# EXPERIMENTAL STUDY OF LOCAL SELECTION ON EVOLUTIONARY COURSE TIMETABLING

PANAGIOTIS ADAMIDIS *

VASSILIS KOSTOGLOU **

* Alexander Technological Education Institute of Thessaloniki
Department of Informatics
P.O. Box 141, 574 00, Thessaloniki, Greece.
Tel: +30.2310.791285, Fax: +30.2310.791290
E-mail: adamidis@it.teithe.gr

** Alexander Technological Education Institute of Thessaloniki
Department of Informatics
P.O. Box 141, 574 00, Thessaloniki, Greece.
Tel: +30.2310.791294, Fax: +30.2310.791294
E-mail: vkostogl@it.teithe.gr

# EXPERIMENTAL STUDY OF LOCAL SELECTION ON EVOLUTIONARY COURSE TIMETABLING

*ABSTRACT*

*Course timetabling is a multi-dimensional NP-Complete problem encountered virtually in every educational institute throughout the world. Evolutionary Algorithms (EAs) have been applied to the course-timetabling problem since early 90s. Solving this problem with EAs, selection traditionally operates on the entire population. This paper studies the effects of local selection EAs on the course-timetabling problem. Here the decision for parent choice is performed locally only. Local selection algorithms operate in parallel on small overlapping neighborhoods. We tested a lot of different configurations in order to enhance our understanding of the effects of neighborhood size and shape.*

*Keywords: Local search, evolutionary algorithms, course timetabling*

# 1. INTRODUCTION

The problem of creating the proper course timetable can be viewed as a multi-dimensional assignment problem in which students, teachers (or faculty members) are assigned to courses, course sections or classes; events (individual meetings between students and teachers) are assigned to classrooms and times. Problem definition and terminology varies from one institute to another (Carter and Laporte , 1998). The timetable must satisfy a number of constraints that concern the capacity and suitability of the rooms, the availability and suitability of the instructors, the relation with other courses, etc.

There have been several attempts to solve this type of problem with Evolutionary Algorithms (EAs) (Adamidis and Arapakis (1999); Burke, Newall, and Weare (1996); Paechter et al. (1998); Paechter, Rankin and Cumming (1994); Ross, Corne and Fang (1994)).

Selection in EAs usually operates globally on the entire population. In nature we rarely find global mating pools. This leads to the introduction of a continuous population structure that is a population of uniformly distributed individuals over a geographic region, which might be linear planar or spatial. In this context, the selection of parents for recombination and the selection of individuals for survival are restricted to a neighborhood i.e. geographically nearby individuals. Genetic information propagates through overlapping neighborhoods and thus this model is also referenced as *diffusion model* to reflect this process. They're also referred as *finely-grained* or *massively parallel EAs* since they can be easily implemented on a massively parallel system.

Diffusion models have already been introduced in EAs; some examples of these can be found in Collins and Jefferson (1991); De Jong and Sarma (1995); Gorges-Schleuter (1998); Sarma and De Jong (1996, 1997). Besides their nice properties concerning parallelization, the local selection EAs have a different behavior compared to their global selection counterparts. It is claimed that they promote global diversity and especially in those cases where we have a multi-modal, nonlinear environment frequently give better results Collins and Jefferson (1991); Gorges-Schleuter (1998).

In order to implement such systems one must decide on the neighborhood size, its shape, and the particular selection algorithm to be used. Only little research has been done on diffusion models applied to timetabling problems. For example, Turner et al. (1996) describe the "tribe method", a system that improves the efficiency with which an EA can obtain multiple distinct solutions to a timetabling problem.

## 2. PREVIOUS WORK ON EVOLUTIONARY COURSE TIMETABLING

The basic element of a course timetabling problem is a set of events $E = \{e_1, e_2,...,e_n\}$. There are also a set of times $T = \{t_1, t_2,...,t_s\}$, a set of places $P = \{p_1, p_2,...,p_m\}$, and a set of 'agents' $A = \{a_1, a_2,...,a_t\}$. Each member of $E$ is a unique event requiring assignment of a time, a place, and an agent (lecturer, tutor, technician etc.). An *assignment* is a four-tuple (e,t,p,a) such as $e \in E$, $t \in T$, $p \in P$, and $a \in A$, with the interpretation "event $e$ starts at time $t$ in place $p$ involving agents $a$". A lecture timetable is simply a collection of $n$ assignments, one for each event. The problem is to find a timetable that satisfies, or minimally violates a collection of constraints Ross, Corne and Fang (1994).

There are several kinds of constraints, the most common being an 'edge constraint' between two events, which states that a given pair of events must not overlap in time. There are several other kinds of constraints such as unary, capacity, and agent constraints (Corne, Ross and Fang (1994); Ross, Corne and Fang (1994)).

In applying an EA to a problem, central considerations are the choice of representation, the design of the fitness function and the genetic operators used to evolve the population.

There are essentially two different types of representation that have been used in evolutionary timetabling. The direct representation encodes the actual timetable, where an individual is a vector of symbols $nxe$, where $e$ is the number of events and $n$ is the number of genes for each event (Corne, Ross and Fang (1994); Ross, Corne and Fang (1994)). The implicit representation, on the other hand, encodes a set of instructions as to how the timetable should be build (Corne, Ross and Fang (1994); Ross, Corne and Fang (1994)).

The algorithm used has its origins in our work presented in Adamidis and Arapakis (1999), where we used a "panmictic" population. The problem remains the same i.e. the course timetable of the Dept. of Informatics of Alexander TEI of Thessaloniki.

In brief, each individual represents an entire weekly timetable and is initially assigned random values within some range. An individual is an array *n*x*e*, where *e* is the number of events and *n* is the number of genes for each event. There are 180 different events to be scheduled (columns of the array). An individual may be created using both recombination and mutation, or one of them, or no operator at all. We use multiple-point recombination and the number of cutting points depends on the number of events. The mutation operator used has the advantage of avoiding creating unfeasible solutions. It replaces the value of a gene with some allelic value within the proper range. We've also used elitism.

## 3. LOCAL SELECTION EVOLUTIONARY ALGORITHMS

One way to think of the local selection methods is that they introduce a rather crude distance bias in which individuals within the neighborhood are "visible" and those outside "invisible" from the point of view of interaction. One can imagine other forms of distance bias in which the probability of interaction decreases as a function of the distance. This induces both a neighborhood size and shape.

There are a variety of local selection EAs that have been proposed and studied (see, for example Collins and Jefferson (1991); De Jong and Sarma (1995); Gorges-Schleuter (1998); Sarma and De Jong (1996, 1997)). In these EAs the population is distributed over a grid-like topology, and selection, mating, reproduction, etc. operate in a distributed fashion within local overlapping neighborhoods. These spatially structured EAs behave quite differently than the more familiar and better understood "panmictic" EAs where interactions are allowed between any individuals. The main source of these differences is due to the effects of local (rather than global) selection (Sarma and De Jong (1996, 1997)).

We assume a two-dimensional toroidal grid as the spatial population structure. Each grid point contains one individual of the population. The EA selects parents from the neighborhood of that grid

point in order to produce offspring, which replace the current individual assigned to that grid point. The overlapping neighborhoods offer a means for migration of genetic information throughout the whole population. The amount of overlap (and as a result the flow of information) is a function of the neighborhood size and shape.

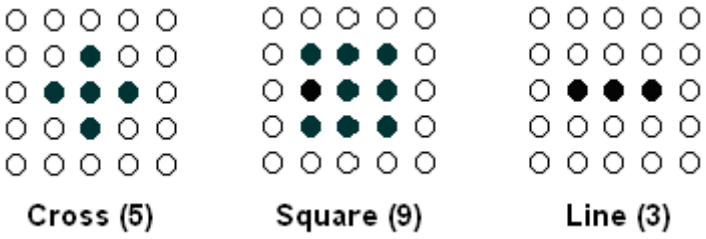


**FIGURE 1:** NEIGHBORHOOD SHAPES

Figure 1 illustrates three neighborhood shapes used in this paper. The number in parenthesis gives the size of the neighborhood. We used several neighborhood sizes ranging from 5 to 19 for cross-shaped neighborhoods, 9 to 100 for square-shaped and 3 to10 for line-shaped neighborhoods.

The selection methods used on the local neighborhoods are typically the same ones used for "panmictic" EAs. In this paper we use tournament selection.

## 4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

### *4.1 Implementation*

Some problem constraints are already encoded in the representation of the problem (Adamidis and Arapakis (1999)). The constraints used and the penalty values for each constraint violation are given in Table 1. A view of the interface and the whole set of available constraints (some of them used and some not) are shown in Figure 2.

The algorithm terminates when a number of generations is completed, or the global optimum is reached (i.e. no constraints are violated), or when the best individual is not improved for a certain number of generations. In all our experiments the algorithm terminates when the best individual does not improve for 500 generations.

**TABLE 1:** CONSTRAINTS AND PENALTY VALUES

| **Description** | **Penalty values (0-1)** |
|---|---|
| Agent 1 is the same as agent 2 | 0.8 |
| Different agents in couple teachings | 0.9 |
| Couple teachings on the same day | 0.6 |
| Two events of the same semester overlap in time | 0.5 |
| Place overlap | 0.9 |
| More than one event assigned to an agent at the same time | 0.9 |
| Each hour that exceeds an agent's weekly hour limit | 0.5 |
| An empty hour in the daily schedule of a semester | 0.1 |

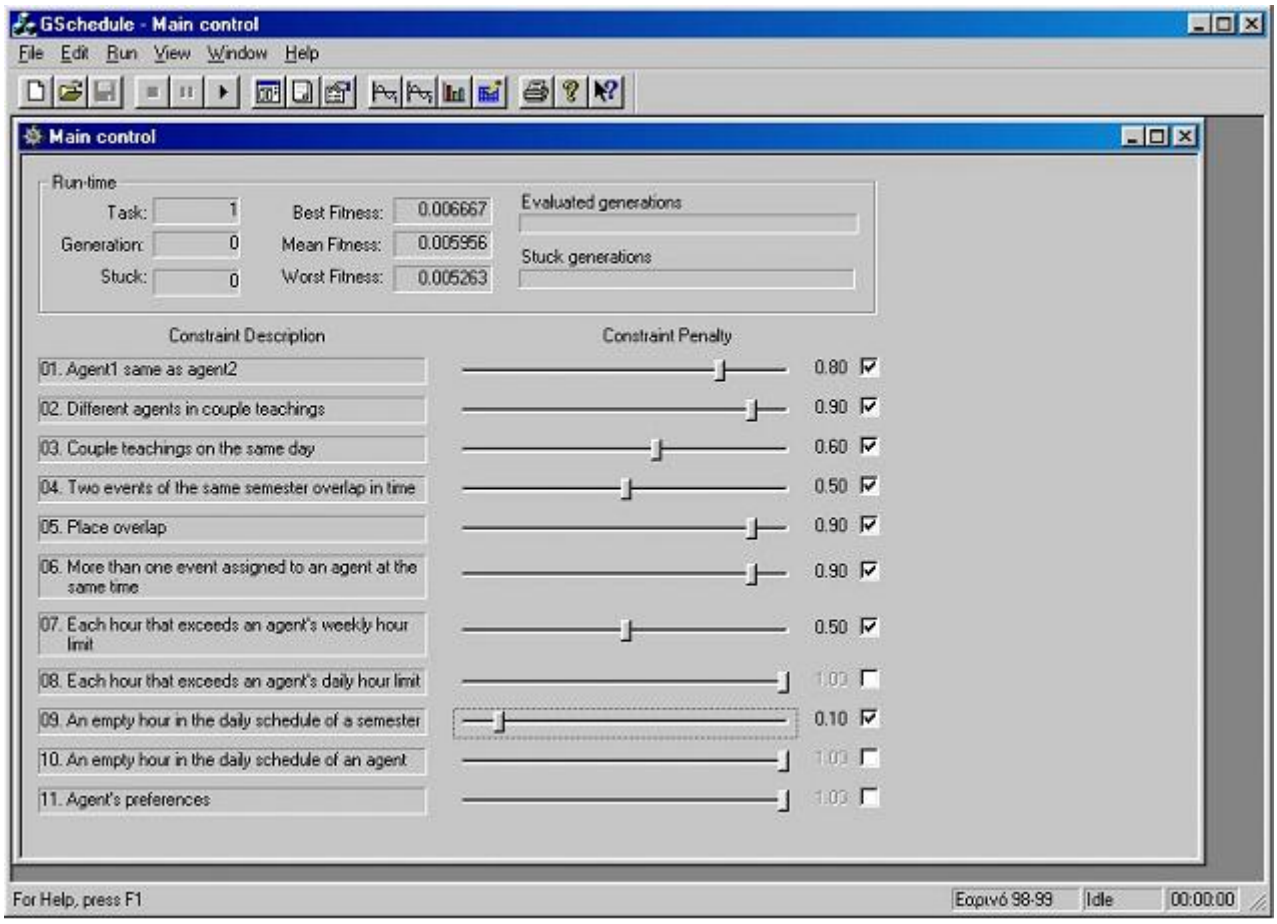


**FIGURE 2:** THE USER INTERFACE

We use integer-valued encoding of each gene, with values within some range. The genetic operators used are recombination and mutation. We also use multiple-point recombination. The mutation operator used here replaces the value of a gene with some allelic value within the proper range for the specific

assignment. More on the representation and the operators used in this paper, as well as on the evaluation function can be found in Adamidis and Arapakis (1999).

The neighborhood shapes that we used here as mentioned earlier are: Cross, Square, and Line (see Figure 1). For each shape we used 8 different sizes (24 different neighborhood configurations):

- Cross: 5, 7, 9, 11, 13, 15, 17, 19
- Square: 9, 16, 25, 36, 49, 64, 81, 100
- Line: 3, 4, 5, 6, 7, 8, 9, 10

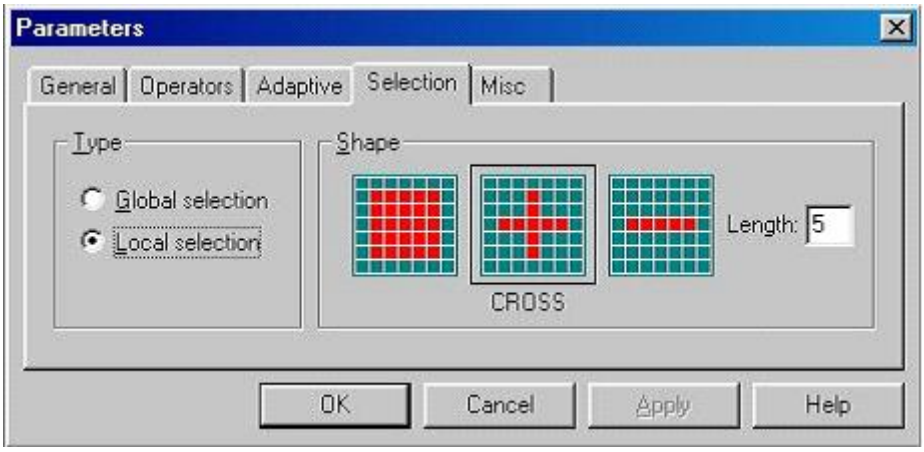


**FIGURE 3:** SETTING NEIGHBORHOOD SHAPE AND SIZE

Figure 3 shows the programming interface that allows us to choose the neighborhood shape and size. For each neighborhood configuration we used 4 different recombination and mutation rates. Table 2 gives the different operator configurations.

**TABLE 2:** CONFIGURATIONS OF OPERATOR RATES

| Rate configuration | Mutation | Recombination |
|---|---|---|
| 1 | 0.2 | 0.8 |
| 2 | 0.4 | 0.6 |
| 3 | 0.6 | 0.4 |
| 4 | 0.8 | 0.2 |

*4.2 Experimental Results*

We used three different population sizes: 50, 100, 150 individuals. (The population size with 50

individuals can be tested only for the first three neighborhood sizes).

The results from the different local selection configurations are compared with the global selection ("panmictic") algorithm.

Figures 4, 5, and 6 show the effect of different operator rates and different population sizes on the behavior of the three neighborhood shapes, for half of the used neighborhood sizes. The behavior is the same for the rest neighborhood sizes as well. The results are the average of 50 runs. We observe that we get better results with the fourth operator rate configuration, which has a high mutation rate and a small recombination probability. This agrees with the behavior of the global selection algorithm ("panmictic" population), which has also been reported by Adamidis and Arapakis (1999). Better results are also obtained using the bigger population size with 150 individuals.

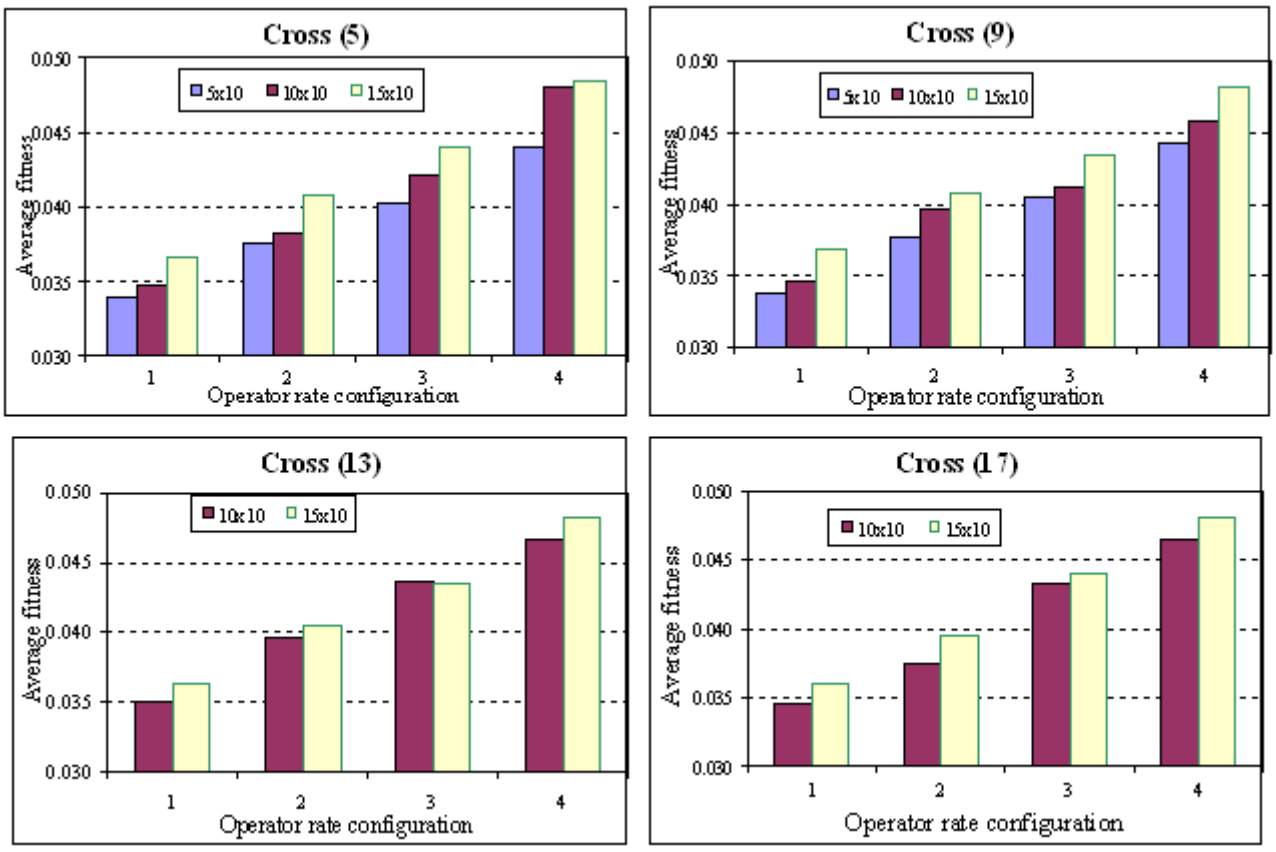


**FIGURE 4:** BEHAVIOR OF THE CROSS SHAPED NEIGHBORHOOD

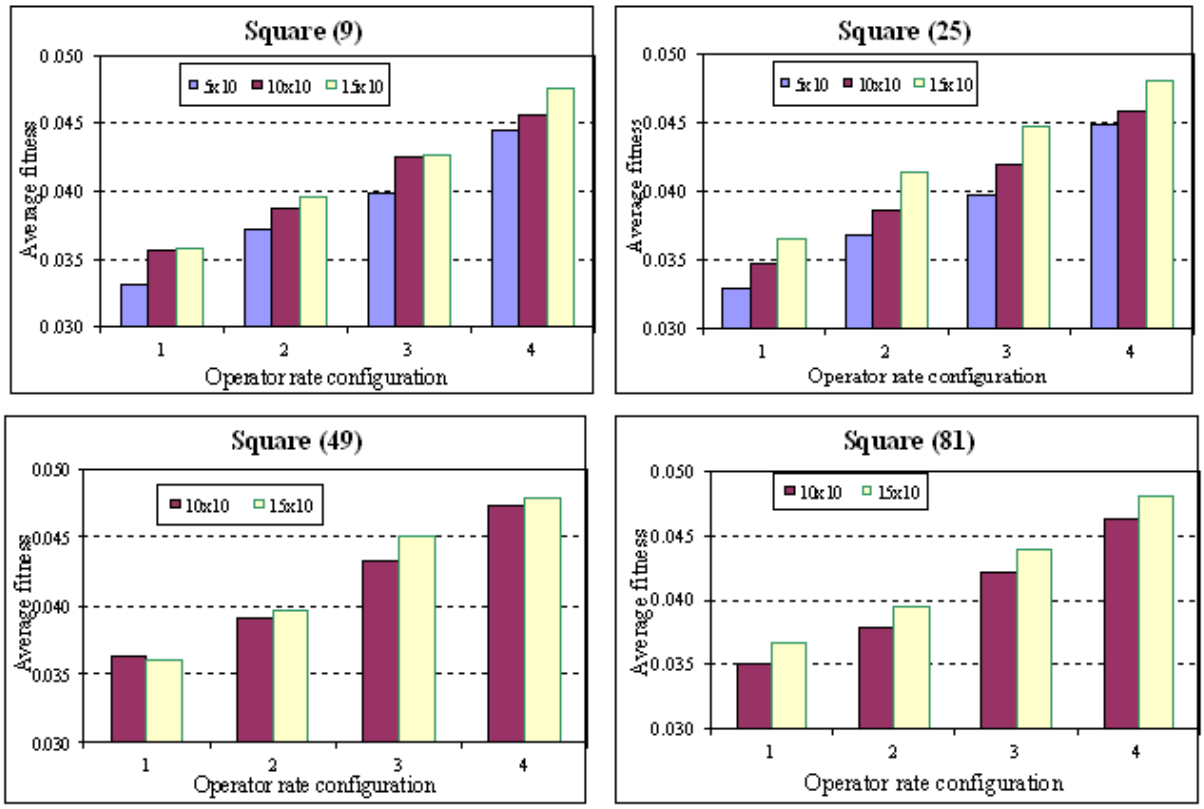


**FIGURE 5:** BEHAVIOR OF THE SQUARE SHAPED NEIGHBORHOOD

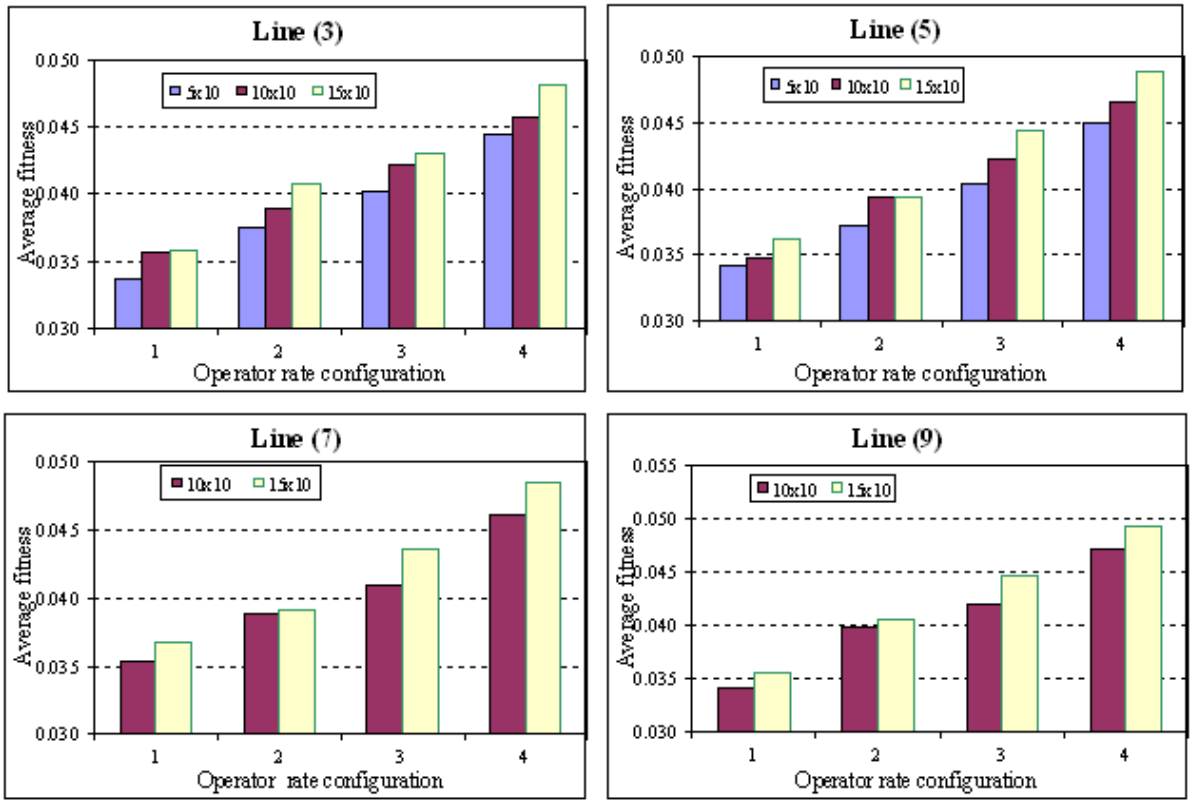


**FIGURE 6:** BEHAVIOR OF THE LINE SHAPED NEIGHBORHOOD

A question that we can ask is: “Is it better to use dense or sparse neighborhood shapes?” Figure 7 illustrates the behavior of the three neighborhood shapes with the same size (9 individuals), for the two operator rate configurations with best performance (i.e. 3 and 4), and for two population sizes (100 and 150 individuals). We observe that the average performance of sparse neighborhood shapes is better since line neighborhood has a better performance than cross and square shaped neighborhoods. The densest shape (square) has the worse performance.

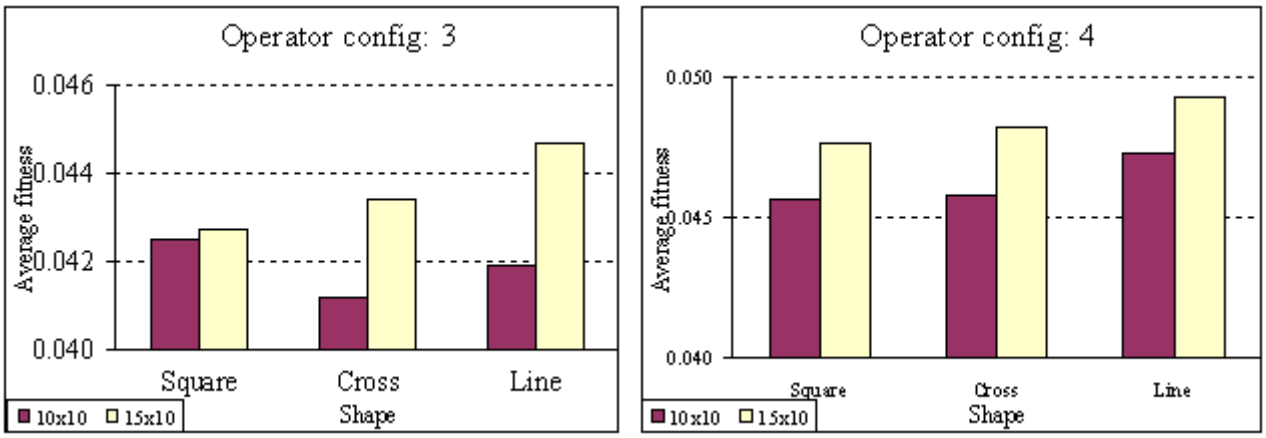


**FIGURE 7:** BEHAVIOR OF THE LINE SHAPED NEIGHBORHOOD

Another issue is the neighborhood size. Our results (Figure 8) show no clear advantage of the use of a large or small neighborhood size, and this is true for all neighborhood shapes either sparse or dense. Even more, local selection algorithms do not show a better performance over global selection algorithms on the timetabling problem.

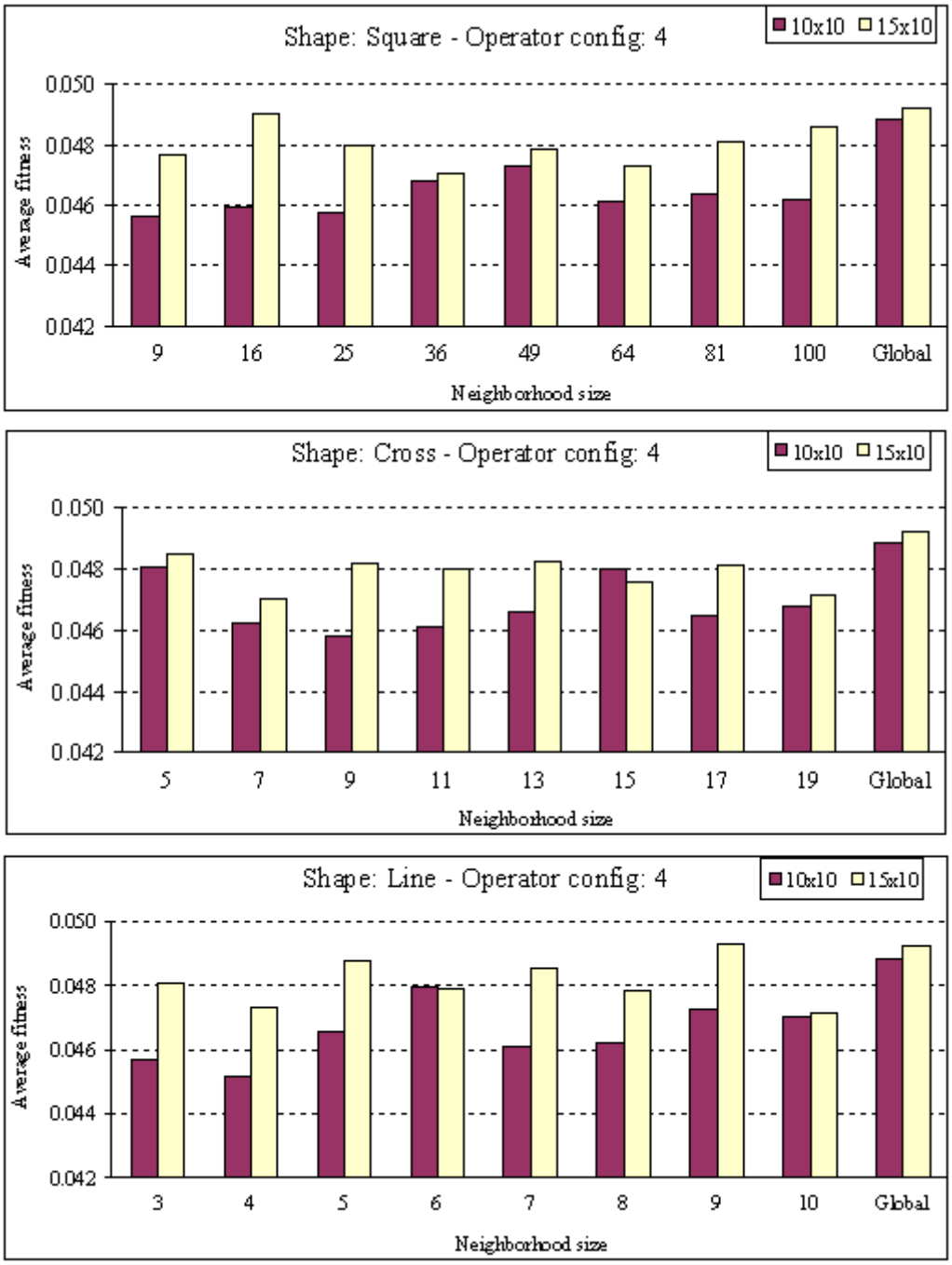


**FIGURE 8:** EFFECT OF NEIGHBORHOOD SIZE

## 5. DISCUSSION

This paper presents an experimental study of the effects of local selection EAs on a real course timetabling problem using three different neighborhood shapes (square, cross and line). Usually selection

is studied alone i.e. no mutation or recombination is active. In this paper we also studied the effects of different operator configurations on the different local selection schemes. We tested a lot of different genetic operator probabilities. Our results indicate that it's better to use a high mutation rate and a small recombination probability and this holds for all three neighborhood shapes and all neighborhood sizes that we have studied.

The analysis of the three local selection neighborhood shapes, suggest the use of sparser neighborhood shapes. Of the three neighborhood shapes studied, the line shaped neighborhood appears to have the best performance.

The analysis of our results on the neighborhood size is not clearly in favor of a large or small neighborhood size. This is something that seems to need more experimentation.

In this paper we used constant mutation rate and recombination probabilities. An interesting open question is what will be the effect of adaptive operator probabilities on the performance of the different neighborhood shapes and sizes. In addition, it would also be interested to experiment on the effects of population re-initialization to local selection.

## REFERENCES


Adamidis, P. and Arapakis, P. (1999). "Evolutionary Algorithms in Lecture Timetabling." In Proceedings of 1999 Congress on Evolutionary Computation (CEC'99). IEEE, pp. 1145-1151.

Burke, E.K., Newall, J.P. and Weare, R. F. (1996). "A Memetic Algorithm for University Exam Timetabling." In Burke, E. and Ross, P. (eds.), Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1153, Springer-Verlag.

Carter M. W. and Laporte G. (1998). "Recent Developments in Practical Course Timetabling." In: Burke, E., Carter, M. (eds.), Practice and Theory of Automated Timetabling II, Lecture Notes in Computer Science 1408. Springer-Verlag, 3–19.

Collins R.J. and Jefferson D.R. (1991). "Selection in Massively Parallel Genetic Algorithms." In Belew R.K. and Booker L.B. (eds.), Proceedings of the 4th International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 249-256.

Corne, D., Ross, P. and Fang, H.-L. (1994). "Fast Practical Evolutionary Timetabling", In Fogarty T.C. (ed.), Proceedings of the AISB Workshop on Evolutionary Computation, Lecture Notes in Computer Science 865. Springer-Verlag.

De Jong K. and Sarma J. (1995). “On Decentralizing Selection Algorithms.” In Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 17-23.

Gorges-Schleuter M. (1998). “A Comparative Study of Global and Local Selection in Evolution Strategies.” In Eiben A.E., Bäck T., Schoenauer M. and Schwefel, H.-P. (eds.), Parallel Problem Solving from Nature – PPSN V, Lecture Notes in Computer Science 1498, Springer-Verlag, pp. 366–377.

Paechter B., Rankin R.C., Cumming A. and Fogarty T.C. (1998). “Timetabling the Classes of an Entire University with an Evolutionary Algorithm.” In Eiben A.E., Bäck T., Schoenauer M. and Schwefel, H.-P. (eds.), Parallel Problem Solving from Nature – PPSN V, Lecture Notes in Computer Science 1498. Springer-Verlag, pp. 865–874.

Paechter B., Rankin R.C. and Cumming A. (1998). “Improving a Lecture Timetabling System for University-Wide Use.” In Burke, E. and Carter, M. (eds.), Practice and Theory of Automated Timetabling II, Lecture Notes in Computer Science 1408, Springer-Verlag, pp. 156–165

Ross P., Corne D. and Fang H.-L. (1994). “Successful Lecture Timetabling with Evolutionary Algorithms.” Working paper, Dept. of AI, Univ. of Edinburgh, UK.

Sarma J. and De Jong K. (1997). “An Analysis of Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm.” In Bäck T. (ed.), Proceedings of the 7th International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 181-186.

Sarma J. and De Jong K. (1996). “An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms.” In Voigt H.-M., Ebeling W., Rechenberg I. and Schwefel, H.-P. (eds.), Parallel Problem Solving from Nature – PPSN IV, Lecture Notes in Computer Science 1141, Springer-Verlag, pp. 236–244.

Turner, A., Corne, D., Ritchie G. and Ross, P. (1996). “Obtaining Multiple Distinct Solutions with Genetic Algorithm Niching Methods.” In Voigt H.-M., Ebeling W., Rechenberg I. and Schwefel, H.-P. (eds.), Parallel Problem Solving from Nature – PPSN IV, Lecture Notes in Computer Science 1141, Springer-Verlag, pp. 451-460.