Distributed Steady-State Neuro-Evolutionary Path Planning in Non-Stationary Environments Using Adaptive Replacement

Gerry Dozier
Department of Computer Science and Software Engineering
Auburn University
Auburn, AL 36849-5347
email: gvdozier@eng.auburn.edu

Abstract

Recently, there has been an increasing interest in applying evolutionary computation to path planning. To date, these evolutionary path planners have been single agent planners. In real-world environments where the knowledge of obstacles is naturally distributed, it is possible for single agent path planners to become overwhelmed by the volume of information needed to be processed in order to develop accurate paths quickly in non-stationary environments. This paper presents an adaptive replacement strategy that increases the effectiveness of steady-state evolutionary path planning in slow, constantly changing environments.

1 Introduction

Recently, there has been a growing interest in the use of evolutionary computation (EC) for path planning [18]. This interest has lead to the development of a number of exciting and successful evolutionary path planners. However, to date, all of these planners have been single agent path planners that operate in stationary environments. In real-world situations, it is possible for single agent path planners to become overwhelmed by the sheer volume of information needed to be processed in order to develop accurate paths quickly. In the case of a multi-agent environment, the reliance on centralized planning will result in an inefficient utilization of the knowledge distributed among the agents. Thus, there is a need for multi-agent path planning protocols for distributed path planning. These planning protocols must also be able to develop paths for non-stationary environments.

The Distributed Path Planning Problem (DPPP) can be stated as follows. Given an environment (R, S, G, A, O) where R represents a "client" point agent, S represents the starting point, G represent the destination point, and A represents a set of distributed "planning" agents, where each $a_i \in A$ has knowledge of a subset of obstacles, in O, located near the region that a_i occupies: discover a collision free path from S to G using the "planning" agents in A that some "client" agent, R, can traverse.

In this paper, we compare a number of steady-state distributed neuro-evolutionary computations (DNECs) [5] for path planning in non-stationary environments. The DNECs evolve candidate paths (CPs) that are represented as radial basis function networks (RBFNs) [10]. By evolving, RBFNs the DNECs are able to dramatically reduce the number of nodes (or via points) needed to represent smooth CPs. The major contribution of this paper is the presentation of a new adaptive replacement strategy (ARS). Our results show that this new ARS is superior to static replacement strategies (SRS) for steady-state evolutionary search.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of distributed EC, RBFNs, and steady-state ECs for non-stationary problems. In Section 3, the DNECs and the ARS are described in detail. In Section 4, the test suite of five randomly generated path planning problems is described and in Section 5, our preliminary results are presented. Section 6 provides a brief discussion and our conclusions are presented in Section 7.

2 Background

2.1 Distributed ECs

Distributed Evolutionary Computations (DECs) typically fall into one of three categories: function-based [2, 14], domain-based [1, 13], and variable-based [9, 12]. Function-based DECs distribute tasks (or functions) of the evolutionary process (selection, procreation, evaluation) among k processors in order to speed-up the processing time of a single EC.

Domain-based DECs (DBDECs) distribute a population of $\mathcal{P}$ candidate solutions (CSs) among k processors. There are two types of DBDECs: coarse-grained and fine-grained. In coarse-grained DBDECs, k populations of $\frac{\mathcal{P}}{k}$ CSs are maintained. Thus, k ECs are executed in parallel. Periodically, selected CSs are allowed to migrate to other populations. Fine-grained DBDECs usually assign one CS to each processor. The populations or demes overlap as CSs are only allowed to mate and compete for survival within their geographical neighborhoods.

In variable-based DECs (VBDECs), the structure representing CSs is distributed. Let V represent the variables that form the structure of the CSs of a problem. VBDECs, distribute $\frac{|V|}{k}$ variables among k processors. Each processor uses an EC to evolve a population of $\mathcal{P}$ partial CSs. These types of DECs are also known as distributed co-evolutionary computations because the partial CSs can be viewed as k species working together to solve a problem in symbiotic fashion.

2.2 Steady-State Evolutionary Search in Non-Stationary Environments

Research in the area of evolutionary search in non-stationary environments, to date, has focused primarily on two approaches: memory-based[7] and mutation-based [3, 4, 8, 13, 16]. Memory-based approaches enhance ECs with additional structures that enable them to remember past solutions while mutation-based approaches focus on using "higher than normal" mutation rates in an effort to track non-stationary optima. In this paper we investigate the possibility of a third approach that can be used by steady-state evolutionary search in non-stationary environments.

In [17], the authors introduce a simple modification to steady-state search that allows it to effectively track optima in non-stationary environments consisting of small changes that occur with low frequency. Rather than replacing the worst individual each iteration [15] they replaced the oldest.

The adaptive replacement strategy presented in this paper is a generalization of the "replace the oldest" strategy. The parameter that is adapted during search is, γ, the number of fitness-based replacements before one age-based replacement. Thus, the "replace the oldest" strategy can be seen as $\gamma = 0$. The value assigned to γ is based on the amount "stress"[3] the EC is under as the environment changes.

2.3 RBFNs and Path Construction

The construction of CPs can be viewed as an interpolation problem where the objective is to develop a continuous function that passes through n specified nodes. By using interpolation, fewer nodes are needed in order to represent a smooth path. In fact, only $n = |O| + 2$ nodes need to be specified, where O is the set of obstacles and where the other two nodes represent the start and goal nodes. Therefore one can perform path planning by evolving nodes for interpolation RBFNs.

Consider a sequence of n nodes, $[(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$. An interpolation RBFN representing this sequence can be expressed as $f(x) = \Sigma_{i=1}^{n} \phi(x, x_i) w_i$, where $\phi(x, x_i) = exp(\frac{(x-x_i)^2}{2\sigma^2})$ and where w_i are weights. In order to properly pass through the nodes, a set of values must to be discovered for the weights. This can be accomplished by solving the following linear equations for the set of weights (where $\sigma = \frac{1}{\sqrt{2n}}$, and where ϕ must be non-singular) [10]:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \phi(x_1, x_1) & \phi(x_1, x_2) & \cdots & \phi(x_1, x_n) \\ \phi(x_2, x_1) & \phi(x_2, x_2) & \cdots & \phi(x_2, x_n) \\ \vdots & \vdots & \vdots & \vdots \\ \phi(x_n, x_1) & \phi(x_n, x_2) & \cdots & \phi(x_n, x_n) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

3 The DNECs

3.1 An Evolutionary Agent Protocol

The DNECs were developed for evolving CPs through an environment composed of one-dimensional, non-stationary obstacles. The obstacles are represented by a vertical line segment with an upper and lower bound. Associated with each obstacle in the environment is an evolutionary agent located at the midpoint of the obstacle. Each agent has an x coordinate that remains constant and uses an EC to evolve a population of offsets that are added to the upper or lower bounds of the their associated obstacle to form feasible y coordinates. Thus, each offset of an EC represents one node of a distributed CP. The values on the x and y axes range from 0.0 to 1.0.

Figure 1 shows a pseudocode version of the protocol used by an evolutionary agent. Each agent uses two random number generators rnd_1 and rnd_2. The random number generator, rnd_1, is used for synchronizing the tournament selection algorithm [11]. If the seed for each evolutionary agent is identical then the selection of parents will be synchronized (i.e., all the offsets corresponding to a distributed parent will be selected by the evolutionary agents). If the seeds of the agents are different, then an asynchronous form of tournament selection will be used (i.e., each agent may select offsets corresponding to different distributed CPs to be crossed and/or mutated).

After the seed has been given to an evolutionary agent, the protocol proceeds as follows. An initial population of $\mathcal{P}$ offsets is randomly generated[1]. Each offset is then evaluated by sending it to the 'Evaluate' procedure which takes as arguments the address of some agent that will be evaluating the length of the distributed CP, the population of offsets, the index of the offset to be evaluated, and the time of birth of the offset (tob). This procedure converts the offset into a y value and then sends a message to the evaluator agent containing the candidate node and the index of the distributed CP that the node belongs to. The evaluator agent replies with a message which contains the index of the distributed CP as well as the overall path length associated with the CP. This path length is assigned to the offset as its fitness[2]. After a CP is evaluated the position of the evolutionary agent (along with the position of its associated obstacle) is changed.

After the initial generation of offsets has been generated and evaluated, the evolutionary agent begins its iterative loop of selection, procreation and replacement. In this loop, the timer (t) is incremented and two parents are selected via (synchronous or asynchronous) tournament selection.

After two parent offsets have been selected, one offset in the population must be selected to be replaced. This is an important part of the protocol. One cannot focus on selecting the worst fit individual [17] which we refer to as fitness-based replacement. Instead a balance must be negotiated between fitness-based replacement and age-based replacement. In age-based replacement, the oldest individual is selected to be replaced. We define the fitness-based replacement rate, γ, to be the number of consecutive fitness-based replacements executed before a single age-based replacement is executed[3].

Once an offset is selected to be replaced, the parent offsets then create a child offset, δ_{ic}, through the use of three operators. The first operator is BLX-0.0 crossover [6] and works as follows: $\delta_{ic} = rnd_2(\delta_{ip}, \delta_{iq})$, where rnd_2 returns a uniform random number within the interval $[\delta_{ip}, \delta_{iq}]$. The second and third operators use only the first parent selected and are as follows: $\delta_{ic} = rnd_2(\delta_{ip}, 0.0)$ and $\delta_{ic} = \delta_{ip} + rnd_2(\delta_{ip}, 0.0)$. Each

```
EvolutionaryAgent_i(EvaluatorAgent, Seed){
    t = 0;
    Randomly Generate n offsets (δ_ij);
    for j = 1 to n {
        t = t + 1;
        Evaluate(EvaluatorAgent,δ_i,j,t);
        updateTheOffsets(δ_i);
    }
    while (t ≤ NumberOfIterations) {
        t = t + 1;
        p = selectParentFrom(δ_i,Seed,rnd_1);
        q = selectParentFrom(δ_i,Seed,rnd_1);
        w = selectToDie(δ_i);
        if (rnd_2(0.0, 1.0) < 0.5)
            δ_ic = rnd_2(δ_ip, δ_iq);
        else if (rnd_2(0.0, 1.0) < 0.5)
            δ_ic = rnd_2(δ_ip, 0.0);
        else
            δ_ic = δ_ip + rnd_2(δ_ip, 0.0);
        δ_ic = δ_ic + N(0, s);
        Mutate(δ_ic,μ,rnd_2);
        δ_iw = δ_ic;
        Evaluate(EvaluatorAgent,δ_i,w,t);
        updateTheOffsets(δ_i);
    }
}

Evaluate(EvaluatorAgent, δ_i, k, t){
    if (δ_ik < 0.0)
        sendTo(EvaluatorAgent,(x_i, δ_ik + lb_i),k);
    else
        sendTo(EvaluatorAgent,(x_i, δ_ik + ub_i),k);
    receiveFrom(EvaluatorAgent,k,PathLength);
    f_ik = PathLength;
    tob_ik = t;
}

Mutate(δ_ik, μ, rnd_2){
    if (rnd_2(0.0, 1.0) ≤ μ)
        δ_ik = −δ_ik;
}
```

Figure 1: Evolutionary Agent Protocol

offspring undergoes a Gaussian disturbance, $N(0, s)$. The child offset is then mutated using sign mutation which flips the sign of the offset. After sign mutation, the child replaces the offset selected to die and is evaluated. After each offspring is evaluated the coordinates of the evolutionary agent are changed and the population of offsets is updated so that the distributed paths will remain stationary.

3.2 An Adaptive Replacement Strategy

Instead of using a constant γ, it would be more advantageous to adapt γ during search. The reasons for this stem from the fact that the EC's perception of environmental change will vary during evolution. If

[1] Offset are always checked to make sure that their associated y value is within [0.0,1.0].

[2] Since the associated path length of an offset is assigned as its fitness, the lower the fitness the better the offset.

[3] In [17], the authors only investigate the use of $\gamma = 0$.

the environment remains relatively stationary, then a large γ would be more effective than a smaller one. However, as the rate of change of the environment increases the γ value should decrease to prevent the selection algorithm from selecting individuals based on old, out-dated fitnesses.

An effective adaptive replacement strategy (ARS) would be one that attempts to balance the convergence of the population with the perceived rate of change of the environment. This ratio can then be used as the γ value for a specified amount of time.

The ARS can be derived using the following notation. Let:

x_k, denote the best individual in the population at time k[4],

$\varepsilon(x_k, m)$, denote the fitness associated with x_k at time m, where $m \geq k$,

$\frac{df_p}{dt}$, denote the change in the best fitness of the population with respect to time

$$= \frac{\varepsilon(x_{k-\Delta t}, k-\Delta t) - \varepsilon(x_k, k)}{\Delta t},$$

$\frac{df_b}{dt}$, denote the change in of the previous best individual's fitness over time

$$= \frac{\varepsilon(x_{k-\Delta t}, k-\Delta t) - \varepsilon(x_{k-\Delta t}, k)}{\Delta t},$$

$\frac{df_p}{df_b}$, denote the ratio of the population convergence (with respect to the best fitness in the population) to the perceived change in the environment,

therefore,

$$\gamma = \begin{cases} \Delta t & \text{if } df_b \geq 0 \\ |\frac{df_p}{df_b}| & \text{if } df_p \geq 0 \text{ and } df_b < 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

In Equation 1, three triggers [3, 4] are shown. The first trigger causes the EC to use fitness-based replacement exclusively. Any time the ARS DNEC perceives that the environment is stationary or that the previous best individual's path length decreases[5] as the environment changes then $\gamma = \Delta t$.

[4] In this paper one time unit is equivalent to one CP evaluation or one iteration of the distributed steady-state EC.

[5] This can happen when by updating an offset the value of the offset goes from nonnegative to negative or vice versa.

The second trigger allows γ to be assigned intermediate values within the interval $[0, \Delta t]$. As long as the rate of population convergence is greater than the perceived rate of change of the environment then a non-zero value will be assigned to γ. If the case arises where $|\frac{df_p}{df_b}| \geq \Delta t$ then γ is set equal to Δt.

Finally, the third trigger is used to envoke the "replace the oldest" strategy. Our hypothesis is that when this strategy is used, especially with smaller population sizes (5-20 individuals), a type of hyper-diversity [3, 4] will result.

4 The Test Suite

Our test suite consisted of five randomly generated non-stationary path planning problems. For each problem, the number of equally spaced obstacles was randomly selected from the interval, [4,8]. Problems 1 and 5 consisted of 4 obstacles while Problems 2-4 consisted of 8 obstacles. The length of each obstacle was randomly selected from the interval [0.15,0.35]. For each of the five problems the start and destination points were (0.0,0.5) and (1.0,0.5).

Each obstacle, $o \in O$, slides parallel to the y-axis. The obstacles move at a constant rate with the y-coordinate of each obstacle changing every time a distributed CP is evaluated. The rate at which an obstacle moves is determined by $\Delta_o = \frac{2(1-\omega_o)}{g-2}$, where ω_o represents the length of o and g represents the maximum number of path evaluations allowed. The obstacles initially move upwards until they reach the edge of the top boundary of the environment and then move downward until they reach the bottom boundary of the environment. At this point, they being moving upwards again. After g path evaluations all obstacles will have returned to their original starting positions.

5 Results

A total of 48 DNECs were compared. These DNECs were distinguished by three attributes: type of selection, which was taken from the set, $\{a, s\}$, (where a denotes asynchronous selection and s denotes synchronous selection), population size, which was taken from the set, $\{5, 10, 20, 160\}$, and the value of γ which was taken from the set $\{0, 1, 2, 4, 8, *\}$, where $*$ denotes the use of the ARS.

The other parameter settings for the 48 DNECs were as follows. The usage rates for the three operators were 0.5, 0.25, and 0.25. The standard deviation of the Gaussian disturbance of the DNECs was set to 0.03 ($s = 0.03$), and the sign mutation rate was set to 0.05 ($\mu = 0.05$). The length, l, of a CP was determined as follows: $l = \Sigma_{i=0}^{19} dst(\frac{i}{20}, f(\frac{i}{20}), \frac{i+1}{20}, f(\frac{i+1}{20}))$, where $dst(b,c,e,f)$ returns the distance between points (b,c) and (e,f). For the eight DNECs using the ARS, Δt was set to 5 and at $t=0$ γ was initialized to 1.

In our study, the 48 DNECs were divided into classes based on the selection method and the population size used. Each of the DNECs was run a total of 31 times on each of the problems with $g = 2000$. For each class the best performing SRS DNEC was determined by comparing instances on each of the five problems and choosing the DNEC that had the greater number of statistically significant average off-line performances. For these comparisons a paired Student's t-test was used. Any $|t| \geq 1.7$ signaled a statistically significant difference in average performance.

In Tables 1 and 2, the performances of the synchronous and asynchronous DNECs are compared. Each cell of the tables records the average off-line performance along with the standard deviation of the average off-line performance recorded in parentheses. For each class, the best performing SRS DNEC is highlighted in boldface and is compared with an ARS DNEC using the Student's t-test. The t values are recorded on the row beneath the recorded performances of the ARS DNECs.

In Table 1, notice that the for the synchronous SRS DNECs, the best γ decreases as the population size increases. We believe this to be because the DNECs with larger population sizes converge at a slower rate. Thus, the faster the speed of convergence the higher γ can (and will need to) be. Our results suggest that the best setting for γ is $\frac{40}{\mathcal{P}}$.

When comparing the best SRS DNECs with the ARS DNECs, the differences in the performances are all statistically significant in favor of the ARS DNECs except when the population size is 160 (see Problem 5).

The observations made concerning the performances of the synchronous DNECs are similar for the performances of the asynchronous DNECs in Table 2. Once again the best setting is $\gamma = \frac{40}{\mathcal{P}}$ for the asynchronous SRS DNECs. As was seen in Table 1, the asynchronous ARS DNECs have statistically significant better performances on each of the five problems except when the population size is large (see Problem 5).

Although the performances of the DNECs are all close, the best overall synchronous and asynchronous SRS DNECs were $(s,5,8)$ and $(a,5,8)$. This was primarily because they had significantly better performances than their counterparts with larger populations on the problems with only four obstacles, Problems 1 and 5. When these two DNECs were compared neither had a statistically significant better performance on any of the problems. Similarly, for the same reason, the best overall ARS DNECs were $(s,5,*)$ and $(a,5,*)$. When their performances are compared the former has a statistically significant better performance on Problem 2 while the latter has a statistically significant better performance on Problem 4.

Figure 3 shows a plot of the γ values of $(s,5,*)$ for each of the 5 problems. For each plot, the x-axis corresponds to the number of evaluations (time) and the y-axis corresponds to the value of γ. Notice that for each of the plots a variety of values for γ can be seen.

6 Discussion

Of the performances of the 48 DNECs, those that consistently perform the worst have $\gamma = 0$, the "replace the oldest" strategy. This seems to have more to do with the population size than anything else. Using the "replace the oldest" strategy in combination with a small population will cause the best individuals to be removed too quickly from the population. These individuals will not have enough opportunities to influence the direction of the search. As a result, the population will experience an influx of diversity. This is especially the case when the selective pressure of the selection algorithm is low with respect to the rate at which the best individuals leave the population. Thus, lower γ values allow for a greater influx of diversity.

7 Conclusions

In this paper, we have shown how RBFNs and EC can be combined and used effectively for distributed path planning in non-stationary environments. More importantly we have developed an adaptive replacement strategy that is triggered by the convergence of the EC as well as the perceived change in the environment.

Our results show that γ rate is an important parameter in determining the effectiveness of the DNECs. Also the results show that adapting γ during evolution is superior to using a static value for γ. In this study, the type of selection used (synchronous or asynchronous) was not a factor in the performance of the DNECs. Our future research will be devoted towards developing DNECs for environments with more complex obstacles and obstacle motions.

References

[1] Baluja, S. (1993). Structure and Performance of Fine-Grain Parallelism in Genetic Search, *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 155-162.

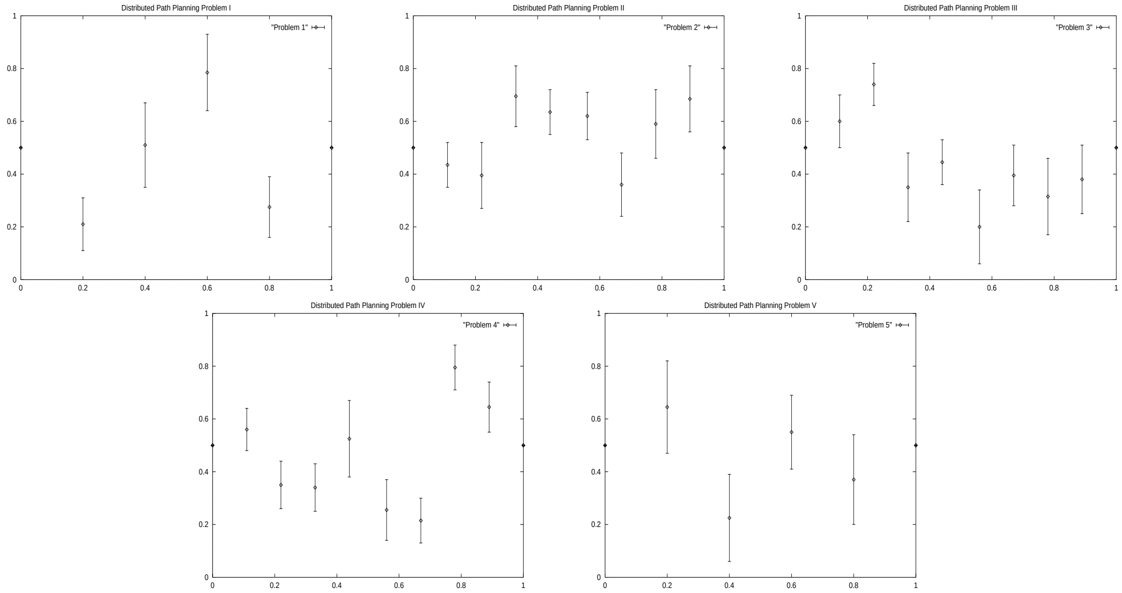

Figure 2: The Test Suite

Distributed Path Planning Problems

Algs	1	2	3	4	5
$(s,5,0)$	1.409 (0.023)	2.295 (0.046)	2.379 (0.042)	2.325 (0.048)	1.454 (0.021)
$(s,5,1)$	1.236 (0.013)	1.999 (0.039)	2.081 (0.025)	2.003 (0.040)	1.297 (0.022)
$(s,5,2)$	1.177 (0.012)	1.858 (0.035)	1.935 (0.038)	1.845 (0.043)	1.238 (0.018)
$(s,5,4)$	1.157 (0.014)	1.747 (0.052)	1.784 (0.042)	1.696 (0.052)	1.226 (0.012)
$(s,5,8)$	**1.200 (0.017)**	**1.676 (0.047)**	**1.681 (0.058)**	**1.612 (0.047)**	**1.273 (0.030)**
$(s,5,*)$	1.124 (0.012)	1.550 (0.065)	1.580 (0.057)	1.513 (0.045)	1.185 (0.020)
	$t=21.08$	$t=8.16$	$t=6.19$	$t=10.90$	$t=16.07$
$(s,10,0)$	1.258 (0.022)	2.003 (0.041)	2.087 (0.039)	2.015 (0.042)	1.321 (0.023)
$(s,10,1)$	1.163 (0.014)	1.801 (0.029)	1.872 (0.046)	1.757 (0.037)	1.231 (0.017)
$(s,10,2)$	1.161 (0.013)	1.709 (0.037)	1.746 (0.050)	1.645 (0.040)	1.249 (0.017)
$(s,10,4)$	**1.216 (0.019)**	**1.655 (0.046)**	**1.670 (0.060)**	**1.610 (0.050)**	**1.313 (0.029)**
$(s,10,8)$	1.293 (0.038)	1.645 (0.086)	1.678 (0.066)	1.601 (0.082)	1.415 (0.039)
$(s,10,*)$	1.140 (0.015)	1.532 (0.056)	1.582 (0.070)	1.478 (0.051)	1.204 (0.019)
	$t=19.06$	$t=9.66$	$t=5.04$	$t=10.08$	$t=16.38$
$(s,20,0)$	1.210 (0.019)	1.885 (0.039)	1.970 (0.036)	1.848 (0.036)	1.277 (0.023)
$(s,20,1)$	1.186 (0.016)	1.738 (0.040)	1.754 (0.045)	1.672 (0.042)	1.279 (0.022)
$(s,20,2)$	**1.226 (0.018)**	**1.677 (0.044)**	**1.708 (0.063)**	**1.621 (0.066)**	**1.341 (0.040)**
$(s,20,4)$	1.279 (0.025)	1.668 (0.067)	1.698 (0.061)	1.621 (0.081)	1.416 (0.042)
$(s,20,8)$	1.324 (0.027)	1.736 (0.104)	1.752 (0.066)	1.666 (0.070)	1.495 (0.050)
$(s,20,*)$	1.171 (0.015)	1.524 (0.086)	1.586 (0.057)	1.500 (0.046)	1.243 (0.022)
	$t=11.81$	$t=8.90$	$t=9.00$	$t=10.66$	$t=12.74$
$(s,160,0)$	**1.315 (0.038)**	**1.842 (0.074)**	**1.876 (0.088)**	**1.782 (0.086)**	**1.422 (0.052)**
$(s,160,1)$	1.389 (0.048)	1.810 (0.075)	1.804 (0.109)	1.780 (0.095)	1.476 (0.066)
$(s,160,2)$	1.427 (0.048)	1.799 (0.092)	1.841 (0.108)	1.771 (0.087)	1.523 (0.046)
$(s,160,4)$	1.447 (0.062)	1.792 (0.103)	1.840 (0.087)	1.764 (0.100)	1.586 (0.084)
$(s,160,8)$	1.490 (0.111)	1.840 (0.092)	1.848 (0.148)	1.795 (0.117)	1.601 (0.095)
$(s,160,*)$	1.300 (0.026)	1.594 (0.088)	1.645 (0.114)	1.588 (0.083)	1.405 (0.042)
	$t=2.57$	$t=13.82$	$t=8.12$	$t=14.65$	$t=1.36$

Table 1: The Performances of the Synchronous DNECs on the Path Planning Problems

Distributed Path Planning Problems

Algs	1	2	3	4	5
$(a,5,0)$	1.401 (0.026)	2.272 (0.043)	2.354 (0.043)	2.220 (0.034)	1.451 (0.022)
$(a,5,1)$	1.240 (0.019)	1.989 (0.040)	2.070 (0.042)	2.000 (0.037)	1.300 (0.015)
$(a,5,2)$	1.180 (0.015)	1.863 (0.035)	1.931 (0.037)	1.839 (0.043)	1.241 (0.022)
$(a,5,4)$	1.158 (0.013)	1.736 (0.044)	1.786 (0.049)	1.726 (0.047)	1.220 (0.013)
$(a,5,8)$	**1.199 (0.021)**	**1.657 (0.040)**	**1.684 (0.041)**	**1.634 (0.061)**	**1.277 (0.021)**
$(a,5,*)$	1.129 (0.017)	1.510 (0.066)	1.582 (0.060)	1.541 (0.054)	1.180 (0.018)
	$t=14.80$	$t=10.16$	$t=8.09$	$t=6.38$	$t=22.29$
$(a,10,0)$	1.261 (0.022)	2.010 (0.039)	2.084 (0.034)	2.016 (0.038)	1.328 (0.021)
$(a,10,1)$	1.162 (0.011)	1.803 (0.038)	1.858 (0.045)	1.750 (0.032)	1.230 (0.014)
$(a,10,2)$	1.168 (0.014)	1.706 (0.049)	1.775 (0.046)	1.658 (0.055)	1.246 (0.016)
$(a,10,4)$	**1.226 (0.026)**	**1.655 (0.064)**	**1.675 (0.045)**	**1.609 (0.036)**	**1.314 (0.027)**
$(a,10,8)$	1.281 (0.025)	1.653 (0.077)	1.677 (0.065)	1.600 (0.071)	1.403 (0.036)
$(a,10,*)$	1.142 (0.013)	1.541 (0.056)	1.584 (0.051)	1.483 (0.043)	1.201 (0.018)
	$t=16.64$	$t=9.05$	$t=7.33$	$t=13.68$	$t=21.84$
$(a,20,0)$	1.206 (0.019)	1.900 (0.031)	1.971 (0.035)	1.848 (0.032)	1.277 (0.023)
$(a,20,1)$	1.188 (0.017)	1.733 (0.039)	1.756 (0.044)	1.664 (0.045)	1.284 (0.029)
$(a,20,2)$	**1.219 (0.016)**	**1.695 (0.049)**	**1.693 (0.049)**	**1.626 (0.053)**	**1.332 (0.036)**
$(a,20,4)$	1.279 (0.026)	1.675 (0.071)	1.715 (0.070)	1.657 (0.071)	1.409 (0.030)
$(a,20,8)$	1.329 (0.022)	1.736 (0.101)	1.700 (0.065)	1.679 (0.100)	1.462 (0.033)
$(a,20,*)$	1.175 (0.021)	1.515 (0.076)	1.560 (0.051)	1.471 (0.036)	1.241 (0.027)
	$t=8.66$	$t=11.46$	$t=12.04$	$t=13.60$	$t=14.38$
$(a,160,0)$	**1.311 (0.029)**	**1.857 (0.068)**	**1.879 (0.095)**	**1.807 (0.071)**	**1.410 (0.047)**
$(a,160,1)$	1.381 (0.043)	1.829 (0.121)	1.847 (0.084)	1.789 (0.072)	1.493 (0.055)
$(a,160,2)$	1.432 (0.075)	1.824 (0.126)	1.824 (0.098)	1.792 (0.099)	1.494 (0.057)
$(a,160,4)$	1.468 (0.066)	1.836 (0.121)	1.839 (0.073)	1.787 (0.109)	1.560 (0.090)
$(a,160,8)$	1.516 (0.143)	1.830 (0.095)	1.856 (0.112)	1.800 (0.114)	1.591 (0.083)
$(a,160,*)$	1.291 (0.023)	1.636 (0.109)	1.671 (0.117)	1.601 (0.088)	1.407 (0.045)
	$t=3.07$	$t=10.86$	$t=7.78$	$t=10.50$	$t=0.23$

Table 2: The Performances of the Asynchronous DNECs on the Path Planning Problems

[2] Brown, D. E., Huntley, C. L., and Spillane, A. R. (1989). A Parallel Genetic Heuristic for the Quadratic Assignment Problem, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 406-415.

[3] Cobb, H. G. (1990). An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments, *NRL Memorandum Report 6760.*

[4] Cobb, H. G., and Grefenstette, J. J. (1993). Genetic Algorithms for Tracking Changing Environments, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp.523-530.

[5] Davis, Lawrence. (1991). Handbook of Genetic Algorithms. Van Nostrand Reinhold, 115 Fifth Avenue New York, NY 10003.

[6] Eshelman, L. J. and Shaffer, J. D. (1993). Real-Coded Genetic Algorithms and Interval-Schemata, in *Foundations of Genetic Algorithms II*, pp. 187-202, ed. L. Darrell Whitley, Morgan Kaufman Publishers.

[7] Goldberg, D. E. and Smith, R. E. (1987). Nonstationary Function Optimization Using Genetic Dominance and Diploidy, *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 59-68.

[8] Grefenstette, J. J. (1992). Genetic Algorithms for Changing Environments, *Proceedings of Parallel Problem Solving from Nature - PPSN II*, pp.137-144.

[9] Husbands, P., and Mill, F. (1991). Simulated Co-Evolution as The Mechanism for Emergent Planning and Scheduling, *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 264-270.

[10] Jang, J.-S. R., Sun, C.-T. and Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall.

[11] Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd Edition, Springer-Verlag.

[12] Potter, M.A., De Jong, K.A., and Grefenstette, J.J. (1995). A Coevolutionary Approach to Learning Sequential Decision Rules, *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 366-372.

[13] Sarma, J. and De Jong, K. (1999). The Behavior of Spatially Distributed Evolutionary Algorithms in Non-Stationary Environments, *Proceedings the 1999 Genetic and Evolutionary Computation Conference (GECCO-99)*, pp. 572-578.

[14] Stanley, T. J., and Mudge, T. (1995). A Parallel Genetic Algorithm for Multiobjective Microprocessor Design, *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 597-604.

[15] Syswerda, G. (1991). A Study of Reproduction in Generational and Steady-State Genetic Algorithms, *Foundations of Genetic Algorithms*, pp. 94-101.

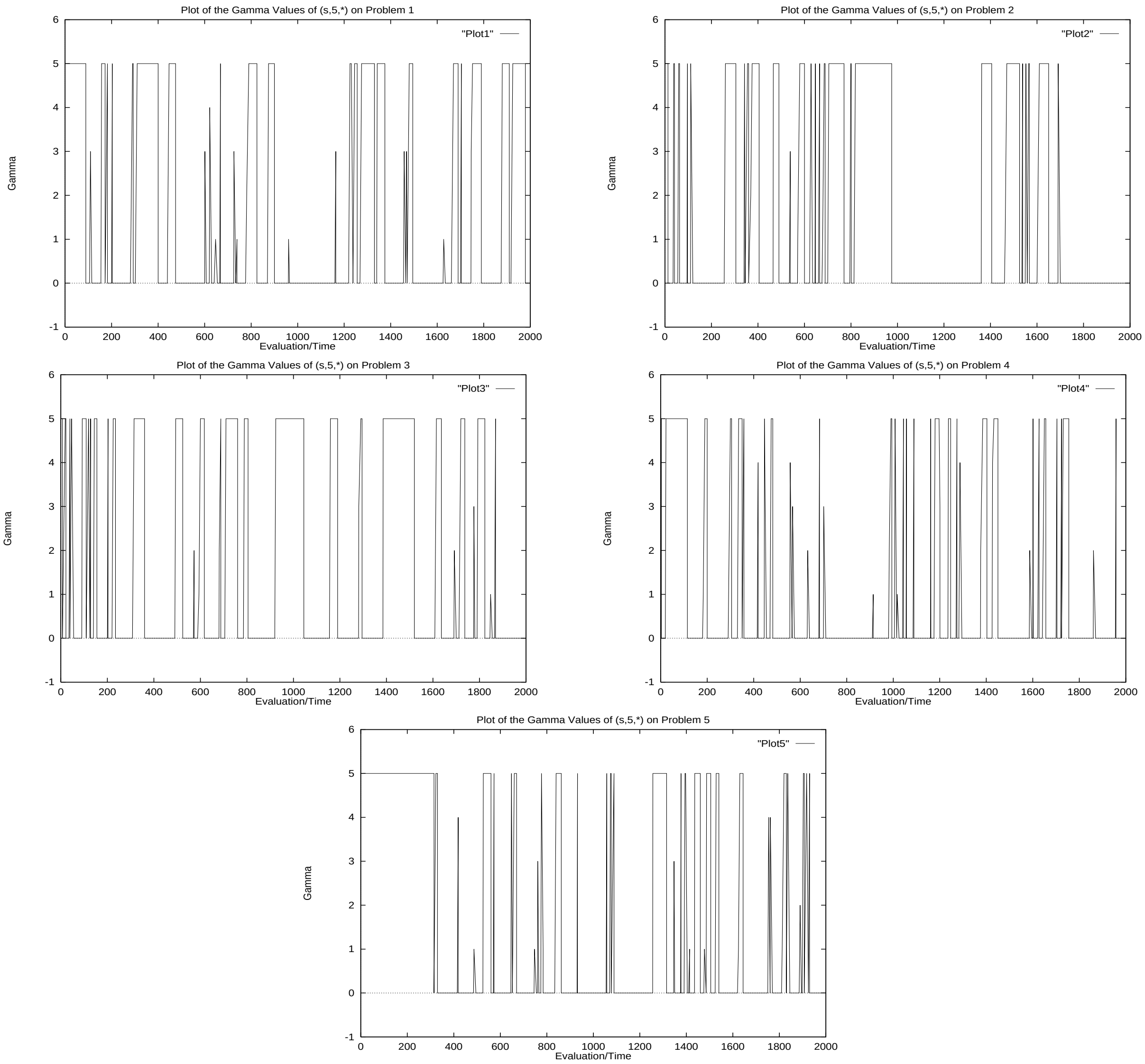

Figure 3: The γ Values for the 5 Path Planning Problems

[16] Vavak, F., Fogarty, T. C., and Jukes, K. (1996). A Genetic Algorithm with Variable Range of Local Search for Tracking Changing Environments, *Proceedings of Parallel Problem Solving from Nature - PPSN IV*, pp. 376-385.

[17] Vavak, F. and Fogarty, T.C. (1996). Comparison of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments, *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 192-195.

[18] Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K. (1997). Adaptive Evolutionary Planner/Navigator for Mobile Robots, *IEEE Transactions on Evolutionary Computation*, pp. 18-28, Vol. 1, No. 1, April 1997.