Summer 2010

An artificial life approach to evolutionary computation: from mobile cellular algorithms to artificial ecosystems

Shivakar Vulli

AN ARTIFICIAL LIFE APPROACH TO EVOLUTIONARY COMPUTATION: FROM MOBILE CELLULAR ALGORITHMS TO ARTIFICIAL ECOSYSTEMS

by

SRINIVASA SHIVAKAR VULLI

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2010

Approved by

S. Agarwal, Co-advisor
S. Madria, Co-advisor
D. Tauritz

ABSTRACT

This thesis presents a new class of evolutionary algorithms called *mobile cellular evolutionary algorithms* (mcEAs). These algorithms are characterized by individuals moving around on a spatial population structure. As a primary objective, this thesis aims to show that by controlling the population density and mobility in mcEAs, it is possible to achieve much better control over the rate of convergence than what is already possible in existing cellular EAs. Using the observations and results from this investigation into selection pressure in mcEAs, a general architecture for developing agent-based evolutionary algorithms called Artificial Ecosystems (AES) is presented. A simple agent-based EA is developed within the scope of AES is presented with two individual-based bottom-up schemes to achieve dynamic population sizing. Experiments with a test suite of optimization problems show that both mcEAs and the agent-based EA produced results comparable to the best solutions found by cellular EAs.

ACKNOWLEDGMENTS

My first thanks and immense gratitude goes to Dr. Sanjeev Agarwal, without whose constant guidance, support and mentoring, this thesis would not have been possible. Over the course of the last four-and-a-half years, I have learned a great deal about a lot of things from him. Most of all, a thirst for knowledge and passion for investigation and experimentation. He has been a great advisor to me who always gave me time and freedom to explore a variety of research areas.

I would like to thank Dr. Sanjay Madria for accepting to be my department co-advisor for this thesis. He has been most generous in accommodating me into his busy schedule and for this I am forever thankful.

I would like to thank Dr. Daniel Tauritz for agreeing to be on my thesis committee. He has been most helpful in helping me identify the thesis topic and the outline during his Advanced Evolutionary Computation course. He accommodated me with numerous discussions and giving pointers on several occasions. His help with this thesis is greatly appreciated.

I would also like to thank my friends, Sarat Sreepathi and Vamsi Sripati for all their support, feedback and the numerous discussions throughout my graduate studies. A special thanks to all the people I have known through ARIA lab for making my stay in Rolla a memorable time.

I would like to express my eternal gratitude to my parents for giving me the independence and opportunity to pursue my dreams and always believing I can achieve them. Finally, I would like to thank my fiancee, Bindu Laxminarayan, for her selfless and unwavering love, support, and patience during all these years of graduate school.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ALGORITHMS

1. INTRODUCTION

Evolutionary computation is a subfield of computational intelligence and involves the use of metaheuristic optimization techniques such as evolutionary algorithms, swarm intelligence, and artificial immune systems, among others. Of these, evolutionary algorithms is the most widely accepted as representative of the field. Evolutionary algorithms (EAs) are loosely based on the metaphor of biological evolution, typically encoding biological phenomena such as selection, reproduction, and mutation (called *genetic operators*) into black-box procedures to solve some search and/or optimization problem. These algorithms typically start with a set of initial solutions encoded into the genotypes of a population of individuals and iteratively *evolve* better solutions by the application of the genetic operations on the population or on the individuals. One important aspect of biological evolution that is typically ignored by evolutionary algorithms is the spatial structure of populations. While spatially structured populations in evolutionary algorithms have been investigated in the forms of island models (Sprave, 1999) and cellular evolutionary algorithms (Alba and Dorronsoro, 2008a), they have generally received much less attention than evolutionary algorithms with no population structure (called *panmictic models*) (Tomassini, 2005).

It is interesting to note that *panmixia* (globally mixing populations with no population structure), while easy to model and achievable in laboratory settings, does not exist and is in fact impossible to achieve in the natural world. Evolutionary biologists have long known the significance of spatial structure in evolution. Even in one of the earliest works on biological evolution, Charles Darwin (Darwin, 1900) noted that geographic separation and in particular isolation together with occasional migration played a significant role in the differences between evolved traits of the same species. While geographic isolation with occasional migration[1] is the underlying philosophy of island model EAs, it is not representative of evolution in the natural world. Populations of different or same species are not isolated in the natural world[2]. Migration

[1] Migration in island models is generally accomplished according to some predetermined topology.

[2] While islands in the natural world do exhibit the *isolation with occasional migration* phenomena, it can safely be claimed that this is not the general case.

among populations does not happen in sporadic bursts, but rather in a continuous fashion due to locally mixing populations. Cellular evolutionary algorithms (cEAs) attempt to capture this continuous dissemination of genetic information by enforcing a rigid spatial structure on the population. Genetic operators such as selection, reproduction and mutation are executed on small neighborhoods or *demes*[3]. While cEAs do achieve continuous flow of genetic information throughout the population (throughout the world of the cEA), due to the rigid structuring of the population, migration between two unconnected demes is not possible. Hybrid island-cellular EAs (Cantu-Paz, 2000) have been proposed to incorporate both migration and continuous genetic information flow into EAs, with isolated populations structured in the rigid manner typical of cEAs and a communication topology between these populations to facilitate migration.

The primary motivation for research into spatially structured EAs is that they seem to alleviate the major problem often attributed to panmictic EAs (Tomassini, 2005; Skolicki, 2007; Alba and Dorronsoro, 2008a), namely, premature convergence due to lack of genetic diversity. In panmictic EAs, due to global mixing of the population, the best individual in the current generation (which may not be the global optimum solution) has equal probability to influence any member of the population. This global influence results in the decline of diversity in the population. Decline in diversity of the population means that sufficient genetic information is not available to generate new solutions[4], resulting in convergence to points in the fitness landscape which may not be optimal. In structured populations, however, the current best individual can only influence evolution in its own deme or island and therefore information from the best individual would take several generations before influencing the entire population, enabling the population to maintain more diversity.

In this thesis, a new class of evolutionary algorithms, termed *mobile cellular EAs* (mcEAs) are proposed to facilitate the migration of individuals between demes while maintaining a population structure to preserve genetic diversity. This is accomplished by allowing individuals to move around their environment (the world of the mcEA) with genetic operators being applied in the deme local to an individual. The deme of

[3]A Deme is a subpopulation, typically much smaller than the total population, which is subjected to the selection operator as a unit rather than as individuals.

[4]New solutions in either/both, genotypic and phenotypic space

an individual changes as it moves around in the environment, however, the structure of the deme remains the same. As one of its objectives, this thesis aims to show that by controlling the mobility it is possible to achieve much better control over the rate of convergence in mcEAs than what is already possible by manipulating the *ratio*[5] (Alba and Troya, 2000) in cEAs.

Another motivation for the study of mcEAs is to provide common ground for comparing and investigating existing evolutionary techniques with the newly emerging field of agent-based evolutionary computation (Sarker and Ray, 2010). While agent-based models are routinely used in the study of social (Axelrod, 1997; Epstein and Axtell, 1996), economic (Tesfatsion, 2002), ecological (Grimm and Railsback, 2005) and biological (Coakley et al., 2006) phenomena, very few attempts have been made to use these for evolutionary search and optimization. Although agent-based models are difficult to validate and model analytically, in many of the problem domains they are particularly useful in explaining real-world observations when mathematical modeling of the phenomena is difficult to develop at best. As an extension to mcEAs, this thesis presents a conceptual description and architecture for developing agent-based EAs. This architecture, called *artificial ecosystems* (AES), draws inspiration from artificial life (ALife) experiments such as Tierra (Ray, 1991), Avida (Adami and Brown, 1994) and Framesticks (Komosinski and Rotaru-Varga, 2001), which aimed at providing a decentralized platform for studying artificial evolution. The architecture consists of multiple evolving and non-evolving species interacting (based on spatially local interaction rules) in an spatial environment. While existing agent-based EAs do justice in capturing this decentralized bottom-up approach of capturing emergent phenomena[6], almost all of them are developed in an ad-hoc fashion for a specific application. Most of these algorithms involve a large number of simulation/algorithmic parameters with no justifiable theoretical and/or philosophical reasons. This thesis aims to use the results from the study of mcEAs to develop an architecture, or at

[5]It has been demonstrated by Alba and Troya (2000) that ratio of neighborhood size to the population size, simply called the *ratio*, is what influences the rate of convergence in cEAs.

[6]In agent-based EAs, the emergent phenomena is to evolve optimal solutions to the given optimization problem using locally interacting agents, with no explicit notion of fitness function and/or fitness evaluation.

the least, guidelines for developing coherent agent-based EAs which are true to the philosophy of artificial life research from which they claim to have been derived.

1.1. MAJOR CONTRIBUTIONS

While agent-based EAs are not new, very little if any work has been done in a systematic description of the building blocks of these algorithms. To the best of the author's knowledge an attempt to identify mobility of these agents as a contributing factor for the performance of the algorithms has never been done. This thesis therefore presents an original work and the following are considered as its major contributions:

1. **The theoretical description and analysis of selection pressure in mobile cellular evolutionary algorithms**: As stated before, the first objective of this thesis is to show that by introducing the notion of individual mobility into cEAs (to create mcEAs), we can control the rate of convergence at a finer level than what is already possible in cEAs, resulting in improved performance. This is accomplished by investigating selection pressure in mcEAs.

2. **The development of artificial ecosystems (AES) as an architectural description for developing agent-based EAs**: Staying true to the decentralized, bottom-up modeling approach of agent-based models (and in general ALife), instantiations of AES are developed using individual level genetic operators. The interactions between individuals of an AES then result in observable emergent system-level (global-level) phenomena. While very little effort is put into developing new or special variational operators, this thesis does present decentralized bottom-up ways to achieve selection and replacement in AES in the form of two population dynamics schemes.

3. **The development of single species & landscape (SSL) model as a basic agent-based EA**: Using the knowledge gained in developing mcEAs and the population dynamics schemes for AES, a special case of AES, consisting of a single species in a landscape is developed as a basic agent-based EA.

1.2. ORGANIZATION OF THE THESIS

A brief overview of the organization of this thesis is as follows. In Section 2, the current state-of-the-art in theoretical modeling of cellular EAs is presented. Section 3

introduces mobile cellular EAs and presents an empirical study of selection pressure in the same. The section concludes with experimental results using a test suite of combinatorial optimization problems.

Section 4 presents a survey and discussion on existing agent-based evolutionary algorithms. In Section 5 two population dynamics models inspired by ALife for agent-based EAs are presented. Section 6 presents *Artificial Ecosystems* (AES), the general architecture for developing agent-based EAs. A simple single species algorithm for function optimization is discussed within the scope of AES. Two variations of this algorithm are developed using the two population dynamics models developed in Section 5. Experiments are conducted on a test suite of problems and the results are presented. A conceptual description for developing AES instances with multiple species is also presented in Section 6.

Section 7 concludes this thesis with a discussion on open problems and future directions of research.

2. CELLULAR EVOLUTIONARY ALGORITHMS

Evolutionary algorithms iteratively evolve new (and hopefully better) solutions by applying selection, recombination and mutation operators on a population of individuals. In evolutionary algorithms with no spatial population structure, during the *selection* of parents, the probability of an individual being paired up with any other individual of the population for *reproduction* is equal. It is argued that this global mixing or *panmixia* is responsible for the loss of genetic diversity over the course of the evolutionary algorithm resulting in premature convergence of the population to a sub-optimal solution. It should be noted that in unstructured populations, the *selection* operator can be considered to be a centralized mechanism, i.e., the selection of an individual at any stage of the *selection* process is independent of the selection of another individual in the population.

In cellular evolutionary algorithms (cEAs), however, the population is structured into local neighborhoods, called *demes*. Genetic operators, such as selection and recombination, are applied on these *demes* instead of the entire population. Here, the selection of an individual at any stage of the *selection* process, depends on whether or not the individual is a member of the current demes under consideration. This decentralized application of genetic operators results in reducing the rate at which genetic information is propagated through the population and helps in preserving the diversity of the population by removing the possibility of a local optimal solution exerting equal influence over all the individuals in the population. For a population P, the neighborhood is defined by the neighborhood function N as follows

$$N : P \rightarrow \mathcal{P}(P) \tag{1}$$

where $\mathcal{P}(P)$ is a set of subsets of P.

This neighborhood function associates each individual i with a set of individuals $N(i)$ called its neighborhood. Typically, this is a symmetric function with

$$j \in N(i) \implies i \in N(j) \qquad \forall i, j \in P$$

For cEAs, the neighborhood is typically much smaller than the size of the population, i.e., $|N(i)| << |P| \quad \forall i \in P$, whereas for panmictic EAs, the neighborhood of each individual is the population itself, i.e., $N(i) = P \quad \forall i \in P$. This neighborhood structuring is the primary difference between panmictic and structured population evolutionary algorithms. Even with other genetic operators and replacement schemes being the same, the structure of neighborhood is shown to have significant effect on the performance of the evolutionary algorithm (Alba and Dorronsoro, 2008b). While population topologies such as array (Giacobini et al., 2004b), ring (Rudolph, 2000) and grid (Whitley, 1993, 1995) have also been explored, the torus structure (Giacobini et al., 2004a) is most commonly used. This is especially true for the agent-based evolutionary algorithm discussed in later sections of this thesis. To this end, only the torus population structure is discussed in greater detail in this thesis. Figure 2.1 shows a toroidal population structure with several neighborhood models.

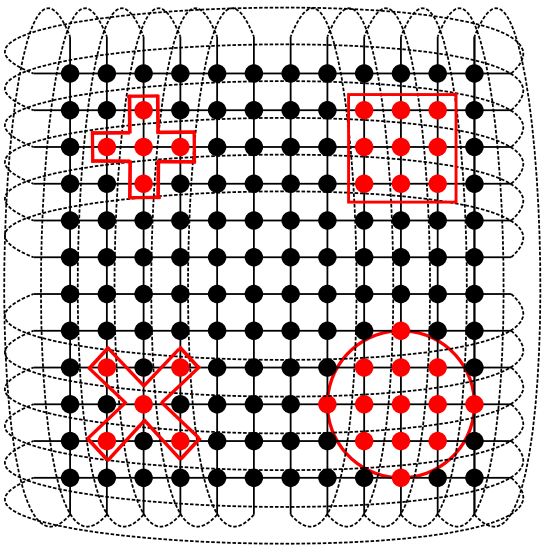

Figure 2.1: A population structured on a 2-D torus with several neighborhood topologies.

2.1. SELECTION PRESSURE AND TAKEOVER TIME

While criteria such as time-to-solution and solution quality can be used to characterize the performance of an evolutionary algorithm, additional criteria are

required to understand and characterize the behavior of the same. Goldberg and Deb (1991), introduced the concept of takeover time to study the effects of selection schemes on the behavior of genetic algorithms. Takeover time is defined as the expected number of applications (generations) of some selection method until the population of the EA consists entirely of copies of the best individual, in the absence of variational operators and assuming that the initial population consists of a single copy of the best individual (Rudolph, 2000). The takeover time is directly related to the selection pressure of a selection scheme. Shorter takeover times indicate higher selection pressure while longer takeover times indicate lower selection pressure. This selection pressure indicates the trade off between exploration and exploitation of the search landscape that all evolutionary algorithm try to balance. Lower selection pressure indicates higher exploration at the expense of increased execution time where as higher selection pressure indicate higher exploitation with an increase in possibility of premature convergence to a sub-optimal solution. Plotting the number of copies of the best individual in each generation, depicts the general behavior of a selection method. Referred to as *growth curves*, these plots are generally used to compare the selection pressure of different selection schemes.

2.2. STRUCTURAL RATIO

Sarma and De Jong (1996, 1997) studied the effects of the size and structure of the neighborhood on the selection pressure of different selection schemes in cEAs. They observed that the selection pressure is not independently influenced by the sizes and structures of the population and neighborhood, but by the ratio of the neighborhood radius to the radius of the population topology. Although the authors and subsequent investigators simply termed this as the *ratio*, in this thesis this is re-termed and referred to from here on as *structural ratio* for clarity. Equations 2-4 show the calculation of the neighborhood & topology radii and structural ratio, respectively.

$$r_n = \sum_{i=1}^{n^*} \frac{\sqrt{(x_i - x^*)^2 + (y_i - y^*)^2}}{n^*} \quad (2)$$

$$r_t = \sum_{i=1}^{n} \frac{\sqrt{(x_i - x)^2 + (y_i - y)^2}}{n} \quad (3)$$

$$r = \frac{r_n}{r_t} \quad (4)$$

where r_n and r_t are the radius of neighborhood and topology, n^* and n are the number of individuals in a neighborhood and population, x^* and y^* are the mean location of a neighborhood and x and y are the mean location of the topology given by Equation 5.

$$\begin{aligned} x^* &= \frac{\sum_{i=1}^{n^*} x_i}{n^*} \quad x = \frac{\sum_{i=1}^{n} x_i}{n} \\ y^* &= \frac{\sum_{i=1}^{n^*} y_i}{n^*} \quad y = \frac{\sum_{i=1}^{n} y_i}{n} \end{aligned} \quad (5)$$

Subsequently, Alba and Troya (2000) demonstrated that selection pressure would be the same for different population and neighborhood sizes and topologies as long as the structural ratio remained the same. This is an important result for characterizing cEAs, since the selection pressure of seemingly different topologies and neighborhood shapes can now be compared both empirically and theoretically. The structural ratio and selection pressure are directly proportional to each other. A small structural ratio indicates lower selection pressure, thus promoting exploration of the search space, while a large structural ratio indicates a high selection pressure.

2.3. MODELING SELECTION PRESSURE: A BRIEF SURVEY

Theoretical modeling of selection pressure in cEAs has received lot of attention over the years and a wealth of literature in this area is available. In this section, several seminal efforts in characterizing the selection pressure in cEAs are revisited.

Sarma and De Jong (1996) were one of the first to analyze the effect of the size and shape of the neighborhood on the selection pressure of a cEA. They investigated

the effect of several neighborhood topologies including $L5$ (Von-neumann neighborhood), $C9$ (Moore neighborhood), $L9$ (2-pixels on each of NEWS directions), and $C13$ (13 closest cells) on two selection schemes, namely, fitness proportional selection and linear ranking selection. They observed that the selection intensity for a given selection scheme was similar for all neighborhoods of a similar neighborhood radius (given by Equation 2). Throughout the work the size of the grid was kept constant at 32×32 toroidal grid. They also proposed a simple quantitative model for selection intensity based on the well-known logistic equation as:

$$P_{b.t} = \frac{1}{1 + \left(\frac{1}{P_{b,0}} - 1\right) e^{-at}} \tag{6}$$

where $P_{b,t}$ is the proportion of the best individuals in the population at time t, and a is the coefficient of growth.

While the simple logistic model is able to characterize the trend of selection pressure, Gorges-Schleuter (1999) noted that the logistic equation does not hold for spatially structured populations with local selection schemes. She investigated the selection pressure in ring and torus population structures and found linear and quadratic growth equations, respectively. Giacobini et al. (2003a) later extended these quadratic growth equations to bounded populations with synchronous updates. Sprave (1999) proposed a hypergraph-based unified model for studying selection pressure in all non-panmictic populations, i.e., any spatial structures such as cEAs or island models.

Alba and Troya (2000) demonstrated that the structural ratio is in fact the governing parameter for the selection pressure in cEAs and not the radius of the neighborhood itself. Numerical experiments were conducted to demonstrate the utility of dynamically changing the ratio in improving the performance of cEAs on several well-known test problems. The ratio was changed by altering the shape of the population topology and also the size and shape of the selection neighborhood. They concluded that a thin-grid with low-ratio was more suitable for solving multi-modal and/or epistatic problems and that dynamically changing the structural ratio is useful in altering the algorithm's behavior from exploitative to explorative and vice-versa.

Rudolph (2000) investigated the selection intensity in array and ring population structures in the case of synchronous update and developed a quantitative model for

selection intensity in the same using a graph-based approach. Giacobini et al. (2003b) extended this investigation to include several asynchronous update schemes, namely, *fixed line sweep* (LS) in which the cells in the grid are updated sequentially, *fixed random sweep* (FRS) in which a fixed random order of cells is selected at the first iteration and cells are updated according to this order throughout the course of the simulation, *new random sweep* (NRS) in which a random order of cells is generated each iteration, and finally, *uniform choice* (UC) in which the next cell is selected uniformly at random and with replacement. Two selection schemes, namely, binary tournament selection and linear ranking selection were used for the experiments. In both the selection schemes, it was noted that fixed line sweep induced the most selection pressure, where as synchronous update induced the least selection pressure, followed by uniform choice, new random sweep and fixed random sweep. A numerical investigation on the solution quality of synchronous and asynchronous update schemes can be found in Alba et al. (2002). Giacobini et al. (2004a, 2005a,b) later investigated the selection pressure due to several asynchronous update schemes in regular lattice, toroidal, random and small-world structured populations.

Alba and Dorronsoro (2008c) proposed a non-parameterized probabilistic approach to modeling selection pressure in synchronous cEAs. This new method was compared against the three existing methods, namely, the logistic model, the hypergraph model and the modified logistic model (with quadratic growth equation). They noted that the probabilistic model produced the best fit with empirical observations followed by the modified logistic, hypergraph and logistic model, respectively. Also this model was independent of the structural ratio and only incorporated the probabilities that a given individual would have one or two best individuals in its neighborhood. Three selection schemes, namely, roulette wheel, binary tournament, and linear ranking selection were considered with Von-neumann neighborhood as the only neighborhood topology investigated.

2.4. EMPIRICAL ANALYSIS OF SELECTION PRESSURE

In this section, empirical analysis of selection pressure in cEAs is conducted. The selection pressure in cEAs due to synchronous and two asynchronous update schemes

are studied using several neighborhood topologies and structural ratios. Figure 2.2 shows the different neighborhood topologies (and their radius) used in the study.

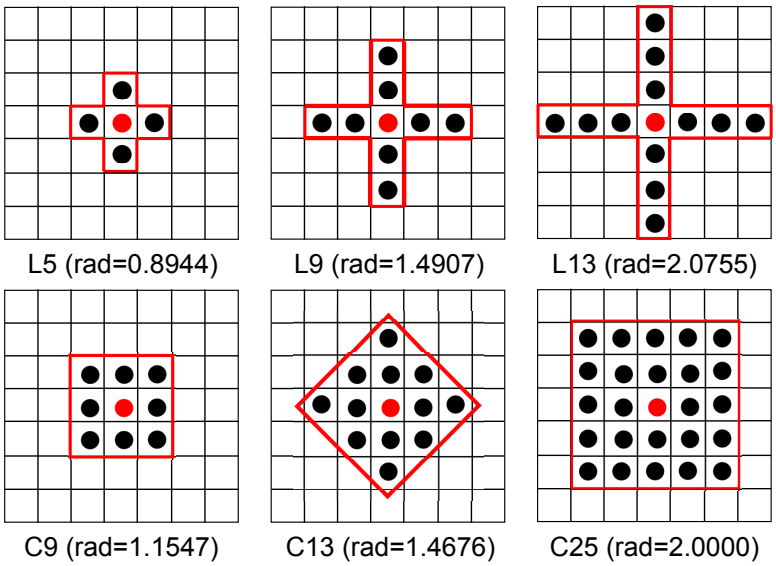

Figure 2.2: Neighborhood topologies used.

2.4.1. Experimental Setup. Experiments were conducted for synchronous and two asynchronous update schemes, namely, *new random sweep* and *uniform choice*. A toroidal population structure is used due to its popularity and to facilitate comparison in later sections. In *new random sweep* a random order of cells (individuals) is generated each time step and the cells are updated in the generated order. In *uniform choice* update scheme, a cell is selected uniform randomly from the population and updated. This process is repeated n times, where n is the population size. For parent selection, the individual in the cell being updated is always considered as one of the parents. The other parent is selected uniform randomly from the neighborhood of the current cell. The best of the two parents is simply *copied* into the offspring. For replacement, *replace-worst-if-better* policy is used by which the offspring replaces the worst individual in the neighborhood, if the offspring is better. If multiple worst individuals are present in the neighborhood, one of them is replaced uniformly at random. When considering selection pressure, only two fitness

values can exist, either an individual is best (fitness = 1) or the individual is not the best (fitness = 0). Therefore, an offspring replaces an individual from the neighborhood if the offspring is a copy of the best and the individual is not the best. The particular choices in the population structures, update schemes, selection policy and tournament size were made in order to facilitate comparison with mobile cellular EAs in later chapters.

Additionally, comparison with selection pressure in a steady-state panmictic EA is also presented for completion. Binary tournament selection with a tournament size of 32 is used for selecting parents in each time step. This mimics the same selection probability for the best individual in the early time steps of a simulation run. Averages of 100 independent runs are reported as results.

Table 2.1 shows the list of parameters used for the selection pressure study. Table 2.2 shows the radius and the resulting structural ratio for each neighborhood topology used. For random number generation, Mersenne twister[7] was used throughout the thesis.

Table 2.1: Parameters used for selection pressure study.

Population size	1024
Grid size (for cEAs)	32×32 (rad = 13.058)
Neighborhoods	see Table 2.2
Parent selection (for panmictic EA)	Binary tournament + binary tournament
Parent selection (for cEA)	Central selection + uniform random
Tournament size (for panmictic EA)	32
Replacement	*replace-worst-if-better*
Time step size	1024 offsprings generated

2.4.2. Results. Figure 2.3 shows the general trend of selection pressure due to the update scheme and neighborhood topology.

[7]Implementation used can be downloaded from http://www-personal.umich.edu/~wagnerr/MersenneTwister.html.

Table 2.2: Neighborhoods used for the selection pressure study (for 32x32 torus population structure).

Label	Radius	Structural ratio
L5	0.8944	0.0685
L9	1.4907	0.1142
L13	2.0755	0.1589
C9	1.1547	0.0884
C13	1.4676	0.1124
C25	2.0000	0.1532

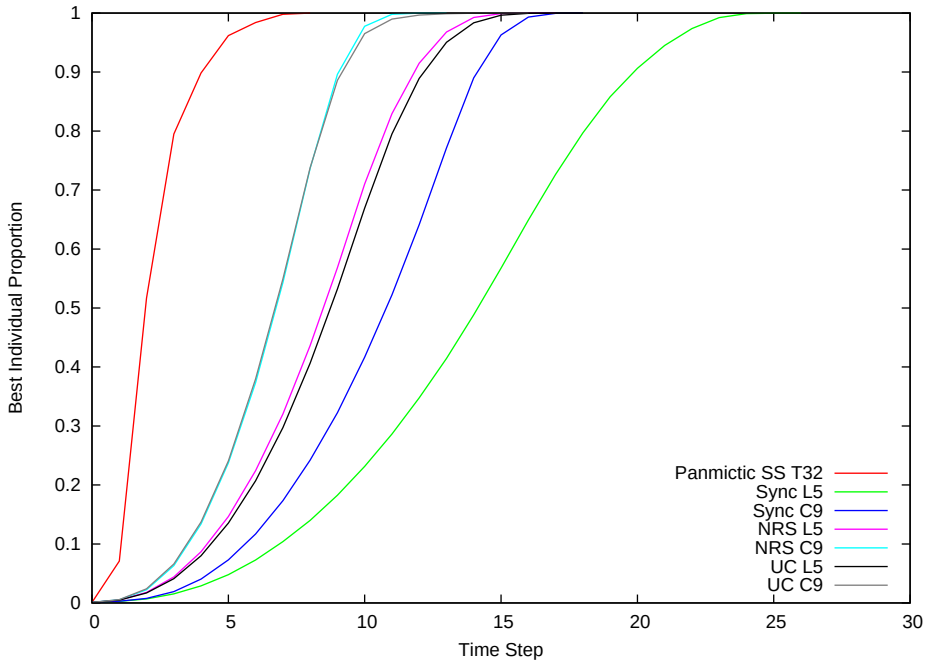

Figure 2.3: Growth curves for various update schemes and neighborhood topologies.

The highest selection pressure is induced by the panmictic EA, while the lowest selection pressure is induced by the synchronous cEA with the smallest structural ratio. It should be noted that for the same structural ratio, both *new random sweep* and *uniform choice* update scheme perform similar with the similarity being high for smaller structural ratios.

Figures 2.4, 2.5 and 2.6 show the effect of structural ratio on the selection pressure for the three update schemes.

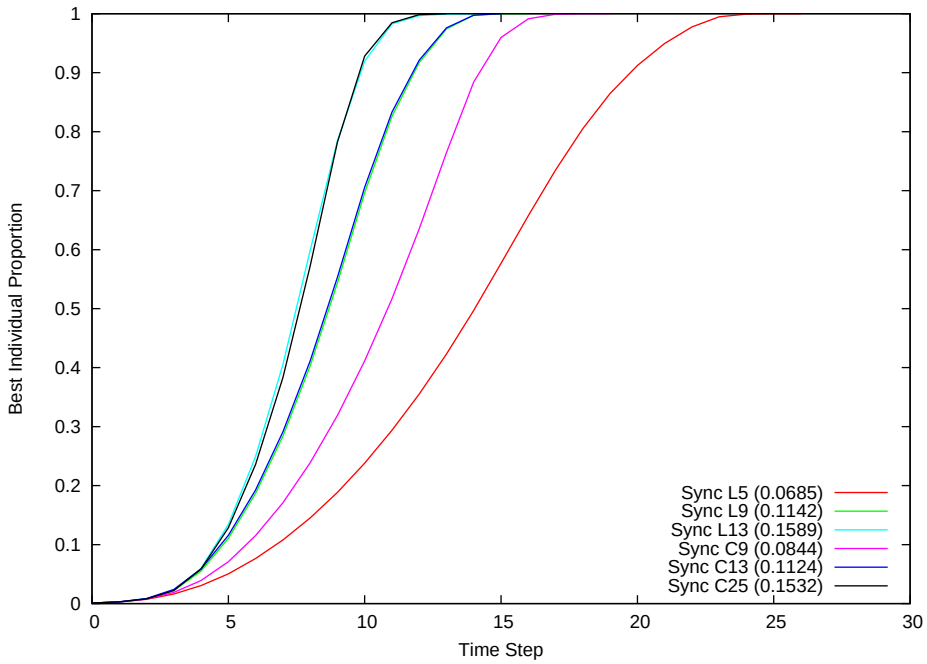

Figure 2.4: Effect of structural ratio in synchronous update.

In all three cases, neighborhood topologies resulting in similar structural ratio induce similar selection pressure. The neighborhood topology pairs $(L9, C13)$ and $(L13, C25)$ demonstrate this phenomena.

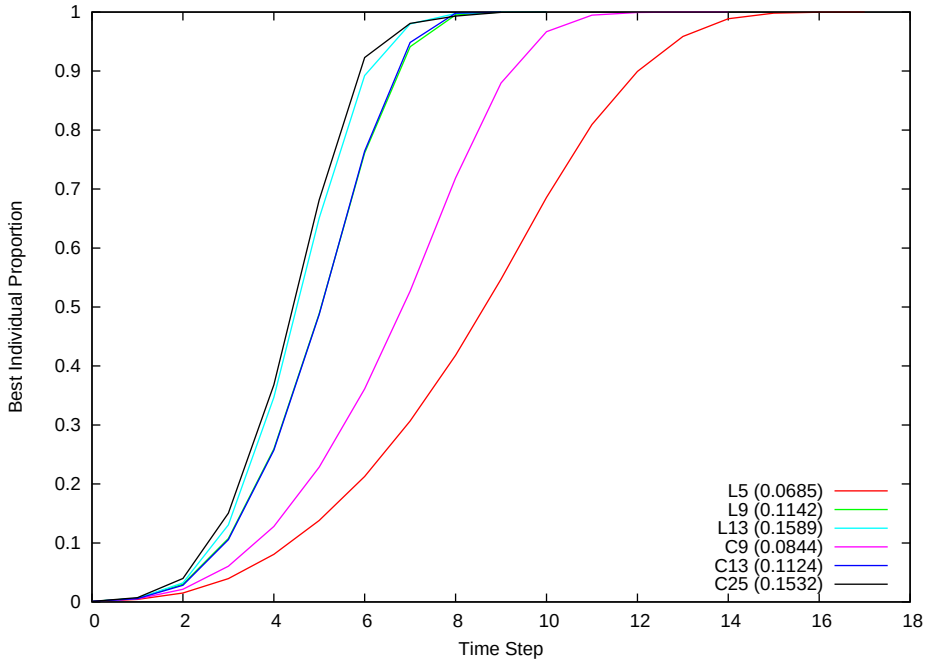

Figure 2.5: Effect of structural ratio in *new random sweep* asynchronous update.

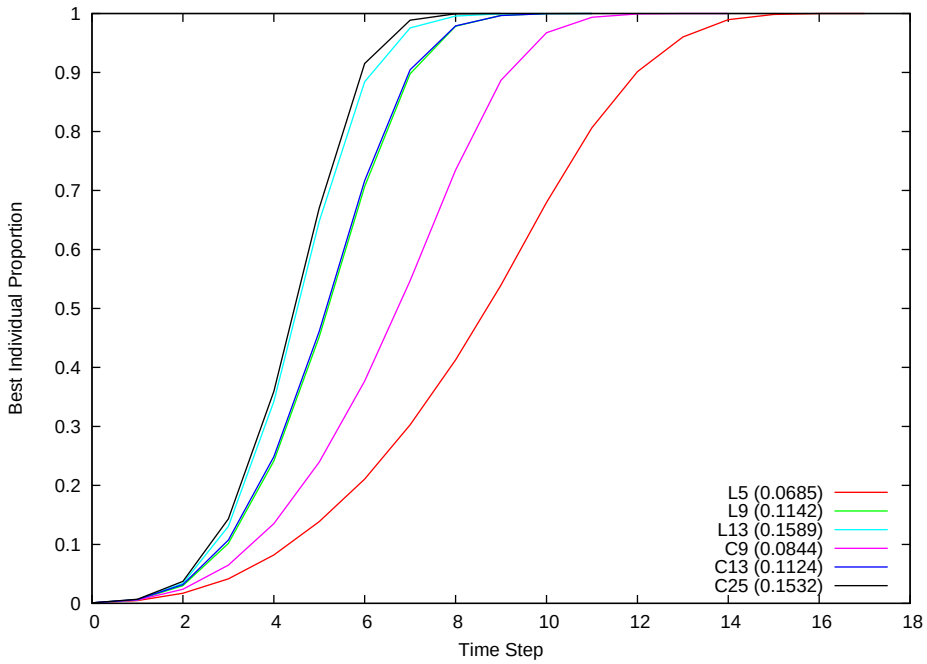

Figure 2.6: Effect of structural ratio in *uniform choice* asynchronous update.

3. MOBILE CELLULAR EVOLUTIONARY ALGORITHMS

Cellular EAs have fine grained population structure, with each individual placed on a vertex of a topology (say a 2-D torus). Selection and variation happens only in local neighborhoods defined by some topology (e.g., L5, C9, etc.). It should be noted that the number of vertices in the population topology is equal to the population size, i.e., no vertex is vacant. In order to introduce the notion of mobility into cEAs, this thesis introduces a new population structure where the number of vertices is greater than the population size, leaving some vertices vacant. This introduces a notion of population density into the algorithm. Individuals are free to move from their current location to another vacant vertex according to some *mobility function*. Such an algorithm with population density less than one and a *mobility function* which allows individuals to move around the population topology is termed as *Mobile Cellular Evolutionary Algorithm* (mcEA).

Algorithms 3.1 and 3.2 list the pseudocode for a typical synchronous cEA and synchronous mcEA, respectively.

Algorithm 3.1 Pseudocode for a synchronous cEA.

```
1:  GenerateInitialPopulation(pop)
2:  Copy(pop, auxpop)
3:  Evaluation(pop)
4:  while !StopCondition() do
5:    for individual ← 1 to popSize do
6:      neighbors ← CalculateNeighborhood(topology, Position(individual))
7:      parents ← Selection(neighbors)
8:      offspring ← Recombination(parents)
9:      offspring ← Mutation(offspring);
10:     Evaluation(offspring);
11:     Replacement(Position(individual), auxpop, offspring)
12:   end for
13:   Copy(auxpop, pop);
14: end while
```

Algorithm 3.2 Pseudocode for a synchronous mcEA.

```
1:  GenerateInitialPopulation(pop)
2:  Copy(pop, auxpop)
3:  Evaluation(pop)
4:  while !StopCondition() do
5:    for individual ← 1 to popSize do
6:      neighbors ← CalculateNeighborhood(topology, Position(individual))
7:      parents ← Selection(neighbors)
8:      offspring ← Recombination(parents)
9:      offspring ← Mutation(offspring);
10:     Evaluation(offspring);
11:     Replacement(Position(individual), auxpop, offspring)
12:     Movement(Position(individual), auxpop)
13:   end for
14:   Copy(auxpop, pop);
15: end while
```

Notice that the only difference between the two listings is the *Movement* operator which moves the individual from its current location in the population topology to another based on some predetermined/dynamic mobility function.

Asynchronous versions of these algorithms can be generated by removing the auxiliary population (auxpop) and allowing the *Replacement* and *Movement* operators to update the population (pop) for each individual before moving on to the next.

3.1. POPULATION DENSITY

As described previously, population density is defined as the ratio of the population size to the number of cells/vertices in the population topology.

For a population size of n and number of cells in the population topology equal to V, population density is given by

$$d = \frac{n}{V} \tag{7}$$

For the current investigation, each cell is constrained to be occupied by only one individual at a given time. This results in a population density which is strictly less than one (i.e., $n < V$) for any mcEA considered.

3.2. MOTION FUNCTION

For the mobility of individuals, a wide variety of motion functions can be defined. To keep the thesis to a reasonable length, some constraints are placed on the movement of the individuals and only a single motion function is investigated.

The following characteristics are considered for the current study:

- An individual can *move* into and *occupy* only a vacant cell. (Consequence of the constraint to keep the population density strictly less than one.)
- Only a single motion function, namely, *random-hop* is considered. According to this function, given a mobility radius, r_m, an individual may randomly move from the current location to any cell whose euclidean distance is within r_m to the current location, in a single hop.
- In each time step, an individual selects only one cells in the mobility radius to move to. If the cell is occupied, then the individual simply remains in its current location.

Note that the mobility radius, r_m, and the neighborhood radius, r_n are not necessarily the same.

Constraining the mobility to a single location selection each time step induces a bias towards the individual not moving. However, if multiple selections are allowed for the individuals to select a vacant location to move into, an additional parameter is introduced into the algorithm. Therefore, this constraint was placed, to avoid the influence of another parameter and interdependency with other parameters.

Algorithm 3.3 lists the pseudocode for the *random-hop* motion function.

3.3. SELECTION PRESSURE: EMPIRICAL STUDY

In this section, an empirical study is conducted to investigate the effect of population density and the selected motion function on the selection pressure in mcEAs.

Algorithm 3.3 Pseudocode for the *random-hop* motion function.

```
1: procedure Movement(individual, grid, r_m)
2:
3: currXLoc ← individual->xLoc
4: currYLoc ← individual->yLoc
5: locsInRadius ← GetLocationsInRadius(grid, currXLoc, currYLoc, r_m)
6: nextloc ← RandomInt(0, length(locsInRadius))
7: if !IsOccupied(grid, locsInRadius[nextLoc]) then
8:    individual->xLoc ← locsInRadius[nextLoc]->xLoc
9:    individual->yLoc ← locsInRadius[nextLoc]->yLoc
10: end if
```

3.3.1. Experimental Setup: Population Density. Table 3.1 lists the parameters used to study the effects of population density on the selection pressure of mcEAs, where n is the population size.

Table 3.1: Parameters used to investigate the effects of population density.

Grid size	32×32, 64×64
Neighborhood	L5, C9, L9, C13
Mobility function	*random-hop*
Mobility radius	1 (L5)
Parent selection	Central selection + uniform random
Replacement	*replace-worst-if-better*
Time step	n offsprings generated

3.3.2. Results: Population Density. Figures 3.1 and 3.2 show the effect of population density on the selection pressure of an mcEA for synchronous and *new random sweep* update schemes, respectively. Each curve in the plots is labeled as <algorithm> <selection neighborhood> <grid size> <(d=density)>.

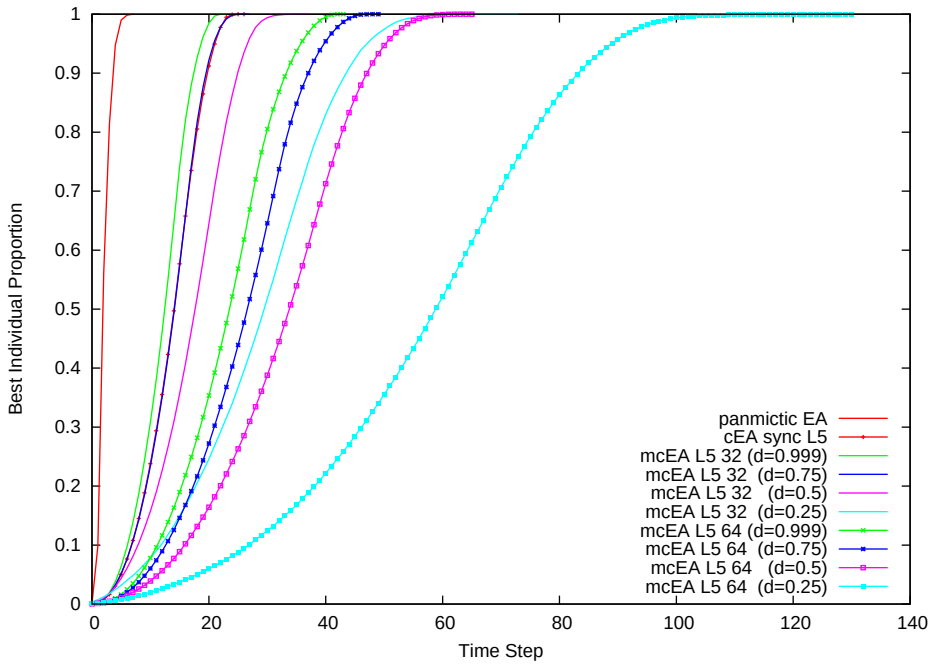

Figure 3.1: Effect of population density on synchronous mcEA.

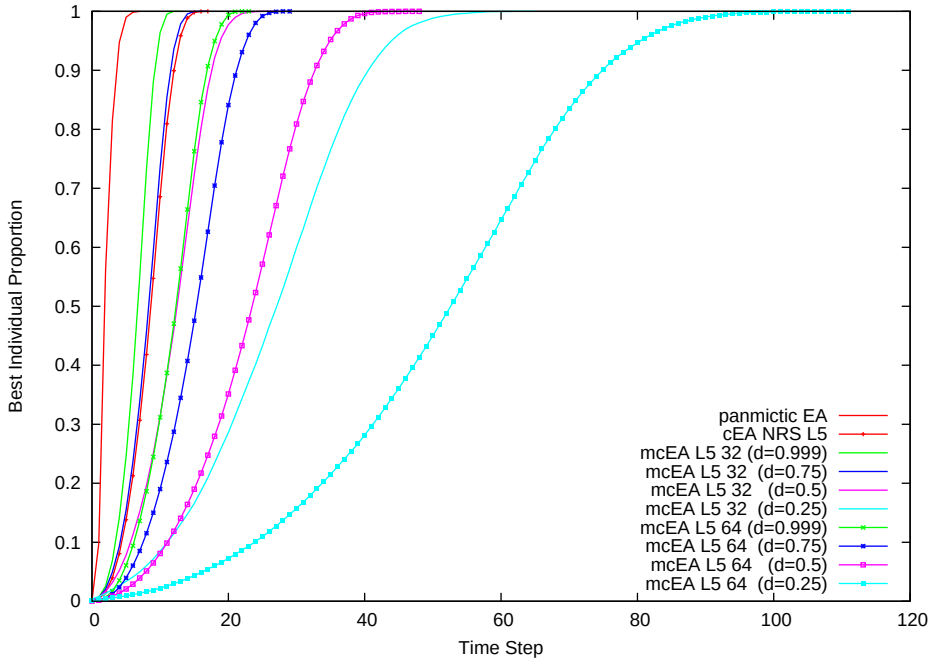

Figure 3.2: Effect of population density on asynchronous NRS mcEA.

In both cases, the selection pressure induced due to population density was always lower than that of a panmictic EA or cellular EA. It is interesting to note that the selection pressure for very high population density (d=0.999), was higher than that in cellular EAs. This is due to the fact that even with such a high population density few vacant cells are available (in case of d=0.999 only one vacant cell) for the individuals to move around. This small number of vacant cells together with a mobility of 1 facilitates migration from one neighborhood to another increasing the flow of good solutions to other neighborhoods. This is an indication of the effect of mobility on selection pressure.

With the decrease in population density, the selection pressure is lowered, due to the fact that a low mobility radius was used which requires the individuals to move for multiple generations in order to reach another neighborhood. However, it can be generally concluded from these results that for a given neighborhood, grid size and mobility radius, selection pressure is directly proportional to the population density, starting from slightly higher selection pressure than cEAs for high population density to low selection pressure for lower population densities.

Figures 3.3 and 3.4 show the effect of population density with respect to selection neighborhood for synchronous and *new random sweep* update schemes, respectively. The results follow the general trend described above. Additionally, as can be expected, a larger structural ratio increases the selection pressure in mcEA following the trend from cellular EA described in Section 2.4.2.

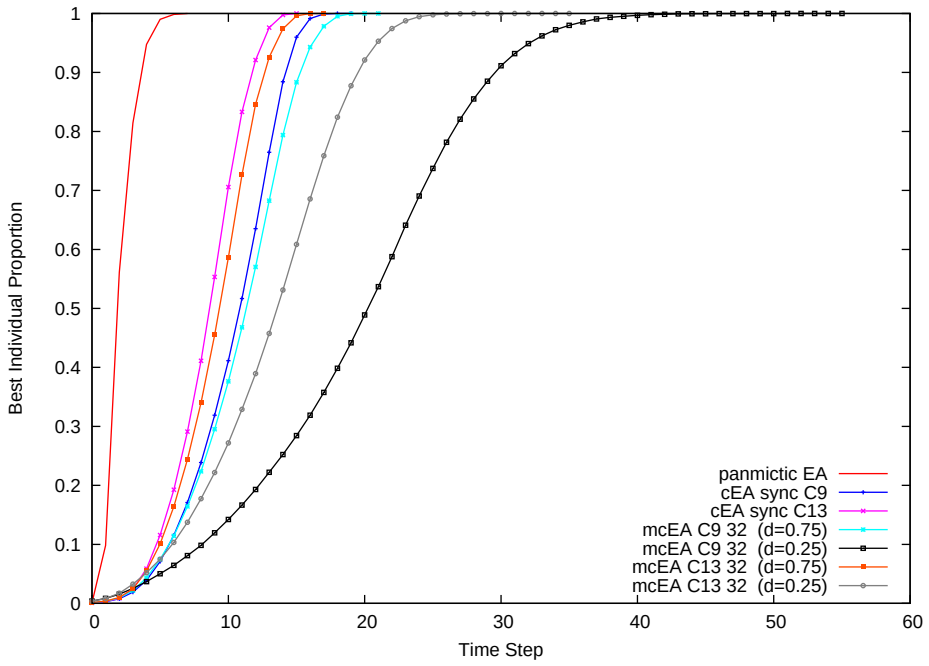

Figure 3.3: Effect of population density with selection neighborhood on synchronous mcEA.

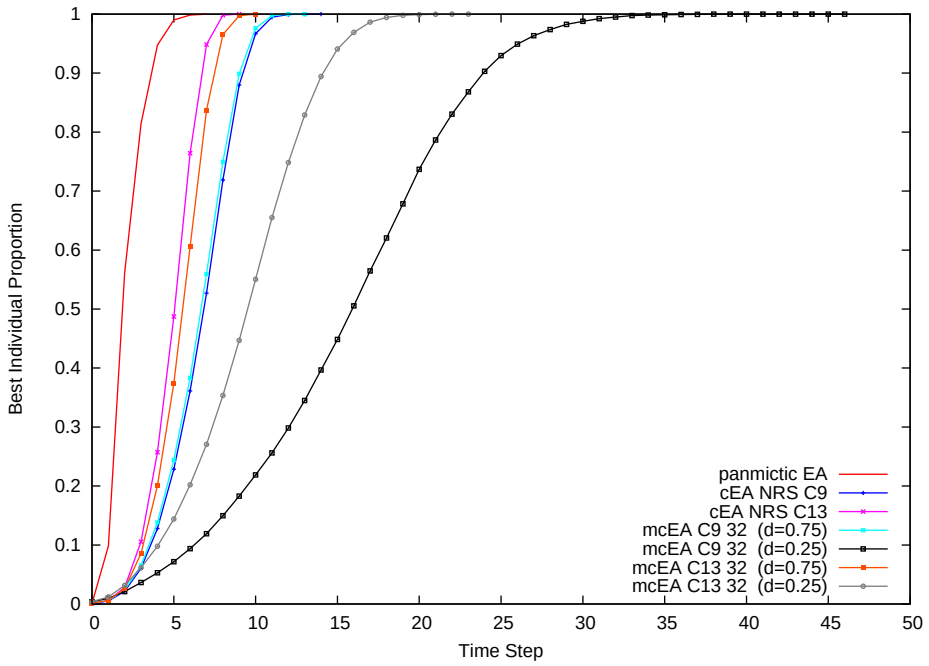

Figure 3.4: Effect of population density with selection neighborhood on NRS mcEA.

3.3.3. Experimental Setup: Mobility Function. Table 3.2 lists the parameters used to study the effects of mobility function on the selection pressure of mcEAs, where n is the population size. Figures 3.5 and 3.6 show the effect of mobility

Table 3.2: Parameters used to investigate the effects of mobility function.

Grid size	32×32, 64×64
Neighborhood	C9, C13
Mobility function	*random-hop*
Mobility radius	$1, 2, 3, 4, 8, 6, 13$
Parent selection	Central selection + uniform random
Replacement	*replace-worst-if-better*
Time step	n offsprings generated

radius on selection pressure of an mcEA for synchronous update scheme with $C9$ and $C13$ selection neighborhoods.

In both the cases a clear general trend is visible. Lower mobility induces lower selection pressure due to slower "mixing" of individuals from different neighborhoods and higher mobility induces higher selection pressure due to faster "mixing" of individuals from different neighborhoods. However, it is interesting to note that when the mobility radius is equal to the grid side (in case of 32×32 grid), the selection pressure induced is significantly lower than that in panmictic EA of the same population size. This is due to the two characteristics of the mcEA used. Firstly, although individuals can move to almost anywhere in the grid, selection and reproduction still happen within individuals of a small deme size, namely, $C9$. Secondly, *random-hop* mobility function does not guarantee that the individual will move the distance equal to the mobility radius. An individual selects a vacant cell within the mobility radius to move. These two characteristics restrict the global mixing of individuals to much less than that in panmictic EA. Also it should be noticed that for all mobility radii, the selection pressure induced is greater than that induced by a synchronous cEA with the same grid size and selection neighborhood.

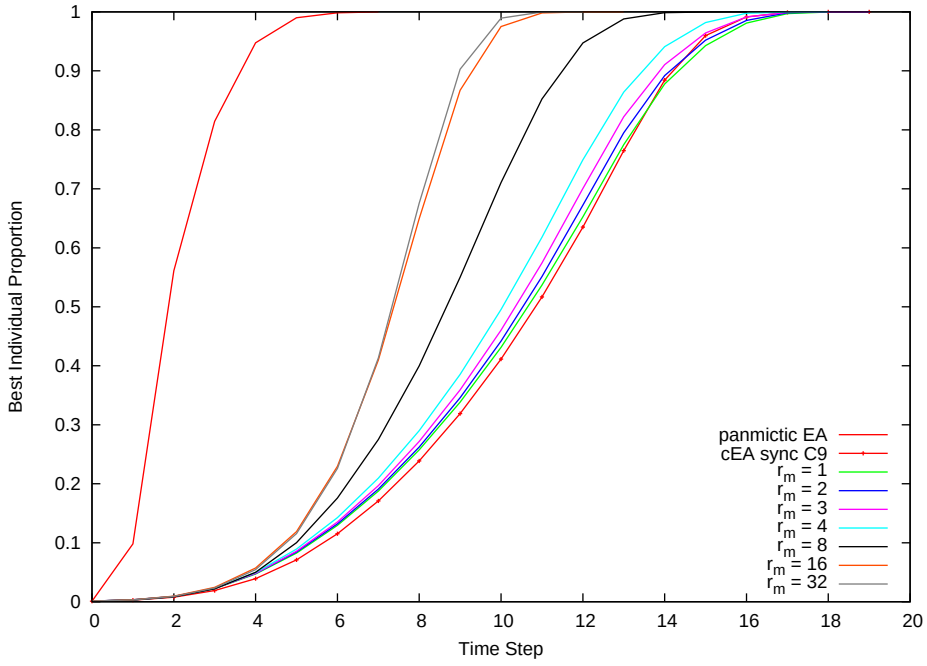

Figure 3.5: Effect of mobility radius on selection pressure in a synchronous mcEA with grid size of 32×32, population density 0.9 and selection neighborhood of $C9$.

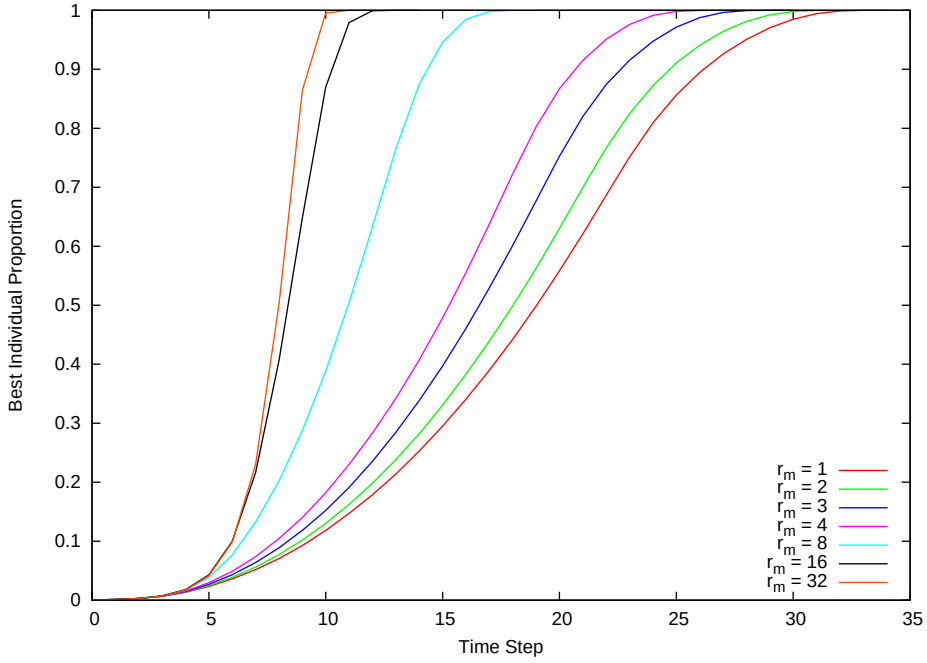

Figure 3.6: Effect of mobility radius on selection pressure in a synchronous mcEA with grid size of 64×64, population density 0.5 and selection neighborhood of $C13$.

Figures 3.7 and 3.8 show the effect of mobility radius on selection pressure of an mcEA for *new random sweep* update scheme with $C9$ and $C13$ selection neighborhoods.

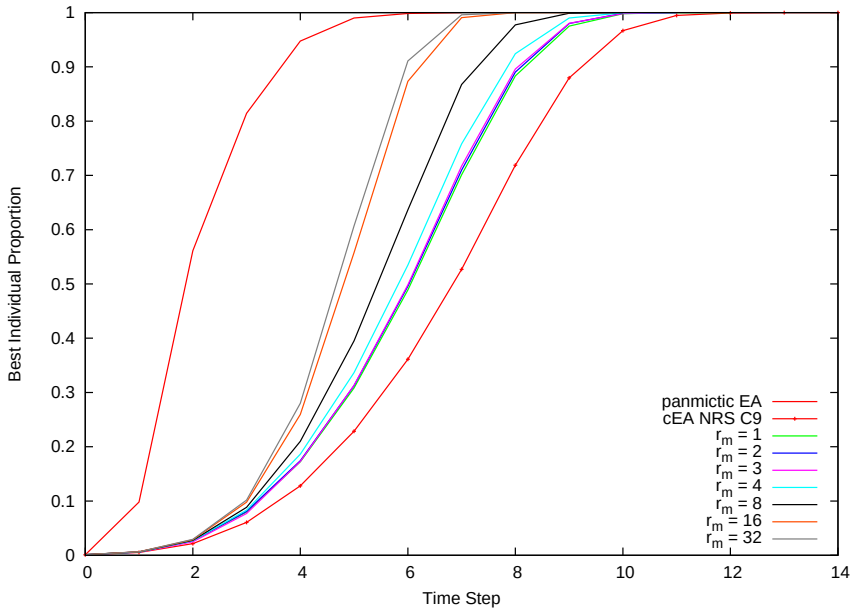

Figure 3.7: Effect of mobility radius on selection pressure in an asynchronous NRS mcEA with grid size of 32×32, population density 0.9 and selection neighborhood of $C9$.

Even in the case of NRS update scheme, the general trend is similar to that of synchronous case. Again in the case with 32×32 grid, the selection pressure induced by mcEA for all mobility values is lower than that of panmictic EA and greater than that of cEA.

3.4. EXPERIMENTS WITH TEST PROBLEMS

To analyze the performance of mcEA in optimization tasks, experiments were conducted on a test suite of different classes of problems. In this section the test problems used and the results from the experiments are presented.

3.4.1. Test Suite. For the current study, three problems, namely, massively multimodal deceptive problem (MMDP), frequency modulation sounds (FMS) and

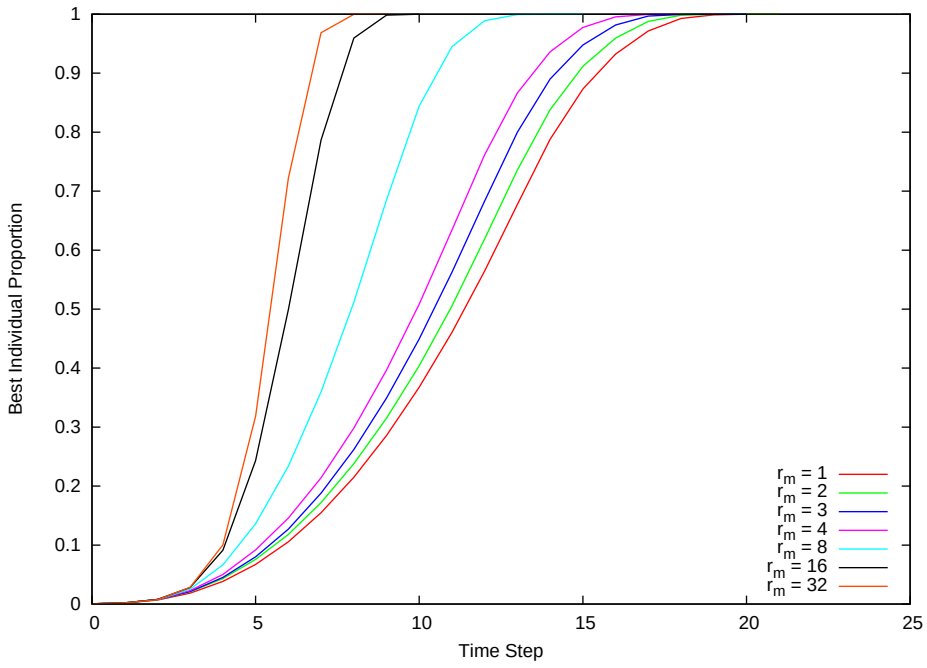

Figure 3.8: Effect of mobility radius on selection pressure in an asynchronous NRS mcEA with grid size of 64×64, population density 0.5 and selection neighborhood of $C13$.

problem generator, P-PEAKS, are used to investigate the performance of mcEAs. These problems represent the three classes of problems which are generally considered difficult for evolutionary computation. These classes are deception, multimodality and epistasis. All the problems are combinatorial in nature. While FMS is a minimization task, MMDP and P-PEAKS are maximization problems. The problems are explained in detail in Appendix A.

3.4.2. Experimental Setup. Table 3.3 shows the parameter values used for the experiments. The parameters are selected as follows to facilitate comparison with agent-based EAs developed in the later sections.

A selection neighborhood of $C9$ was used as it is one of the most popular neighborhoods used in both cEAs and agent-based EA. Two update schemes, namely, synchronous and NRS were used. While synchronous update scheme is popular in cEAs, NRS is very common in agent-based EAs. Although it is not explicitly noted

Table 3.3: Parameters used for the test suite.

Grid size	32×32
Update scheme	synchronous, NRS
Neighborhood	C9
Population density	$0.9, 0.75, 0.5$
Mobility function	*random-hop*
Mobility radius	$1, 2, 3, 4, 8, 16$
Parent selection	Central selection + uniform random
Recombination	DPX (double point crossover)
Mutation	Bit-flip mutation
Replacement	*replace-worst-if-better*
Time step	n offsprings generated
Chromosome length (L)	240 (MMDP) 192 (FMS) 100 (P-PEAKS)
Probability of crossover (P_c)	0.9
Probability of mutation (P_m)	$1/L = 0.0042$ (MMDP) $10/L = 0.052$ (FMS) $1/L = 0.01$ (P-PEAKS)

in most agent-based EAs existing in literature, the common update scheme is follows that in each time step, individuals in random order with respect to one another execute their behavior set. This is exactly what NRS achieves. Parent selection was restricted to central selection (the individual representing the current grid cell being updated) and a random individual in the selection neighborhood. Although this is quite uncommon in cEAs, where tournament selection is generally used, it is similar to decentralized mate selection in agent-based EAs where each individual, selects a mate by itself to reproduce. Double-point crossover was selected due to the simplicity of the recombination scheme. It is a popular crossover scheme along with one-point crossover for binary encoded EAs and is commonly used in both panmictic and cellular EAs. However, very few agent-based EAs exist which have attempted to solve binary encoded optimization problems and therefore the commonality of this recombination scheme cannot be claimed. The replacement scheme used, replaces the worst individual in the neighborhood by the offspring produced if the offspring is better than the said worst individual. This is a popular replacement scheme in cEA.

Time to convergence of the best and average fitness is measured in *generations* than in number of fitness evaluations which is commonly used in evolutionary computation, to facilitate comparison with agent-based EAs. In the current study, a single generation is said to have passed when each individual in the population has had a chance to reproduce. In cellular EAs, this results in the generation of number of offsprings equal to the size of the population. In mcEAs, however, when no individuals are present in the neighborhood of the current individual, reproduction does not happen. Therefore in the case of mcEAs, in a generation, the number of offspring generated is less than or at most equal to the number of offsprings generated in the cellular EA case. However, the main aim of this thesis is to provide a characterization of mcEAs and agent-based EA and not to discuss the actual performance of these algorithms. Surely, better variational operators for the problems considered in specific and mobile cellular EAs in general can be developed to suit the behaviors of the algorithms. Therefore, reporting *generations* as the time-to-solution is justified. Also the algorithms are terminated when the average fitness of the population is equal to best fitness of the population within an error of 10^{-5}. This is done to include the takeover time of the best individual in the presence of variational operators. This provides a much better estimate of the time for the population to converge to a solution (whether local or global optimal) and is strong indicator of the selection pressure induced by the algorithm. However, in the case of massively multimodal deceptive problem (MMDP), the algorithm is also terminated if the number of generations exceed 30,000 generations. The MMDP search landscape consists of a large number of local optimal solutions and population convergence to a single peak within the error margin resulted in finding the global optimal solution each and every time. To reduce this effect, the limit of 30,000 generations was empirically selected. The values for probability of recombination and mutation were the values most commonly used in literature for the problems considered.

3.4.3. Results. Table 3.4 shows the average time to convergence (generations) in 100 runs on each problem. The labels for mcEA results are generated as "<s, n>_<density>_<mobility>", with an "*s*" in the second field indicating synchronous update scheme and an "*n*" indicating NRS update scheme.

Table 3.4: Average number of generations on each problem.

Algorithm	FMS	MMDP	P-PEAKS
cEA sync	182.55 ± 34.578	30000 ± 0.000	74.29 ± 6.293
s_0.9_1	175.92 ± 33.805	30000 ± 0.000	73.81 ± 6.019
s_0.9_2	180.86 ± 38.507	30000 ± 0.000	73.37 ± 5.705
s_0.9_3	171.59 ± 32.804	30000 ± 0.000	72.11 ± 5.474
s_0.9_16	162.90 ± 28.358	30000 ± 0.000	66.32 ± 4.183
s_0.75_1	174.99 ± 30.107	30000 ± 0.000	75.74 ± 6.933
s_0.75_2	173.32 ± 35.009	30000 ± 0.000	73.04 ± 5.224
s_0.75_3	177.37 ± 37.567	30000 ± 0.000	72.06 ± 6.058
s_0.75_16	161.22 ± 25.733	30000 ± 0.000	63.69 ± 3.805
s_0.5_1	188.28 ± 39.588	30000 ± 0.000	82.15 ± 7.235
s_0.5_2	190.68 ± 52.130	30000 ± 0.000	78.86 ± 7.654
s_0.5_3	181.43 ± 38.113	30000 ± 0.000	73.84 ± 6.888
s_0.5_16	163.00 ± 29.005	30000 ± 0.000	66.68 ± 4.431
cEA NRS	138.96 ± 30.944	30000 ± 0.000	55.12 ± 5.174
n_0.9_1	158.30 ± 44.187	30000 ± 0.000	53.80 ± 2.740
n_0.9_2	123.81 ± 24.765	30000 ± 0.000	55.83 ± 7.208
n_0.9_3	130.51 ± 17.348	30000 ± 0.000	57.65 ± 4.442
n_0.9_16	128.23 ± 17.180	30000 ± 0.000	49.28 ± 2.440
n_0.75_1	134.30 ± 16.364	30000 ± 0.000	55.90 ± 3.932
n_0.75_2	139.49 ± 21.365	30000 ± 0.000	55.23 ± 2.858
n_0.75_3	136.81 ± 22.710	30000 ± 0.000	55.10 ± 4.598
n_0.75_16	126.84 ± 12.664	30000 ± 0.000	48.41 ± 2.980
n_0.5_1	152.92 ± 29.524	30000 ± 0.000	65.12 ± 4.254
n_0.5_2	156.11 ± 24.837	30000 ± 0.000	61.45 ± 6.003
n_0.5_3	135.33 ± 14.682	30000 ± 0.000	61.33 ± 7.087
n_0.5_16	138.59 ± 39.483	30000 ± 0.000	51.30 ± 3.888

The following general trends which closely agree with the trends in selection pressure study can be observed from the average number of generations to solution:

- *New random sweep* update scheme induces higher selection pressure on the individuals than *synchronous* update scheme, reducing the time to convergence.
- For the same update scheme higher population densities in mcEA facilitates faster convergence than lower population densities. It can be noticed that for population densities 0.9 and 0.75 the average number of generations in mcEAs is lower than that in synchronous cEA, and for lower population density, namely,

0.5, the average number of generations is higher than that of the synchronous cEA.

- Following the trend in selection pressure study. Higher mobility induces a high selection pressure on individual thereby reducing the time to convergence than with lower mobility.

Table 3.5 shows the best, worst and mean fitness of 100 runs with the synchronous update scheme.

The following observations can be made from the quality of solutions with the synchronous update scheme:

- On the FMS problem, the best solutions was found by mcEA with density 0.9. On an average, the best solution was found by mcEA with density 0.9 and mobility radius 3, followed by the cellular EA. Although in terms of best solutions, other configuration of mcEA also performed well, their average performance was worse than these two cases. It is interesting to note that these two configurations induced the highest selection pressure of all the configuration. In fact, configurations with lower selection pressure are expected to perform better on the FMS problem due to its strong epistatic nature, however, from the results it appears that the selection pressure induced by lower population densities is too low to facilitate sufficient exploitation of the fitness landscape by the recombination operator.

- On the MMDP problem, all the configurations were able to find the global optimal solution. However, in the average case the best performance was achieved by mcEA with density 0.75. It appears that the exploration-to-exploitation trade off was well-balanced for this configuration. However, it should be noted that the number of generations expended for the MMDP problem was 30000, which is very high in terms of search effort. This high search effect could be the reason for all configurations to find the global optimal solution. However, even at the end of this massive search effort, no configuration achieved the primary stopping condition of low difference in average and best fitness of the population. This shows that all the configurations were able to maintain a good

population diversity on one of the representative problems of the deceptive class of problems.

- No interesting observations can be made from the P-PEAKS problem. All configurations including synchronous cEA were able to find the global optimal solution, in 100% of the runs. This shows that the medium-high epistatic instance selected for the study was not as strong as the epistatis in the FMS problem case.

Table 3.6 shows the best, worst and mean fitness of 100 runs with the asynchronous *new random sweep* update scheme.

The following observation can be made from the quality of solutions with the NRS update scheme:

- Although all the configuration found good solutions on the FMS problem, no configuration of mcEA performed better than the cellular EA in the average case. However, the best solution was found by mcEA with density 0.5 and mobility radius 2, which represents a very low selection pressure.

- On the MMDP problem, low density configuration ($d = 0.5$), performed the best. It is interesting to not that while in the synchronous case, low density configuration performed poorly, they performed better with the NRS update scheme. This is due to the fact that when using NRS update scheme, the selection pressure of the configuration is increased, which improves the performance of lower density configurations.

- Again no iteresting observation can be made from the P-PEAKS problem as all the configuration could find the global optimal solution in 100% of the runs.

In general, on the test problems considered, one or the other configuration of mcEA performed on par with cellular EAs. This strongly supports the primary claim of this thesis that an mcEA can be used as a highly tunable evolutionary algorithm in which the parameters can be appropriately tuned for a particular problem.

Table 3.5: Quality of solution on each problem using synchronous update scheme.

Algorithm	FMS			MMDP			P-PEAKS		
	best $\times(10^{-6})$	worst	mean	best	worst	mean	best	worst	mean
cEA sync	1.581	17.430	6.285 ± 5.917	40.000	39.001	39.729 ± 0.251	1.000	1.000	1.000 ± 0.000
s_0.9_1	0.099	18.397	8.379 ± 5.982	40.000	39.641	39.989 ± 0.061	1.000	1.000	1.000 ± 0.000
s_0.9_2	0.022	20.800	7.519 ± 6.322	40.000	39.125	39.456 ± 0.054	1.000	1.000	1.000 ± 0.000
s_0.9_3	0.013	17.099	5.819 ± 5.396	40.000	39.641	39.996 ± 0.035	1.000	1.000	1.000 ± 0.000
s_0.9_16	13.708	18.462	7.337 ± 6.056	40.000	39.023	39.443 ± 0.033	1.000	1.000	1.000 ± 0.000
s_0.75_1	1.756	17.714	8.265 ± 5.880	40.000	40.000	40.000 ± 0.000	1.000	1.000	1.000 ± 0.000
s_0.75_2	0.019	19.650	8.362 ± 6.083	40.000	40.000	40.000 ± 0.000	1.000	1.000	1.000 ± 0.000
s_0.75_3	4.402	18.725	7.195 ± 6.613	40.000	39.641	39.964 ± 0.108	1.000	1.000	1.000 ± 0.000
s_0.75_16	1.806	18.717	6.797 ± 6.270	40.000	39.331	39.932 ± 0.133	1.000	1.000	1.000 ± 0.000
s_0.5_1	0.048	23.284	9.713 ± 6.749	40.000	39.360	39.864 ± 0.225	1.000	1.000	1.000 ± 0.000
s_0.5_2	82.108	19.604	8.732 ± 6.656	40.000	39.641	39.928 ± 0.147	1.000	1.000	1.000 ± 0.000
s_0.5_3	14.252	21.533	9.307 ± 6.822	40.000	39.466	39.964 ± 0.110	1.000	1.000	1.000 ± 0.000
s_0.5_16	0.046	19.908	8.360 ± 6.570	40.000	39.332	39.233 ± 0.231	1.000	1.000	1.000 ± 0.000

Table 3.6: Quality of solution on each problem using NRS update scheme.

Algorithm	FMS			MMDP			P-PEAKS		
	best $\times(10^{-6})$	worst	mean	best	worst	mean	best	worst	mean
cEA NRS	14.715	17.940	5.392 ± 5.837	40.000	39.001	39.762 ± 0.259	1.000	1.000	1.000 ± 0.000
n_0.9_1	64.658	15.190	6.169 ± 5.927	40.000	39.201	39.964 ± 0.110	1.000	1.000	1.000 ± 0.000
n_0.9_2	91.963	20.360	7.566 ± 6.220	40.000	39.332	39.854 ± 0.052	1.000	1.000	1.000 ± 0.000
n_0.9_3	54.665	18.052	6.717 ± 6.070	40.000	39.151	39.993 ± 0.074	1.000	1.000	1.000 ± 0.000
n_0.9_16	54.863	17.630	5.687 ± 6.226	40.000	39.641	39.966 ± 0.090	1.000	1.000	1.000 ± 0.000
n_0.75_1	29.511	16.485	9.698 ± 6.294	40.000	40.000	40.000 ± 0.000	1.000	1.000	1.000 ± 0.000
n_0.75_2	53.546	18.658	11.069 ±7.156	40.000	40.000	40.000 ± 0.000	1.000	1.000	1.000 ± 0.000
n_0.75_3	92.006	11.843	5.408 ± 4.802	40.000	39.345	39.766 ± 0.124	1.000	1.000	1.000 ± 0.000
n_0.75_16	31.153	15.849	8.851 ± 6.826	40.000	39.442	39.965 ± 0.113	1.000	1.000	1.000 ± 0.000
n_0.5_1	23.165	19.168	11.295 ±6.555	40.000	39.630	39.952 ± 0.245	1.000	1.000	1.000 ± 0.000
n_0.5_2	1.408	17.625	5.850 ± 4.425	40.000	39.820	39.973 ± 0.127	1.000	1.000	1.000 ± 0.000
n_0.5_3	93.027	20.818	9.596 ± 7.236	40.000	39.532	39.930 ± 0.227	1.000	1.000	1.000 ± 0.000
n_0.5_16	88.572	18.772	8.987 ± 7.721	40.000	39.230	39.645 ± 0.185	1.000	1.000	1.000 ± 0.000

4. AGENT-BASED EVOLUTIONARY ALGORITHMS

Recently, there has been a growing interest in the use of agent-based models (sometimes called multi-agent systems) for evolutionary computation. Agent-based models follow a decentralized computing approach with traditional roots in artificial life (ALife) models. While ALife models are used to study evolution in a synthetic environment, agent-based EAs use the evolution of the digital organisms for solving optimization problems. In this section a brief survey on some popular models is presented. While most of the agent-based EAs in existence are aimed at multi-objective optimization, few models for function optimization also exist.

In one of the earliest seminal works on agent-based EAs, Laumanns et al. (1998) presented a spatial predator-prey approach for multi-objective optimization. The model consists of a 2-D toroidal grid, with preys representing one possible decision vector placed on each of the vertex. Predators are sparsely placed on the same grid. Predators move around the grid and evaluate each prey with respect to a single specific objective. In each iteration, a predator will catch the worst prey in the neighborhood of its current location and kill it. The vacant vertex is then filled by recombination of the individuals adjacent to it. Since each predator kills prey with respect to only one objective over time only the prey which have survived from all the predators exist in the grid. These prey represent the Pareto-optimal solutions since they survived the "evaluation" of all the predators. They demonstrated the effectiveness of this scheme on a number of test problems and noted that agent-based EAs represent an alternative approach in multi-objective optimization.

Deb (2001) and later Grimme and Schmitt (2006) revised the predator-prey model with a few modifications. Firstly, the restriction of one predator to one objective assignment was replaced with each predator selects the worst prey with respect to a weighted sum of all objective, allowing each predator to steer the prey population to a specific region on the Pareto-front. Secondly, a weighted intermediate recombination operator was specifically designed for reproduction in multi-objective optimization. With these changes, improved results were reported.

Ursem (1999) presented a multinational evolutionary algorithm for finding multiple solutions, both global optimal and local optimal solutions, in the fitness landscape. Inspired by the political interactions of nations in the real world, the algorithm consists of individuals from multiple nations living in the search space. While the algorithm itself is not spatially structured, individuals form nations in the search space. With rules for *migration* from one nation to another, *merging* two nations and *mating* among individuals of the same nation, the algorithm is highly dynamic with a large number of tunable parameters. The algorithm was evaluated using several test functions and was reported to have found more optimal (global and local) solutions than even with fitness-sharing which is commonly used to preserve diversity among individuals of an EA.

Thomsen et al. (2000) later proposed a religion based spatial model for function optimization. In this algorithm, each individual in a 2-D torus belongs to one of several religions. In each time step, an individual randomly walks to a new location and tries to convert individual of another religion to his own. Conversion is based on the fitness of the individuals. Since mating is restricted among individuals of the same religion, religions with individuals with better fitness increase in number while worse fitness individuals are reduced. Experiments with this model were conducted using several test function and compared against panmictic and cellular EAs. It was reported that on most of the test problems, the religion-based method performed far better than either of the other algorithms.

Socha and Kisiel-Dorohinicki (2002) presented a evolutionary multi-agent system for multi-objective optimization. In the algorithm, autonomous agents are placed on a 2-D toroid and move around in the grid. Each agent is associated with some energy level of *life energy*. Each behavior in the agent's behavior set require some energy to execute with reproduction requiring highest energy. An agent dies when it runs out of energy. During each time step, an agent initiates communication with another agent and requests the quality of solution with respect to each objective. The second then responds with its quality. If either of the agents is dominated, the dominated agent transfers some energy to the dominant agent. This way better solutions (representing the Pareto front) receive energy from other agents and reproduce while dominated agents lose energy and ultimately die due to lack of energy. The

algorithm was evaluated on a number of multi-objective problems and it was reported that results were promising. However, do lack of any mechanism to avoid crowding not all of the Pareto-front was covered. This was noted a possible future research direction.

Berry and Vamplew (2003, 2005) presented the combative accretion model for multi-objective optimization. In this model, agents moved around on 2-D torus and combat among each other. In each time step, an agent fights one of the other agents in its neighborhood. Pareto-dominance based comparison between the two agents is made and if one of the dominated, the size of the dominant agent is increased and the dominated agent is decreased. If neither one is dominated, then both their sizes are increased and if the individuals are equal, one of them is killed by a global process. If the size of an agent falls below a certain threshold the agent dies. When an agent is killed by a dominant agent by causing it to fall below threshold, the dominant agent is said to have contributed in reproduction. When an agent is killed, it is replaced by a new individual created from an *accretion pool* of genetic material. An agent is removed from the environment and placed in the accretion pool in two cases, either when the size of the agent crosses a certain threshold or when it has contributed to reproduction a set number of times. The algorithm was compared against some of the best panmictic algorithms for multi-objective optimization and was found to produce comparable Pareto-fronts with lower number of fitness evaluations (lower computational effort).

Amato and Farina (2005) presented an ALife inspired spatial evolutionary algorithm for dynamic multi-objective optimization. In this algorithm, agents more around on a 2-D torus. In each time step, an agent *meets* with another agent with a dynamic probability. If the agent does not meet with any other agent, the agent undergoes a uni-sexual reproduction to spawn an offspring. With two agents *meet* they can either fight based on Pareto-dominance or undergo bisexual reproduction to produce an offspring. Interestingly a population density based dynamic population sizing is used to determine the probability of *meeting* at each time step. However, this is not locally calculated, as parameters are usually calculated in agent-based EAs, but a global function sets this probability to the ratio of current population size to the desired population size. The probability of bisexual reproduction is set as the

probability of not meeting. This population sizing scheme was shown to generate a stable equilibrium population size.

Drezewski and Siwik (2008) presented a single species agent-based algorithm and a host-parasite based algorithm for multi-objective optimization. Both the algorithms utilize the notion of energy and Pareto-dominance based *energy-sharing* similar to other algorithms described above. It was observed that while the single species based algorithm did poorly, the host-based algorithm performed comparable to existing panmictic multi-objective optimization algorithms requiring lower number of fitness evaluations.

While certainly some interesting works and appreciable results have been presented in agent-based evolutionary algorithms (abEAs), most of the algorithms discussed here are developed in an ad-hoc fashion with large number of parameters and irrelevant ideas. Although, the field is rather young when compared to evolutionary computation itself, no concrete justification has been presented for choosing particular values for parameters in many of these works. Even when presented, the parameter values can only be justified for a small set/class of problems and cannot be easily tuned for other problems.

This thesis aims to provide a baseline for agent-based algorithms by investigating the two important aspects of agent-based algorithms, namely, mobility and bottom-up evolutionary operators and dynamic parameter tuning. Bottom-up indicates that information available from local neighborhoods is used to execute local operators which result in a globally observable phenomena. While mobility was studied in the form of mobile cellular EAs, one evolutionary parameter, namely offspring sizing is investing in the following sections in the form of bottom-up population dynamics.

5. POPULATION DYNAMICS FOR AGENT-BASED ALGORITHMS

As discussed in Section 4, evolutionary operators such as selection, recombination, and mutation are incorporated into an individual's behavior set which are executed utilizing local information gathered by the individual. A common characteristic of agent-based EAs is that the population size is dynamic and does not remain constant across all iterations. In this section two bottom-up population dynamics schemes are presented which mimic the popular survival schemes in evolutionary computation. The first, called Age-Based Population Dynamics (ABPD) is a non-elitist scheme in which individuals are replaced proportional to their age. The second is an elitist scheme called Combative Population Dynamics (CPD) in which the best individual is never eliminated/killed. While both elitist and non-elitist schemes are commonly used in evolutionary computation, ABPD is a typical example of ALife inspired schemes.

5.1. AGE-BASED POPULATION DYNAMICS

In age-based population dynamics (ABPD), dynamic offspring size is determined by each individual based on the density of individuals it has experienced throughout its lifetime. The following parameters are introduced into the algorithm to accomplish ABPD.

Let N_i and N be the initial population size, i.e., population size at time step zero and the desired equilibrium population size, respectively. Let r be the interaction radius (selection radius) in which represents the local neighborhood of the individuals. I_a denotes the number of cell (area) in the interaction radius. A new parameter termed *probability of death* (P_d) is introduced, which denotes the finite probability that an individual would die in any time step. As a consequence the average life time of an individual would be $1/P_d$. As mentioned earlier, the concept of probability of death makes this scheme non-elitist wherein the best individual even if found, is not guaranteed to survive till the end of the simulation. However, due to this constant death and reproduce cycle, higher population diversity is maintained which aids in solving highly multimodal and epistatic problems.

The desired density of the population is assumed to be known to all individuals due to genetic memory and is given by

$$G_d = \frac{N}{S^2} \tag{8}$$

where S^2 is the area (number of cells) in the environment.

Since an individual can only interact within its neighborhood, in the ABPD scheme, each individual keeps a moving average estimate of the local population density around it. This estimated density, d for each individual each given by

$$d_t = d_{t-1} + \lambda(O_t - d_{t-1}) \tag{9}$$

where d_t and d_{t-1} are estimated population densities at time t and $t-1$, respectively. Here O_t is the directly observed density at the time instance t and λ is the confidence coefficient for the directly observed density. Since ALife systems are inherently noisy due to complex inter-dependencies between stochastic components,this confidence coefficient is usually kept low (~ 0.2)

Since P_d is the individual probability of death, the following information is locally available to each individual:

- The current deviation in local population size is given by $G_d I_a - d_t I_a$
- The average number of natural deaths in the local population is equal to $d_t I_a P_d$
- The estimated number of survivors to the next time step is equal to $d_t I_a - d_t I_a P_d$

Therefore, the per-capita offsprings needed is given by

$$\hat{b} = \frac{G_d I_a - d_t I_a + d_t I_a P_d}{d_t I_a - d_t I_a P_d} \tag{10}$$

simplifying and rearranging terms

$$\hat{b} = \frac{G_d}{d_t(1 - P_d)} - 1 \tag{11}$$

Since G_d is calculated as a constant and P_d is an input, the offspring size is only inversely proportional to the estimated density d_t, during the course of the simulation.

Figure 5.1 shows the stability of ABPD scheme for various initial population sizes and a desired population size of 1000 individuals.

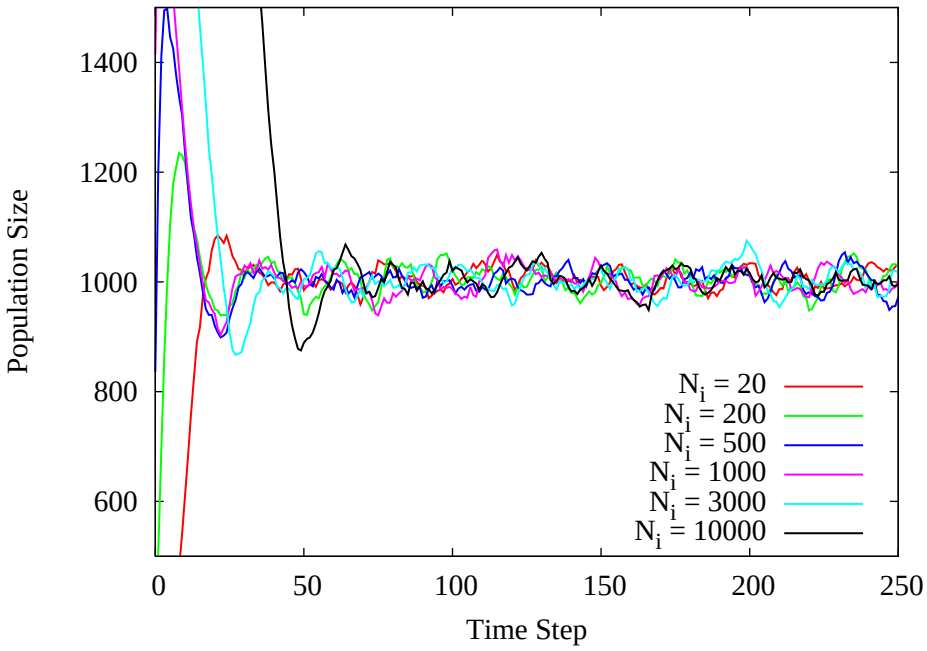

Figure 5.1: ABPD convergence for various initial population sizes. The desired population size is 1000.

The population size quickly reaches and stays around the desired population density. The overshoot and undershoot artifacts in the figure are due to the fact that only first order relationships between the variables are considered here. A more non-linear model which takes into account the per-capita changes in death rate due to one individual's death would result in a much smoother albeit slower response to the density change.

5.2. COMBATIVE POPULATION DYNAMICS

In this elitist scheme, an additional behavior, *combat*, is added to the behavior set of the individuals. In each iteration, each individual based on the observed population density fights one of the individuals in its interaction radius. The outcome of

each battle is that the weaker (less fit) individual is killed. The winner increments a counter, N_k of offsprings to be produced to compensate for the loss. If the fitness of both individuals is the same, then the combat ends in a draw and no individual dies. An individual only fights another individual if its observed population is greater than the desired density (genetic density). Notice that in this scheme not only does the individual with best fitness never die, but will produce more offsprings during the course of the simulation than others.

Following the discussion for ABPD, based on the observed population density, the following information is locally available to an individual:

- The current deviation in local population size is given by $G_d I_a - d_t I_a$
- The number of weaker individuals killed by the individual is given by N_k^i
- The estimated number of survivors to the next time step according to this individual is equal to $d_t I_a - N_k^i$

Therefore, the number of offsprings to be produced by an individual i is given by

$$\hat{b}^i = (\frac{G_d}{d_t} - 1) + N_k^i \tag{12}$$

Figure 5.2 shows the stability of CPD scheme for various initial population sizes and a desired population size of 1000 individuals.

It can be observed that even with this scheme the population quickly reaches a stable size. However the equilibrium population size is slightly higher (10%) than the desired population size, this is due to the simple theoretical modeling of the population deficit. The number of individuals eliminated by an individual which is later killed before it can reproduce is not taken into account. As with ABPD, a non-linear estimate of the population deficit would be required to accurately reach the desired population level. However, the important aim of this preliminary investigation into bottom-up population dynamics model is to establish schemes which are able to maintain an equilibrium population size which has been achieved in both the cases.

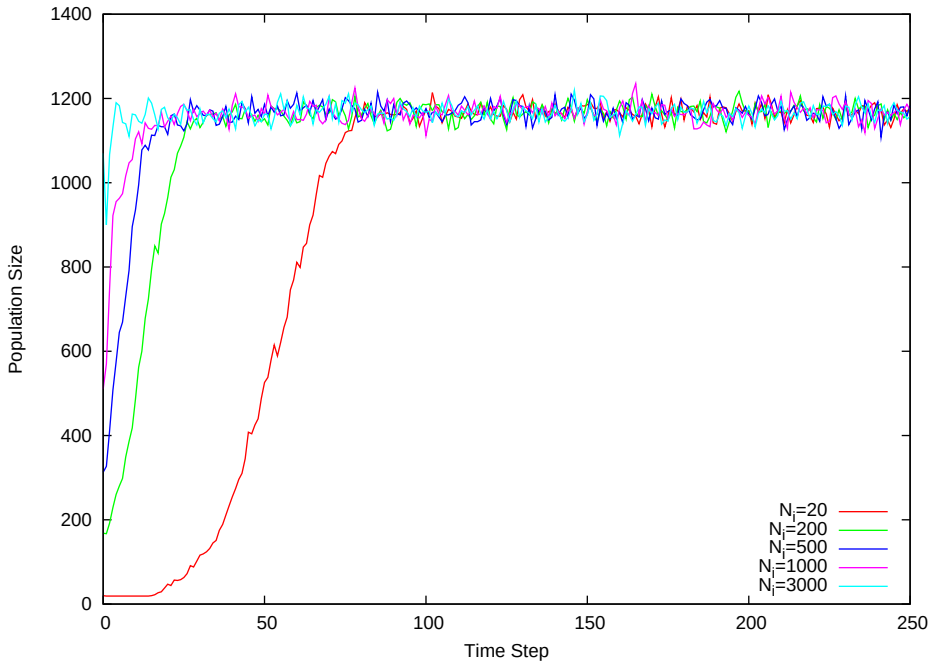

Figure 5.2: CPD convergence for various initial population sizes. The desired population size is 1000.

Another interesting artifact in the plot, is that even for high population values, the population quickly drops in the first time step. This is due to the combative scheme selected. This could lead to a very drastic decrease in diversity in the population. This problem is left as an open problem for future investigation.

6. ARTIFICIAL ECOSYSTEMS

Numerous examples of relatively closed ecosystems can be found in nature, which sustain and regulate themselves even in the presence of enormous uncertainty and variability in their environment. Inspired by these naturally occurring ecosystems, this section, presents a novel approach for developing agent-based evolutionary algorithms. Termed as Artificial Ecosystems, this approach utilizes the observations from experiments with mobile cellular EAs and bottom-up population dynamics schemes presented in the previous section to develop agent-based evolutionary algorithms.

Figure 6.1 shows the schematic representation of AES.

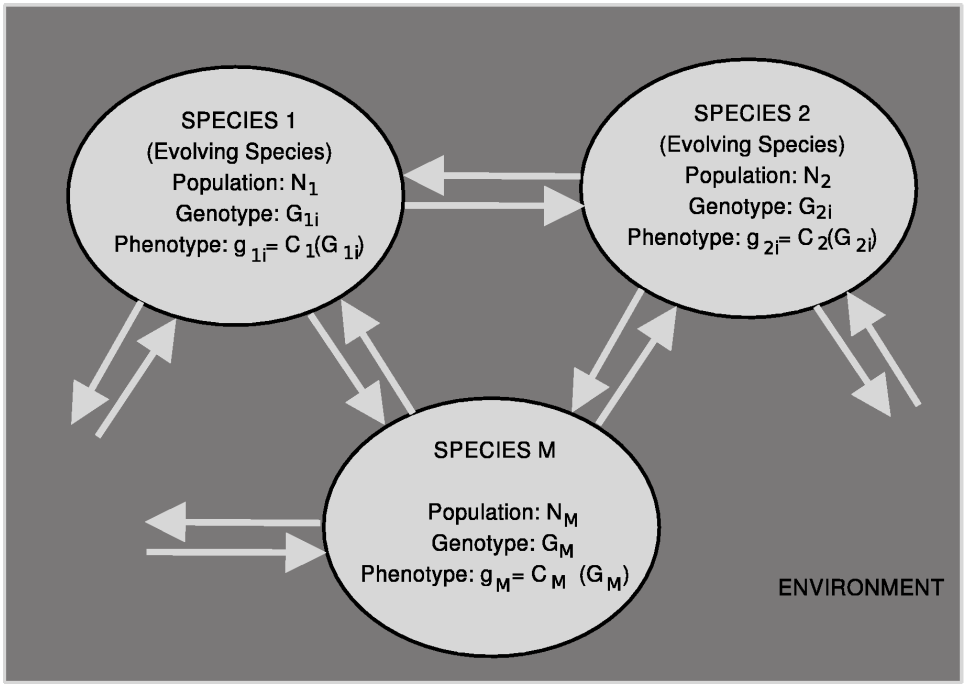

Figure 6.1: Schematic representation of AES approach.

Individuals of an AES are modeled as autonomous mobile agents similar to ALife simulations with selection and parameter control being incorporated into the

behavior set of the agents. Each individual has several associated properties such as age, mortality rate, ability to reproduce, etc. These individuals live in spatially structured environment either alone or along side or other species. In Figure 6.1, arrows between any two species indicate the interspecific interactions such as predation, parasitoidism, grazing, symbiotism and other similar behaviors which can occur between individuals of different species. Apart from these, intraspecific interactions (not depicted in figure) which may occur between individuals of the same species such as mate selection, social and territorial dominance can be modeled where required. Apart from providing a spatial structure to the population, the environment can also be used to model additional information about the problem, e.g., change in fitness function as in the case of dynamic multiobjective optimization. Effects of the environment on each species varies and are depicted by the open ended arrow from each species.

For an instance of AES, the parameter set of the problem to be optimized is mapped as adaptable characteristics (genome) of one or more of the evolving species. Not all species in AES need to be evolving. One or more non-evolving species can also inhabit the ecosystem. They still participate in the evolutionary dynamics of the ecosystem due to interactions with other evolving species. Examples of such non-evolving species are as follows. Consider the satisfiability problem, in which a given solution needs to satisfy certain number of constraints. In this situation, the solution vector is mapped to a evolving species as the solution needs to be evolved. However, as the given constraints are absolute and need no evolving, the constraints are mapped to a non-evolving species. One can think of the non-evolving species as having the best survival algorithm in the ecosystem and only other species need to adapt taking the non-evolving species into consideration. As another example, consider the predator-prey ecosystem for multiobjective optimization (Grimme and Schmitt, 2006). The prey represent the decision vector which needs to be evolve a solution on the Pareto front. The predators kill a prey based on its relative fitness in its neighborhood. Fitness of the preys is improved due to evolutionary pressure, but the predators' algorithm to hunt does not need any alteration and therefore predators have no evolvable characteristics. This kind of decentralization between the parameter set to be evolved and the constraints on the solution would be beneficial in cases where

the fitness landscape is highly chaotic and high local optima to global optima ratio. In such cases, due to spatial dispersion the constraints are not always enforced on the solution enabling constraint violating solutions to participate and hopefully assist the search process.

6.1. SINGLE SPECIES AND LANDSCAPE MODEL

In this section, the simplest possible AES model with a single species and landscape (environment) is presented for function optimization. Generally, individual-based ecological models (Grimm and Railsback, 2005) and several agent-based models discussed in Section 4 explicitly model resources of the environment for individuals to consume, grow and reproduce. In the single species and landscape (SSL) model however, the bottom-up population dynamics schemes presented in Section 5 take into account this resource modeling. The population growth models assume that there is always sufficient resources to sustain the desired population density, thereby alleviating the need to explicitly model the resources.

Algorithm 6.1 lists the pseudocode for a SSL model.

Algorithm 6.1 Psuedocode for single species and landscape AES.

```
1: GenerateLandscape(landscape)
2: GenerateInitialPopulation(pop)
3: while !StopCondition() do
4:    for all individual ∈ pop do
5:       individual→Move()
6:       individual→Interact()
7:       individual→SelectMate()
8:       individual→Reproduce()
9:       individual→Death()
10:   end for
11: end while
```

The parameters to be optimized are mapped as the genotype of the individuals and the function value corresponding to the parameter set are mapped to the phenotype of the individual. Each individual is associated with five behaviors which it executes each time step. The five behaviors are as follows:

- In *Move*, an individual moves to a vacant location in its interaction radius.
- In *Interact*, the individual estimates the local population density by counting the number of other individuals in its interaction radius.
- In *SelectMate*, the individual selects another individual to mate in its selection neighborhood (which can be same as the interaction radius).
- In *Reproduce*, the individual reproduces according to the population dynamics scheme selected.
- *Death* can occur due to age in the case of age-based population dynamics or due to combat in the case of combative population dynamics scheme.

6.1.1. Experimental Setup. The same test suite used for evaluating mobile cellular EAs is used for evaluating the SSL model, namely, FMS, MMDP and P-PEAKS problems. Table 6.1 shows the parameter values used for this study. In the case of *combative population dynamics*, a simulations is stopped in the same as with mcEA evaluation, i.e., when the absolute difference between the average fitness of the population and the best fitness of the population is less than 10^{-5}. However, in the case of *age-based population dynamics* the simulation is stopped after 1500 generations. This is due to the fact that ABPD is a non-elitist scheme which does not guarantee the survival of the best solution forever. Therefore the simulation is run for the specified 1500 generations and mean of 100 runs are reported as results.

6.1.2. Results. Tables 6.2 and 6.3 show the average number of generations and fitness values achieved by the two population dynamics based SSL models.

Table 6.1: Parameters used for the test suite.

Grid size	32×32
Update scheme	Each individual updates each time step
Neighborhood	C9
Initial population size	1000
Desired population size	1000
Population density	0.97656
Observed density confidence λ	0.2
Offspring sizing (population dynamics)	ABPD, CPD
Death rate (for ABPD)	0.05
Maximum generations (for ABPD)	1500
Mobility function	*random-hop*
Mobility radius	1
Mate selection	Uniform random
Recombination	DPX (double point crossover)
Mutation	Bit-flip mutation
Time step	All individuals updated
Chromosome length (L)	240 (MMDP) 192 (FMS) 100 (P-PEAKS)
Probability of crossover (P_c)	1.0
Probability of mutation (P_m)	$1/L = 0.0042$ (MMDP) $10/L = 0.052$ (FMS) $1/L = 0.01$ (P-PEAKS)

Table 6.2: Average number of generations on each problem.

Algorithm	FMS	MMDP	P-PEAKS
SSL (ABPD)	1500 ± 0.000	1500 ± 0.000	1500 ± 0.000
SSL (CPD)	734.452 ± 56.334	30000.000 ± 0.000	512.987 ± 100.298

It is interesting to note that in the CPD case, the number of generations on FMS and PPEAKS problems is significantly higher than any configuration of the mcEA presented in the Section 3. This indicates that very low selection pressure is

induced by the combative population dynamics scheme. As mentioned earlier, the ABPD scheme is a non elitist scheme which constantly replaces the population with new individuals and cannot guarantee convergence of the population to a small region in the search landscape and therefore, the algorithm was run for a specified number of generations and results are presented from the final state of the population.

Table 6.3: Quality of solution on each problem.

Algorithm	FMS			MMDP			P-PEAKS		
	best $\times(10^{-6})$	worst	mean	best	worst	mean	best	worst	mean
SSL (ABPD)	27.371	30.568	29.495 ± 1.0036	38.360	36.000	37.378 ± 0.6887	0.800	0.710	0.748 ± 0.026
SSL (CPD)	3.873	10.223	2.334 ± 1.839	40.000	40.000	40.000 ± 0.000	1.000	1.000	1.000 ± 0.000

As expected, ABPD based SSL algorithm, performed far worse than any configuration of mcEA or CPD based SSL algorithm. ABPD based algorithm, did not find the best solution[8] for any of the three problems, including the P-PEAKS problem which was easily solved by all the configuration of mcEA.

The CPD based algorithm on the other hand performed very well produced results comparable to the best solutions found by cellular and mobile cellular EAs. On both, the MMDP and the P-PEAKS problems, CPD based algorithm was able to find the global optimal solution in 100% of the runs. On FMS problem, however, the algorithm slightly under performed in the average case.

Based on these results, combative population dynamics shows promise for static function optimization while age-based population dynamics seems appropriate for dynamic optimization problems where the fitness function changes over time. The

[8]Or did not sustain the best solution till the end of the simulation.

investigation into dynamic optimization using ABPD based SSL algorithm is left for future work.

6.2. MULTI-SPECIES AES FOR OPEN ENDED PROBLEMS

In this section, a theoretical discussion on the efficacy of the AES approach to spatially asynchronous co-evolutionary experiments is presented. As AES is inherently based on individual interaction, co-evolutionary class of algorithms can be easily implemented to solve problems with vague or no absolute fitness measures or problems where more complex interacting sub-components need to be designed or optimized simultaneously.

As a example, a competitive co-evolutionary predator-prey ecosystem is described here to solve, the problem of pattern recognition in 2-D signals. The objective of the problem is to identify distinct spatio-temporal patterns in the input signal.

Let the predator-prey ecosystem have three components - environment, predators and prey. Assuming no other information about the input signal is provided, there is no absolute fitness function to evaluate the fitness of a pattern solution. Here the input signal is mapped to the environment as a landcover in which the predators and prey "live". The prey genotype represents the parameters of a general 2-D signal, i.e., prey represent a possible pattern in the input signal and therefore represent solutions to the problem. Using the density dependent reproduction and movement behaviors, the prey move about in the environment interacting with each other. The parameter set mapped to the prey project a phenotype which is a 2-D signal. The predator has the "visual acuity" to distinguish prey signal from the background. Therefore the preys whose phenotype closely matches with the landcover will receive more camouflage than those whose phenotype does not match the landcover. Predators can start with some arbitrary threshold to distinguish prey from land cover. As the prey which are "easily" visible to the predator are killed, predators find it difficult to sustain the killing spree. Predators whose visual acuity is improved due to their reproduction cycle would be able to hunt more prey. This evolutionary arms-race for survival would not only provide the patterns in the input signal (from the prey genotype), but also provide an approximate algorithm to detect the patterns in similar signals (from the predator genotype).

While the example presented here utilizes a antagonistic interaction between the two species (Predators kill prey), other forms of coevolution such as competitive, in which multiple species compete for the same resource and cooperative, in which species have to co-exist for mutual benefit, can also be achieved under this framework.

A comparison of AES with existing panmictic and spatially structured coevolutionary techniques is an important future direction of this research.

7. CONCLUSIONS AND FUTURE WORK

This thesis presented a new class of evolutionary algorithms called mobile cellular EAs (mcEAs). These algorithms are characterized by individuals which move around on a spatial population structure. Selection pressure in these algorithms was investigated and the results pertaining to the effects of population density and mobility were presented. It was observed that selection pressure in mcEAs was much more tunable that what was possible with existing cellular EAs. Experiments with a test suite of combinatorial problems also support this claim.

This thesis also presented a general architecture for developing agent-based EAs called Artificial Ecosystems (AES). Individual-based bottom-up reproduction schemes were developed along with a simple agent-based algorithm for function optimization. Experiments with a test suite of problems showed promising results for the architecture and hopes to inspire a new way of developed agent-based evolutionary algorithms.

Although significant results for mobile cellular EAs and the AES framework, it is none the less a preliminary investigation into both and further investigation is required to fully characterize their behaviors. In the current work, only a few possible combinations of update schemes, neighborhoods and mobility functions for mcEAs were investigated. Further investigation into these parameters is necessary to characterize mcEAs. For the AES framework, the developed population dynamics schemes only considered first-order relationships among parameters. It is believed that better performing schemes can be developed if higher-order relationships and inter-dependencies between behaviors are taken into account. Population dynamics is an important aspect of agent-based algorithms and certainly warrants further investigation. While the simple AES example, demonstrates the efficacy of the framework for function optimization, examples of multi-species instantiations of AES need to be developed to demonstrate coevolutionary mechanisms. Finally, better understanding of the performance characteristics of the algorithms presented in this thesis can be only evaluated using high-quality test and real-world problems using state-of-the-art selection and recombination operators.

Another important future scope of this research is developing parallel implementations of mobile cellular EAs and AES framework. Parallelization on both Beowulf type configurations and Graphics Processing Units (GPUs) are being considered to study complex optimization problems involving large search spaces. Existing implementations of cellular EAs and agent-based models on these architectures should facilitate in fast development of this future direction.

APPENDIX

TEST SUITE FOR ALGORITHM EVALUATION

A.1. FREQUENCY MODULATION SOUNDS (FMS)

The Frequency Modulation Sounds (FMS) parameter identification problem was proposed by Tsutsui et al. (1997) as a difficult problem for evolutionary algorithms to optimize. The problem involves identifying parameters of a model (say $y(t)$) to a basic sound generator $y_0(t)$. The goal of any optimization algorithm is therefore to minimize the sum of squared errors (SSE) given by

$$f_{FMS}(\vec{x}) = \sum_{t=0}^{100} (y(t) - y_0(t))^2 \tag{A.1}$$

The problem involves evolving 6 parameters $\vec{x} = (a_1, w_1, a_2, w_2, a_3, w_3)$ in order to fit the evolved model $y(t)$ to the original model $y_0(t)$, given by Equations A.2 and A.3 respectively.

$$y(t) = a_1 sin(w_1 t\theta + a_2 sin(w_2 t\theta + a_3 sin(w_3 t\theta))) \tag{A.2}$$

$$y_0(t) = 1.0 sin(5.0 t\theta + 1.5 sin(4.8 t\theta + 2.0 sin(4.9 t\theta))) \tag{A.3}$$

In the current work, each parameter is encoded with 32 bits, resulting in a chromosome of length 192 bits. When calculating fitness, this binary representation is scaled into the range -6.4 to +6.35. Also, $\theta = 2\pi/100$ to match the 100 time steps used to calculate the SSE of a given solution. The resulting problem is highly multimodal with strong epistasis with minimum value (global optimal) being $f_{FMS} = 0.0$.

A.2. MASSIVELY MULTIMODAL DECEPTIVE PROBLEM (MMDP)

The MMDP problem has be specifically designed by Goldberg et al. (1992) to be a difficult problems for EAs to solve. The problem is made up of k subproblems

of 6 bits each. This is a maximization problem with the global optimum equal to k. To achieve this global optimum, each subproblem needs to be composed to either all zeros or all ones. Each subproblem is given a fitness value based on the number of ones in it according to Table A.1. The fitness of the entire solution is given by Equation A.4.

Table A.1: 6-bit Subproblem Value

Number of ones	Subproblem value
0	1.000000
1	0.000000
2	0.360384
3	0.640576
4	0.360384
5	0.000000
6	1.000000

$$f_{MMDP}(\vec{x}) = \sum_{i=1}^{k} subfunction(x_i) \tag{A.4}$$

The number of local optima is very large (22^k) compared to the total number of global optimal solutions (2^k). Thus the degree of multimodality is defined by k. A considerably large instance with $k = 40$ is considered in the current study. This results in a chromosome of length 240 bits.

A.3. MULTIMODAL PROBLEM GENERATOR - P-PEAKS

This problem generator was used to study the epistasis on EAs in De Jong et al. (1997). The P-PEAKS generator generates a set of P random N-bit strings that represent the location of the P peaks in the search space of N dimensions. An arbitrary bit string is evaluated by calculating the number of bits it has in common with the nearest of the P peaks. The nearest peak is located in Hamming space. The

fitness of a bit string is calculated using Equation A.5. Weakly/strongly epistatic problems can be generated with small/large number of peaks. In the current study, $P = 100$ peaks in a search space of $N = 100$ dimensions is used, which represents a medium-high epistasis problem. This results in a chromosome of length 100.

$$f_{P_PEAKS}(\vec{x}) = \frac{1}{N} \max_{i=1}^{P} \{N - HammingD(\vec{x}, Peak_i)\} \tag{A.5}$$

BIBLIOGRAPHY

Adami, C. and Brown, C. T. (1994). Evolutionary Learning in the 2D Artificial Life Systems Avida. In Brooks, R. and Maes, P., editors, *Proceedings of Artificial Life IV*, pages 377–381. MIT Press.

Alba, E. and Dorronsoro, B. (2008a). *Cellular Genetic Algorithms*, volume 42 of *Operations Research/Computer Science Interfaces Series*. Springer US.

Alba, E. and Dorronsoro, B. (2008b). *Cellular Genetic Algorithms*, volume 42 of *Operations Research/Computer Science Interfaces Series*, chapter On the Effects of Structuring the Population, pages 37–46. Springer US.

Alba, E. and Dorronsoro, B. (2008c). *Cellular Genetic Algorithms*, volume 42 of *Operations Research/Computer Science Interfaces Series*, chapter Some Theory: A Selection Pressure Study on cGAs, pages 47–69. Springer US.

Alba, E., Giacobini, M., and Tomassini, M. (2002). *Parallel Problem Solving from Nature PPSN VII*, volume 2439/2002 of *Lecture Notes in Computer Science*, chapter Comparing Synchronous and Asynchronous Cellular Genetic Algorithms, pages 601–610. Springer Berlin / Heidelberg.

Alba, E. and Troya, J. M. (2000). *Parallel Problem Solving from Nature PPSN VI*, volume 1917/2000 of *Lecture Notes in Computer Science*, chapter Cellular Evolutionary Algorithms: Evaluating the Influence of Ratio, pages 29–38. Springer Berlin / Heidelberg.

Amato, P. and Farina, M. (2005). *Soft Computing: Methodologies and Applications*, volume 32/2005 of *Advances in Soft Computing*, chapter An ALife-Inspired Evolutionary Algorithm for Dynamic Multiobjective Optimization Problems, pages 113–125. Springer Berlin / Heidelberg.

Axelrod, R. (1997). *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton University Press.

Berry, A. and Vamplew, P. (2003). A Simplified Artificial Life Model for Multiobjective Optimisation: A Preliminary Report. In *Proceedings of Congress on Evolutionary Computation*.

Berry, A. and Vamplew, P. (2005). *Evolutionary Multi-Criterion Optimization*, volume 3410/2005 of *Lecture Notes in Computer Science*, chapter The Combative Accretion Model - Multiobjective Optimisation Without Explicit Pareto Ranking, pages 77–91. Springer Berlin / Heidelberg.

Cantu-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms*, volume 1 of *Genetic Algorithms and Evolutionary Computation*. Springer US.

Coakley, S., Smallwood, R., and Holcombe, M. (2006). From Molecules to Insect Communities - How Formal Agent Based Computational Modelling is Uncovering New Biological Facts. *Scientiae Mathematicae Japonicae*, 64:185–198.

Darwin, C. (1900). *The Origin of Species: By Means of Natural Selection*. D. Appleton and Company.

De Jong, K. A., Potter, M. A., and Spears, W. M. (1997). Using Problem Generators to Explore the Effects of Epistasis. In *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 338–345.

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience.

Drezewski, R. and Siwik, L. (2008). *Advances in Evolutionary Algorithms*, chapter Agent-Based Co-Evolutionary Techniques for Solving Multi-Objective Optimization Problems, pages 468–498.

Epstein, J. M. and Axtell, R. (1996). *Growing Artificial Societies: Social Science From the Bottom Up*. MIT Press.

Giacobini, M., Alba, E., Tettamanzi, A., and Tomassini, M. (2004a). Modeling Selection Intensity for Toroidal Cellular Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO '04*, pages 1138–1149.

Giacobini, M., Alba, E., and Tomassini, M. (2003a). Selection Intensity in Asynchronous Cellular Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO '03*, pages 955–966.

Giacobini, M., Tomassini, M., and Tettamanzi, A. (2003b). Modeling Selection Intensity for Linear Cellular Evolutionary Algorithms. In *Proceedings of the Sixth International Conference on Artificial Evolution, Evolution Artificielle*, pages 345–356.

Giacobini, M., Tomassini, M., and Tettamanzi, A. (2004b). *Artificial Evolution*, volume 2936/2004 of *Lecture Notes in Computer Science*, chapter Modeling Selection Intensity for Linear Cellular Evolutionary Algorithms, pages 345–356. Springer Berlin / Heidelberg.

Giacobini, M., Tomassini, M., and Tettamanzi, A. (2005a). Takeover Time Curves in Random and Small-World Structured Populations. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO '05*, pages 1333–1340.

Giacobini, M., Tomassini, M., Tettamanzi, A. G. B., and Alba, E. (2005b). Selection Intensity in Cellular Evolutionary Algorithms for Regular Lattices. *IEEE Transactions on Evolutionary Computation*, 9(5):489–505.

Goldberg, D. E. and Deb, K. (1991). *Foundations of Genetic Algorithms*, chapter A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, pages 69–93. Morgan Kaufmann.

Goldberg, D. E., Deb, K., and Horn, J. (1992). Massively Multimodality, Deception and Genetic Algorithms. In *Proceedings of PPSN II*, pages 37–46.

Gorges-Schleuter, M. (1999). An Analysis of Local Selection in Evolution Strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999)*, volume 1, pages 847–854.

Grimm, V. and Railsback, S. F. (2005). *Individual-Based Modeling and Ecology*. Princeton University Press.

Grimme, C. and Schmitt, K. (2006). Inside a Predator-Prey Model for Multi-Objective Optimization: A Second Study. In *Genetic and Evolutionary Computation Conference (GECCO '06)*, pages 707–714.

Komosinski, M. and Rotaru-Varga, A. (2001). Comparison of Different Genotype Encodings for Simulated 3D Agents. *Artificial Life*, 7(4):395–418.

Laumanns, M., Rudolph, G., and Schwefel, H.-P. (1998). *Parallel Problem Solving from Nature PPSN V*, volume 1498/1998 of *Lecture Notes in Computer Science*, chapter A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study, pages 241–249. Springer Berlin / Heidelberg.

Ray, T. S. (1991). *Scientific Excellence in Supercomputing: The IBM 1990 Contest Prize Papers*, chapter Evolution and Optimization of Digital Organisms, pages 489–531. The Baldwin Press.

Rudolph, G. (2000). On Takeover Times in Spatially Structured Populations: Array and Ring. In *Proceedings of the Second Asia-Pacific Conference on Genetic Algorithms and Applications*, pages 144–151.

Sarker, R. A. and Ray, T. (2010). *Agent-Based Evolutionary Search*, volume 5 of *Adaptation, Learning, and Optimization*, chapter Agent-Based Evolutionary Approach: An Introduction, pages 1–12. Springer US.

Sarma, J. and De Jong, K. A. (1996). *Parallel Problem Solving from Nature PPSN IV*, volume 1141/1996 of *Lecture Notes in Computer Science*, chapter An Analysis of the Effect of the Neighborhood Size and Shape on Local Selection Algorithms, pages 236–244. Springer Berlin / Heidelberg.

Sarma, J. and De Jong, K. A. (1997). An Analysis of Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 181–186.

Skolicki, Z. M. (2007). *An Analysis of Island Models in Evolutionary Computation*. PhD thesis, George Mason University.

Socha, K. and Kisiel-Dorohinicki, M. (2002). Agent-Based Evolutionary Multiobjective Optimization. In *Proceedings of Congress of Evolutionary Computation*.

Sprave, J. (1999). A Unified Model of Non-panmictic Population Structures in Evolutionary Algorithms. In *Proceedings of Congress on Evolutionary Computation CEC '99*, pages 1384–1391.

Tesfatsion, L. (2002). Agent-Based Computational Economics: Growing Economies from the Bottom Up. *Artificial Life*, 8(1):55–82.

Thomsen, R., Rickers, P., and Krink, T. (2000). *Parallel Problem Solving from Nature - PPSN VI*, chapter A Religion-Based Spatial Model For Evolutionary Algorithms, pages 817–826. Springer Berlin / Heidelberg.

Tomassini, M. (2005). *Spatially Structured Evolutionary Algorithms*. Springer.

Tsutsui, S., Ghosh, A., Corne, D., and Fujimoto, Y. (1997). A Real Coded Genetic Algorithm with an Explorer and Exploiter Populations. In *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 238–245.

Ursem, R. K. (1999). Multinational Evolutionary Algorithms. In *Proceedings of Congress on Evolutionary Computation*, pages 1633–1640.

Whitley, D. (1993). Cellular Genetic Algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA)*, page 658.

Whitley, D. (1995). *Applications of Modern Heuristic Methods*, chapter A Review of Simple and Cellular Genetic Algorithms, pages 55–67.

VITA

Srinivasa Shivakar Vulli (Shivakar), was born on 23rd November, 1982 in Hyderabad, India. He received his Bachelor's degree in Mechanical Engineering from Nagarjuna University, India, in May 2004. He joined Missouri University of Science and Technology (formerly University of Missouri – Rolla) in Spring 2006. He received his Master's degree in Mechanical Engineering from Missouri S&T in May, 2008. He continued his education at Missouri S&T pursuing a Master of Science in Computer Science. During the course of his study, he worked in the capacity of Graduate Research Assistant with Airborne Reconnaissance and Image Analysis (ARIA) laboratory. His research was funded by the Department of Electrical and Computer Engineering and the Intelligent Systems Center at Missouri S&T. His areas of interest include data mining, machine learning, robotics, evolutionary algorithms, multi-agent systems and high performance computing.

Shivakar received his Master's degree in Computer Science from Missouri S&T in August, 2010.