cgPGAs. This function is less time consuming than RANN training.

In this case, each set comprises 4500 experiments. Epoch values ranging from 1 to 500 generations were used. The population values used were: 50, 100 and 200 individuals per island. The results are the mean values from 100 runs.

The results certify the improved performance of CoPDEB over cgPGAs. Independent of the epoch value and the population size, the performance of CoPDEB is better (Figure 5).

It is also clear that the performance of both CoPDEB and cgPGAs, depends on the epoch value used. The mean error improves in both paradigms, as the epoch value increases. The results show a high linear correlation between epoch values used and mean errors. The tendency of mean errors to decrease as epoch increases is greater for CoPDEB. This can be seen in the slopes of the linear regression lines (Table 2). The rate of improvement of the CoPDEB is larger compared to cgPGAs.

The rate of improvement of the CoPDEB performance with epoch is also increased, as population per island is increased. This is not the case for cgPGAs.

Table 2. Slopes of the linear regression lines for different population sizes.

	50	**100**	**200**
CoPDEB	-0.00851	-0.00965	-0.0104
cgPGA	-0.00727	-0.00336	-0.00433

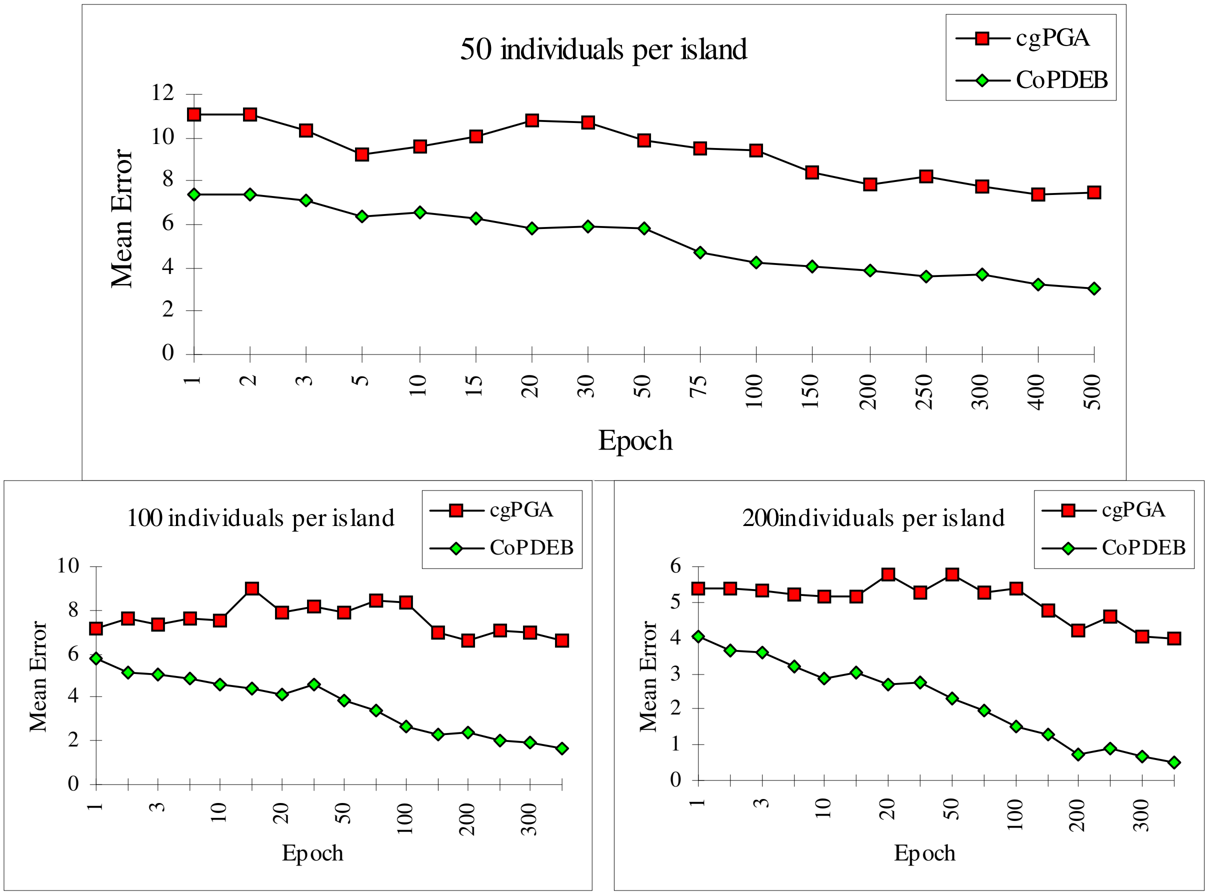

Figure 5. Performance of CoPDEB and cgPGAs on the real function optimization

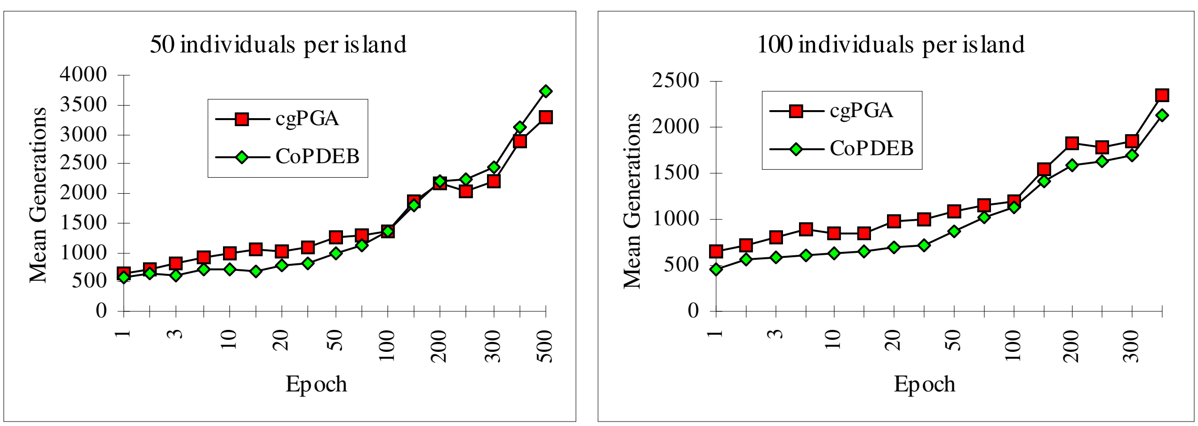

Figure 6. Average number of generations (Rastrigin Function)

As the epoch value increases, the number of generations to converge is also increased. (Figure 6).

Another aspect of our work concerns the population size of the different populations both in cgPGAs and CoPDEB. Obviously, a larger population results in a more dense sampling of the search space and a better estimate. The drawback to the use of larger populations is that the rate of improvement or convergence is slower when measured by the number of evaluations performed.

Our results agree with the pioneering work of Grosso (1985), concerning the cgPGA, i.e., the rate of improvement is faster for smaller populations. On the other hand, it is easier for smaller populations to stuck in local optima. The same results hold in the case of CoPDEB.

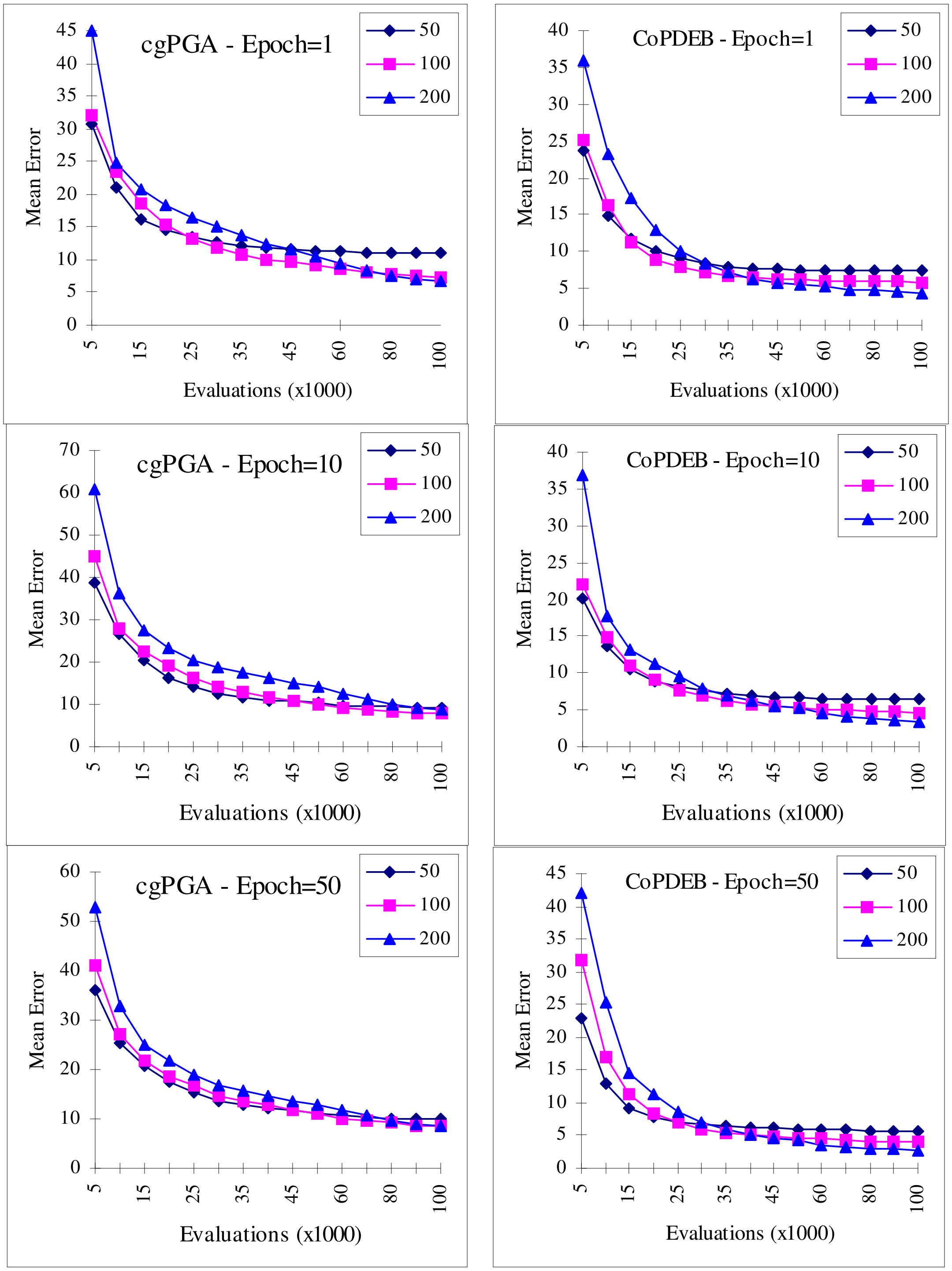

Figure 7. Improvement of mean error for cgPGAs and CoPDEB

Figure 7 shows the improvement of the mean error for three population sizes (50, 100, and 200) above critical population size defined by Goldberg et al. (1995). Initially, performance with population size 50 is better than the other two, but it becomes stuck in local optima and is surpassed by population size 100, which is later surpassed by population size 200. Though Figure 7 shows the performance only for epoch values 1, 10, and 50, the results keep the same form at all other epoch values.

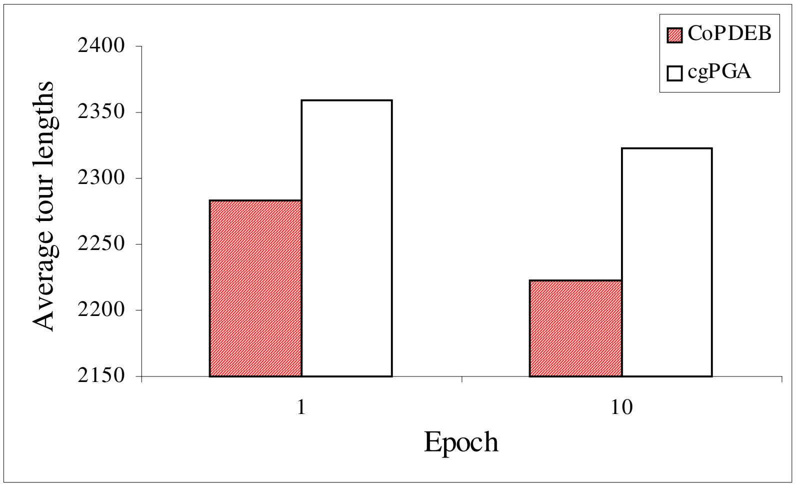

Figure 8. Results from TSP problem

The final problem (TSP) was used in order to check the performance superiority of CoPDEB over cgPGAs in combinatorial problems. Epoch values 1 and 10 were tested. The population of each island was 50 individuals. The results are the mean values from 50 runs.

As we have already pointed out, the CoPDEB configuration used is not optimal in solving the TSP problem. Better performance of CoPDEB over cgPGAs is observed in this problem too. Performance using epoch value 10 was better than epoch value 1. (Figure 8)

7. Discussion - Conclusions

By introducing migration, differences in the various subpopulations are exploited, which in fact represent a source of genetic diversity. If migration is too infrequent (large *epoch* value), it may not be enough to prevent each population from prematurely converging (Whitley, 1991). Migrating individuals too often, or not often enough, degrade performance (Starkweather, Whitley & Mathias, 1990).

Our results propose that infrequent migration helps to avoid premature convergence of all populations, though some of them may have converged prematurely at a local optimum. Without migration at all, these populations would have converged even earlier. When communication is infrequent, populations investigate different parts of the search space and contain different genetic information, which when introduced to converged populations, help them escape from local optima.

When migration becomes more often, populations are directed to search on some part of the search space, loosing their diversity, eventhough initially it helps populations avoid premature convergence. Belding, (1995) has also reported loss of diversity, in case of high frequency of migration, for cgPGAs. This holds true in our results for both cgPGA and CoPDEB. Though, very frequent communication helps quickly exploit promising areas of the search space, it can also direct all populations to the same areas of the search space, thus loosing useful genetic material and information about the search space, and leading to premature convergence of all populations.

If we do not consider the number of generations needed to reach the lower error, we could suggest to use an *epoch* value as big as possible. Considering the number of generations, we observe that large epoch values have an immediate effect on the number of generations. The selection of the appropriate epoch value, depends on the problem and the time that we have to find a good solution. If the problem is easy enough we can use a small value, since in this case, the global optimum will be found in a smaller generation number. A small epoch value must also be used in the case that we need a good solution in a very short time. For difficult problems and when we are not interested in the elapsed time, we can use a big epoch value. We have to use big epoch values, in cases that we need to find the global optimum.

Another issue is population size. As Goldberg et al. (1995) say "the optimal prescription of deme size must be dependent on our choice of inter-deme communication and machine architecture". According to our results, increasing the population size was seen to improve long-term performance at the expense of slower improvement and larger number of function evaluations. These conclusions agree with earlier work concerning cgPGAs (Cantú-Paz & Mejía-Olvera, 1994). Thus, a large population size can be used to improve off-line performance, but a small population size (but above critical population size) is required when good on-line performance is sought.

Selecting an intermediate epoch value and population size, could be a choice for any case.

When someone tries to construct optimal parameter settings for different populations, it is of great importance to assemble populations with settings oriented either to exploration or to exploitation of the search space. This will give the opportunity to the best solutions to be handled by different operators. In the most cases a parameter setting is not optimal for the whole search space. It may be optimal for a part of the search space but not for another. Having this in mind, we can think of two distinct cases for choosing optimal settings for any number of populations:

- when the solution to a specific problem is sought, then one should use as much *a priori* knowledge as possible, in order to construct different populations with operators mechanisms and parameters found to tackle successfully the problem at hand.
- when a general tool is constructed, or when little is known about the nature of the problem, the integration of a variety of tools, in the same evolutionary engine is important. Elitism and hill-climbing operator could be used in all populations. Different encoding on different populations, eventhough it is not explored in our configuration of CoPDEB, is quite helpful. The use of different types of crossover (e.g., two-point, multi-point, uniform crossover, etc.) on different populations is also helpful. It is profitable to use either small or large mutation rates in accordance with the crossover probabilities. Operator probabilities could be adaptive, but not against the peculiarity of each population.

In both cases adaptive connection topology can lead to better results and performance.

The results of our work with CoPDEB have shown it to be capable of improving the performance of the cgPGA model, achieving speedup and providing better exploration and exploitation of the search space.

It can also relive the burden of selection of operators, parameter and mechanisms.

Many features can still be investigated, such as different selection mechanisms, different encodings, adaptive population size, migration strategies, different topologies, etc.

References

Adamidis, P. (1994). Review of Parallel Genetic Algorithms Bibliography, Internal Technical Report, Automation and Robotics Lab., Dept. of Electrical and Computer Eng., Aristotle University of Thessaloniki, Greece, November (http://www.control.ee.auth.gr/~panos/papers/pga_review.ps.gz)

Adamidis, P., & Petridis, V. (1996). Co-operating Populations with Different Evolution Behavior. *Proceedings of the Second IEEE Conference on Evolutionary Computation*

Bäck, T. (1993) Oprimal mutation rates in genetic search. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp.2-8). San Mateo, CA: Morgan Kaufmann.

Bäck, T. (1994a). Parallel Optimization of Evolutionary Algorithms. In Y. Davidor, H-P. Schwefel and R. Manner (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature* (pp.418-427), vol 866 of Lecture Notes in Computer Science, Springer-Verlag, 1994

Bäck, T. (1994b). Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms, *Proceedings of the First IEEE Conf. on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, vol I, (pp.57-62)

Bäck, T., Hoffmeister, F. (1991). Extended selection mechanisms in genetic algorithms. In R-K. Belew and L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp.92-99). San Mateo, CA: Morgan Kaufmann.

Bakirtzis, A., Petridis, V., & Kazarlis, S. (1994). A genetic algorithm solution to the economic dispatch problem. *IEE Proceedings - Gener. Transm. Distrib.,* vol 141 (4), (pp. 377-382).

Belding, T.C., (1995). The Distributed Genetic Algorithm Revisited. In D. Eshelman (Ed.) *Procceedings of the Sixth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufman.

Beyer, H-G. (1993). Optimization of large-scale order problems by the Evolution Strategy. In R. Schumacher (Ed.), *One Year KSR1 at the University of Mannheim - Results & Experiences*. Technical Report RUM 35/93, University of Mannheim, December.

Bianchini, R., & Brown, C., (1993). Parallel genetic algorithms on distributed-memory architectures. In S.Atkins and A.S. Wagner(Eds.), *Transputer Research and Applications, Proceedings of the Sixth Conference of the North American Transputer Users Group* (pp. 67-82). Berlin: IOS Press. (ftp://ftp.cs.rochester.edu/pub/systems_papers/93.tr436.parallel_genetic_ algorithms.ps.Z)

Bramlette M.F. (1991). Initialization, mutation and selection methods in genetic algorithms for function optimization. In R-K. Belew and L.B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, (pp.100-107). San Mateo, CA: Morgan Kaufmann

Braun, H.. (1990). On travelling salesman problems by genetic algorithms. In H.-P.Schwefel and R.Manner (Eds.), *Proceedings of the First Workshop on Parallel Problem Solving from Nature*, vol. 496 of Lecture Notes in Computer Science (pp.129-133). Berlin : Springer-Verlag.

Bui, T., & Moon, B. (1994).A new genetic approach for the travelling salesman problem. *Proceedings of the IEEE World Congress on Computational Intelligence, First IEEE Conference on Evolutionary Computation*, vol I, (pp. 7-12).

Cantú-Paz, E. (1995). A Summary of Research on Parallel Genetic Algorithms. IlliGAL Report No. 95007, Illinois Genetic Algorithms Lab., Dept. of General Eng., University of Illinois at Urbana-Champaign, July. (ftp://gal4.ge.uiuc.edu:pub/papers/IlliGALs/95007.ps.Z)

Cantú-Paz, E., & Mejía-Olvera, M., (1994). Experimental results in distributed genetic algorithms. In *International Symposium on Applied Corporate Computing*, (pp.99-108). Montery, Mexico

Cohoon, J.P., Martin, W.N., & Richards, D.S. (1990). Genetic Algorithms and punctuated equilibria in VLSI. In H.-P.Schwefel and R.Manner (Eds.), *Proceedings of the First Workshop on Parallel Problem Solving from Nature*, vol. 496 of Lecture Notes in Computer Science (pp.134-144). Berlin : Springer-Verlag.

Cohoon, J.P., Martin, W.N., & Richards, D.S. (1991). A Multi-population Genetic Algorithm for Solving the K-Partition Problem on Hyper-cubes. In R-K. Belew and L.B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, (pp.244-248). San Mateo, CA: Morgan Kaufmann

Cui, J., & Fogarty, T.C. (1992). Optimization by Using a Parallel Genetic Algorithm on a Transputer Computing Surface. In M. Varelo, E. Onate, M. Jane, J.L. Larriba, & B. Suarez (Eds.), *Parallel Computing and Transputer Applications, Transputer and Occam Engineering Series*, vol. 28 (pp.246-254). Amsterdam: IOS Press.

Davidor, Y. (1991). A Naturally Occuring Niche & Species Phenomenon: The Model and First Results. In R-K. Belew and L.B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp.257-263). San Mateo, CA: Morgan Kaufmann.

Davidor, Y., Yamada, T., & Nakano, R. (1993). The ECOlogical framework II: Improving GA performance at virtually zero cost. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp.171-176). San Mateo, CA: Morgan Kaufmann.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In J.D. Schaffer(Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 61-69) San Mateo, CA: Morgan Kaufmann.

Davis, L. (Ed.) (1991). *Handbook of genetic algorithms*. New York: Van Norstrand.

De Jong K.A., & Spears W.M. (1992). A formal analysis of the role of multi-point crossover in Genetic Algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1), 1-26.

de la Maza, M., & Tidor B. (1993). An analysis of selection procedures with particular attention paid to proportional and Boltzmann selection. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp.124-131). San Mateo, CA: Morgan Kaufmann.

Dekker, L., & Ribeiro Filho, J.L. (1993). The GAME Virtual Machine Architecture. In J. Stender (Ed.), *Parallel Genetic Algorithms: Theory and Applications* (pp.93-110). Brainware GmbH, Berlin: IOS Press

Dorigo, M., & Maniezzo, V. (1993). Parallel Genetic Algorithms: Introduction and Overview of Current Research. In J. Stender (Ed.), *Parallel Genetic Algorithms: Theory and Applications* (pp.5-42). Brainware GmbH, Berlin: IOS Press

Dzubera, J., & Whitley, D. (1994). Advanced correlation analysis of operators for the traveling salesman problem. In Y.Davidor, H-P. Schwefel, and R. Manner (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature* (pp.68-77). Berlin: Springer-Verlag.

East, I., & Macfarlane, D. (1993). Implementation in Occam of Parallel Genetic Algorithms on Transputer Networks. In J. Stender (Ed.), *Parallel Genetic Algorithms: Theory and Applications* (pp.43-64). Brainware GmbH, Berlin: IOS Press

Fogarty, T.C. (1989). Varying the probability of mutation in the Genetic Algorithm. In J.D. Schaffer(Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 104-109) San Mateo, CA: Morgan Kaufmann.

Fogarty, T.C., & Huang, R. (1990). Implementing the Genetic Algorithm on Transputer Based Parallel Processing Systems. In H.-P.Schwefel and R.Manner (Eds.), *Proceedings of the First Workshop on Parallel Problem Solving from Nature*, vol. 496 of Lecture Notes in Computer Science (pp.145-149). Berlin : Springer-Verlag.

Goldberg, D.E. (1989a). *Genetic algorithms in search optimization and machine learning*. Reading, MA: Addison-Wesley

Goldberg, D.E. (1989b). Sizing Populations for Serial and Parallel Genetic Algorithms. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp.70-79). San Mateo, CA: Morgan Kaufmann

Goldberg, D.E., & Deb K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In G. Rawlins (Ed.), *Foundations of Genetic Algorithms* (pp.69-93). San Mateo, CA: Morgan Kaufmann

Goldberg, D.E., Kargupta, H., Horn, J., & Cantu-Paz, E. (1995). Critical Deme Size for Serial and Parallel Genetic Algorithms. IlliGAL Report No. 95002, Illinois Genetic Algorithms Lab., Dept. of General Eng., University of Illinois at Urbana-Champaign, January. (ftp://gal4.ge.uiuc.edu:pub/papers/IlliGALs/95002.ps.Z)

Gordon, V., & Whitley, D. (1993). Serial and Parallel Genetic Algorithms as Function Optimizers. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp.177-183). San Mateo, CA: Morgan Kaufmann.

Gorges-Schleuter, M. (1989). ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp.422-427). San Mateo, CA: Morgan Kaufmann.

Grosso, P. (1985). Computer simulations of genetic adaptation: parallel subcomponenet interaction in multilocus model. PhD. Thesis, University of Michigan

Hauser, R., & Manner, R. (1994). Implementation of Standard Genetic Algorithm on MIMD Machines. In Y. Davidor, H-P. Schwefel, and R. Manner (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature,* vol 866 of Lecture Notes in Computer Science, (pp.504-513). Berlin: Springer-Verlag.

Homaifar, A., Guan, S., Liepins, G.E. (1993) A new approach on the traveling salesman problem by genetic algorithms. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp.460-466). San Mateo: Morgan Kaufmann.

Jog, P., Suh, J.Y., & Van Gucht, D. (1989). The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the travelling salesman problem. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp.104-109). San Mateo, CA: Morgan Kaufmann

Kazarlis, S., Bakirtzis, A., & Petridis, V. (1994). A genetic algorithm solution to the unit commitment problem.

Levine, D. (1994). A Parallel Genetic Algorithm for the Set Partitioning Problem. Ph.D. Thesis, Mathematics and Computer Science Division, Argonne National Laboratory, May. (info.mcs.anl.gov: pub/tech_reports/reports/ANL9423.ps.Z)

Lin, S.-C., Punch III, W.F., & Goodman, E.D. (1994). Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach, *Parallel & Distributed Processing 94*, Dallas TX

Macfarlane, D., & East, I. (1989). An Investigation of Several Parallel Genetic Algorithms. In S.J. Turner (Ed), *Tools and Techniques for Transputer Applications, Proceedings of the 12th Occam User Group Technical Meeting* (pp.60-67). Amsterdam : IOS Press

Maresky, J.G. (1994). On Effective Communication in Distributed Genetic Algorithms. MSc. Thesis, Institute of Computer Science, Hebrew Univ. of Jerusalem, Israel. (ftp://ftp.huji.ac.il/pub/misc/jonathan/msc-maresky.ps.gz)

Marin, F.J., Trelles-Salazar, O., & Sandoval, F. (1994). Genetic Algorithms on LAN-Message Passing Architectures Using PVM: Application to the Routing Problem. In Y. Davidor, H-P. Schwefel, and R. Manner (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature - PPSN III*, vol 866 of Lecture Notes in Computer Science, (pp.534-543). Berlin: Springer-Verlag.

Mathias, K., & Whitley, D. (1992). Genetic Operators, the fitness landscape and the traveling salesman problem. In R.Manner and B.Manderick (Eds.), *Proceedings of the Second Workshop on Parallel Problem Solving from Nature* (pp.219-228). Amsterdam: Elsevier Science Publishers.

McClurkin, G.D., Geary, R.A., & Durrani T.S. (1990). An Investigation into the Parallelising of Genetic Algorithms (Extended Abstract). In D.J. Prichard and C.J. Scott (Eds), *Applications of Transputers 2, Proceedings of the Second International Conference on Applications of Transputers* (pp.581-7). Amsterdam: IOS Press.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press

Mühlenbein, H., (1989). Parallel genetic algorithms, population genetics, and combinatorial optimization. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 416-421). San Mateo, CA: Morgan Kaufmann.

Mühlenbein, H., (1991). Evolution in Time and Space - The Parallel Genetic Algorithm. In G.E. Rawlins (Ed.), *Foundations of Genetic Algorithms* (pp.316-337). San Mateo, CA: Morgan Kaufmann.

Mühlenbein, H., (1992). How genetic algorithms really work: I. mutation and hillclimbing. In R.Manner and B.Manderick (Eds.), *Proceedings of the Second Workshop on Parallel Problem Solving from Nature* (pp.15-25). Amsterdam: Elsevier Science Publishers.

Mühlenbein, H., Gorges-Schleuter, M., & Kramer, O. (1988). Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7, 65-85.

Mühlenbein, H., Schomisch, M., & Born, J., (1991a). The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17, 619-632.

Mühlenbein, H., Schomisch, M., & Born, J. (1991b). The parallel genetic algorithm as function optimizer. In R-K. Belew and L.B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 271-278). San Mateo, CA: Morgan Kaufmann.

Munetomo, M., Takai, Y., & Sato, Y. (1994). A Genetic Approach to Dynamic Load Balancing in a Distributed Computing System. *Proceedings of the IEEE World Congress on Computational Intelligence, First IEEE Conference on Evolutionary Computation*, vol I, (pp. 418-421).

Norman, M.G. (1988). A genetic approach to topology optimization for multiprocessor architectures. Technical Report ECSP-TR-7, Dept. of Physics, Univ. of Edinburgh.

Peterson, C. (1990). Parallel distributed approaches to combinatorial optimization: Benchmark studies on travelling salesman problem. *Neural Computation* 2 (261-269).

Petridis, V., & Kazarlis, S. (1994). Varyiing quality function in genetic algorithms and the cutting problem. *Proceedings of the IEEE World Congress on Computational Intelligence, First IEEE Conference on Evolutionary Computation*, vol I, (pp. 166-169).

Petridis, V., Kazarlis, S., & Papaikonomou, A. (1993). A Genetic Algorithm for Training Recurrent Neural Networks. *Proceedings of International Joint Conference on Neural Networks*. (pp. 2706-2709). IEEE

Petridis, V., & Papaikonomou, A. (1994). Recurrent Neural Networks as Pattern Generators. *Proceedings of the IEEE International Conference on Neural Networks, IEEE World Congress on Computational Intelligence,* vol II, (pp. 872-875).

Pettey, C.C., & Leuze, M.R. (1989). A Theoretical Investigation of a Parallel Genetic Algorithm. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 398-405). San Mateo, CA: Morgan Kaufmann.

Pham, Q.T. Competitive evolution: A natural approach to operator selection, School of Chemical Eng. and Industrial Chemistry, Univ. of New South Wales, Sydney, Australia

Potts, J.C., Giddens, T.D., & Yadav, S.B. (1994). The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artificial Selection. *IEEE Transactions on Systems, Man, and Cybernetics*, 24 (1), 73-86.

Schaffer, J.D., Caruana, R.A., Eshelman, L.J., & Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp.51-60). San Mateo, CA: Morgan Kaufmann.

Schlierkamp-Voosen, D., & Mühlenbein, H. (1994). Strategy adaptation by competing subpopulations. In Y. Davidor, H-P. Schwefel, and R. Manner (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature.,* vol 866 of Lecture Notes in Computer Science (pp.199-208). Berlin: Springer-Verlag.

Seredynski, F., (1994a). Dynamic Mapping and Load Balancing with Parallel Genetic Algorithms. *Proceedings of the IEEE World Congress on Computational Intelligence, First IEEE Conference on Evolutionary Computation*, vol II, (pp. 834-839).

Seredynski, F., (1994b). Loosely Coupled Distributed Genetic Algorithms. In Y.Davidor, H-P. Schwefel and R. Manner (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature* (pp.514-523), vol 866 of Lecture Notes in Computer Science. Berlin: Springer-Verlag.

Shonkwiler, R. (1993). Parallel Genetic Algorithms. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp.199-205). San Mateo, CA: Morgan Kaufmann.

Spears W.M. (1993). Crossover or mutation. In L.D. Whitley (Ed.), *Foundations of Genetic Algorithms 2*. San Mateo, CA: Morgan Kaufmann

Spears W. M., & De Jong K.A. (1991). . An analysis of multi-point crossover. In G. Rawlins (Ed.), *Foundations of Genetic Algorithms* (pp.301-315). San Mateo, CA: Morgan Kaufmann

Srinivas, M., & Patnaik, L.M. (1994). Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 24 (4).

Starkweather, T., Whitley, D., & Mathias, K. (1990). Optimization using distributed genetic algorithms. In H.-P.Schwefel and R.Manner (Eds.), *Proceedings of the First Workshop on Parallel Problem Solving from Nature*, vol. 496 of Lecture Notes in Computer Science (pp. 176-185). Berlin : Springer-Verlag.

Syswerda, G. (1989). Uniform crossover in gentic algorithms. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 2-9). San Mateo, CA: Morgan Kaufmann.

Tamaki, H., Kita, H., Shimizu, N., Maekawa, K., Nishikawa, Y. (1994). A comparison study of genetic codings for the travelling salesman problem. *Proceedings of the IEEE World Congress on Computational Intelligence, First IEEE Conference on Evolutionary Computation*, vol I, (pp. 1-6).

Tamaki, H., & Nishikawa, Y. (1992). A Paralleled Genetic Algorithm based on a Neighborhood Model and its Application to the Jobshop Scheduling. In R.Manner and B.Manderick (Eds.), *Proceedings of the Second Workshop on Parallel Problem Solving from Nature* (pp.573-582). Amsterdam: Elsevier Science Publishers.

Tanese, R. (1987). Parallel Genetic Algorithms for a hypercube. In J.J Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and their Applications* (pp.177-183). Cambridge, MA: Lawrence Erlbaum Associates

Tanese, R. (1989). Distributed Genetic Algorithms. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp.434-439). San Mateo, CA: Morgan Kaufmann.

Tang, A. Y.-C., & Leung, K.-S. (1994). A modified edge recombination operator for the travelling salesman problem. In Y. Davidor, H-P. Schwefel, and R. Manner (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature* (pp.180-188). Berlin: Springer-Verlag.

Thierens, D., & Goldberg, D.E. (1993). Mixing in genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp.38-45). San Mateo, CA: Morgan Kaufmann.

Törn, A., & Zilinskas, A. (1989). Global Optimization. *Lecture Notes in Computer Science*, vol 350, Berlin: Springer-Verlag.

Valenzuela, C.L., Jones, A.J. (1993). Evolutionary divide and conquer (I): A novel genetic approach to the TSP. *Evolutionary Computation* 1(4), 313-333

Voigt, H-M., Santibanez-Koref, I., & Born, J. (1992). Hierarchically Structured Distributed Genetic Algorithms. In R.Manner and B.Manderick (Eds.), *Proceedings of the Second Conference on Parallel Problem Solving from Nature* (pp.155-164). Amsterdam: Elsevier Science Publishers.

Whitley, D. (1993). A Genetic Algorithm Tutorial, Technical Report CS-93-1-3, Dept. of Computer Science, Colorado State University, March 10, 1993

Whitley, D., Starkweather, T., & Fuquay D'Ann. (1989) Scheduling problems and traveling salesmen: The genetic edge recombination operator. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp.133-140). San Mateo, CA: Morgan Kaufmann