Empirical Investigation of the Benefits of Partial Lamarckianism

Christopher R. Houck – chouck@eos.ncsu.edu
Jeffery A. Joines – jjoine@eos.ncsu.edu
Michael G. Kay[1] – kay@eos.ncsu.edu
James R. Wilson – jwilson@eos.ncsu.edu

Box 7906, Dept. of Industrial Engineering
North Carolina State University
Raleigh N.C. 27695-7906
(919) 515-2008 (919) 515-1543 FAX
(Version 10-May 22, 1997)

Abstract—Genetic algorithms (GAs) are very efficient at exploring the entire search space; however, they are relatively poor at finding the precise local optimal solution in the region in which the algorithm converges. Hybrid genetic algorithms are the combination of improvement procedures, which are good at finding local optima, and genetic algorithms. There are two basic strategies for using hybrid GAs. In the first, Lamarckian learning, the genetic representation is updated to match the solution found by the improvement procedure. In the second, Baldwinian learning, improvement procedures are used to change the fitness landscape, but the solution that is found is not encoded back into the genetic string. This paper examines the issue of using partial Lamarckianism, i.e., the updating of the genetic representation for only a percentage of the individuals, as compared to pure Lamarckian and pure Baldwinian learning in hybrid GAs. Multiple instances of five bounded nonlinear problems, the location-allocation problem, and the cell formation problem were used as test problems in an empirical investigation. Neither a pure Lamarckian nor a pure Baldwinian search strategy was found to consistently lead to quicker convergence of the genetic algorithm to the best known solution for the series of test problems. Based upon a minimax criteria, i.e., minimizing the worst case performance across all test problem instances, the 20% and 40% partial Lamarckianism search strategies yielded the best mixture of solution quality and computational efficiency.

Keywords–Lamarckian Evolution, Baldwin Effect, Local Improvement Procedures, Hybrid GAs

1 – Corresponding author

1. Introduction

Genetic algorithms (GAs) are a powerful set of global search techniques that have been shown to produce very good results for a wide class of problems. Genetic algorithms search the solution space of a function through the use of simulated “Darwinian” evolution. In general, the fittest individuals of any population tend to reproduce and survive to the next generation, thus improving successive generations. However, inferior individuals can, probabilistically, survive and also reproduce. Genetic algorithms have been shown to solve linear and nonlinear problems by exploring all regions of the state space and exponentially exploiting promising areas through mutation, crossover, and selection operations applied to individuals in the population. Whereas traditional search techniques use characteristics of the problem (objective function) to determine the next sampling point (e.g., gradients, Hessians, linearity, and continuity), GAs make no such assumptions. Instead, the next sampled points are determined based on stochastic sampling/decision rules, rather than a set of deterministic decision rules. Therefore, evaluation functions of many forms can be used, subject to the minimal requirement that the function can map the population into a totally ordered set. A more complete discussion of genetic algorithms, including extensions and related topics, can be found in the books by Davis [4], Goldberg [7], Mitchell [22], and Michalewicz [20].

Hybrid genetic algorithms use local improvement procedures as part of the evaluation of individuals. For many problems there exists a well developed, efficient search strategy for local improvement, e.g., hill-climbers for nonlinear optimization. These local search strategies compliment the global search strategy of the genetic algorithm, yielding a more efficient overall search strategy. The issue of whether or not to re-code the genetic string to reflect the result of the local improvement has given rise to research into Lamarckian search strategies, i.e., re-coding the string to reflect the improvement, and Baldwinian search strategies, i.e., leaving the string unaltered. In one investigation, Whitley et al. [31] found that, for three test problems, utilizing a Baldwinian search strategy was superior in terms of finding the optimal solution for the Schwefel function, while Lamarckian search performed better on the Griewank function, and both strategies worked equally well on the Rastrigin function. Although most approaches to hybrid genetic algorithms use either a pure Lamarckian or a pure Baldwinian search strategy, neither is optimal for all problems. Therefore, this work concentrates on a partial Lamarckian strategy, where neither a pure

Lamarckian nor pure Baldwinian search strategy is used; rather, a mix of Lamarckian and Baldwinian search strategies are used.

This paper presents an empirical analysis of the use of genetic algorithms utilizing local improvement procedures to solve a variety of test problems. Section 2 describes how GAs can be hybridized by incorporating local improvement procedures into the algorithm. Incorporating local improvement procedures gives rise to the concepts of Larmarckian evolution, the Baldwin Effect, and partial Lamarckianism, all of which are explained in Section 2. Also, the concept of a one-to-one genotype to phenotype mapping is reviewed, where genotype refers to the space the GA searches while the phenotype refers to the space of the actual problem. The concepts introduced in Section 2 are applied to three different classes of problems described in Section 3: five different multimodal nonlinear function optimization problems, the location-allocation problem, and the manufacturing cell formation problem. Section 4 presents the results of a series of empirical tests of the various search strategies described in Section 2.

2. Hybridizing GAs with Local Improvement Procedures

Many researchers [1,4,16,20,27] have shown that GAs perform well for global searching because they are capable of quickly finding and exploiting promising regions of the search space, but they take a relatively long time to converge to a local optimum. Local improvement procedures, e.g., pairwise switching for permutation problems and gradient descent for unconstrained nonlinear problems, quickly find the local optimum of a small region of the search space, but are typically poor global searchers. They have been incorporated into GAs in order to improve the algorithm's performance through individual "learning." Such hybrid GAs have been used successfully to solve a wide variety of problems [4,16,27].

There are two basic models of evolution that can be used to incorporate learning into a GA: the Baldwin Effect and Lamarckian evolution. The Baldwin Effect allows an individual's fitness (phenotype) to be determined based on learning, i.e., the application of local improvement. Like natural evolution, the result of the improvement does not change the genetic structure (genotype) of the individual. Lamarckian evolution, in addition to using learning to determine an individual's fitness, changes the genetic structure of the individual to reflect the result of the learning. Both

"Baldwinian learning" and "Lamarckian learning" have been investigated in conjunction with hybrid GAs.

2.1 Baldwin Effect

The Baldwin Effect, as utilized in genetic algorithms, was first investigated by Hinton and Nolan [10] using a flat landscape with a single well representing the optimal solution. Individuals were allowed to improve by random search, which in effect transformed the landscape to include a funnel around the well. Hinton and Nolan showed that, without learning, the genetic algorithm fails; however, when hybridized with the random search, the GA is capable of finding the optimum.

Whitley et al. [31] demonstrate that "exploiting the Baldwin Effect need not require a needle in a haystack and improvements need not be probabilistic." They show how using a local improvement procedure (a bitwise steepest ascent algorithm performed on a binary representation) can in effect change the landscape of the fitness function into flat landscapes around the basins of attraction. This transformation increases the likelihood of allocating more individuals to certain attraction basins containing local optima. In a comparison of Baldwinian and Lamarckian learning, Whitley et al. [31] show that utilizing either form of learning is more effective than the standard GA approach without the local improvement procedure. They also argue that, while Lamarckian learning is faster, it may be susceptible to premature convergence to a local optimum as compared with Baldwinian learning. Three numerical optimization problems were used to test this conjecture; however, the results were inconclusive. They believe that these test functions were too easy and that harder problems may in fact show that a Baldwinian search strategy is preferred.

2.2 Lamarckian Evolution

Lamarckian evolution forces the genotype to reflect the result of the local search. This results in the inheritance of acquired or learned characteristics that are well adapted to the environment. The improved individual is placed back into the population and allowed to compete for reproductive opportunities. However, Lamarckian learning inhibits the schema processing capabilities of genetic algorithms [9,31]. Changing the genetic information in the chromosomes results in a loss of inherited schema, altering the statistical information about hyperplane partitions implicitly contained in the population. Although schema processing may not be an issue given the higher order

cardinality representations for the test problems used in this paper, the problem of premature convergence remains the same.

Baldwinian learning, on the other hand, certainly aggravates the problem of multiple genotype to phenotype mappings. A genotype refers to the composition of the values in the chromosome or individual in the population, whereas a phenotype refers to the solution that is constructed from a chromosome. In a direct mapping, there is no distinction between genotypes and phenotypes. For example, to optimize the function $5\cos(x_1) - \sin(2x_2)$, a typical representation for the chromosome would be a vector of real numbers (x_1, x_2), which provides a direct mapping to the phenotype. However, for some problems, a direct mapping is not possible or desired [24]. The most common example of this is the TSP problem with an ordinal representation. Here, the genotype is represented by an ordered list of n cities to visit, e.g., ($c_{[1]}$,$c_{[2]}$,...,$c_{[n]}$). However, the phenotype of the TSP is a tour, and any rotation of the chromosome yields the same tour; thus, any rotation of a genotype results in the same phenotype. For example, the two tours (1,2,3,4) and (3,4,1,2) have different genotypes since their gene strings are different, but both strings represent the same tour and thus have the same phenotype.

It has been noted that having multiple genotypes map to the same phenotype may confound the GA [12,24]. This problem also occurs when a local improvement procedure is used in conjunction with a GA. Consider the example of optimizing $\sin(x)$. Suppose a simple gradient based LIP is used to determine the fitness of a chromosome. Then any genotype in the range [$-\pi/2$, $3\pi/2$] will have the same phenotype value of 1 at $\pi/2$.

2.3 Partial Lamarckian

Hybrid genetic algorithms need not be restricted to operating in either a pure Baldwinian or pure Lamarckian manner. Instead, a mix of both strategies, or what is termed partial Lamarckianism, could be employed together. For example, a possible strategy is to update the genotype to reflect the resulting phenotype in 50% of the individuals. Clearly, a 50% partial Lamarckian strategy has no justification in natural evolution but, for simulated evolution, it provides for an additional search strategy.

Orvosh and Davis [26] advocated the use of a 5% rule for updating individuals when employing repair functions in genetic algorithms to solve constrained optimization problems. All infeasible

solutions generated by the genetic algorithm are repaired to the feasible domain in order to determine their fitness. The 5% rule dictates that 5% of the infeasible individuals have their genetic representation updated to reflect the repaired feasible solution. This partial updating was shown on a set of combinatorial problems to be better than either no updating or always updating. However, Michalewicz and Nazhiyath [21] determined that a higher percentage, 20–25%, of update did better when using repair functions to solve continuous constrained nonlinear programming problems.

Previous research has concentrated either on the comparison of pure Lamarckian and pure Baldwinian search, or the effectiveness of partial repair for constrained optimization. This work examines the use of partial Lamarckianism with regard to the use of local search on a set of bounded nonlinear and combinatorial optimization test problems.

3. Experimentation

To investigate the trade-off between disrupted schema processing, (or premature convergence to a local optima) in Lamarckian learning and multiple genotype mapping to the same phenotype in Baldwinian learning, a series of experiments using local improvement procedures as evaluation functions were run on several different test problems. For each test problem, the GA was run with varying levels of Lamarckianism, i.e., updating the genotype to reflect the phenotype, as well as a pure genetic approach (i.e., no local improvement). The GA was run with no Lamarckianism (i.e., a pure Baldwinian search strategy), denoted as 0% update, as well as 100% Lamarckianism (i.e., a pure Lamarckian search strategy). Combinations of both strategies (i.e., partial Lamarckian search strategies) were used to determine if there was a trend between the two extremes, and if combinations of Baldwinian learning and Lamarckian learning were beneficial. In these experiments, individuals were updated to match the resulting phenotype with a probability of 5, 10, 20, 40, 50, 60, 80, 90, and 95%. These values were chosen in order to examine the effects at a broad variety of levels, but focusing attention at each extreme. For each problem instance, the GA was replicated 30 times, with common random seeds across the different search strategies. The genetic algorithm and parameters used in these experiments are described in Appendix A. The genetic algorithm is of a form advocated by Michalewicz [20] and uses a floating point representation with three real-valued crossovers and five real-valued mutations. However, for the cell formation problems, the GA is modified to work with integer variables. The GA was terminated when either

the optimal solution was found or after one million function evaluations were performed. Using function evaluations takes into account the cost of using a LIP (i.e., the number of function evaluations generated by the LIP), thus allowing a direct comparison between the hybrid GA methods and the pure GA (i.e., no local improvement).

Multiple instances of seven different test problems were used in this investigation to ensure that any effect observed (1) held across different classes of problems as well as different sized instances of the problem and (2) was not due to some unknown structure of the specific class of problem selected, specific instance chosen, or the particular local improvement procedure used. The first five test problems are bounded nonlinear optimization test problems taken from the literature, while the last two problems are the location-allocation and manufacturing cell formation problems. For each of the nonlinear optimization problems, three different size instances (n = 2, 10, 20) were used in the study. For both the location-allocation and cell formation problems, two different sized instances were used.

3.1 Brown's Almost Linear Function

The first test problem comes from a standard test set of nonlinear optimization problems [23] and is as follows:

$$f(x) = \sum_{i=1}^{n} (f_i(x))^2, \qquad \text{for } x_i \in [-25, 25] \tag{1}$$

where

$$f_i(x) = \begin{cases} x_i + \sum_{j=1}^{n} x_j - (n+1), & \text{for } i = 1, \ldots, n-1 \\ \prod_{j=1}^{n} x_j - 1, & \text{for } i = n \end{cases}$$

The function is not linearly separable and has the basic form of a nonlinear least squares problem. Three different instances of this problem were used for this investigation: a 2-dimensional

instance (Brown-2), a 10-dimensional instance (Brown-10), and a 20-dimensional instance (Brown-20).

3.2 Corana Function

The next test problem consists of three instances of the modified Corana problem: a 2-, 10-, and 20-dimensional instance (Corana-2, Corana-10, and Corana-20, respectively). This problem comes from a family of continuous nonlinear multimodal functions developed by Corana et al. [3] to test the efficiency and effectiveness of simulated annealing as compared to stochastic hill-climbing and Nelder-Mead optimization methods. This parametrized family of test functions contains a large number of local minima. The function can be described as an *n*-dimensional parabola with rectangular pockets removed and with the global optimum always occurring at the origin. These functions are parameterized with the number of dimensions (*n*), the number of local optima, and the width and height of the rectangular pockets. As formulated in Corana et al. [3], the function proved to be too easy: all the learning methods found the optimal solution every time with the same amount of computational effort. Therefore, the first condition in the definition of $g(x_i)$ below was modified so that rectangular troughs, instead of pockets, are removed from the function. This was accomplished by eliminating the restriction that all, instead of any, x_i satisfy $k_i s_i - t_i \le x_i \le k_i s_i + t_i$ and all k_i are not zero. The modified Corana function is as follows:

$$f(x) = \sum_{i=1}^{n} c_i g(x_i), \qquad \text{for } c_i \in \bar{c} \text{ and } x_i \in [-10000, 10000] \tag{2}$$

where

$$g(x_i) = \begin{cases} 0.15 z_i^2, & \text{if } k_i s_i - t_i \le x_i \le k_i s_i + t_i, \text{ for any integer } k_i \\ x_i^2, & \text{otherwise} \end{cases}$$

$$z_i = \begin{cases} k_i s_i + t_i & \text{if } k_i < 0 \\ 0 & \text{if } k_i = 0 \\ k_i s_i - t_i & \text{if } k_i > 0 \end{cases}$$

$$\bar{c} = (1, 1000, 10, 100, 1, 10, 100, 1000, 1, 10, \ldots \\ 100, 1000, 1, 10, 100, 1000, 1, 10, 100, 1000)$$

$$s_i = 0.2$$

$$t_i = 0.05$$

3.3 Griewank Function

The Griewank function is the third test problem, again with three instances of 2, 10, and 20 dimensions, respectively (Griewank-2, Griewank-10, and Griewank-20). The Griewank function is a bounded optimization problem used by Whitley et al. [31] to compare the effectiveness of Lamarckian and Baldwinian search strategies. The Griewank function, like Brown's almost linear function, is not linearly separable, and is as follows:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) - 1, \text{ for } x_i \in [-512, 511] \quad (3)$$

3.4 Rastrigin Function

The next test problem, with three instances of 2, 10, and 20 dimensions (Rastrigin-2, Rastrigin-10, and Rastrigin-20), is also a bounded minimization problem taken from Whitely et al. [31]. The functional form is as follows:

$$f(x) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right), \text{ for } x_i \in [-5.12, 5.11] \quad (4)$$

3.5 Schwefel Function

The last nonlinear test problem, with three instances of 2, 10, and 20 dimensions (Schwefel-2, Schwefel-10, and Schwefel-20), is also a bounded minimization problem taken from Whitley et al. [31]. The functional form is as follows:

$$f(x) = nV + \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right), \text{ for } x_i \in [-512, 511] \quad (5)$$

where

$$V = -\min\{-x\sin(\sqrt{|x|}) : x \in [-512, 511]\}$$

V depends on system precision; for our experiments, V = 418.9828872721625.

3.6 Sequential Quadratic Programming

For these five nonlinear optimization test problems, a sequential quadratic programming (SQP)[1] optimization method was used as the LIP. SQP is a technique used to solve nonlinear programming problems of the following form:

$$\text{Minimize } f(x), \qquad \text{s.t. } x \in X$$

where, for n_i inequality and n_e equality constraints, X is the set of points $X \in \Re^n$ satisfying

$$bl \le x \le bu$$
$$g_j(x) \le 0 \qquad j = 1, \ldots, n_i$$
$$h_j(x) = 0 \qquad j = 1, \ldots, n_e$$

with $bl \in \Re^n$; $bu \in \Re^n$; $f : \Re^n \to \Re$ smooth; $g_j : \Re^n \to \Re, j = 1, \ldots, n_i$ nonlinear and smooth; and $h_j : \Re^n \to \Re, j = 1, \ldots, n_e$ nonlinear and smooth.

SQP works by first computing the descent direction by solving a standard quadratic program using a positive definite estimate of the Hessian of the Lagrangian. A line search is then performed to find the feasible minimum of the objective along the search direction. This process is repeated until the algorithm converges to a local minimum, or exceeds a certain number of iterations. For the purposes of this study, SQP was stopped after 25 iterations if it had not found a local optimum. A full description of the SQP used in this study can be found in [17].

3.7 Location-Allocation Problem

The continuous location-allocation problem, as described in [13], is used as the next test problem. The location-allocation (LA) problem, a type of nonlinear integer program, is a multifacility location problem in which both the location of *n* new facilities (NFs) and the allocation of the flow requirements of *m* existing facilities (EFs) to the new facilities are determined so that the total transportation costs are minimized. The location-allocation problem is a difficult optimization problem because its objective function is neither convex nor concave, resulting in multiple local minima [2]. Optimal solution techniques are limited to small problem instances (less than 25 EFs

1. SQP is available in the Matlab environment [8], and as Fortran and C code [17].

for general l_p distances [18] and up to 35 EFs for rectilinear distances [19]). Two rectilinear distance instances of this problem were used in the investigation: a 200 EF by 20 NF instance (LA-200) and a 250 EF by 25 NF instance (LA-250), both taken from [13].

Given m EFs located at known points a_j, $j = 1,..., m$, with associated flow requirements w_j, $j = 1,..., m$, and the number of NFs, n, from which the flow requirements of the EFs are satisfied, the location-allocation problem can be formulated as the following nonlinear program:

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} d(X_i, a_j) \tag{6}$$

$$\text{subject to} \quad \sum w_{ij} = w_j, \qquad j = 1, \ldots, m$$

$$w_{ij} \geq 0, \qquad i = 1, \ldots, n, j = 1, \ldots, m$$

where the unknown variable locations of the NFs, X_i, $i = 1,..., n$, and the flows from each NF i to each EF j, w_{ij}, $i = 1,..., n$, $j = 1,..., m$, are the decision variables to be determined, and $d(X_i, a_j)$ is the distance between NF i and EF j. An infinite number of locations are possible for each NF i since each X_i is a point in a continuous, typically two-dimensional, space.

The genetic algorithm approach to solve this problem, described in detail in [13], is used in these experiments. The individuals in the GA are a vector of real values representing the starting locations for the new facilities. Based upon these locations for the new facilities, the optimal allocations are found by allocating each existing facility to the nearest new facility. The total cost of the resulting network is then the weighted sum of the distances from each existing facility to its allocated new facility.

The alternate location-allocation method developed by Cooper [2] is used as the LIP for this problem. This method quickly finds a local minimum solution given a set of starting locations for the new facilities. This procedure works by starting with a set of new facility locations; it then determines an optimal set of allocations based on those locations, which for this uncapacitated version of the problem reduces to finding the closest new facility to each existing facility. The optimal new facility locations for these new allocations are then determined by solving n single facility

location problems, which is in general a nonlinear convex optimization problem. This method continues until no further allocation changes can be made.

3.8 Cell Formation Problem

The manufacturing cell formation problem, as described in [14], is the final test problem. The cell formation problem, a type of nonlinear integer programming problem, is concerned with assigning a group of machines to a set of cells and assigning a series of parts needing processing by these machines into families. The goal is to create a set of autonomous manufacturing units that eliminate intercell movements (i.e., a part needing processing by a machine outside its assigned cell). Two instances of this problem were in the investigation: a 22 machine by 148 part instance (Cell-1) and a 40 machine by 100 part instance (Cell-2).

Consider an m machine by n part cell formation problem instance with a maximum of k cells. The problem is to assign each machine and each part to one and only one cell and family, respectively. Joines et al. [14] developed an integer programming model with the following set variable declarations that eliminate the classical binary assignment variables and constraints:

$x_i = l$, if machine i is assigned to cell l

$y_j = l$, if part j is assigned to family l

The model was solved using a GA. In the GA, the individuals are a vector of integer variables representing the machine and part assignments ($x_1, x_2,..., x_m, y_1, y_2,..., y_n$). This representation has been shown to perform better than other cell formation methods utilizing the same objective [14]. The following "grouping efficacy" measure (Γ) is used as the evaluation function:

$$\text{Maximize } \Gamma = \frac{e - e_o}{e + e_v} \tag{7}$$

where

$$e = e_o + e_v$$

$$e_v = \sum_{i=1}^{k} D_i - d_i$$

Grouping efficacy tries to minimize a combination of both the number of exceptional elements, e_o (i.e., the number of part operations performed outside the cell), and the process variation, e_v (i.e., the discrepancy of the parts' processing requirements and the processing capabilities of the cell). Process variation for each cell *i* is calculated by subtracting from the total number of operations a cell can perform, D_i, the number part operations that are performed in the cell, d_i. The total process variation is the sum of the process variation for each cell. Exceptional elements decrease Γ since the material handling cost for transportation within a cell is usually less than the cost of the transportation between cells. Process variation degrades the performance of the cell due to increased scheduling, fixturing, tooling, and material handling costs.

A greedy one-opt switching method, as described in [16], was used as the LIP. Given an initial assignment of machines and parts to cells and families, respectively, the LIP uses a set of improvement rules developed by Ng [25] to determine the improvement in the Γ objective function if machine *i* is switched from its current cell to any of the remaining $k - 1$ cells. The procedure repeats this step for each of the *m* machines as well as each of the *n* parts, where each part is switched to $k - 1$ different families.

4. Results

For the 30 replications of each problem instance, both the mean number of function evaluations and the mean final function values were determined. To ensure the validity of the subsequent analysis of variance (ANOVA) and multiple means tests, each data set was tested first for normality and second for homogeneity of the variances among the different search strategies. The Shapiro-Wilk test for normality was performed on the results at $\alpha = 0.05$. Since there is no established test for variance homogeneity [29], visual inspection was used to determine if there were significant variance differences. If there were significant differences and there existed a strong linear relationship between the means and standard deviations (i.e., a linear regression correlation greater than 0.8), a logarithmic transformation was used [29]. For the number of functional evaluations, standard deviations equal to zero (which correspond to the search strategy terminating after one million functional evaluations), were removed from the linear regression since these values were not due to randomness.

For those results passing the Shapiro-Wilk test for normality, an ANOVA was conducted for both the number of function evaluations and the final functional value. If the ANOVA showed a significant effect, $\alpha > 0.01$, the means of each of the 12 different search strategies considered, namely, no local improvement (referred to as –1), pure Baldwinian (referred to 0), pure Larmarckian (100), and nine levels of partial Lamarckian (referred to as 5, 10, 20, 40, 50, 60, 80, 90, and 95, respectively), were compared using three different statistical methods. The three methods, each performing a multiple means comparison, were the Duncan approach to minimize the Bayesian loss function [5]; Student-Newman-Keuls (SNK), a multiple range test using a multiple stage approach; and an approach developed by Ryan [28], Einot and Gabriel [6], and Welsch [30] (REGW), which is also a multiple stage approach that controls the maximum experimentwise error rate under any complete or partial hypothesis. Each of the tests were conducted using the general linear models routine in SAS v6.09 [29]. All three of these tests were used because there is no consensus test for the comparison of multiple means.

The ranking of both the number of function evaluations to reach the optimal solution and the ranking of the final fitness are given in Tables I through VII for each of the 19 test problem instances considered in this paper: the five nonlinear test problems (Brown, Corana, Griewank, Rastrigin, and Schwefel), each at 2, 10 and 20 dimensions; the two location-allocation problems (LA-200 and LA-250); and the two cell formation problems (Cell-1 and Cell-2). Each of the three multiple means comparisons, Duncan's, SNK, and REGW, were performed on each problem instance. In the tables, if all three of the multiple means comparison methods do not agree, the results of each are presented; otherwise, the combined results for all of the methods agreeing are presented. The means and standard deviations of the final fitness value and the number of functional evaluations for each search strategy are shown in Tables XI-XVII of Appendix B for all test problem instances.

The multiple means composition methods are statistical tests that provide information on sets of means whose differences are statistically significant. For example, consider the results in Table I for the Brown-10 instance with respect to final fitness. All three multiple means tests agree that the –1 (no local improvement) search strategy yields significantly worse results than any of the other strategies. Similarly, the Baldwinian strategy (0% Lamarckian) is significantly better than the –1 strategy, but significantly worse than any of the other strategies. Finally, the other strategies

(5–100% Lamarckian) yield significantly the same final fitness value. In some instances, the tests are unable to place a level into one group and range of groups is presented. For example, the 20% partial Lamarckian strategy for the Brown-20 instance (Table I) belongs to two both groups C and D (i.e., C-D). The strategies assigned to the best group are shaded. The rank indicates the rank of the various means, and moving from left to right, the means get better. Each of the test problems is discussed in detail in the following sections.

Table I: Multiple Means Comparison for Brown-'s Almost Linear Function

Prob. Inst.	Crit.													
Brown-2	#Evals	Rank	–1	0	5	10	20	40	50	60	80	90	95	100
		All	A	B	B	B	B	B	B	B	B	B	B	B
	Fitness	Rank	–1	0	5	10	20	40	50	60	80	90	95	100
		All	A	A	A	A	A	A	A	A	A	A	A	A
Brown-10	#Evals	Rank	–1	0	5	10	20	40	50	60	80	90	95	100
		All	A	B	C	D	E	F	F	F	F	F	F	F
	Fitness	Rank	–1	0	5	10	20	40	50	60	80	90	95	100
		All	A	B	C	C	C	C	C	C	C	C	C	C
Brown-20	#Evals	Rank	0	–1	5	10	20	40	60	80	50	100	95	90
		Duncan	A	A	B	C	C	D	D	D	D	D	D	D
		SNK/REGW	A	A	B	C	C-D	D	D	D	D	D	D	D
	Fitness	Rank	–1	0	5	100	20	40	50	60	95	90	80	10
		All	A	B	C	C	C	C	C	C	C	C	C	C

4.1 Brown's Almost Linear Function

This problem benefits greatly from the use of SQP. As shown in Table I, the optimal solution was readily found by all partial Lamarckian search strategies employing SQP as their local improvement procedure. The GA employing no local improvement took considerable longer to find the optimal solution for the 2-dimensional instance and failed to find the optimal for the other two instances. The pure Baldwinian search also terminated after one million function evaluations for the larger instances. For the 2-dimensional instance, there is no statistically significant difference in the number of function evaluations required to obtain the optimal solution for all strategies employing SQP. However, the 10- and 20-dimensional instances show that, while the strategies employing a lower percentage of partial Lamarckianism find the optimal, they require a signifi-

cantly greater number of function evaluations.

Table II: Multiple Means Comparison for Corana Function

Prob. Inst.	Crit.												
Corana-2	#Evals	Rank	–1	40	0	50	100	5	90	95	10	20	80	60
		All	A	B	B	B	B	B	B	B	B	B	B	B
	Fitness	Rank	–1	5	0	10	20	40	50	60	80	90	95	100
		All	A	A	A	A	A	A	A	A	A	A	A	A
Corana-10	#Evals	Rank	0	5	–1	10	20	40	50	60	90	80	100	95
		Duncan	A	A	A	A	B	C	C	D	D-E	D-E	D-E	E
		SNK/REGW	A	A	A	A	B	C	C	D	D	D	D	D
	Fitness	Rank	0	5	10	–1	20	40	50	60	80	90	95	100
		All	A	B	C	C	D	D	D	D	D	D	D	D
Corana-20	#Evals	Rank	0	40	20	50	5	10	–1	60	80	100	90	95
		Duncan	A	A	A	A	A	A	A	A	A-B	B	B	C
		SNK	A	A	A	A	A	A	A	A-B	A-B	B-C	B-C	C
		REGW	A	A	A	A	A	A	A-B	A-C	A-C	B-D	C-D	D
	Fitness	Rank	0	5	10	20	40	-1	50	60	80	90	95	100
		Duncan/SNK	A	B	C	D	E	F	F	G	H	H-I	I	I
		REGW	A	B	C	D	E	F	F-G	H	H	H-I	H-I	I

4.2 Corana Function

As shown in Table II, the 2-dimensional Corana function instance is easily solved by all variants of the GA. The 10-dimensional instance shows an interesting result. Only the higher levels of partial Lamarckianism were able to find the optimal solution, while pure Baldwinian search found the worst solutions. For the 20-dimensional instance, no strategy was able to locate the optimal value every time; however, the final function value shows more statistically significant variation (i.e., the lower the percentage of Lamarckian learning, the worse the final functional value until 90% is reached, after which all strategies yield similar results). The 90, 95, and 100% Lamarckian strategies found the optimal approximately half the time. It is interesting to note that the GA not employing local search (–1) outperforms the lower Lamarckian levels including pure Baldwinian learning. A possible explanation for this is that the Corana function exhibits numerous local minima and extreme discontinuity wherever a rectangular local minimum has been placed in the

hyper-dimensional parabola, making this problem troublesome for SQP. Therefore, the other strategies waste valuable computational time mapping the same basin of attraction.

Table III: Multiple Means Comparison for Griewank Function

Prob. Inst.	Crit.												
Griewank-2	#Evals	Rank	50	0	80	5	40	90	10	60	100	20	95	-1
		Duncan	A	A	A	A	A	A-B	A-B	A-B	A-B	A-B	A-B	B
		SNK/REGW	A	A	A	A	A	A	A	A	A	A	A	A
	Fitness	Rank	–1	80	90	10	100	20	50	5	0	60	95	40
		All	A	B	B	B	B	B	B	B	B	B	B	B
Griewank-10	#Evals	Rank	–1	100	95	90	80	60	50	40	20	10	5	0
		Duncan	A	B	B	B-C	C	D	D	E	F	F-G	G	G
		SNK	A	B	B-C	B-C	C	D	D	E	F	F-G	G	G
		REGW	A	B	B	B	B	C	C-D	D	E	E-F	E-F	F
	Fitness	Rank	–1	5	0	10	20	40	50	60	80	90	95	100
		All	A	B	B	B	B	B	B	B	B	B	B	B
Griewank-20	#Evals	Rank	–1	100	95	90	80	60	50	40	20	10	5	0
		Duncan	A	B	B	B-C	B-C	B-D	C-D	D-E	E-F	F	F	F
		SNK	A	B	B	B	B-C	B-C	B-C	C-D	D-E	D-E	E	E
		REGW	A	B	B	B-C	B-C	B-D	B-D	C-E	D-F	E-F	E-F	F
	Fitness	Rank	–1	5	0	10	20	40	50	60	80	90	95	100
		All	A	B	B	B	B	B	B	B	B	B	B	B

4.3 Griewank Function

This problem benefits greatly from the use of SQP. As shown in Table III, the optimal solution was readily found by all variants of the GA. While the quality of the solutions was similar, only Duncan's method found any difference for the 2-dimensional instance, with the GA employing no local improvement and several higher levels of Lamarckianism being more efficient. For the 10- and 20-dimensional instances, the optimal solution was readily found by all search strategies employing SQP as their local improvement procedure; whereas, the no local improvement GA terminated after one million function evaluations. However, there is a statistically significant difference in the number of function evaluations required. Unlike all other problems, the greater

amount of Lamarckianism, the greater number of function evaluations required to locate the optimal solution.

4.4 Rastrigin Function

As shown in Table IV, all strategies found the optimal solution for the 2-dimensional instance; however, no local improvement took significantly longer. Similarly, all strategies, except for pure Baldwinian search, found the optimal solution for the 10-dimensional instance. There is statistically significant evidence that moderate and high levels of Lamarckianism (> 50%) yield quicker convergence. For the 20-dimensional instance, a similar trend is present. All strategies except pure Baldwinian search yielded optimal results; for this size instance, a Lamarckian level of greater than 40% yields the quickest convergence. It should be noted that no local improvement outperformed several of the lower levels including pure Baldwinian search. As the problem size increases, more computational time is used in the initialization of the population, not allowing Baldwinian learning to overcome the problem of a one-to-one genotype to phenotype mapping before it terminates.

Table IV: Multiple Means Comparison for Rastrigin Function

Prob. Inst.	Crit.													
Rastrigin-2	#Evals	Rank	-1	100	90	10	0	95	50	5	20	80	40	60
		All	A	B	B	B	B	B	B	B	B	B	B	B
	Fitness	Rank	–1	5	0	10	20	40	50	60	80	90	95	100
		All	A	A	A	A	A	A	A	A	A	A	A	A
Rastrigin-10	#Evals	Rank	0	5	10	-1	20	40	50	60	80	90	100	95
		All	A	B	B	C	C	D	D-E	E	E	E	E	E
	Fitness	Rank	0	-1	5	10	20	40	50	60	80	90	95	100
		All	A	B	B	B	B	B	B	B	B	B	B	B
Rastrigin-20	#Evals	Rank	0	5	10	20	-1	40	60	50	100	80	95	90
		Duncan	A	B	C	D	E	E-F	E-G	E-G	F-G	F-G	G	G
		SNK/REGW	A	B	C	D	E	E-F	E-F	E-F	E-F	E-F	F	F
	Fitness	Rank	–1	5	0	10	20	40	50	60	80	90	95	100
		All	A	B	B	B	B	B	B	B	B	B	B	B

4.5 Schwefel Function

As shown in Table V, every strategy found the optimal solution for the 2-dimensional instance; with no local improvement taking significantly longer . The 10- and 20-dimensional instances show faster convergence for greater degrees of Lamarckianism. Low levels of Lamarckianism do not consistently yield optimal solutions while, with 20% or greater Lamarckianism, the optimal solution is consistently located.

Table V: Multiple Means Comparison for Schwefel Function

Prob. Inst.	Crit.													
Schwefel-2	#Evals	Rank	–1	100	90	95	80	50	40	60	20	10	0	5
		All	A	A	A	A	A	A	A	A	A	A	A	A
	Fitness	Rank	–1	5	0	10	20	40	50	60	80	90	95	100
		All	A	A	A	A	A	A	A	A	A	A	A	A
Schwefel-10	#Evals	Rank	0	5	10	–1	20	40	50	60	80	90	100	95
		Duncan	A	B	B	B-C	B-C	C	C	C	C	C	C	C
		SNK	A	B	B	B-C	B-C	B-C	C	C	C	C	C	C
		REGW	A	B	B-C	B-D	B-D	B-D	C-D	D	D	D	D	D
	Fitness	Rank	0	–1	5	10	20	40	50	60	80	90	95	100
		All	A	B	B	B	B	B	B	B	B	B	B	B
Schwefel-20	#Evals	Rank	0	-1	5	10	20	60	40	50	100	80	90	95
		Duncan	A	B	B	B-C	B-D	C-E	C-E	D-E	D-E	D-E	E	E
		SNK	A	B	B-C	B-C	B-D	C-D	C-D	C-D	C-D	C-D	D	D
		REGW	A	B	B-C	B-D	B-E	B-E	C-E	D-E	D-E	D-E	E	E
	Fitness	Rank	0	-1	5	10	20	40	50	60	80	90	95	100
		All	A	B	B	B	B	B	B	B	B	B	B	B

4.6 Location-Allocation Problem

As shown in Table VI, both location-allocation problem instances showed the same trend seen in the second cell formation problem instance (Cell-2 in Table VII). With any use of local improvement, the GA found the optimal solution before the one million function evaluation ceiling was

reached. However, the Baldwinian strategy (0%) required a significantly greater number of function evaluations than any strategy employing any form of Lamarckianism.

Table VI: Multiple Means Comparison for Location-Allocation Problem

Prob. Inst.	Crit.													
LA-200	#Evals	Rank	–1	0	5	10	20	90	40	60	100	50	80	95
		Duncan	A	B	C	C-D	D-E	E	E	E	E	E	E	E
		SNK/REGW	A	B	C	C-D	C-D	D	D	D	D	D	D	D
	Fitness	Rank	–1	0	5	10	20	40	50	60	80	90	95	100
		All	A	B	C	C	C	C	C	C	C	C	C	C
LA-250	#Evals	Rank	–1	0	5	10	20	100	40	90	50	60	95	80
		Duncan	A	B	C	C-D	C-D	D-E	D-E	D-E	D-E	D-e	D-E	E
		SNK	A	B	C	C	C	C	C	C	C	C	C	C
		REGW	A	B	C	C-D	C-D	C-D	C-D	C-D	C-D	C-D	C-D	D
	Fitness	Rank	–1	0	5	10	20	40	50	60	80	90	95	100
		All	A	B	C	C	C	C	C	C	C	C	C	C

4.7 Cell Formation Problem

As shown in Table VII, the first cell formation problem instance, Cell-1, is a case where using the local improvement routine with any amount of Lamarckian learning resulted in finding the optimal solution; whereas, when using no local improvement or a strictly Baldwinian search strategy, the algorithm did not find the optimal solution and terminated after one million function evaluations, with the Baldwinian strategy finding better solutions than the no local improvement strategy. The second cell formation instance, Cell-2, is more interesting. In terms of fitness, any strategy employing a moderate amount of Lamarckianism (>20%) found the optimal solution, whereas, the strategies employing either a Baldwinian strategy or using no local improvement did not consistently find the optimal solution. This problem instance also shows the familiar pattern of increasing levels of Lamarckian learning resulting in fewer function evaluations to reach the optimal solution. A possible explanation for the poor performance of a Baldwinian search strategy is that most of the one million function evaluations are used in the initialization of the population. Therefore, a strategy employing pure Baldwinian learning does not have enough functional evaluations left to overcome the multiple genotype to phenotype mapping problem.

Table VII: Multiple Means Comparison for Cell Formation Problem

Prob. Inst.	Crit.													
Cell-1	#Evals	Rank	0	–1	50	80	40	5	60	20	10	90	100	95
		Duncan	A	A	B	B	B-C	B-C	B-D	B-D	B-D	B-D	C-D	D
		SNK	A	A	B	B	B	B	B	B	B	B	B	B
		REGW	A	A	B	B-C	B-C	B-C	B-C	B-C	B-C	B-C	B-C	C
	Fitness	Rank	–1	0	5	10	40	80	20	90	50	95	60	100
		All	A	B	C	C	C	C	C	C	C	C	C	C
Cell-2	#Evals	Rank	0	–1	5	10	20	60	40	50	95	90	80	100
		Duncan	A	A	A-B	A-C	B-D	B-D	B-D	B-D	C-D	D	D	D
		SNK	A	A	A-B	A-C	A-C	A-C	A-C	B-C	B-C	C	C	C
		REGW	A	A-B	B	B-C	B-C	B-C	B-C	B-C	B-C	B-C	B-C	C
	Fitness	Rank	–1	0	5	20	10	40	50	60	90	80	100	95
		Duncan	A	B	C	C	C-D	C-D	C-D	C-D	C-D	C-D	D	D
		SNK	A	B	C	C	C	C	C	C	C	C	C	C
		REGW	A	A	B	B	B	B	B	B	B	B	B	B

4.8 Summary of Results

To concisely present the main results of these statistical tests shown in detail in Tables I–VII, Tables VIII and IX show rankings of each search strategy with respect to the fitness of the final result and the number of function evaluations required, respectively. For each strategy, the rankings were determined by finding the group which yielded the best results for each test problem instance as determined by the SNK means analysis. If the strategy was placed into a single group, that group's rank was used; however, if the method was placed into several overlapping groups, the method was placed into the group with the best rank. Therefore, these rankings represent how many groups of means are significantly better for each test problem instance. The SNK multiple means test was arbitrarily chosen for these rankings.

Table VIII shows the rankings for the final fitness of the solutions returned by each of the search

Table VIII: Rank of Solutions (SNK)

Problem Instance	Search Strategy											
	–1	0	5	10	20	40	50	60	80	90	95	100
Brown-2†	1	1	1	1	1	1	1	1	1	1	1	1
Brown-10	3	2	1	1	1	1	1	1	1	1	1	1
Brown-20	3	2	1	1	1	1	1	1	1	1	1	1
Corana-2†	1	1	1	1	1	1	1	1	1	1	1	1
Corana-10	2	4	3	2	1	1	1	1	1	1	1	1
Corana-20	4	2	8	7	6	5	4	3	2	1	1	1
Griewank-2†	2	1	1	1	1	1	1	1	1	1	1	1
Griewank-10	2	1	1	1	1	1	1	1	1	1	1	1
Griewank-20	2	1	1	1	1	1	1	1	1	1	1	1
Rastrigin-2†	1	1	1	1	1	1	1	1	1	1	1	1
Rastrigin-10†	2	1	1	1	1	1	1	1	1	1	1	1
Rastrigin-20	2	1	1	1	1	1	1	1	1	1	1	1
Schwefel-2†	1	1	1	1	1	1	1	1	1	1	1	1
Schwefel-10	1	2	1	1	1	1	1	1	1	1	1	1
Schwefel-20	1	2	1	1	1	1	1	1	1	1	1	1
LA-200	3	2	1	1	1	1	1	1	1	1	1	1
LA-250	3	2	1	1	1	1	1	1	1	1	1	1
Cell-1	3	2	1	1	1	1	1	1	1	1	1	1
Cell-2	2	2	1	1	1	1	1	1	1	1	1	1

†- Failed Shapiro-Wilk test, or ANOVA did not show significant effect at $\alpha = 0.01$

strategies, where 1 represents the best rank and is shaded, 2 represents the next best rank, etc. All of the strategies employing at least 20% Lamarckian learning consistently found the optimal solution. However, the pure Baldwinian (0%) and the 5 and 10% partial Lamarckian strategies did

find significantly worse solutions for several test problem instances. The GA employing no local improvement procedure (–1) is included to provide a comparison with how efficiently the local search procedure utilizes function evaluations as compared to a pure genetic sampling approach. For most of these test problem instances, the use of the local improvement procedure (LIP) significantly increases the efficiency of the genetic algorithm.

Table IX shows the rankings for the number of function evaluations required to locate the final functional value returned by each of the search strategies. The table shows an interesting trend: neither the pure Lamarckian (100) nor pure Baldwinian (0) strategies consistently yielded the best performance; this was also observed by Whitley et al. [31]. Using a binary representation and a bitwise steepest ascent LIP, Whitley et al. [31] demonstrated that a Baldwinian search strategy was preferred for the 20-dimensional Schwefel function while the results in Table IX show a preference for a Lamarckian search strategy when using a floating point representation and SQP. Also, a Lamarckian strategy was preferred in Whitley et al. [31] for the 20-dimensional Griewank instance, whereas in this study a Baldwinian search strategy is preferred.

The results of this study and those of Whitley et al. [31], together, show that a combination of the problem, the representation, and the local improvement procedure (LIP) all can influence the effectiveness of either a pure Lamarckian or pure Baldwinian search strategy. However, with respect to the representations and LIPs used in this study, a partial Lamarckian strategy tends to perform well across all of the problems considered. Taking a minimax approach, i.e., minimizing the worst possible outcome, the 20% and 40% partial Lamarckian search strategies provide the best results. The worst ranking of each search strategy, in terms of convergence speed, is shown as the last row of Table IX. Both the 20% and 40% strategies yield, at worst, the third best convergence to the optimal solution. The other search strategies, 0% and 5% partial Lamarckian did not consistently find the optimal. Higher amounts of Lamarckianism, 50% to 100%, converge slower to the optimal solution. Since no single search strategy is dominant in terms of solution quality and computational efficiency, the worst case performance of the search strategy can be minimized across the entire set of test problems examined by employing a 20% to 40% partial Lamarckian strategy. The no local improvement search strategy (–1) is shown for comparison to demonstrate the effectiveness of hybrid genetic algorithms.

Table IX: Rank of Convergence Speed (SNK)

Problem instance	Search Strategy											
	–1	0	5	10	20	40	50	60	80	90	95	100
Brown-2	2	1	1	1	1	1	1	1	1	1	1	1
Brown-10	6	5	4	3	2	1	1	1	1	1	1	1
Brown-20	4	4	3	2	1	1	1	1	1	1	1	1
Corana-2	2	1	1	1	1	1	1	1	1	1	1	1
Corana-10	4	4	4	4	3	2	2	2	2	1	1	1
Corana-20	3	3	3	3	3	3	3	2	2	1	1	2
Griewank-2†	1	1	1	1	1	1	1	1	1	1	1	1
Griewank-10	7	1	1	1	2	3	4	4	5	5	5	6
Griewank-20	5	1	1	1	1	2	3	3	3	4	4	4
Rastrigin-2	2	1	1	1	1	1	1	1	1	1	1	1
Rastrigin-10	3	5	4	4	3	2	1	1	1	1	1	1
Rastrigin-20	2	6	5	4	3	1	1	1	1	1	1	1
Schwefel-2†	1	1	1	1	1	1	1	1	1	1	1	1
Schwefel-10	1	3	2	2	1	1	1	1	1	1	1	1
Schwefel-20	3	4	2	2	1	1	1	1	1	1	1	1
LA-200	3	2	1	1	1	1	1	1	1	1	1	1
LA-250	3	2	1	1	1	1	1	1	1	1	1	1
Cell-1	2	2	1	1	1	1	1	1	1	1	1	1
Cell-2	2	2	1	1	1	1	1	1	1	1	1	1
Worst Rank	7	6	5	4	3	3	4	4	5	5	5	6

†- Failed Shapiro-Wilk test or ANOVA did not show significant effect at $\alpha = 0.01$

5. Conclusions

Local improvement procedures were shown to enhance the performance of a genetic algorithm across a wide range of problems. Even though Lamarckian learning disrupts the schema processing of the genetic algorithm and may lead to premature convergence, it reduces the problem of not

having a one-to-one genotype to phenotype mapping. In contrast, while Baldwinian learning does not affect the schema processing capabilities of the genetic algorithm, it results in a large number of genotypes mapping to the same phenotype. Except for the Griewank-10 and Griewank-20 problem instances, this may explain why Baldwinian search was the worst strategy for several of the test problems. In the empirical investigation conducted, neither pure Baldwinian nor pure Lamarckian search strategies were consistently effective. For some problems, it appears that by forcing the genotype to reflect the phenotype, the GA converges more quickly to better solutions, as opposed to leaving the chromosome unchanged after it is evaluated. However, for Griewank-10 and Griewank-20, Lamarckian strategies require more computational time to obtain the optimal. Therefore, a partial Lamarckian strategy is suggested.

The results of this paper have opened up several promising areas for future research. This research has shown that local search improves the GAs' ability to find good solutions. If nothing is known about the particular problem under consideration, a minimax approach is suggested to determine the amount of partial Lamarckianism. Future research could investigate the use of an adaptive approach to determine the amount of partial Lamarckianism to use. Also, it is possible that, instead of a single percentage of Lamarckianism applied during the entire run, different amounts of Lamarckianism over the course of the run would be more beneficial. Even though Lamarckian learning helps to alleviate the problem of one-to-one genotype to phenotype mapping, it can still apply local searches to many points in the same basin of attraction. Another area of future research would involve determining if a local search is even necessary for a new individual (i.e., has the current basin already been mapped). Therefore, computational time is wasted mapping out the same basin of attraction which then can be used to explore new regions. We are currently investigating combining several global optimization techniques with GAs for the purpose of introducing memory into the search strategy.

Acknowledgments

The authors would like to thank the editor and the referees for their insightful comments and suggestions, which resulted in a much improved paper. This research was supported, in part by the National Science Foundation under Grant DMI-9322834.

References

[1] P. Chu and J. Beasley. A genetic algorithm for the generalized assignment problem. Technical Report, The Management School Imperial College London, 1995.

[2] L. Cooper. The transportation-location problems. *Operations Research*, 20:94–108, 1972.

[3] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Transactions on Mathematical Software*, 13(3):262–280, 1987.

[4] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[5] D. Duncan. t-tests and intervals for comparisons suggested by the data. *Biometrics*, 31:339–359, 1975.

[6] I. Einot and K. Gabriel. A study of the powers of several methods of multiple comparisons. *Journal of the American Statistical Association*, 70:351, 1975.

[7] D. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison Wesley, Reading, MA, 1989.

[8] A. Grace. *Optimization Toolbox User's guide*. MathWorks, Natick, MA, 1992.

[9] F. Gruau and D. Whitley. Adding learning to the cellular development of neural networks. *Evolutionary Computation*, 1:213–233, 1993.

[10] G. Hinton and S. Nolan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[11] J. Holland. *Adaptation in Natural and Artificial Systems*. PhD thesis, University of Michigan, 1975.

[12] A. Homaifar, S. Guan, and G. E. Liepins. A new approach on the traveling salesman problem by genetic algorithms. In *Proceedings of the 5th ICGA*, pp. 460–466, Champaign, IL, 1993.

[13] C. R. Houck, J. A. Joines, and M. G. Kay. Comparison of genetic algorithms, random restart, and two-opt switching for solving large location-allocation problems. *Computers and Operations Research*, 23(6):587–596, 1996.

[14] J. Joines, C. Culbreth, and R. King. Manufacturing cell design: An integer programming model employing genetic algorithms. *IIE Transactions*, 28(1):69–85, 1996.

[15] J. Joines and C. Houck. On the use of non-stationary penalty functions to solve constrained optimization problems with genetic algorithms. In *1994 IEEE International Symposium Evolutionary Computation*, pp. 579–584, Orlando, Fl, 1994.

[16] J. Joines, R. King, and C. Culbreth. The use of a local improvement heuristic with a genetic algorithm for manufacturing cell design. Technical Report NCSU-IE 95^{-06}, Dept. of Industrial Engineering North Carolina State University, 1995.

[17] C. Lawrence, J. L. Zhou, and A. L. Tits. User's guide for CFSQP version 2.4: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical Report TR-94016r1, University of Maryland, Electrical Engineering Department and Institute for Systems Research Univer-

sity of Maryland, College Park, MD 20742, 1994.

[18] R. Love and H. Juel. Properties and solution methods for large location-allocation problems with rectangular distances. *Journal of the Operations Research Society*, 33:443–452, 1982.

[19] R. Love and J. Morris. A computational procedure for the exact solution of location-allocation problems with rectangular distances. *Naval Research Logistics Quarterly*, 22:441–453, 1975.

[20] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolutionary Programs*. Springer-Verlang, New York, 3rd ed., 1996.

[21] Z. Michalewicz and G. Nazhiyath. Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Proceedings of the 2nd IEEE International Conference on Evolutionary Computation*, 2:647–651, 1995.

[22] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[23] J. J. More, B. Garbow, and K. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, 1981.

[24] R. Nakano. Conventional genetic algorithm for job shop problems. In *Proc. of the 4th ICGA*, pp. 474–479, San Mateo, CA, 1991.

[25] S. Ng. Worst-case analysis of an algorithm for cellular manufacturing. *European Journal of Operational Research*, 69(3):384–398, 1993.

[26] D. Orvosh and L. Davis. Using a genetic algorithm to optimize problems with feasibility constraints. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, vol. 2, pp. 548–553, 1994.

[27] J.-M. Renders and S. Flasse. Hybrid methods using genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 26(2):243–258, 1996.

[28] T. Ryan. Significance tests for multiple comparison of proportions, variances, and other statistics. *Psychological Bulletin*, 57:318–328, 1960.

[29] SAS Institute Inc, Cary, NC. *SAS/STAT User's Guide*, 4th ed., 1990.

[30] R. Welsch. Stepwise multiple comparison procedures. *Journal of the American Statistical Association*, 72:359, 1977.

[31] D. Whitley, S. Gordon, and K. Mathias. Larmarckian evolution, the Baldwin effect and function optimization. In Y. Davidor, H. Schwefel, and R. Manner, editors, *Parallel Problem Solving from Nature-PPSN III*, pp. 6–15. Springer-Verlag, 1994.

Appendix A: Description of Genetic Algorithm

The genetic algorithm (GA) used in the experiments of Section 3 is summarized in Figure 1. Each of its major components is discussed in detail below.

1. Supply a population P_0 of N individuals and respective function values
2. $i \leftarrow 1$
3. $P_i' \leftarrow$ selection_function(P_{i-1})
4. $P_i \leftarrow$ reproduction_function(P_i')
5. Evaluate(P_i)
6. $i \leftarrow i+1$
7. Repeat steps 3 – 6 until termination
8. Print out best solution found

Figure 1: Genetic Algorithm

The use of a genetic algorithm requires the determination of six fundamental issues: chromosome representation, selection function, the genetic operators making up the reproduction function, the creation of the initial population, termination criteria, and the evaluation function. The rest of this appendix briefly describes each of these issues relating to the experiments of Section 3.

A.1 Solution Representation

For any GA, a chromosome representation is needed to describe each individual in the population of interest. The representation scheme determines how the problem is structured in the GA and also determines the genetic operators that are used. Each individual or chromosome is made up of a sequence of genes from a certain alphabet. An alphabet could consist of binary digits (0 and 1), floating point numbers, integers, symbols (i.e., A, B, C, D), matrices, etc. In Holland's original design [11], the alphabet was limited to binary digits. Since then, problem representation has been the subject of much investigation. It has been shown that more natural representations are more efficient and produce better solutions [20]. In these experiments, a real-valued representation was used for the five nonlinear problems and the location-allocation problem, while an integer representation was used for the cell formation problem.

A.2 Selection Function

The selection of individuals to produce successive generations plays an extremely important role in a genetic algorithm. There are several schemes for the selection process: roulette wheel selec-

tion and its extensions, scaling techniques, tournament, elitist models, and ranking methods [7,20]. Ranking methods only require the evaluation function to map the solutions to a totally ordered set, thus allowing for minimization and negativity. Ranking methods assign P_i based on the rank of solution i after all solutions are sorted. A normalized geometric ranking selection scheme [15] employing an elitist model was used in these experiments. Normalized geometric ranking defines P_i for each individual by:

$$P\,[\text{Individual } i \text{ is chosen}] \;=\; q'\,(1-q)^{\,r-1} \tag{8}$$

where

q = probability of selecting the best individual
r = rank of the individual, where 1 is the best
N = population size

$$q' \;=\; \frac{q}{1-(1-q)^{N}}$$

A.3 Genetic Operators

Genetic operators provide the basic search mechanism of the GA. The application of the two basic types of operators (i.e., mutation and crossover, and their derivatives depends on the chromosome representation used). Operators for real-valued representations, i.e., an alphabet of floats, are described in Michalewicz [20]. For real parents, $\bar{X} = (x_1, x_2, ..., x_n)$ and $\bar{Y} = (y_1, y_2, ..., y_n)$, each of the operators used in the experiments of this paper are described briefly below, Note, for the cell formation problem, the operators were modified to produce integer results [14]. Let a_i and b_i be the lower and upper bound, respectively, for each variable x_i (or y_i), and let $\bar{X}'$ (or $\bar{Y}'$) represent the changed parent or offspring produced by the genetic operator.

Uniform mutation: Randomly select one variable, j, and set it equal to a random number drawn from a uniform distribution $U(a_i, b_i)$ ranging from a_i to b_i:

$$x'_i \;=\; \begin{cases} U(a_i, b_i), & \text{if } i = j \\ x_i, & \text{otherwise} \end{cases} \tag{9}$$

Multi-uniform mutation: Apply eq. (9) to all of the variables in the parent $\bar{X}$.

Boundary mutation: Randomly select one variable, *j*, and set it equal to either its lower or upper bound, where $r = U(0,1)$:

$$x'_i = \begin{cases} a_i, & \text{if } i = j, r < 0.5 \\ b_i, & \text{if } i = j, r \geq 0.5 \\ x_i, & \text{otherwise} \end{cases} \tag{10}$$

Non-uniform mutation: Randomly select one variable, *j*, and set it equal to a non-uniform random number based upon eq. (11).

$$x'_i = \begin{cases} x_i + (b_i - x_i) f(G), & \text{if } i = j, r_1 < 0.5 \\ x_i - (x_i + a_i) f(G), & \text{if } i = j, r_1 \geq 0.5 \\ x_i, & \text{otherwise} \end{cases} \tag{11}$$

where

$$f(G) = \left(r_2 \left(1 - \frac{G}{G_{max}} \right) \right)^b$$

r_1, r_2 = a uniform random number between (0,1)

G = current generation

G_{max} = maximum number of generations

b = a shape parameter

Multi-non-uniform mutation: Apply eq. (11) to all of the variables in the parent $\bar{X}$.

Simple crossover: Generate a random number *r* from a discrete uniform distribution from 2 to $(n - 1)$ and create two new individuals ($\bar{X}'$ and $\bar{Y}'$) using eqs. (12) and (13), where *n* is the number of parameters.

$$x'_i = \begin{cases} x_i, & \text{if } i < r \\ y_i, & \text{otherwise} \end{cases} \tag{12}$$

$$y'_i = \begin{cases} y_i, & \text{if } i < r \\ x_i, & \text{otherwise} \end{cases} \tag{13}$$

Arithmetic crossover: Produce two complimentary linear combinations of the parents, where $r = U(0,1)$:

$$\begin{aligned} \overline{X'} &= r\bar{X} + (1-r)\,\bar{Y} \\ \overline{Y'} &= (1-r)\,\bar{X} + r\bar{Y} \end{aligned} \tag{14}$$

Heuristic crossover: Produce a linear extrapolation of the two individuals. This is the only operator that utilizes fitness information. A new individual, $\overline{X'}$, is created using eq. (15), where $r = U(0,1)$ and $\bar{X}$ is better than $\bar{Y}$ in terms of fitness. If $\overline{X'}$ is infeasible, (i.e., *feasibility* equals 0 as given by eq. (16)), then generate a new random number r and create a new solution using eq. (15); otherwise stop. To ensure halting, after t failures, let the children equal the parents and stop.

$$\begin{aligned} \overline{X'} &= \bar{X} + r\,(\bar{X} - \bar{Y}) \\ \overline{Y'} &= \bar{X} \end{aligned} \tag{15}$$

$$feasibility = \begin{cases} 1, & \text{if } x_i' \geq a_i, x_i' \leq b_i \quad \text{for all } i = 1, \ldots, n \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

A.4 Initialization of Population

As indicated in step 1 of Figure 1, the GA must be provided with an initial population. The most common methods are to (1) randomly generate solutions for the entire population or (2) seed some individuals in the population with "good" starting solutions. The experiments of Section 3 used a random initial population with common random numbers across all search strategies for each replication.

A.5 Termination Criteria

The GA moves from generation to generation selecting and reproducing parents until a termination criterion is met. Several stopping criteria currently exist and can be used in conjunction with

another. For example, a specified maximum number of generations, convergence of the population to virtually a single solution, lack of improvement in the best solution over a specified number of generations, etc. The stopping criterion used in the experiments in Section 3 was to terminate when either the known optimal solution value was found or the specified maximum number of function evaluations was exceeded.

A.6 Evaluation Functions

As stated earlier, evaluation functions of many forms can be used in a GA, subject to the minimal requirement that the function can map the population into a totally ordered set. The GA employing no local improvement used the evaluation functions of Section 3 directly. The hybrid-GA used in the pure Baldwinian, pure Lamarckian, and partial Lamarckian search strategies used the local improvement procedure for each particular problem as its evaluation function. Each of the local improvement procedures then used the evaluation functions of Section 3.

A.7 GA Parameters

Table X lists the values of the parameters of the GAs used in the experiments of Section 3. The parameter values associated with the mutation operators represent the number of individuals that will undergo that particular mutation while the values associated with crossover represent the pairs of individual.

Table X: Parameter Values of the GAs Used in the Experiments

Parameter	Parameter Value
Uniform mutation	4
Multi-Uniform mutation	4
Non-Uniform mutation	4
Multi-Non-Uniform mutation	6
Boundary mutation	4
Simple crossover	2
Arithmetic crossover	2
Heuristic crossover	2
Probability of selecting the best, q	0.08
Population size, N	80
Maximum number of function evaluations	1,000,000
Maximum number iterations for SQP	25

Appendix B: Means and Standard Deviations for Each Search Strategy

The means and standard deviations of the final fitness value and the number of functional evaluations for each search strategy are shown in Tables XI-XVII for all instances. Also, the number of times (maximum 30) the search strategy found the optimal solution is given in the tables.

Table XI: Means and Standard Deviations for the Brown Problem

Problem Instance	Search Strategy	Final Fitness Value		# Times Optimal	# Functional Evaluations	
		Mean	Standard Dev.		Mean	Standard Dev.
Brown-2	-1	$9.33333e^{-07}$	$2.5371e^{-07}$	30	$9.5529e^{+04}$	$6.7213e^{+04}$
	0	$0.00000e^{+00}$	0.0000	30	$3.1070e^{+03}$	$1.5220e^{+02}$
	5	0.00000	0.0000	30	$3.1026e^{+03}$	$1.6197e^{+02}$
	10	0.00000	0.0000	30	$3.0946e^{+03}$	$1.6261e^{+02}$
	20	0.00000	0.0000	30	$3.0856e^{+03}$	$1.7367e^{+02}$
	40	0.00000	0.0000	30	$3.0389e^{+03}$	$1.6929e^{+02}$
	50	0.00000	0.0000	30	$3.0193e^{+03}$	$1.8582e^{+02}$
	60	0.00000	0.0000	30	$2.9950e^{+03}$	$1.8076e^{+02}$
	80	0.00000	0.0000	30	$2.9282e^{+03}$	$1.6989e^{+02}$
	90	0.00000	0.0000	30	$2.9092e^{+03}$	$1.6929e^{+02}$
	95	0.00000	0.0000	30	$2.8798e^{+03}$	$1.6445e^{+02}$
	100	0.00000	0.0000	30	$2.8681e^{+03}$	$1.5299e^{+02}$
Brown-10	-1	$1.26602e^{+00}$	1.9554	0	$1.0000e^{+06}$	0.0000
	0	$5.03208e^{-01}$	$4.7555e^{-01}$	7	$8.5243e^{+05}$	$3.0670e^{+05}$
	5	$1.00000e^{-07}$	$3.0513e^{-07}$	30	$2.8470e^{+04}$	$1.5704e^{+04}$
	10	$3.33333e^{-08}$	$1.8257e^{-07}$	30	$1.8565e^{+04}$	$1.0994e^{+04}$
	20	0.00000	0.0000	30	$1.3169e^{+04}$	$4.2345e^{+03}$
	40	0.00000	0.0000	30	$9.2201e^{+03}$	$1.7223e^{+03}$
	50	0.00000	0.0000	30	$8.9320e^{+03}$	$1.5899e^{+03}$
	60	0.00000	0.0000	30	$8.6745e^{+03}$	$1.6312e^{+03}$
	80	0.00000	0.0000	30	$8.1481e^{+03}$	$9.7597e^{+02}$
	90	0.00000	0.0000	30	$7.8687e^{+03}$	$8.2605e^{+02}$
	95	0.00000	0.0000	30	$7.7787e^{+03}$	$7.7405e^{+02}$
	100	0.00000	0.0000	30	$7.6381e^{+03}$	$6.6600e^{+02}$
Brown-20	-1	$1.65499e^{+01}$	$4.3312e^{+00}$	0	$1.0000e^{+06}$	0.0000
	0	$1.10978e^{+01}$	$2.5511e^{+01}$	0	$1.0013e^{+06}$	$8.6148e^{+02}$
	5	$1.00000e^{-07}$	$3.0513e^{-07}$	30	$2.7049e^{+04}$	$2.4537e^{+04}$
	10	$3.33333e^{-08}$	$1.8257e^{-07}$	30	$1.4136e^{+04}$	$7.3162e^{+03}$
	20	$3.33333e^{-08}$	$1.8257e^{-07}$	30	$1.0893e^{+04}$	$5.1028e^{+03}$
	40	$3.33333e^{-08}$	$1.8257e^{-07}$	30	$6.7153e^{+03}$	$2.7935e^{+03}$
	50	$6.66667e^{-08}$	$2.5371e^{-07}$	30	$5.7384e^{+03}$	$1.3383e^{+03}$
	60	$3.33333e^{-08}$	$1.8257e^{-07}$	30	$6.1421e^{+03}$	$1.4538e^{+03}$
	80	0.00000	0.0000	30	$5.7521e^{+03}$	$1.3410e^{+03}$
	90	$3.33333e^{-08}$	$1.8257e^{-07}$	30	$5.3642e^{+03}$	$1.1904e^{+03}$
	95	$6.66667e^{-08}$	$2.5371e^{-07}$	30	$5.4270e^{+03}$	$1.1955e^{+03}$
	100	$6.66667e^{-08}$	$2.5371e^{-07}$	30	$5.5075e^{+03}$	$1.3284e^{+03}$

Table XII: Means and Standard Deviations for the Corana Problem

Problem Instance	Search Strategy	Final Fitness Value		# Times Optimal	# Functional Evaluations	
		Mean	Standard Dev.		Mean	Standard Dev.
Corana-2	-1	4.66667e^{-07}	5.0742e^{-07}	30	5.1410e^{+04}	1.8823e^{+04}
	0	0.00000	0.0000	30	2.5613e^{+03}	1.9251e^{+03}
	5	0.00000	0.0000	30	2.4311e^{+03}	1.7278e^{+03}
	10	0.00000	0.0000	30	2.3331e^{+03}	1.6358e^{+03}
	20	0.00000	0.0000	30	3.7342e^{+03}	7.8971e^{+03}
	40	0.00000	0.0000	30	2.8112e^{+03}	2.8586e^{+03}
	50	0.00000	0.0000	30	2.4239e^{+03}	1.7661e^{+03}
	60	0.00000	0.0000	30	1.9940e^{+03}	1.5739e^{+03}
	80	0.00000	0.0000	30	2.1187e^{+03}	1.5760e^{+03}
	90	0.00000	0.0000	30	2.2695e^{+03}	1.8779e^{+03}
	95	0.00000	0.0000	30	2.2323e^{+03}	1.8967e^{+03}
	100	0.00000	0.0000	30	2.3534e^{+03}	1.8996e^{+03}
Corana-10	-1	3.78311e^{+01}	6.9463e^{+01}	0	1.0000e^{+06}	0.0000
	0	5.46673e^{+06}	6.3360e^{+06}	0	1.0038e^{+06}	3.0048e^{+03}
	5	2.39987e^{+04}	3.9151e^{+04}	1	1.0023e^{+06}	1.5628e^{+04}
	10	4.56015e^{+02}	1.0328e^{+03}	6	9.6985e^{+05}	8.2647e^{+04}
	20	1.18519e^{+01}	5.8604e^{+01}	20	7.6512e^{+05}	2.3101e^{+05}
	40	1.01250e^{-01}	5.5457e^{-01}	29	4.4581e^{+05}	1.8076e^{+05}
	50	0.00000	0.0000	30	4.0596e^{+05}	1.4588e^{+05}
	60	0.00000	0.0000	30	2.8710e^{+05}	1.1819e^{+05}
	80	0.00000	0.0000	30	2.5739e^{+05}	1.1249e^{+05}
	90	0.00000	0.0000	30	2.6730e^{+05}	8.0431e^{+04}
	95	0.00000	0.0000	30	2.1381e^{+05}	7.4134e^{+04}
	100	0.00000	0.0000	30	2.5520e^{+05}	6.6527e^{+04}
Corana-20	-1	9.38237e^{+02}	1.1534e^{+03}	0	1.0000e^{+06}	0.0000
	0	2.34023e^{+08}	1.2459e^{+08}	0	1.0062e^{+06}	4.1383e^{+03}
	5	3.02898e^{+07}	4.4236e^{+07}	0	1.0040e^{+06}	2.9309e^{+03}
	10	5.47664e^{+06}	8.5726e^{+06}	0	1.0038e^{+06}	2.5784e^{+03}
	20	9.09493e^{+05}	1.2609e^{+06}	0	1.0054e^{+06}	3.2298e^{+03}
	40	5.71348e^{+04}	1.4106e^{+05}	0	1.0061e^{+06}	3.9037e^{+03}
	50	1.95620e^{+03}	4.1795e^{+03}	0	1.0044e^{+06}	3.0620e^{+03}
	60	2.51093e^{+03}	1.1494e^{+04}	4	9.9150e^{+05}	4.5846e^{+04}
	80	5.44879e^{+01}	1.7228e^{+02}	7	9.7488e^{+05}	6.1539e^{+04}
	90	1.95900e^{+01}	6.5789e^{+01}	11	9.5661e^{+05}	1.0050e^{+05}
	95	6.17831	1.4902e^{+01}	15	9.2491e^{+05}	1.1067e^{+05}
	100	2.83835	5.7170	12	9.5727e^{+05}	8.9149e^{+04}

Table XIII: Means and Standard Deviations for Griewank Problem

Problem Instance	Search Strategy	Final Fitness Value		# Times Optimal	# Functional Evaluations	
		Mean	Standard Dev.		Mean	Standard Dev.
Griewank-2	-1	3.69823e^{-03}	3.7610e^{-03}	15	5.0653e^{+05}	5.0193e^{+05}
	0	7.39600e^{-04}	2.2567e^{-03}	27	3.2468e^{+05}	3.1199e^{+05}
	5	7.39600e^{-04}	2.2567e^{-03}	27	3.0965e^{+05}	3.0753e^{+05}
	10	9.86133e^{-04}	2.5571e^{-03}	26	2.9192e^{+05}	3.1061e^{+05}
	20	4.93067e^{-04}	1.8764e^{-03}	28	2.5194e^{+05}	2.4883e^{+05}
	40	4.93067e^{-04}	1.8764e^{-03}	28	2.9217e^{+05}	2.6033e^{+05}
	50	7.39600e^{-04}	2.2567e^{-03}	27	3.5088e^{+05}	3.1584e^{+05}
	60	4.93067e^{-04}	1.8764e^{-03}	28	2.6093e^{+05}	2.5242e^{+05}
	80	9.86133e^{-04}	2.5571e^{-03}	26	3.3917e^{+05}	3.3349e^{+05}
	90	1.23267e^{-03}	2.8034e^{-03}	25	2.9896e^{+05}	3.2910e^{+05}
	95	7.39600e^{-04}	2.2567e^{-03}	27	2.2906e^{+05}	2.7131e^{+05}
	100	9.86133e^{-04}	2.5571e^{-03}	26	2.8729e^{+05}	3.1562e^{+05}
Griewank-10	-1	5.25369e^{-02}	2.9763e^{-02}	1	9.7013e^{+05}	1.6368e^{+05}
	0	0.00000	0.0000	30	4.7135e^{+03}	1.7993e^{+02}
	5	0.00000	0.0000	30	4.8088e^{+03}	2.8770e^{+02}
	10	0.00000	0.0000	30	5.0444e^{+03}	3.7042e^{+02}
	20	0.00000	0.0000	30	5.4296e^{+03}	5.9641e^{+02}
	40	0.00000	0.0000	30	6.2652e^{+03}	6.4218e^{+02}
	50	0.00000	0.0000	30	6.8425e^{+03}	8.7743e^{+02}
	60	0.00000	0.0000	30	7.2035e^{+03}	9.1656e^{+02}
	80	0.00000	0.0000	30	8.2834e^{+03}	1.1450e^{+03}
	90	0.00000	0.0000	30	8.8033e^{+03}	1.2008e^{+03}
	95	0.00000	0.0000	30	9.0910e^{+03}	1.1866e^{+03}
	100	0.00000	0.0000	30	9.3120e^{+03}	1.1467e^{+03}
Griewank-20	-1	4.20434e^{-02}	4.4794e^{-02}	4	9.0575e^{+05}	2.6030e^{+05}
	0	0.00000	0.0000	30	7.8540e^{+03}	1.9988e^{+02}
	5	0.00000	0.0000	30	8.0354e^{+03}	2.8841e^{+02}
	10	0.00000	0.0000	30	8.2087e^{+03}	3.8040e^{+02}
	20	0.00000	0.0000	30	8.4880e^{+03}	5.3300e^{+02}
	40	0.00000	0.0000	30	9.1453e^{+03}	6.8650e^{+02}
	50	0.00000	0.0000	30	9.4776e^{+03}	6.6865e^{+02}
	60	0.00000	0.0000	30	9.7216e^{+03}	7.1319e^{+02}
	80	0.00000	0.0000	30	1.0219e^{+04}	7.1239e^{+02}
	90	0.00000	0.0000	30	1.0371e^{+04}	6.5058e^{+02}
	95	0.00000	0.0000	30	1.0554e^{+04}	6.5540e^{+02}
	100	0.00000	0.0000	30	1.0655e^{+04}	5.3251e^{+02}

Table XIV: Means and Standard Deviations for Rastrigin Problem

Problem Instance	Search Strategy	Final Fitness Value		# Times Optimal	# Functional Evaluations	
		Mean	Standard Dev.		Mean	Standard Dev.
Rastrigin-2	-1	5.00000e-07	5.0855e-07	30	6.3136e+03	3.2121e+03
	0	0.00000	0.0000	30	2.1390e+03	2.3572e+01
	5	0.00000	0.0000	30	2.1376e+03	2.5566e+01
	10	0.00000	0.0000	30	2.1391e+03	2.6483e+01
	20	0.00000	0.0000	30	2.1376e+03	3.1725e+01
	40	0.00000	0.0000	30	2.1340e+03	4.3927e+01
	50	0.00000	0.0000	30	2.1377e+03	4.7064e+01
	60	0.00000	0.0000	30	2.1336e+03	4.5957e+01
	80	0.00000	0.0000	30	2.1344e+03	5.3647e+01
	90	0.00000	0.0000	30	2.1401e+03	6.0398e+01
	95	0.00000	0.0000	30	2.1386e+03	6.1656e+01
	100	0.00000	0.0000	30	2.1426e+03	6.4295e+01
Rastrigin-10	-1	1.00000e-06	0.0000	30	7.3554e+04	9.6194e+03
	0	1.12762	1.4006	13	7.1635e+05	3.7994e+05
	5	0.00000	0.0000	30	1.5054e+05	1.0696e+05
	10	0.00000	0.0000	30	1.1710e+05	6.9051e+04
	20	0.00000	0.0000	30	6.8002e+04	2.2303e+04
	40	0.00000	0.0000	30	5.2204e+04	1.1329e+04
	50	0.00000	0.0000	30	4.3796e+04	1.2086e+04
	60	0.00000	0.0000	30	4.0310e+04	1.0581e+04
	80	0.00000	0.0000	30	3.8088e+04	9.2368e+03
	90	0.00000	0.0000	30	3.7627e+04	8.8121e+03
	95	0.00000	0.0000	30	3.5740e+04	8.8191e+03
	100	0.00000	0.0000	30	3.7634e+04	1.0570e+04
Rastrigin-20	-1	1.00000e-06	0.0000	30	2.1561e+05	6.0225e+04
	0	1.21515e+01	7.3257	1	9.9512e+05	3.7836e+04
	5	0.00000	0.0000	30	4.5566e+05	1.5220e+05
	10	0.00000	0.0000	30	3.7480e+05	1.4393e+05
	20	0.00000	0.0000	30	2.9478e+05	1.1289e+05
	40	0.00000	0.0000	30	2.0678e+05	6.5623e+04
	50	0.00000	0.0000	30	1.8398e+05	3.6898e+04
	60	0.00000	0.0000	30	1.9134e+05	5.1640e+04
	80	0.00000	0.0000	30	1.6438e+05	4.0192e+04
	90	0.00000	0.0000	30	1.4991e+05	3.8368e+04
	95	0.00000	0.0000	30	1.5426e+05	3.8706e+04
	100	0.00000	0.0000	30	1.6482e+05	3.5259e+04

Table XV: Means and Standard Deviations for Schwefel Problem

Problem Instance	Search Strategy	Final Fitness Value		# Times Optimal	# Functional Evaluations	
		Mean	Standard Dev.		Mean	Standard Dev.
Schwefel-2	-1	$4.33333e^{-07}$	$5.0401e^{-07}$	30	$6.7829e^{+03}$	$3.4377e^{+03}$
	0	0.00000	0.0000	30	$8.3627e^{+03}$	$8.9766e^{+03}$
	5	0.00000	0.0000	30	$8.3500e^{+03}$	$8.9631e^{+03}$
	10	0.00000	0.0000	30	$8.3915e^{+03}$	$9.0084e^{+03}$
	20	0.00000	0.0000	30	$8.3996e^{+03}$	$9.0260e^{+03}$
	40	0.00000	0.0000	30	$8.4540e^{+03}$	$9.1100e^{+03}$
	50	0.00000	0.0000	30	$8.4480e^{+03}$	$9.0932e^{+03}$
	60	0.00000	0.0000	30	$8.4446e^{+03}$	$9.0945e^{+03}$
	80	0.00000	0.0000	30	$8.5156e^{+03}$	$9.2095e^{+03}$
	90	0.00000	0.0000	30	$8.5299e^{+03}$	$9.2290e^{+03}$
	95	0.00000	0.0000	30	$8.5241e^{+03}$	$9.2185e^{+03}$
	100	0.00000	0.0000	30	$8.5262e^{+03}$	$9.2175e^{+03}$
Schewefel-10	-1	$1.00000e^{-06}$	0.0000	30	$7.5922e^{+04}$	$1.3120e^{+04}$
	0	$6.63306e^{-01}$	1.0560	19	$4.9627e^{+05}$	$4.2578e^{+05}$
	5	0.00000	0.0000	30	$1.1988e^{+05}$	$7.8612e^{+04}$
	10	0.00000	0.0000	30	$1.1426e^{+05}$	$6.1898e^{+04}$
	20	0.00000	0.0000	30	$7.9676e^{+04}$	$3.7094e^{+04}$
	40	0.00000	0.0000	30	$7.7312e^{+04}$	$4.6399e^{+04}$
	50	0.00000	0.0000	30	$7.3009e^{+04}$	$4.8608e^{+04}$
	60	0.00000	0.0000	30	$7.1344e^{+04}$	$4.7217e^{+04}$
	80	0.00000	0.0000	30	$7.3951e^{+04}$	$5.4221e^{+04}$
	90	0.00000	0.0000	30	$7.1995e^{+04}$	$5.3185e^{+04}$
	95	0.00000	0.0000	30	$6.8266e^{+04}$	$4.8523e^{+04}$
	100	0.00000	0.0000	30	$6.9889e^{+04}$	$4.7395e^{+04}$
Schewefel-20	-1	$4.40933e^{-04}$	$1.1698e^{-03}$	23	$4.6027e^{+05}$	$3.6124e^{+05}$
	0	8.14151	9.0376	11	$7.0960e^{+05}$	$4.0809e^{+05}$
	5	0.00000	0.0000	30	$3.6296e^{+05}$	$1.8691e^{+05}$
	10	0.00000	0.0000	30	$3.3455e^{+05}$	$1.5009e^{+05}$
	20	0.00000	0.0000	30	$2.8962e^{+05}$	$9.5471e^{+04}$
	40	0.00000	0.0000	30	$2.4478e^{+05}$	$8.7211e^{+04}$
	50	0.00000	0.0000	30	$2.3312e^{+05}$	$8.1234e^{+04}$
	60	0.00000	0.0000	30	$2.4751e^{+05}$	$9.2517e^{+04}$
	80	0.00000	0.0000	30	$2.3880e^{+05}$	$1.2116e^{+05}$
	90	0.00000	0.0000	30	$2.2414e^{+05}$	$1.1503e^{+05}$
	95	0.00000	0.0000	30	$2.2112e^{+05}$	$1.1259e^{+05}$
	100	0.00000	0.0000	30	$2.4558e^{+05}$	$1.3543e^{+05}$

Table XVI: Means and Standard Deviations for Location-Allocation Problem

Problem Instance	Search Strategy	Final Fitness Value		# Times Optimal	# Functional Evaluations	
		Mean	Standard Dev.		Mean	Standard Dev.
LA-200	-1	7.75238e+08	9.0360e+06	0	1.0000e+06	7.5943e+00
	0	7.6064E+08	6.3320e+05	0	4.0881e+05	4.3267e+05
	5	7.6042e+08	0.00000	30	3.8905e+04	2.0744e+04
	10	7.6042e+08	0.00000	30	3.3372e+04	2.7074e+04
	20	7.6042e+08	0.00000	30	2.5369e+04	1.6848e+04
	40	7.6042e+08	0.00000	30	1.6566e+04	5.6508e+03
	50	7.6042e+08	0.00000	30	1.6787e+04	7.5832e+03
	60	7.6042e+08	0.00000	30	1.7623e+04	8.8216e+03
	80	7.6042e+08	0.00000	30	1.6273e+04	5.6618e+03
	90	7.6042e+08	0.00000	30	1.7641e+04	5.9482e+03
	95	7.6042e+08	0.00000	30	1.6480e+04	7.7134e+03
	100	7.6042e+08	0.00000	30	1.8010e+04	8.4490e+03
LA-250	-1	914266e+08	1.1527e+07	0	1.0000e+06	7.6177e+00
	0	8.93803e+08	1.7180e+06	13	5.3239e+05	3.9781e+05
	5	8.93043e+08	0.00000	30	6.9669e+04	6.0673e+04
	10	8.93043e+08	0.00000	30	9.2689e+04	9.7886e+04
	20	8.93043e+08	0.00000	30	5.7390e+04	4.6008e+04
	40	8.93043e+08	0.00000	30	5.2337e+04	5.1337e+04
	50	8.93043e+08	0.00000	30	4.2662e+04	4.1346e+04
	60	8.93043e+08	0.00000	30	3.7312e+04	3.1649e+04
	80	8.93043e+08	0.00000	30	2.8483e+04	2.4497e+04
	90	8.93043e+08	0.00000	30	5.1976e+04	8.2919e+04
	95	8.93043e+08	0.00000	30	4.1953e+04	5.3571e+04
	100	8.93043e+08	0.00000	30	6.0745e+04	9.6145e+04

Table XVII: Means and Standard Deviations for Cell Formation Problem

Problem Instance	Search Strategy	Final Fitness Value		# Times Optimal	# Functional Evaluations	
		Mean	Standard Dev.		Mean	Standard Dev.
Cell-1	-1	$4.85644e^{-02}$	$2.9701e^{-02}$	0	$9.5896e^{+05}$	$1.8359e^{+05}$
	0	$2.01705e^{-02}$	$1.3371e^{-02}$	0	$1.0040e^{+06}$	$0.0000e^{+00}$
	5	$4.62521e^{-03}$	$7.5925e^{-03}$	18	$4.8973e^{+05}$	$4.4185e^{+05}$
	10	$3.63696e^{-03}$	$6.8736e^{-03}$	21	$4.0218e^{+05}$	$4.2640e^{+05}$
	20	$2.08318e^{-03}$	$2.8383e^{-03}$	17	$5.0916e^{+05}$	$4.6133e^{+05}$
	40	$2.95388e^{-03}$	$4.1654e^{-03}$	17	$4.7390e^{+05}$	$4.9210e^{+05}$
	50	$1.74684e^{-03}$	$2.7329e^{-03}$	18	$4.6613e^{+05}$	$4.7298e^{+05}$
	60	$1.26817e^{-03}$	$2.5732e^{-03}$	21	$3.5049e^{+05}$	$4.5701e^{+05}$
	80	$2.56858e^{-03}$	$4.1571e^{-03}$	18	$4.7031e^{+05}$	$4.7347e^{+05}$
	90	$1.89975e^{-03}$	$3.3351e^{-03}$	19	$3.7619e^{+05}$	$4.7290e^{+05}$
	95	$1.41578e^{-03}$	$3.3167e^{-03}$	24	$2.6205e^{+05}$	$4.0666e^{+05}$
	100	$6.16291e^{-04}$	$1.3342e^{-03}$	24	$2.2410e^{+05}$	$4.0027e^{+05}$
Cell-2	-1	$4.31420e^{-02}$	$3.0110e^{-02}$	0	$1.0000e^{+06}$	$0.0000e^{+00}$
	0	$3.34305e^{-02}$	$1.1680e^{-02}$	0	$1.0040e^{+06}$	$0.0000e^{+00}$
	5	$1.41851e^{-02}$	$1.2382e^{-02}$	6	$8.3525e^{+05}$	$3.4707e^{+05}$
	10	$1.19120e^{-02}$	$1.0142e^{-02}$	9	$7.4112e^{+05}$	$4.1438e^{+05}$
	20	$1.30338e^{-02}$	$1.1518e^{-02}$	10	$6.8779e^{+05}$	$4.4777e^{+05}$
	40	$7.98756e^{-03}$	$8.5386e^{-03}$	12	$6.3550e^{+05}$	$4.6453e^{+05}$
	50	$7.27027e^{-03}$	$7.7368e^{-03}$	13	$5.8844e^{+05}$	$4.7236e^{+05}$
	60	$6.61065e^{-03}$	$6.4923e^{-03}$	12	$6.5049e^{+05}$	$4.5084e^{+05}$
	80	$5.81299e^{-03}$	$9.5796e^{-03}$	19	$4.1762e^{+05}$	$4.6146e^{+05}$
	90	$6.57270e^{-03}$	$1.0109e^{-02}$	18	$4.2704e^{+05}$	$4.8382e^{+05}$
	95	$3.73199e^{-03}$	$4.3847e^{-03}$	17	$5.0059e^{+05}$	$4.7232e^{+05}$
	100	$3.78634e^{-03}$	$5.9721e^{-03}$	19	$4.0647e^{+05}$	$4.6566e^{+05}$