Voronoi-Initializated Island Models for Solving Real-Coded Deceptive Problems

Santiago Muelas, José M. Peña, Víctor Robles and Antonio LaTorre
Computer Architecture Department
Facultad de Informática
Universidad Politécnica de Madrid, Spain
{smuelas, jmpena, vrobles, atorre }@fi.upm.es

ABSTRACT

Deceptive problems have always been considered difficult for Genetic Algorithms. To cope with this characteristic, the literature has proposed the use of Parallel Genetic Algorithms (PGAs), particularly multi-population island-based models. Although the existence of multiple populations encourages population diversity, these problems are still difficult to solve. This paper introduces a new initialization mechanism for each of the populations of the islands based on Voronoi cells. In order to analyze the results, a series of different experiments using several real-value deceptive problems and a set of representative parameters (migration ratio, migration frequency and connectivity) have been chosen. The results obtained suggest that the Voronoi initialization method improves considerably the performance obtained with a traditionally uniform random initialization.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; I.2.m.c [**Artificial Intelligence**]: Miscellaneous: Evolutionary Computing and Genetic Algorithms

General Terms

Algorithms, Experimentation, Performance

Keywords

Parallel Genetic Algorithms, Voronoi Initialization, Migration Topology, Deceptive Problems

1. INTRODUCTION

There are different factors that explain why an optimization problem becomes difficult to solve. Among these factors, deception has been considered one of the most relevant. Some authors even argued that the only problems which pose challenging optimization tasks are those involving some degree of deception [26].

Goldberg [8] presented the concept of deception as an explanation for the behavior of genetic algorithms in certain difficult problems. The effect of deception in a problem misleads the search algorithm away from the global optimum.

With more detail, Whitley [26] explained deception in binary-coded problems using the schemata concept. A deceptive problem can be characterized as a function, which is defined in such a way that the low-order schemata leads the search away from the global optimum toward a suboptimal solution, known as deceptive attractor.

The study of deception in non-binary problems is considered by Goldberg using the perspective of different coding alphabets [9]. Small alphabets, such as binary-coding, can be used to construct any larger alphabet, but they also allow low-order building blocks to be managed. Dealing with low-order building blocks is not always beneficial, and deceptiveness is one of the cases where this situation takes place. Alphabet size has also a significant influence in the provision of initial building blocks. The larger the alphabet is, the bigger the population should be in order to guarantee the appropriate coverage of the solution space.

This discussion is right in the context of discrete alphabets of different sizes, but real-coded genetic algorithms have other considerations. These algorithms often adopt mutation (or even crossover) operators that slightly perturb the obtained value using a small variation [12]. These operators provide self-adaptive features to real-coded genetic algorithms [4], changing a probability distribution based on the distance between parents. This effect compensates the limited provision of initial building blocks.

Although real-value problems could show the same type of misleading behavior than binary-coded problems, when low-order and high-order schemata are contrasted, there also exists the possibility of an internal deceptive trap. A first-order separable function could be deceptive if the basin of attraction of any suboptimal solution is relatively bigger than the basin of attraction of the global optimum [23], characteristic which is featured by some authors as the gradients leading in the wrong direction [27]. This particular problem does not arise in binary-coded genetic algorithms, but it could potentially happen in any k-order alphabet (with $k > 2$). Non-epistatic deception in a high-order alphabet appears when specific operators perform restricted perturbations which are not able to escape from the basin of attraction of the deceptive value.

One alternative to deal with deceptive problems are the

Parallel Genetic Algorithms, particularly the island-based models. Besides the increased diversity of multiple populations, these problems are still difficult to solve. One important aspect of island based PGAs is the initialization of the starting population. This issue is important to provide a good supply of initial individuals to start up the stochastic search. When there are more than one population, the influence of the initial individuals for each of the sub-populations should be also considered. Though unanimously considered as a key issue in all types of evolutionary algorithms, population initialization has not been analyzed in detail, in particular for island models. In most of the cases, these initial populations are created by means of an uniform distribution. This paper proposes the use of a topological tool (Voronoi diagrams) to restrict the initial search space of the different nodes in PGAs.

The rest of the paper is organized as follows: Section 2 presents an overview of parallel genetic algorithms, initialization techniques and migration strategies. This section also surveys some representative approaches specially designed for dealing with deceptive problems. Section 3 details the proposed technique and the rationale behind it. In section 4 the experimental scenario is described in detail. Section 5 presents and comments the results obtained and lists the most relevant facts extracted from this analysis. Finally, section 6 contains the concluding remarks obtained from this study.

2. RELATED WORK

In this section, a selection of the most significant work from the literature is presented. This section begins with a review of some of the existing techniques for dealing with deceptive problems using specialized genetic algorithms, to follow with some concepts about the proposed alternative.

2.1 Dealing With Deceptive Problems

Several algorithms have been proposed to specifically manage deceptive problems. The Structured GA [6] uses a hierarchical encoding and a gene expression mechanism in its overspecified chromosomal representation. In Structured GAs, two schemas are actually used: physical schema and expressed schema. The first is used by the genetic operators and the second for fitness calculation.

A similar approach is used by [17], where redundant representation is also used. This algorithm divides the population into subpopulations only within the stage of selection and reproduction in each generation. The mechanism develops the behavior of finding deceptive hyperplanes and escaping from them using large genetic transitions to the complements to them in the population.

Species conserving genetic algorithm [16] is based on the concept of dividing the population into several species according to their similarity. Each of these species is built around a dominating individual called the species seed. Species seeds found in the current generation are saved (conserved) by moving them into the next generation. This technique is successfully applied to multimodal-deceptive problems.

Holland defined the Cohort GAs [13], to reduce "hitchhiking" – premature convergence of currently low-significance loci located near loci at which good building blocks are found early in the search process. Hitchhiking effect is most severe when the maximum reproduction rate is relatively high, for example, if the expected number of offspring of the best individual in the population is on the order of two or more. Hitchhiking is reduced when fitness is scaled so that the expected number of copies of the fittest individual produced in the next generation is 1.2 or fewer.

Multipopulation parallel genetic algorithms have a beneficial behavior when dealing with deceptive problems. As observed in [26], the island model genetic algorithm performed better than a single population with deceptive type problems. Furthermore, the following trend was observed: the more distributed the population (more islands) is, the better the results on deceptive problems become. Pei [18] compared the behavior of Cohort GAs with simple and parallel island-based distributed GAs. In his tests the island model GAs obtained better results than the Cohort GAs.

2.2 Parallel Genetic Algorithms

When tackling big optimization problems through evolutionary algorithms, the first alternative that arises is the parallelization of the algorithms. This idea has been the subject of many pages in the literature for the particular case of genetic algorithms [5].

There are different ways of carrying out this parallelization task:

- **Master-slave models** (also called centralized models) are a more intuitive line of action which consists in the parallel execution of the evaluation stage of the objective functions. A central node executes the evolutionary algorithm, per se, including the necessary operators, but subsequently distributes the work of each evaluation among the slave nodes. This model can be appropriate for fitness functions with a high computational cost.

- **Island models** (also called coarse-grain or multipopulation) represent a different algorithmic structure. In this case, each node with its own population executes a different genetic algorithm. The independent evolving populations interact with each other through migration strategies that enable an information exchange among populations. These strategies are subject to certain frequencies, migration rates and migration topologies.

- **Cellular models** (also called fine-grain) can be considered as an extreme variant of the previous ones: node populations have only one individual. In these cases, in order to create new individuals, recombination needs to select (from the nearest nodes) the best individual to perform mating with.

2.3 Initialization

The initialization of genetic algorithms is a subject hardly addressed in the literature [14]. Nevertheless, every expert in the field agrees that a bad initialization can make evolution difficult or even stop it from reaching the optimum.

In many cases, the initialization process is based on the application field if an approximate solution to the problem is known. Otherwise, if the solutions can be built through certain techniques, these could be a good starting point to reach the optimum. However, this strategy has some drawbacks: (i) it completely depends on the application field and (ii) it may involve biasing the search process towards certain kinds of solutions (possibly other than the optimal ones).

There is one alternative to the ad-hoc initialization, the use of general schemas applicable to several scenarios [11]. In some cases, other genetic algorithms are used (with different fitness functions) in order to initialize the population [7].

There are also other alternatives for binary-coded individuals, as described in [14] where certain statistical properties (besides uniformity) are given among the starting population (such as uniform coverage and homogeneous blocks).

Island models are especially sensitive to initialization, not only for the reasons stated above, but also because of the possible mutual dependence between the different islands populations (in the island models). There is very little literature on this topic and, therefore, this is one of the aspects to be studied in depth.

Voronoi-based initialization has also been used with other evolutionary algorithms. Saska applied this initialization for path planning and obstacle avoidance optimization with particle swarm optimization algorithms [20]. In his study, Saska observed that with complicated environments, the results obtained with the Voronoi configurations found much better solutions than the traditional random ones.

2.4 Migration Strategy

An important aspect of the performance of the island models is the migration strategy. This is configured through different parameters [5]:

1. Migration frequency: How often (in generations) is information sent?
2. Migration rate: How many individuals are migrated each time?
3. Information selection: What information is selected to migrate?
4. Acceptance policy: How the incoming information and the local algorithm state are combined?
5. Migration topology: Which island sends information to which other?

The information exchanged in the case of island models is a selection of the individuals, sometimes referred to as immigrants. These immigrants are selected from the original population and transmitted to the remote island. This island merges the incoming individuals with the local population.

Close scrutiny of migration parameters [19] has proved that, even though subpopulations with a smaller size have the risk of being trapped in a local optimum, an appropriate migration strategy can stop a suboptimum solution from dominating all populations. This appropriate strategy must be adjusted between the limits of a low interaction (which would practically imply the execution of N independent algorithms) and an excessive interaction (that would lead to the predominance of only one solution). A correct configuration can help to obtain better results with fewer evaluations, but configuring these optimal parameters is not a simple issue [25].

3. CONTRIBUTION

In this paper, the proposed initialization mechanism uses a Voronoi tessellation [1] to define a partition set of the solution space where each island starts its own exploration.

Given a set of reference points $P = p_1, p_2, \ldots, p_n$, the Voronoi region of each point is determined by the following equation:

$$V(p_i) = \{p | p \in R^n, d(p, p_i) < d(p, p_j), \forall j \neq i\} \qquad (1)$$

where d expresses the Euclidean distance between two points.

Therefore, each reference point will determine the initial population of each island. The generation of the reference points is carried out by the master node which sends each reference point to the correspondent island. In order to generate the reference points, a simple iterative procedure that tries to maximize the distance between them is performed. This process is described in the following pseudocode:

```
$GenerateRefPoints(num\_islands)$
$refs \leftarrow GenerateRandomPoints(num\_islands)$
for $i = 1$ to $N$ do
    $refs' \leftarrow GenerateRandomPoints(num\_islands)$
    if $SumMinDistances(refs') > SumDistances(refs)$
    then
        $refs \leftarrow refs'$
    end if
end for

$SumMinDistances(r)$
$sum\_min\_dists \leftarrow 0$
for $i = 1$ to $size(r) - 1$ do
    $min\_dist \leftarrow distance(r_i, r_{i+1})$
    for $j = i + 2$ to $size(r)$ do
        $min\_dist' \leftarrow distance(r_i, r_j)$
        if $min\_dist' < min\_dist$ then
            $min\_dist \leftarrow min\_dist'$
        end if
    end for
    $sum\_min\_dists \leftarrow sum\_min\_dists + min\_dist$
end for
return $sum\_min\_dists$
```

Following equation 1, the initial population of island i will be composed of the points in the solution space which are closer to the ith-reference point than to any other reference point. To create this population, each island samples a uniform distribution to generate tentative individuals. If an individual is outside the Voronoi cell (is closer to another reference point), it is not considered and a new one is generated. An example of a Voronoi initialization obtained for a simple two-dimensional problem with six islands is depicted on Figure 1.

Our approach carries out a systematic initialization procedure following two criteria: (i) coverage of the whole solution space, (ii) no overlap of the solution space explored in each island at the beginning.

The benefits derived from this alternative initialization should be present, more clearly, on early generations. Once the migrations are performed, the population on different islands will be merged and the effect of the initialization will start fading. Before reaching this initial migration phase,

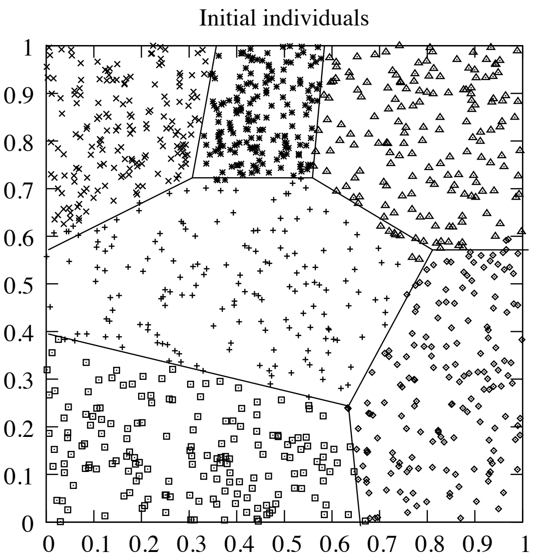

Figure 1: Voronoi island initialization for a two-dimensional problem

the initialization process must provide a good supply of interesting individuals to exchange.

4. EXPERIMENTAL SCENARIO

4.1 Problems

This section introduces the problem set used to evaluate the performance of our contribution. Three deceptive problems with different characteristics have been selected.

4.1.1 *Modified Shekel Function*

This function is based on the one defined in [3]. Its function is given by:

$$shekel - mod(x_0, \ldots, x_{n-1}) = \\ -\sum_{i=0}^{29} \left(\frac{1}{\sum_{j=0}^{4}(x_j - a_{ij})^2 + c_i} \right) \\ 0 \leq x_i \leq 5, i = 1, 2, \ldots, n$$

where a and c are constant matrices commonly used with the Shekel problem [3]. The objective is to minimize this function which has several deceptive optima and a global optimum at (8.0249, 9.1517, 5.1139, 7.62908, 4.5640). This function is multimodal and non separable and has a considerably small basin of attraction of the global optimum.

4.1.2 *Deceptive Function #1*

This function is defined in [10] and is composed of two-dimensional deceptive functions:

$$deceptive1(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n/2} f_{dec}(x_{2i}, x_{2i+1}), \qquad (2)$$

where $x_i \in (0,1)$ and $f_{deceptive}$ is defined as:

$$f_{dec}(x,y) = \begin{cases} 0.8 - \sqrt{\frac{x^2+y^2}{2}} & \text{if } \sqrt{\frac{x^2+y^2}{2}} \leq 0.8 \\ 5 - 5\sqrt{\frac{x^2+y^2}{2}} & \text{otherwise} \end{cases} \qquad (3)$$

The objective is to maximize this function which has $2^{(n/2)-1}$ local optima and only one global optimum located at (1,...,1). This function has an epistasis degree of two and therefore, it cannot be efficiently solved by mutation only. Here it is necessary to learn the linkage between pairs of variables that contribute to the fitness. The number of dimensions, n, has been set to 12.

4.1.3 *Deceptive Function #2*

This deceptive function has been created for this experiment as an example of a linear separable deceptive problem with a small basin of attraction of the global optimum. It is defined as follows:

$$deceptive2(x_0, \ldots, x_{n-a}) = \sum_{i=0}^{n-1} 15 + sin(x_i)(x_i^2 + 2.2^{1-x_i}) \qquad (4)$$

where $x_i \in (-3.5, 4.2)$

The objective is to maximize this function which has the global optimum located at $(-3.5, \ldots, -3.5)$. The number of dimensions has been set to 50.

4.2 Algorithm

With the purpose of analyzing the influence of the proposed strategies on the PGA performance, the following parameters have been chosen:

1. Number of islands. The number of islands has been kept fixed to 64.

2. Topologies and Connectivity. The selected topologies are: Static Hypercube, Ring2 and Dynamic Nearest Neighbor. Double ring and Hypercube are two of the most commonly topologies used on coarse-grained parallelization. Double ring has a connectivity degree of 2 and hypercube of $log_2(numberofislands)$, six in this case. Nearest Neighbor is a dynamic topology. This topology determines the neighborhood of each island according to their minimum Euclidean distance before each migration phase. In order to compute the distances, each island selects a representative (usually the medoid) and sends it to the rest of the islands. For the purpose of comparing topologies with the same connectivities, the nearest neighbor topology has been tested with a connectivity degree of 2 and 6 degrees.

3. Initialization. Voronoi initialization has been tested against a traditional random initialization following a uniform distribution for all the dimensions of the solution space. This initialization is called random for the rest of the paper.

4. Population Size. 1024 individuals of global population (16 per island).

5. Migration rate and frequency. Both migration frequency and migration rate can considerably influence the behavior of parallel evolutionary algorithms ([2, 22, 15]). For this reason, several values have been considered for the migration frequency (migrate every 5,

10, 20 and 40 generations) and for the migration rate (migrate 10, 20 and 40 percent of the individuals).

6. Best-worst migration policy: Islands send a copy of the best individuals and replace the worst individuals of the population.
7. Selection: Tournament Based Selection.
8. Crossover Operator: BLX-α with α=0.5 and 100% of probablity.
9. Mutation Operator: Gaussian with 1% of probability.
10. Full elitism: The set of individuals of the next generation is composed upon the best ones among both parents and children of the current population.

With all these possibilities, 96 tests combinations per problem have been generated and tested against each other as explained in the following section.

4.3 Procedure

For each of the proposed problems, the following experimental procedure is performed:

- All the combinations of the input parameters values are considered.
- For each combination, 50 independent executions are run for 400 generations.
- In order to compare the effects of the island model algorithm, a sequential algorithm configuration with the same global population is also executed.
- The fitness of each configuration is pair-wise compared using a Wilcoxon non-parametric t-test against the others.
- A Ranking Analysis is performed. The goal of this global analysis is to rank the performance of all the 96 combinations per problem. In this analysis, if one configuration is significantly better than other (p-value < 0.01), the winning configuration is granted with +1 "wins" and the losing configuration penalized with -1 "wins". As all the configurations are compared against each other, they are ranked (depending on how many other configuration are better/worse).
- Finally, for each combination of initialization and topology values, the best configuration according to the number of wins is selected. The best Voronoi configurations are compared against the best random ones.

5. ANALYSIS OF THE RESULTS

Tables 1, 2 and 3 display the number of wins obtained for each combination while tables 4, 5, and 6 analyze the performance of the best configurations for each combination of initialization and topology values. From these results some interesting facts should be mentioned:

- All the combinations with the greater number of wins have in common the Voronoi-based initialization method. The effect of this alternative initialization provides several islands with better individuals on early iterations. At the end, the populations are merged and the evolution is roughly the same as with the traditional random initialization method. The evolution of the fitness of the algorithms on later generations (Figure 2) is almost the same for both methods but the influence of the first generations clearly increases the performance of the Voronoi configurations.

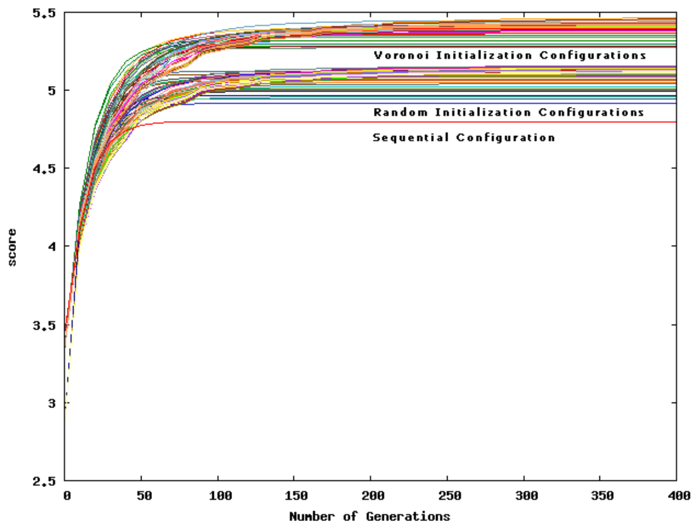

Figure 2: Best Fitness evolution for Deceptive #1 problem

- It can be seen from tables 4, 5, and 6 than in 52 of the 60 possible comparisons, the best Voronoi combinations are statistically better ($p - value$<0.01) than the random initialization counterparts.
- Sequential algorithm performs poorly on these deceptive problems even though it uses the same number of individuals and evaluations.
- With the harder non separable problems (Shekel and Deceptive #1), all Voronoi initialization configurations have obtained significantly better results than the random equivalent ones. With an easier problem as Deceptive #2, the effect of the proposed initialization is not so determinant but has also improved the best random configuration result.
- In [21], the authors recommended scarce linkage (low connectivity) for deceptive problems. This seems to contradict our results where both degrees (two and six) have obtained good results. It must be taken into account that, Sekaj's results were obtained with a considerably reduced number of islands (only 9) with two and five connectivity topologies. Cantú-Paz's study [5] emphasises the influence of the diameter of the topology in addition to the connectivity degree. It can be seen that the diameter of the lowest connectivity degree topologies with 9 islands in Sekaj's work, is comparable with the highest connectivity degree topologies presented here with 64 islands.

6. CONCLUSIONS

This paper presents a new general initialization method for PGAs (island models) aiming to solve deceptive problems. The initialization proposed is based on Voronoi cells which isolate the initial search space of each island and offers the possibility, at least in the earliest generations, to emphasize the search in the starting regions. Several parameter values and three real-coded deceptive problems have been considered in order to analyze the influence of the proposed methods under different conditions. To compare the quality of the results, a rank-based analysis based on the statistical significance of the results has been carried out. The obtained results show that the best results for all the problems are achieved with Voronoi-based configurations. It should be mentioned that these results were obtained with a moderated mutation probability. All the parallel configurations have also outperformed the results from an equivalent sequential algorithm.

7. ACKNOWLEDGMENTS

The authors thankfully acknowledges the computer resources, technical expertise and assistance provided by the *Centro de Supercomputación y Visualización de Madrid (CeSViMa)* and the Spanish Supercomputing Network.

This work was supported by the Madrid Regional Education Ministry and the European Social Fund and financed by the Spanish Ministry of Science TIN2007- 67148.

8. REFERENCES

[1] F. Aurenhammer. Voronoi diagrams, a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23:345–405, 1991.

[2] T. C. Belding. The distributed genetic algorithm revisited. In *Proceedings of the Sixth ICGA*, pages 114–121. Morgan Kaufmann, 1995.

[3] H. Bersini, M. Dorigo, S. Langerman, G. Seront, and L. Gambardella. Results of the first international contest on evolutionary optimisation (1st iceo). In ICEC, pages 611–615, 1996.

[4] H. Beyer and K. Deb. On self-adaptive features in real-parameter evolutionary algorithms. *IEEE TEC*, 5(3):250–270, 2001.

[5] E. Cantú-Paz. *Efficient and accurate parallel genetic algorithms.* Kluwer Academic Publishers, 2001.

[6] D. Dasgupta. Handling deceptive problems using a different genetic search. In *ICEC*, pages 807–811, 1994.

[7] H. de Garis. Genetic programming: artificial nervous systems artificial embryos and embryological electronics. In *PPSN - Proceedings of 1st Workshop*, volume 496, pages 117–123. Springer-Verlag, Berlin, Germany, 1-3 1991.

[8] D. Goldberg. *Genetic Algorithms and Simulated Annealing*, chapter Simple Genetic Algorithms and the Minimal Deceptive Problem, pages 74–88. Morgan Kaufmann, 1987.

[9] D. E. Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, 5:139–167, 1991.

[10] D. E. Goldberg, M. Pelikan, and S. Tsutsui. Getting the best of both worlds: Discrete and continuous genetic and evolutionary algorithms in concert. *Information Sciences*, 156(3-4):147–171, 2003.

[11] J. Grefensette. *Genetic Algorithms and Simulated Annealing*, chapter Incorporating problem specific knowledge into genetic algorithms, pages 42–46. Morgan Kauffmann Publishers, 1987.

[12] F. Herrera, M. Lozano, and A. Sánchez. A taxonomy for the crossover operator for real-coded genetic algorithms. an experimental study. *International Journal of Intelligent Systems*, 18(3):309–338, 2003.

[13] J. Holland. Cohort GAs and hyperplane-defined functions. *Evolutionary Computation*, 8(4):372–391, 2000.

[14] L. Kallel and M. Schoenauer. Alternative random initialization in genetic algorithms. In *Proceedings of the 7th ICGA*, pages 268–275, 1997.

[15] J. R. Koza and D. Andre. Parallel genetic programming on a network of transputers. Technical report, 1995.

[16] J.-P. Li, M. E. Balazs, G. T. Parks, and J. P. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evol. Comput.*, 10(3):207–234, 2002.

[17] K. Ohkura and K. Ueda. An extended framework for overcoming premature convergence. In *Proceedings of the 7th ICGA*, pages 260–267, 1997.

[18] H. Pei and E. Goodman. A comparison of cohort genetic algorithms with canonical serial and island-model distributed GA's. In *GECCO'01*, pages 501–510, 2001.

[19] C. Petty and M. Leuze. A theoretical investigation of a parallel genetic algorithm. In *Proc. 3rd ICGA*, pages 398–405, 1989.

[20] M. Saska, M. Hess, and K. Schilling. Voronoi strains: a spline path planning algorithm for complex environments. In *Proceedings of the 25th IASTED International Multi-Conference*, pages 498–502, 2007. ACTA Press.

[21] I. Sekaj. Robust parallel genetic algorithms with re-initialisation. In *PPSN*, pages 411–419, 2004.

[22] Z. Skolicki and K. De Jong. The influence of migration sizes and intervals on island models. In *GECCO '05*, pages 1295–1302. ACM Press, 2005.

[23] Z. Skolicki and K. A. D. Jong. Improving evolutionary algorithms with multi-representation island models. In *PPSN*, pages 420–429, 2004.

[24] P. A. Whigham and G. Dick. A voronoi-based distributed genetic algorithm. In *Fifteenth Annual Colloquium of the Spatial Information Research Centre*, 2003.

[25] D. Whitley, S. Rana, and R. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–47, 1999.

[26] L. D. Whitley. Fundamental principles of deception in genetic search. In *Foundations of genetic algorithms*, pages 221–241. Morgan Kaufmann, 1991.

[27] D. Wyatt and H. Lipson. Finding building blocks through eigenstructure adaptation. In *GECCO'03*, page 207, 2003.

Table 1: Shekel Ranked Wins

initialization	migration period	migration rate	topology	top.degree	wins	avg	std. dev.
random	-	-	sequential	-	-96	-3.0748	0.00264
random	5	40	hypercube	6	-87	-5.0057	0.00543
random	5	20	hypercube	6	-84	-5.3326	0.00576
random	5	40	ring2	2	-72	-5.5482	0.00585
random	10	10	hypercube	6	-58	-5.4441	0.00566
random	5	20	nearestneighbor	6	-55	-5.4767	0.00563
random	10	20	hypercube	6	-53	-5.4702	0.00564
random	5	10	hypercube	6	-50	-6.0499	0.00616
random	5	20	ring2	2	-50	-6.0623	0.00614
random	40	40	nearestneighbor	2	-50	-6.9415	0.00622
...	...	...	...	...	...	...	...
voronoi	40	20	hypercube	6	56	-9.9827	0.00334
voronoi	40	40	hypercube	6	56	-10.1926	0.00244
voronoi	5	10	nearestneighbor	2	58	-10.0882	0.00292
voronoi	10	10	ring2	2	58	-10.0882	0.00292
voronoi	20	10	nearestneighbor	2	58	-10.0882	0.00241
voronoi	20	10	nearestneighbor	6	58	-10.0882	0.00292
voronoi	20	20	nearestneighbor	6	58	-10.1970	0.00241
voronoi	20	20	ring2	2	58	-10.0882	0.00292
voronoi	40	10	nearestneighbor	6	58	-10.0882	0.00292
voronoi	40	20	nearestneighbor	6	58	-10.1970	0.00241

Table 2: Deceptive #1 Ranked Wins

initialization	migration period	migration rate	topology	top.degree	wins	avg	std. dev.
random	-	-	sequential	-	-96	4.8000	0.0000
random	5	10	hypercube	6	-80	4.9160	0.1218
random	40	20	ring2	2	-73	5.0976	0.1154
random	40	40	ring2	2	-70	5.1182	0.1427
random	5	20	hypercube	6	-69	4.9440	0.1280
random	20	40	ring2	2	-66	5.0949	0.1351
random	40	40	hypercube	6	-63	5.1113	0.1287
random	10	10	hypercube	6	-62	4.9560	0.1473
random	10	40	ring2	2	-62	5.0719	0.1749
random	20	40	hypercube	6	-62	5.0519	0.1388
random	10	20	ring2	2	-61	5.0400	0.1277
. . .	. . .	. . .	. . .	. . .	. . .	. . .	
voronoi	5	40	nearestneighbor	6	60	5.3200	0.1456
voronoi	20	10	nearestneighbor	2	60	5.4391	0.1452
voronoi	40	10	nearestneighbor	2	60	5.4622	0.1103
voronoi	40	20	hypercube	6	63	5.4276	0.1276
voronoi	20	10	hypercube	6	66	5.3840	0.1608
voronoi	10	10	nearestneighbor	2	67	5.4079	0.1337
voronoi	20	20	hypercube	6	67	5.3880	0.1479
voronoi	10	20	nearestneighbor	2	71	5.4478	0.1541
voronoi	5	20	nearestneighbor	2	76	5.4000	0.1340
voronoi	5	10	nearestneighbor	2	80	5.3960	0.1244

Table 3: Deceptive #2 Ranked Wins

initialization	migration period	migration rate	topology	top.degree	wins	avg	std. dev.
random	-	-	sequential	-	-96	1004.4187	6.7504
random	40	10	nearestneighbor	2	-92	1277.4728	11.2885
random	40	10	ring2	2	-91	1280.5346	9.3899
voronoi	40	10	nearestneighbor	2	-89	1282.3516	11.4165
random	40	20	nearestneighbor	2	-86	1284.1447	10.1349
random	40	40	nearestneighbor	2	-83	1285.6679	9.0220
voronoi	40	10	ring2	2	-82	1287.0418	10.6334
voronoi	40	20	nearestneighbor	2	-81	1287.5196	9.8630
voronoi	40	40	nearestneighbor	2	-80	1288.1835	10.3935
random	40	20	ring2	2	-79	1289.9871	11.2163
...	...	...	...	...	...	...	...
voronoi	5	10	hypercube	6	77	1454.1116	13.3271
voronoi	5	10	nearestneighbor	6	79	1458.5036	10.0540
random	5	20	nearestneighbor	6	83	1464.9283	11.6133
random	5	40	nearestneighbor	6	84	1469.0471	9.5926
voronoi	5	40	nearestneighbor	6	87	1473.0825	9.4446
voronoi	5	20	nearestneighbor	6	90	1474.9804	12.5983
random	5	20	hypercube	6	90	1476.0321	8.8718
voronoi	5	20	hypercube	6	91	1478.4239	9.2628
random	5	40	hypercube	6	92	1479.4951	10.2864
voronoi	5	40	hypercube	6	95	1483.5346	11.2146

Table 4: Shekel best configurations comparison

initialization	topology	best avg fit. ± dev	best freq.	best rate	rand-ring	rand-hyp	rand-nn2	rand-nn6	sequential
voronoi	ring	-10.0882 ± 0.0029	20	20	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
voronoi	hypercube	-10.1926 ± 0.0024	40	40	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
voronoi	nearestneighbor2	-10.1970 ± 0.0024	20	10	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
voronoi	nearestneighbor6	-10.1970 ± 0.0024	20	20	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
random	ring	-7.2413 ± 0.0062	40	10	-	-	-	-	-
random	hypercube	-7.4761 ± 0.0063	40	40	-	-	-	-	-
random	nearestneighbor2	-7.1165 ± 0.0418	20	10	-	-	-	-	-
random	nearestneighbor6	-7.0822 ± 0.0063	20	10	-	-	-	-	-
random	sequential	-3.0748 ± 0.0026	-	-	-	-	-	-	-

$\surd$ represents when 50 executions of a Voronoi combination are statistically better than 50 executions of the random confronted combination.

Table 5: Deceptive #1 best configurations comparison

initialization	topology	best avg fit. ± dev	best freq.	best rate	rand-ring	rand-hyp	rand-nn2	rand-nn6	sequential
voronoi	ring	5.4118 ± 0.1238	20	10	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
voronoi	hypercube	5.3880 ± 0.1479	20	20	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
voronoi	nearestneighbor2	5.3960 ± 0.1244	5	10	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
voronoi	nearestneighbor6	5.3200 ± 0.1456	5	40	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
random	ring	5.0200 ± 0.1293	5	20	-	-	-	-	-
random	hypercube	5.0720 ± 0.1443	20	10	-	-	-	-	-
random	nearestneighbor2	5.1160 ± 0.1569	5	10	-	-	-	-	-
random	nearestneighbor6	5.0840 ± 0.1218	5	40	-	-	-	-	-
random	sequential	4.8000 ± 0.0000	-	-	-	-	-	-	-

$\surd$ represents when 50 executions of a Voronoi combination are statistically better than 50 executions of the random confronted combination.

Table 6: Deceptive #2 best configurations comparison

initialization	topology	best avg fit. ± dev	best freq.	best rate	rand-ring	rand-hyp	rand-nn2	rand-nn6	sequential
voronoi	ring	1410.1953 ± 11.4697	5	40	$\surd$	$\times$	$\surd$	$\times$	$\surd$
voronoi	hypercube	1483.5346 ± 11.2146	5	40	$\surd$	$\times$	$\surd$	$\surd$	$\surd$
voronoi	nearestneighbor2	1398.5729 ± 10.7648	5	40	$\times$	$\times$	$\surd$	$\times$	$\surd$
voronoi	nearestneighbor6	1474.9804 ± 12.5983	5	20	$\surd$	$\times$	$\surd$	$\surd$	$\surd$
random	ring	1401.9627 ± 9.9949	5	40	-	-	-	-	-
random	hypercube	1479.4951 ± 10.2864	5	40	-	-	-	-	-
random	nearestneighbor2	1398.5729 ± 10.7648	5	10	-	-	-	-	-
random	nearestneighbor6	1469.0471 ± 9.5926	5	40	-	-	-	-	-
random	sequential	139.4032 ± 6.4659	-	-	-	-	-	-	-

$\surd$ represents when 50 executions of a Voronoi combination are statistically better than 50 executions of the random confronted combination.
$\times$ represents that the 50 executions of a Voronoi combination are not statistically better than the 50 executions of the random confronted combination