Vol. 45 No. SIG 2(TOM 10) IPSJ Transactions on Mathematical Modeling and Its Applications Feb. 2004

*Regular Paper*

# A Study on Parallel Varying Mutation in Deterministic and Self-Adaptive GAs with 0/1 Multiple Knapsack Problems

Hernán E. Aguirre† and Kiyoshi Tanaka†

In this work we study varying mutations applied parallel to crossover in generational deterministic and self-adaptive varying mutation GAs and compare them with the conventional generational model of varying mutations that apply mutation mostly serial to crossover. Experiments are conducted with several subclasses of 0/1 multiple knapsack problems. We found that varying mutation parallel to crossover can be a more effective and efficient framework than the conventional model of varying mutations in both deterministic and self-adaptive GAs to achieve faster convergence velocity and higher convergence reliability. We also found that the conventional model of varying mutations affects negatively the self-adaptive mutation rate control. This strongly suggests that the conventional model of varying mutation GAs may not be appropriate for combining forms of control. Also, we compare deterministic, adaptive, and self-adaptive mutation schedules within the parallel varying mutation model. Best overall performance was achieved by a parallel varying mutation self-adaptive GA.

## 1. Introduction

Parameter control methods modify the values of the strategy parameters during the run of the algorithm by taking into account the actual search process. These methods are an alternative form to the common practice of tuning parameters "by hand" and are considered as one of the most important and promising areas of research in evolutionary algorithms [1]. One of the approaches for parameter control in genetic algorithms (GAs) seeks to combine crossover with (higher) varying mutation rates during the course of a run. Deterministically varying mutation rates over the generations and/or across the representation [2)~4)] and self-adaptive mutation rate schedules have been proposed to control the mutation rate of generational and steady state GAs [4),7),8)]. The principle of self-adaptation incorporates strategy parameters into the representation of individuals evolving simultaneously strategy parameters and object variables. Self-adaptation is regarded as the method having the advantage of reducing the number of exogenous parameters [8)] and is thought to be the most promising way of combining forms of control (parameters co-adaptation) [1)].

From the application of operators standpoint, the varying mutation principles, usually inspired from evolution strategies (ES) [5)] and evolutionary programming (EP) [6)], when applied to generational GAs have been incorporated following the canonical model of GAs [9),10)]. These conventional varying mutation GAs have the advantage of allowing a simple incorporation of the varying mutation principles while usually performing better than canonical GAs. However, since in canonical GAs crossover is applied with probability $p_c$ and then follows mutation, it turns out that in conventional varying mutation GAs higher mutations are mostly applied after crossover. This conventional model of varying mutation GAs raises several important questions regarding the interference between crossover and high mutation, how this affects performance of the algorithm, whether this affect the mutation rate control itself in the case of self-adaptive varying mutation algorithms, and more generally whether this is an appropriate model for combining forms of control (co-adaptation of strategy parameters).

An alternative to conventional varying mutation methods is to design approaches that apply "background" mutation [9),10)] after crossover (or none at all) and higher mutations only parallel to crossover. These approaches could give an efficient framework to achieve better balances for varying mutation and crossover, in which the strengths of the operators can be kept without interfering one with the other, and have the potential of being more suitable models for combining forms of control.

From this point of view, we continue to explore a model of generational GA that applies varying mutations parallel to crossover followed

† Faculty of Engineering, Shinshu University

by "background" mutation, putting the operators in a cooperative-competitive stand with each other by subjecting their offspring to extinctive selection. The model of varying mutation parallel to crossover was proposed in previous reports and its internal structure was studied in depth using an adaptive schedule for varying mutation [11)~13)]. Important structural issues that were studied include the balance for offspring creation between the operator that applies varying mutation and the operator that uses crossover followed by "background" mutation, the ratio between number of parents and number of offspring (extinctive selection pressure), "background" mutation probability after crossover, and the threshold to trigger adaptation in the varying mutation operator. Two mutation strategies to select the bits that will undergo mutation were also investigated for the varying mutation operator. The effect of population size and number of evaluations was observed, too. In Ref. 14) the effectiveness of parallel adaptive varying mutation was explored within distributed GAs.

In this work, the model of varying mutations parallel to crossover is applied to other subclasses of varying mutation GAs (deterministic and self-adaptive varying mutation GAs) and compare it with the conventional generational model of varying mutations GAs across a broad range of difficult, large, and highly constrained 0/1 multiple knapsack problems [15)]. After comparative experiments, we found that varying mutation parallel to crossover can be a more effective and efficient framework than the conventional model in both deterministic and self-adaptive varying mutation GAs to achieve faster convergence velocity and higher convergence reliability. We also found that the conventional model of varying mutations affects negatively the self-adaptive mutation rate control. This strongly suggests that the conventional model of varying mutation GAs may not be appropriate for combining forms of control.

Also, deterministic, adaptive, and self-adaptive mutation schedules are compared within the parallel varying mutation model. Best overall performance was achieved by a parallel varying mutation self-adaptive GA. Some parts of this work have been presented in Refs. 16), 17).

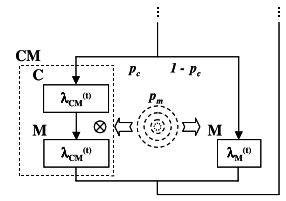



**Fig. 1** A conventional varying mutation GA.

## 2. A Conventional Varying Mutation GA

A conventional varying mutation GA, similar to canonical GAs, applies crossover with probability $p_c$ followed by mutation with probability $p_m$ per bit. In the absence of crossover $(1-p_c)$, mutation is applied alone. From the application of operators standpoint, it can be said that the probability of crossover $p_c$ enables an implicit parallel application of two operators. One of the operators is crossover followed by mutation (CM) and the other one is mutation alone (M). It should be noted that mutation in both CM and M is governed by the same mutation probability $p_m$ and applies the same "bit by bit" mutation strategy. **Figure 1** illustrates the application of operators in a conventional varying mutation GA.

Since $p_c$ is usually set to 0.6, and higher values are often used [1)], it turns out that mutation is mostly applied serial to crossover. In canonical GAs $p_m$ is small, therefore the amount of diversity introduced by mutation either through CM or M is modest. For the same reason, the disruption that mutation causes to crossover in CM is also expected to be small. In varying mutation GAs, however, mutations are higher and the combined effect of crossover and mutation in CM and the effect of mutation alone in M should be carefully reconsidered.

In the case of CM, besides those cases in which crossover and mutation aggregate in a positive manner or are neutral, those cases in which one of the operators is working well but is being hampered by the other should also be taken into account. For example, if mutation rates were high, although crossover could be doing a good job it is likely that some of the just created favorable recombinations would be immediately lost, before they become fix in the offspring, due to the high disruption introduced by mutation. We can think of this case as a mutation interference with crossover in the *creation* of beneficial recombinations. On the

other hand, mutation could be working well but crossover may produce poor performing individuals affecting the survivability of beneficial mutations that can contribute to the search. We can think of this case as a crossover interference with mutation in the introduction of beneficial mutations.

In the case of mutation alone M, its instantaneous effectiveness depends only upon itself and does not diminish the effectiveness of other operator. High mutations in M, when are harmful, will have a negative impact on the *propagation* of beneficial recombinations already present in the parent population, but will not affect their *creation* by crossover as high mutation can do it in CM.

In the following we also refer to conventional varying mutation GAs as *varying mutation serial to crossover*.

## 3. A GA with Parallel Varying Mutation

### 3.1 Parallel Genetic Operators

An alternative to standard varying mutation GAs is to explicitly differentiate the mutation operator applied parallel to crossover from the mutation operator applied after crossover. We explore a model of GA that in addition to crossover followed by background mutation (CM) it also explicitly applies parallel varying mutation [11)~13)]. To clearly distinguish between mutation operators the parallel varying mutation operator is called *Self-Reproduction with Mutation* (SRM). SRM parallel to CM implicitly increases the levels of cooperation to introduce beneficial mutations and create beneficial recombinations. It also sets the stage for competition between operators' offspring. In the following we also refer to this model of GA that applies varying mutation parallel to crossover as GA-SRM.

### 3.2 Extinctive Selection

The model also incorporates the concept of extinctive selection that has been widely used in Evolution Strategies. Through extinctive selection the offspring created by CM and SRM coexist and compete for survival (the number of parents is smaller than the total offspring) and reproduction. Among the various extinctive selection mechanisms available in the EA literature [18)] we chose $(\mu, \lambda)$ Proportional Selection.

The parallel formulation of genetic operators tied to extinctive selection creates a cooperative-competitive environment for the offspring created by CM and SRM. The block diagram of the model is depicted in **Fig. 2**. The number of parents is $\mu$, $\lambda_{CM}$ and $\lambda_{SRM}$ are the number of offspring created by CM and SRM, respectively, and $\lambda = \lambda_{CM} + \lambda_{SRM}$ is the total number of offspring. Both $\lambda_{CM}$ and $\lambda_{SRM}$ are deterministically decided at the beginning of the run.

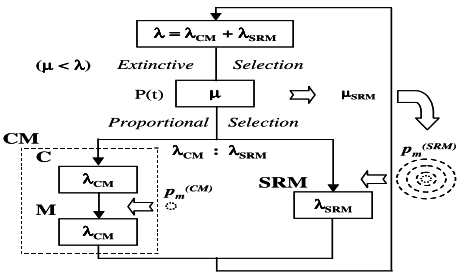



**Fig. 2** A GA with parallel varying mutation.

### 3.3 Mutation Rate Control in SRM

In this work we use deterministic and self-adaptive mutation rate controls in SRM to compare the conventional and parallel model of varying mutations. The deterministic approach implements a time-dependent mutation schedule that reduces mutation rate in a hyperbolic shape, originally proposed in Ref. 4) and expressed by

$$p_m^{(t)} = \left( r_o + \frac{n - r_o}{T - 1} t \right)^{-1} \qquad (1)$$

where $T$ is the maximum number of generations, $t \in \{0, 1, \cdots, T-1\}$ is the current generation, and $n$ is the bit string length. The mutation rate $p_m^{(t)}$ varies in the range $[1/r_o, 1/n]$. In the original formulation $r_o = 2$. Here we included $r_o$ as a parameter in order to study different ranges for mutation. In the deterministic approach the mutation rate calculated at time $t$ is applied to all individuals created by SRM.

To include self-adaptation, each individual incorporates its own mutation probability within the representation. SRM to produce offspring first mutates the mutation probability of the selected individual and then mutates the object variable using the individual's mutated probability. In this work we use the self-adaptive approach originally proposed in Refs. 4), 8), which uses a continuous representation for the mutation rate and mutates the mutation probability of each individual by

$$p_m^{(t)}(i) = \left(1 + \frac{1-p_m^{(t-1)}(i)}{p_m^{(t-1)}(i)} exp(-\gamma N(0,1))\right)^{-1} \quad (2)$$

where $i$ indicates the $i$-th individual, $\gamma$ is a learning rate that control the speed of self-adaptation, and $N(0,1)$ is a normally distributed random number with expectation zero and standard deviation one. Note that individuals selected to reproduce with SRM at generation $t$ could have been created either by SRM or CM at generation $t-1$. Since the mutation rate of each individual is mutated only by SRM, individuals created by CM do not carry an updated mutation rate. Thus, the mutation rate of individuals that were created by CM at generation $t-1$ is first updated by

$$p_m^{(t-1)}(j) = \frac{1}{\mu_{SRM}} \sum_{k=1}^{\mu_{SRM}} p_m^{(t-1)}(k) \quad (3)$$

where $j$ indicates an individual created by CM at $(t-1)$, $k$ indicates the individuals created by SRM at $(t-1)$ that survived extinctive selection, and $\mu_{SRM}$ is the number of offspring created by SRM that survived extinctive selection. In the case that no offspring created by SRM survived extinctive selection, $p_m^{(t-1)}(j)$ is set to the mutation value of the best SRM's offspring. SRM will mutate this updated mutation in order to mutate the object variable.

Also, to compare mutation schedules within the parallel varying mutation model, besides the deterministic and self-adaptive schedules mentioned above, we also use the adaptive dynamic probability (ADP) mutation schedule used in Ref. 13). ADP varies mutation rate each time a normalized mutant's survival ratio $\zeta$ falls under a threshold $\tau$ $(\zeta < \tau)$ by

$$p_m = \begin{cases} p_m \times \beta & (\geq 1/n) \\ 1/n & otherwise \end{cases} \quad (4)$$

where $0 < \beta < 1$ and $n$ is the bit string length. The ratio $\zeta$ is specified by

$$\zeta = \frac{\mu_{SRM}}{\lambda_{SRM}} \cdot \frac{\lambda}{\mu}. \quad (5)$$

The mutation rate $p_m$ varies in the range $[p_m^{(t=0)}, 1/n]$ and, similar to the deterministic approach, it is applied to all individuals created by SRM.

## 4. 0/1 Multiple Knapsacks Problems

In the 0/1 multiple knapsack problem there are $m$ knapsacks and $n$ objects. The capacities of the knapsacks are $c_1, c_2, ..., c_m$. For each object there is a profit $p_i$ $(1 \leq i \leq n)$ and a set of weights $w_{ij}$ $(1 \leq j \leq m)$, one weight per knapsack. If an object is selected its profit is accrued and the knapsacks are filled with the object's weights. The problem consists on finding the subset of objects that maximizes profit without overfilling any of the knapsacks with objects' weights. The 0/1 multiple knapsack problem can be formulated to maximize the function

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} p_i x_i \quad (6)$$

subject to

$$\sum_{i=1}^{n} w_{ij} x_i \leq c_j \qquad (j = 1, ..., m) \quad (7)$$

where $x_i \in \{0,1\}$ $(i = 1, ..., n)$ are elements of a solution vector $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, which is the combination of objects we are interested in finding. Solutions to this problem have a natural binary representation in the GA constructed by mapping each object to a locus within the binary chromosome. A 1 in locus $i$ indicates that the object $i$ is being selected and a 0 otherwise. A solution vector $\boldsymbol{x}$ should guarantee that no knapsack is overfilled and the best solution should yield the maximum profit. An $\boldsymbol{x}$ that overfills at least one of the knapsacks is considered as an infeasible solution.

The 0/1 multiple knapsack problem is a NP-hard combinatorial optimization problem and its importance is well known both from a theoretical and practical point of view. It is a generalization of the 0/1 simple $(m = 1)$ knapsack problem, which can be used as subproblems to solve more complicated ones [19]; also, other combinatorial problems, such the partition problem, can be polynomially transformed into it [20]. Furthermore, well known NP-complete problems, such satisfiability problems (SAT), can be formulated as special instances of a 0/1 multiple knapsack problem [21]. Many practical problems can be formulated as a 0/1 multiple knapsack problem. Some applications of the problem include the capital budgeting problem, allocating processors and databases in a distributed computer system [22], project selection and cargo loading [23], cutting stock problems [24], and maximizing the number

of served clients or the use of bandwidth for ad hoc networks [25].

Test function generators [26] for broad classes of problems are seen as the correct approach for testing the performance of genetic algorithms (GAs). In our study we use a set of problems obtained from the OR-Library[☆] that consists of subclasses of difficult, large, and highly constrained 0/1 multiple knapsack problems. These subclasses of problems were created using a knapsack test problem generator, initially proposed in Ref. 15). The generator itself is based in the procedure suggested in Ref. 27) and its main characteristics are as follows:

(1) The weights $w_{ij}$ are drawn at random from a uniform distribution $U(0, 1000)$.

(2) For each combination of $m$-$n$, the capacities of the knapsacks are set by

$$c_j = \phi \sum_{j=1}^{n} w_{ij}$$

where $\phi$ is the tightness ratio.

(3) The profits of the objects are correlated to the weights of the objects[☆☆] by

$$p_i = \sum_{i=1}^{m} w_{ij}/m + 500q_j$$

where $q_j$ is a real number drawn from a continuous uniform distribution $U(0, 1)$.

With this test problem generator, besides defining the number of knapsacks $m$ and the number of objects $n$, it is also possible to define the tightness ratio $\phi$ between knapsack capacities and object weights. Each combination of $\phi$, $m$, and $n$ defines a subclass of problem. By systematically varying these parameters, 0/1 multiple knapsacks allows us to carefully observe the behavior and scalability of the algorithms in three important aspects that are correlated to the difficulty of a problem: number of constraints $m$, size of the search space $2^n$, and the ratio $\phi$ between the feasible region (knapsacks' capacities) and the whole search space (objects' weights). We use 7 subclasses and 10 random problems in each subclass. **Table 1** shows the combination of values of the parameters $\phi$, $m$, and $n$ used to define the subclasses of problems.

To deal with infeasible solutions a penalty term is introduced into the fitness function as follows

**Table 1** 7 subclasses of problems (10 random problems in each subclass).

| | *Paremeters* | | | |
|---|---|---|---|---|
| *Subclass* | $m$ | $n$ | $\phi$ | *Comment* |
| 1 | 30 | 100 | 0.75 | reducing |
| 2 | | | 0.50 | feasible |
| 3 | | | 0.25 | region |
| 4 | 5 | 100 | 0.25 | increasing |
| 5 | 10 | | | number |
| (3) | 30 | | | constraints |
| (3) | 30 | 100 | 0.25 | increasing |
| 6 | | 250 | | search |
| 7 | | 500 | | space |

$$f(\boldsymbol{x}) = \begin{cases} g(\boldsymbol{x})/(s \cdot max\{o_j\}) & (s > 0) \\ g(\boldsymbol{x}) & (s = 0) \end{cases} \tag{8}$$

where $s$ ($0 \le s \le m$) is the number of overfilled knapsacks and $o_j$ ($> 1$) is the overfilling ratio of knapsack $j$ calculated by

$$o_j = \sum_{i=1}^{n} w_{ij} x_i / c_j. \tag{9}$$

Note that the penalty term of $f$ is a function of both number of violated constraints ($s$) and distance from feasibility ($o_j$).

## 5. Experimental Setup

The following GAs are used in our simulations. A simple canonical GA that applies crossover followed by background mutation, denoted as cGA. Two parallel varying mutation GAs implemented following the GA-SMR model; one with the deterministic varying mutation schedule, denoted as GA-hM, and the other one with self-adaptive varying mutation schedule, denoted as GA-sM. Similarly, two conventional varying mutation GAs with the deterministic (Eq. (1)) and self-adaptive (Eq. (2)) mutation schedules, denoted hGA and sGA, respectively. The GAs use either proportional selection or ($\mu$,$\lambda$) proportional selection. This is indicated by appending to the name of the GA ($\mu$) or ($\mu$,$\lambda$), respectively[☆☆☆]. All algorithms use fitness linear scaling and mating is restricted to $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i \neq j$, so a solution will not cross with itself. Linear scaling is implemented as indicated in Ref. 10) (p.79), where the coefficients $a$ and $b$ that implement the linear transformation $f' = af + b$ are chosen to enforce equality of the raw ($f$) and scaled ($f'$) average fitness values and cause the maximum

☆ http://mscmga.ms.ic.ac.uk/jeb/orlib/info.html

☆☆ The correlation between profits and weights increases the difficulty of the problems [19].

☆☆☆ a simple GA with ($\mu$,$\lambda$) proportional selection is denoted GA($\mu$,$\lambda$)

scaled fitness to be twice the average fitness. For cGA, hGA, and sGA $p_c = 0.6$ and for GA-hM and GA-sM the ratio for offspring creation is set to $\lambda_{CM} : \lambda_{SRM} = 1 : 1$. Background mutation is set to $p_m^{(CM)} = 1/n$. The learning rate for self-adaptation is set to $\gamma = 0.2$

The initial population is randomly initialized with a 0.25 probability for 1s. Results are averaged over 50 runs and the number of generations is set to $T = 5000$.

## 6. Simple GA and Extinctive Selection

Since most varying mutation algorithms combine higher mutations with a kind of extinctive (truncated) selection, it is important to try to asses the contributions to performance of a higher selection pressure and varying mutations. In order to do so, we first observe the effect of extinctive selection on the performance of a simple GA. **Figure 3** plots the fitness of the best-so-far individual over the generations by the canonical cGA(100) and a simple GA using $(\mu, \lambda) = \{(15, 100), (50, 100)\}$ populations.

From this figure we see that extinctive selection alone remarkable improves the solution quality reached by the cGA in this kind of problems. As mentioned before, the problems used in this study are highly constrained with sparse feasible regions where algorithms with penalty functions have a hard time finding feasible solutions [3),15)]. A higher selection pressure in these problems is helping the algorithm to focus the search around the feasible regions. Extinctive selection has also another effect. It increases the convergence speed of the algorithm [18)]. Both GA(15,100) and GA(50,100) are faster than cGA(100). However, a population of (50,100) gives better final results than (15,100). In the following we use the results of GA(50,100) as reference for comparison with the varying mutation algorithms.

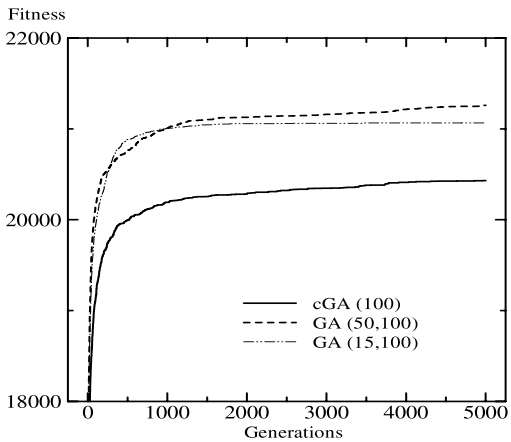



**Fig. 3** Effect of extinctive selection on a simple GA ($m = 30$, $n = 100$, $\phi = 0.25$).

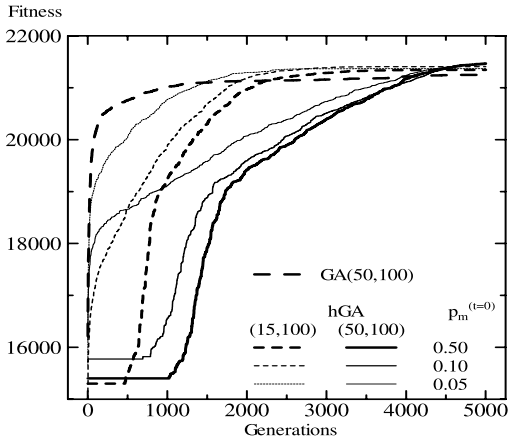
## 7. Comparing the Conventional and Parallel Varying Mutation Models

### 7.1 Deterministic Varying Mutation and Extinctive Selection

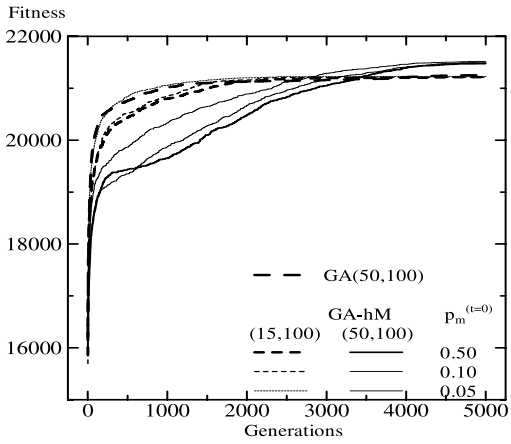
Deterministic mutation varies mutation rates with exactly the same schedule whether it is applied serial (hGA) or parallel to crossover (GA-hM) and therefore is an ideal candidate to isolate and observe the impact of higher mutations in both models of GAs. Experiments are conducted using various populations $(\mu, \lambda) = \{(15, 100), (50, 100), (100, 100)\}$ and initial mutation probabilities $p_m^{(t=0)} = \{0.50, 0.10, 0.05\}$. **Figure 4** (a) and (b) plot the average fitness of the best-so-far individual over the generations illustrating the convergence behavior by hGA and GA-hM, respectively. Results by GA (50,100) are also included for comparison.

From Fig. 4 (a) we can observe that hGA's convergence becomes faster increasing extinctive pressure (reduce $\mu$ while keeping constant $\lambda$). However, better final results were given by ($\mu$,$\lambda$)=(50,100) rather than by ($\mu$,$\lambda$)=(15,100). Setting initial mutation probability to lower values also helps to speed up convergence. These, however, do not help to increase the quality of the final results. Note the initial *flat* periods in which the fitness of the best-so-far individual did not improve. This is a clear indication of the disruption caused by high mutation after crossover.

From Fig. 4 (b) we can see that increasing extinctive selection and reducing initial mutation probability in GA-hM produce similar effects to those remarked for hGA. Looking at both Fig. 4 (a) and Fig. 4 (b) becomes apparent that varying mutation parallel to crossover is less disruptive than varying mutation serial to crossover. Contrary to hGA, in the case of GA-hM there are no initial *flat* periods and in all cases GA-hM converges faster than hGA for similar values of ($\mu$,$\lambda$) and $p_m^{(t=0)}$. Also, as a consequence of this less disruptiveness, the initial value set for varying mutation in GA-hM has a smaller impact on convergence speed than it does in hGA. See for example GA-hM(50,100) for $p_m^{(t=0)} = 0.5$ and $p_m^{(t=0)} = 0.05$ and compare




(a) Serial to Crossover




(b) Parallel to Crossover

**Fig. 4** Deterministic varying mutation ($m = 30$, $n = 100$, $\phi = 0.25$).

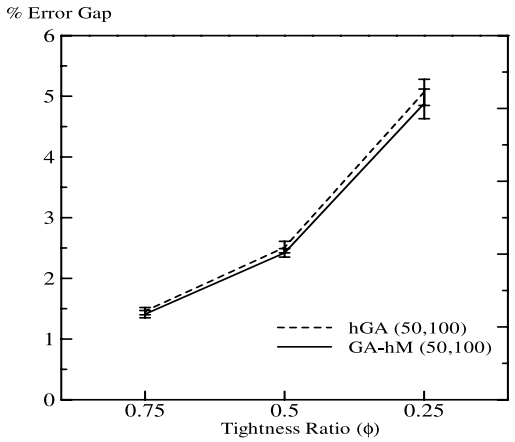



**Fig. 5** Convergence reliability of deterministic varying mutation GAs: reducing the feasible region $\phi = \{0.75, 0.50, 0.25\}$, $m = 30$, $n = 100$.

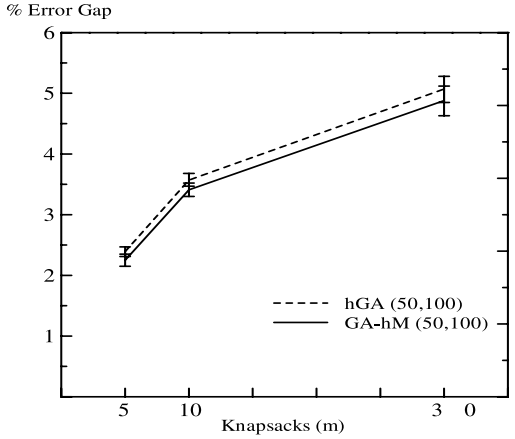



**Fig. 6** Convergence reliability of deterministic varying mutation GAs: increasing the number of constraints $m = \{5, 10, 30\}$, $n = 100$, $\phi = 0.25$.

it with hGA for similar settings. Thus, GA-hM is more robust than hGA to initial settings of mutation rate.

In GA-hM, similar to hGA, a $(\mu, \lambda)$=(50,100) population gives better final results than $(\mu, \lambda)$=(15,100). In fact, note that GA-hM(15,100)'s final quality is not better than GA(50,100)'s that does not apply varying mutations. A (15,100) extinctive selection turns out to be too strong for GA-hM. A less strong selection pressure, such (50,100), gives a better chance to hM's offspring to compete with CM's offspring, which in turn helps to improve the search process.

The quality of the solutions found by the algorithms are measured by the average percentage error gap in a subclass of problems, which is calculated as the normalized difference between the best solutions found and the optimal value given by the linear programming relaxation (LP) (the optimal integer solutions are unknown)[15]. **Figures 5**, **6**, and **7** plot the average percentage error gap by hGA and GA-hM. These figures show the effect on convergence reliability of reducing the feasible region ($\phi$), increasing the number of constraints ($m$), and increasing the search space ($n$), respectively. The vertical bars, overlaying the mean curves, represent 95% confidence intervals.

The statistical significance of the results achieved by hGA and GA-hM is verified conducting a factorial analysis of variance (factorial ANOVA). ANOVA is a procedure for comparing multiple population means. Rejecting the null hypothesis (that the population means are equal, in this case the means achieved by

**Table 2** Factorial ANOVA for hGA and GA-hM.

| *Source* | *SS* | *df* | *MS* | *F* | *Pval* |
|---|---|---|---|---|---|
| GA | 0.18371 | 1 | 0.18371 | 3.260 | 0.0766 |
| $\phi$ | 132.47870 | 2 | 66.23935 | 1175.553 | 0.0000 |
| GA-$\phi$ | 0.05120 | 2 | 0.02560 | 0.454 | 0.6373 |
| Error | 3.04276 | 54 | 0.05635 | | |
| Total | 135.75637 | 59 | | | |
| GA | 0.40017 | 1 | 0.40017 | 6.219 | 0.0157 |
| $m$ | 70.77277 | 2 | 35.38639 | 549.927 | 0.0000 |
| GA-$m$ | 0.00654 | 2 | 0.00327 | 0.051 | 0.9505 |
| Error | 3.47476 | 54 | 0.06435 | | |
| Total | 74.65424 | 59 | | | |
| GA | 0.54150 | 1 | 0.54150 | 8.685 | 0.0047 |
| $n$ | 11.72973 | 2 | 5.86487 | 94.062 | 0.0000 |
| GA-$n$ | 0.00121 | 2 | 0.00060 | 0.010 | 0.9903 |
| Error | 3.36696 | 54 | 0.06235 | | |
| Total | 15.63940 | 59 | | | |

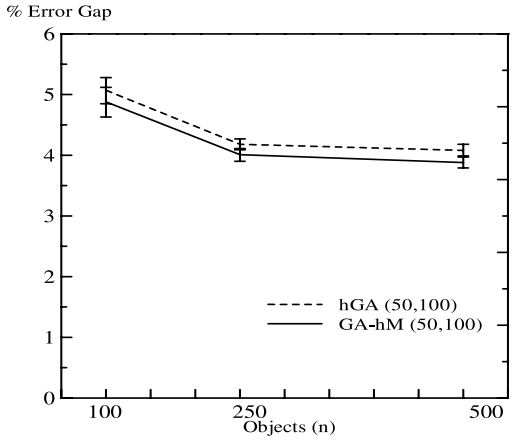



**Fig. 7** Convergence reliability of deterministic varying mutation GAs: increasing the search space $n = \{100, 250, 500\}$, $m = 30$, $\phi = 0.25$.

hGA and GA-hM) implies that at least one of the population means differ from the others. The factorial experiments examine the effects of at least two factors, each having at least two levels. In this case one factor is the type of GA (with levels hGA and GA-hM) and the other factor is a parameter of the problem, for example $\phi$ (with levels 0.75, 0.5 and 0.25). Each factor is tested for a main effect. If the null hypothesis is rejected, it can be concluded that there are differences among the means of the populations corresponding to the factor levels. The factorial ANOVA also provides information about the interactions between factors, see for example Ref. 28).

**Table 2** summarizes the three two-factor factorial ANOVA corresponding to the plots presented in Figs. 5, 6, 7. In Table 2 *Source* indicates the source of variation, *SS* the sum of squares, *df* the degrees of freedom, *MS* is the mean square, which is the result of dividing the sum of squares by its degrees of freedom, $F$ is the ratio between the mean square treatment and the mean square error, and *Pval* is the $p$ *value*, the smallest significant level $\alpha$ that would allow rejection of the null hypothesis (i.e., that the means of hGA and GA-hM are the same).

From Table 2, inspection of the p values reveals that in the case of reducing the feasible region ($\phi$) there is *some indication* of an effect by the GA type factor (conventional/parallel application of deterministic varying mutation). Note that $Pval = 0.0766$ is not much greater than $\alpha = 0.05$. In the cases of increasing the number of constraints ($m$) and the size of the search space ($n$) there are indications of a *strong main effect* by the GA type concluding that the parallel deterministic varying mutation (GA-hM) attains significantly smaller error than the conventional deterministic varying mutation GA (hGA). Notice that the p values 0.0157 and 0.0047, respectively, are considerably less than 0.05. Furthermore, note that in all three cases there is indication of a main effect by the problem difficulty factor ($\phi$, $m$, and $n$). Nothing can be concluded for a possible interaction between GA type and problem difficulty factor ($F < 1$).

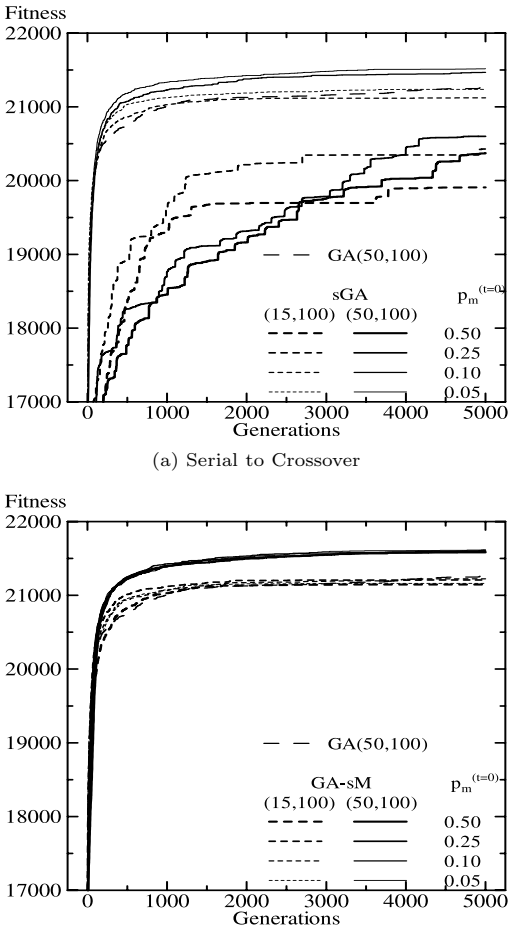
### 7.2 Self-Adaptive Varying Mutation and Extinctive Selection

A self-adaptive scheme uses one mutation rate per individual, which are usually set at $t = 0$ to random values in the range allowed for mutation. Two important ingredients of self-adaptation are the diversity of parameter settings and the capability of the method to adapt the parameters. It has been indicated that some of the implementations of self-adaptation ex-

ploit more the diversity of parameter settings rather than adapting them. However, it has also been argued that the key to the success of self-adaptation seems to consist in using at the same time both a reasonably fast adaptation and reasonably large diversity to achieve a good convergence velocity and a good convergence reliability, respectively [8].

To observe the influence that the conventional/parallel application of varying mutations could have on the self-adaptive capability itself we avoid initial diversity of parameters. Experiments are conducted using populations $(\mu, \lambda) = \{(15, 100), (50, 100)\}$ and mutation ranges of $p_m = [p_m^{min}, p_m^{max}] = [1/n, \{0.50, 0.25, 0.10, 0.05\}]$. In all cases initial mutation for each individual is set to the maximum value allowed for the range, $p_m^{(t=0)} = p_m^{max}$. **Figure 8** (a) and (b) plot the average fitness of the best-so-far individual over the generations illustrating the convergence behavior by sGA and GA-sM, respectively. Results by GA(50,100) are also included for comparison.

From Fig. 4 and Fig. 8 it is worth noting the following. (i) Self-adaptive mutation increases convergence speed compared to deterministic mutation either serial or parallel to crossover. Looking at Fig. 8 (a) and Fig. 4 (a), note that in sGA the initial *flat* periods observed in hGA have disappeared completely. Also, looking at Fig. 8 (b) and Fig. 4 (b) we can see that GA-sM(50,100)'s fitness picks up much earlier than GA-hM(50,100)'s for similar values of $p_m^{(t=0)}$. Between sGA and GA-sM, however, looking at Fig. 8 (a) and (b) note that sGA can match GA-sM's convergence velocity only for small values of $p_m^{(t=0)}$. This is an indication that even in the presence of adaptation the convergence velocity of a conventional varying mutation GA would depend heavily on initial mutation rates, which is not an issue if adaptive mutation is applied parallel to crossover. (ii) Contrary to deterministic varying mutation, convergence reliability of a conventional self-adaptive varying mutation GA could be severely affected, which becomes quite notorious if no initial diversity of parameters is allowed. Note in Fig. 8 (a) that only the configurations of sGA(50,100) having $p_m^{(t=0)} = \{0.10, 0.05\}$ achieved better final results than GA(50,100). On the other hand, the initial lack of diversity of parameters does not affect convergence reliability of GA-sM. Note in Fig. 8 (b) that for the same selection pressure convergence reliability of GA-sM is similar for all values of $p_m^{(t=0)}$. (iii) Similar to deterministic varying mutation, better results are achieved by $(\mu, \lambda) = (50, 100)$ rather than by $(\mu, \lambda) = (15, 100)$.

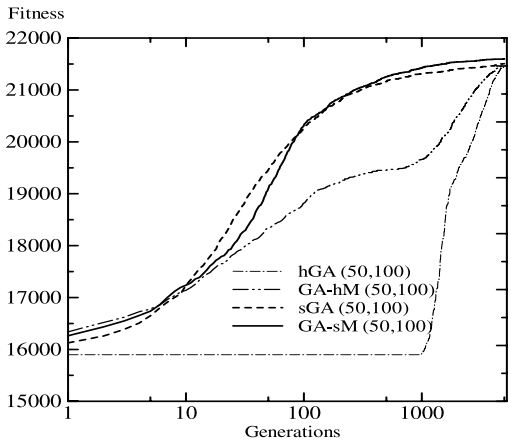



(a) Serial to Crossover

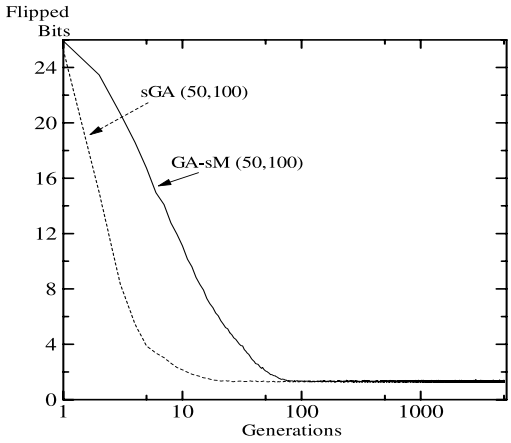
(b) Parallel to Crossover

**Fig. 8** Self-adaptive varying mutation $p_m^{(t=0)}(i) = p_m^{max}$ ($m = 30$, $n = 100$, $\phi = 0.25$).

Next, we allow for initial diversity of parameters setting $p_m^{(t=0)}$ to a random value between the minimum and maximum value allowed for mutation. In this case, the disruption that higher serial mutation causes to crossover becomes less apparent due to the initial diversity of parameters and convergence speed is similar for both sGA and GA-sM. Convergence reliability of sGA also improves. However, the negative impact on reliability remains quite significant for sGA (see below). **Figure 9** (a) and (b) illustrates the fitness transition and the average flipped bits (Log scale) by sGA and GA-




(a) Convergence Velocity




(b) Average Number of Flipped Bits

**Fig. 9** Convergence velocity and average number of flipped bits ($m = 30$, $n = 100$, $\phi = 0.25$). $p_m^{(t=0)} = 0.5$ for hGA and GA-hM. $p_m^{(t=0)}(i) = rand[1/n, 0.5]$ for sGA and GA-SM.

sM both with random initial mutation rates between [1/n,0.50]. Results for hGA and GA-hM are also included in Fig. 9 (a) for comparison. From these figures note that sGA converges to lower fitness and reduces mutation rates faster than GA-sM.

The self-adaptation principle tries to exploit the indirect link between favorable strategy parameters and objective function values. That is, appropriate parameters would lead to fitter individuals, which in turn are more likely to survive and hence propagate the parameter they carry with them to their offspring. A GA that applies varying mutation parallel to crossover as GA-sM can interpret better the self-adaptation principle and achieve higher performance because (i) inappropriate mutation parameters do

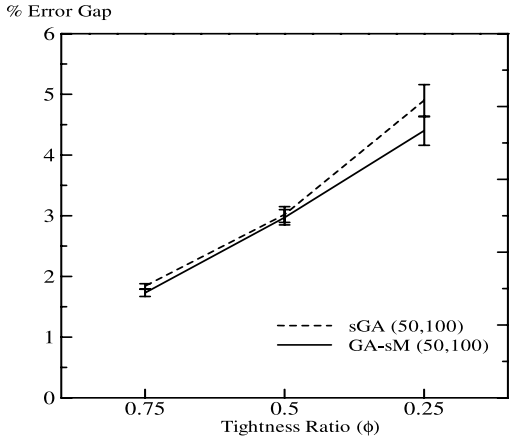



**Fig. 10** Convergence reliability of self-adaptive varying mutation GAs: reducing the feasible region $\phi = \{0.75, 0.50, 0.25\}$, $m = 30$, $n = 100$.

not disrupt crossover, and (ii) it preserves mutation rates (see Eq. (3)) that are being useful to the search. A GA that applies varying mutation serial to crossover as sGA, however, can mislead the mutation rate control because (i) appropriate parameters could be eliminated due to ineffective crossover operations, and (ii) in sGA an appropriate parameter implies parameters that would not affect greatly crossover. Thus, in sGA there is a selective bias towards smaller mutation rates. Algorithms based on a model like GA-SRM, where varying mutations are detached from crossover, (self) adapts mutation rates based only on the instantaneous effectiveness of varying mutations. In the case of conventional varying mutation GAs, however, mutation rates are (self) adapted based on the combined effectiveness of crossover and mutation, which can mislead the mutation rate control negatively affecting performance.

**Figures 10**, **11**, **12** plot the average percentage error gap by sGA and GA-sM showing the effect on performance of reducing the feasible region ($\phi$), increasing the number of constraints ($m$), and increasing the search space ($n$). **Table 3** summarizes the three two-factor factorial ANOVA corresponding to the plots presented in Figs. 10, 11, 12.

From Table 3, inspection of the p values reveals that in all three cases (reducing the feasible region, increasing the number of constraints, and increasing the size of the search space) there are indications of a *strong main effect* by the GA type concluding that the parallel self-adaptive varying mutation GA (GA-sM) at-

**Table 3** Factorial ANOVA for sGA and GA-sM.

| *Source* | *SS* | *df* | *MS* | *F* | *Pval* |
|---|---|---|---|---|---|
| GA | 0.70634 | 1 | 0.70634 | 9.731 | 0.0029 |
| $\phi$ | 82.77031 | 2 | 41.38516 | 570.167 | 0.0000 |
| GA-$\phi$ | 0.59193 | 2 | 0.29596 | 4.078 | 0.0224 |
| Error | 3.91955 | 54 | 0.07258 | | |
| Total | 87.98813 | 59 | | | |
| GA | 0.89060 | 1 | 0.89060 | 12.418 | 0.0009 |
| $m$ | 74.48359 | 2 | 37.24180 | 519.263 | 0.0000 |
| GA-$m$ | 0.69806 | 2 | 0.34903 | 4.867 | 0.0114 |
| Error | 3.87291 | 54 | 0.07172 | | |
| Total | 79.94516 | 59 | | | |
| GA | 3.33233 | 1 | 3.33233 | 42.053 | 0.0000 |
| $n$ | 9.25077 | 2 | 4.62539 | 58.371 | 0.0000 |
| GA-$n$ | 0.08025 | 2 | 0.04013 | 0.506 | 0.6055 |
| Error | 4.27902 | 54 | 0.07924 | | |
| Total | 16.94237 | 59 | | | |

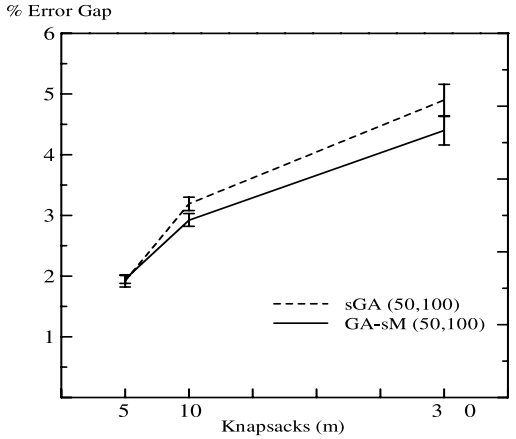



**Fig. 11** Convergence reliability of self-adaptive varying mutation GAs: increasing the number of constraints $m = \{5, 10, 30\}$, $n = 100$, $\phi = 0.25$.

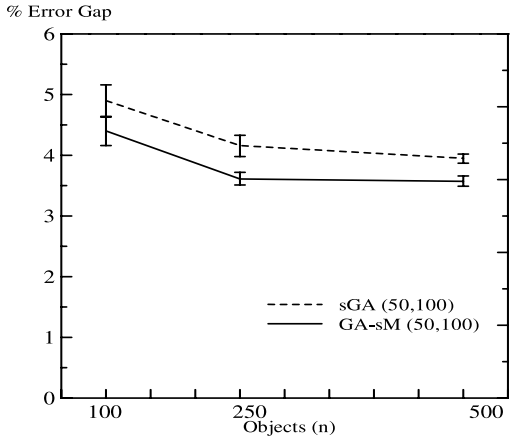



**Fig. 12** Convergence reliability of self-adaptive varying mutation GAs: increasing the search space $n = \{100, 250, 500\}$, $m = 30$, $\phi = 0.25$.

tains significantly smaller error than the conventional self-adaptive varying mutation GA (sGA). Notice that the p values 0.0029, 0.0009, and 0.0000, respectively, are considerably less than 0.05. Furthermore, note that in all three cases there are indications of a main effect by the problem difficulty factor ($\phi$, $m$, and $n$). In other words, increasing the difficulty of the problem makes it more difficult for the self-adaptive GA to find good solutions. In addition, note that the interaction between GA type and problem difficulty factor was significant for ratio $\phi$ and number of constraints $m$ ($F > 1$ and very small $Pval$), which means that the self-adaptive parallel varying mutation GA (GA-sM) not only performs better but also scales up better than the conventional self-adaptive varying mutation GA (sGA) as the difficulty of the problem increases.

## 8. Comparing Mutation Schedules in the Parallel Varying Mutation Model

In Section 7 we compared the conventional and parallel varying mutation models using deterministic and self-adaptive mutation schedules in both models of GAs. In this section we compare different mutation schedules within the parallel model of varying mutation. Besides the deterministic (GA-hM) and self-adaptive (GA-sM) mutation schedules used above we also include results by the adaptive scheme (GA-aM) described in Eq. (4) and used in Ref. 13). For GA-aM the initial mutation rate is set to $p_m^{(t=0)} = 0.5$ and the mutation rate reduction factor is set to $\beta = 0.7$, which allows a smooth reduction of mutation rates es-

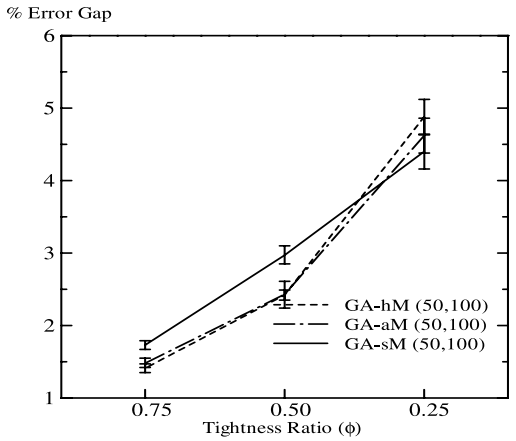



**Fig. 13** Convergence reliability of deterministic, adaptive, and self-adaptive parallel varying mutation GAs: reducing the feasible region $\phi = \{0.75, 0.50, 0.25\}$, $m = 30$, $n = 100$.

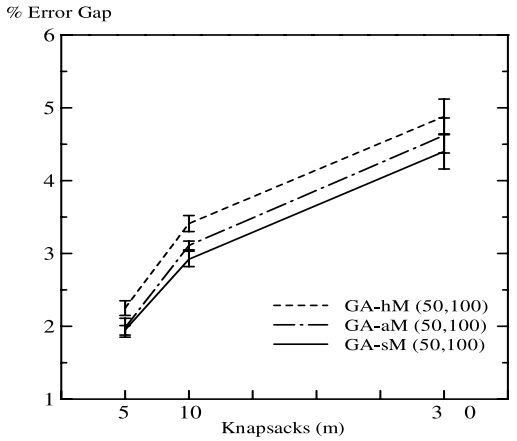



**Fig. 14** Convergence reliability of deterministic, adaptive, and self-adaptive parallel varying mutation GAs: increasing the number of constraints $m = \{5, 10, 30\}$, $n = 100$, $\phi = 0.25$.

pecially in the later stages of the search. The threshold $\tau$ to trigger adaptation is sampled for each subclass of problems. The values of $\tau$ used are 0.58 for subclass 5, 0.60 for subclasses 1, 3 and 4, and 0.64 for subclasses 2, 6, and 7 (for the problem subclass number and its characteristics see Table 1). Note that $\tau$ is not sampled for each problem within a subclass. The other parameters for GA-aM are the same used by GA-hM and GA-sM (see Section 5).

**Figures 13**, **14**, **15** plot the average percentage error gap by GA-hM, GA-aM, and GA-sM showing the effect on performance of reducing the feasible region ($\phi$), increasing the number of constraints ($m$), and increasing the search space ($n$).

From these figures we can see that the overall performance of the self-adaptive GA-sM is better than the performance of the deterministic GA-hM and adaptive GA-aM. The only region where GA-sM performs worse than the two other algorithms is for problems with wide feasible region (Fig. 13, $\phi = 0.75$ and $\phi = 0.5$). A self-adaptive algorithm requires additional resources and is more complex than the other two (one mutation rate per individual instead of the one mutation rate for the whole population). These results suggest that for less difficult problems simpler algorithms like GA-hM and GA-aM would be sufficient to approach the global optimum. However, for difficult problems more complex but well engineered GAs like GA-sM are needed.

The performance of the adaptive GA-aM is better than the deterministic GA-hM for all

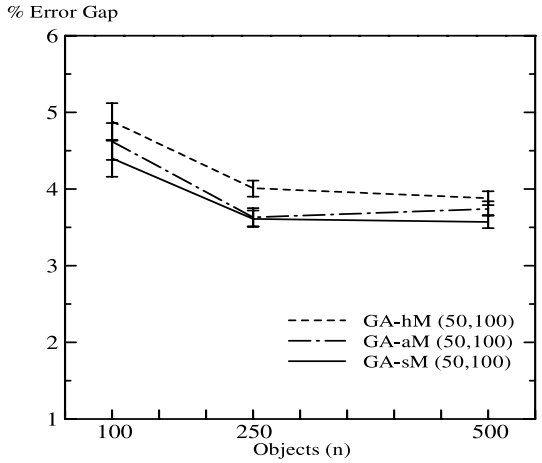



**Fig. 15** Convergence reliability of deterministic, adaptive, and self-adaptive parallel varying mutation GAs: increasing the search space $n = \{100, 250, 500\}$, $m = 30$, $\phi = 0.25$.

subclasses of difficult problems (Fig. 13 $\phi = 0.25$, Fig. 14 and Fig. 15). For more simple problems the performance of both algorithms is similar (Fig. 13, $\phi = 0.75$ and $\phi = 0.5$). It is also important to note that in GA-aM the parameter to trigger adaptation offers a tradeoff. At the expense of fine tuning this parameter we can seek to increase the performance of GA-aM approaching and in some cases performing better than GA-sM. Real world applications sometimes impose additional constraints, like memory usage. In these applications it may not be feasible to use a self-adaptive algorithm and an adaptive algorithm could be the best option.

Finally, it should be mentioned that the per-

formance of GA-hM, despite being the worst among the parallel varying mutation algorithms studied here, is by far better than the performance of a simple GA. For example, for the subclasses of problems $n = \{100, 250, 500\}$, $m = 30$, and $\phi = 0.25$ (Fig. 15) the percentage error gaps achieved by GA-hM(50,100) are {4.88, 4.01, 3.88}, respectively, while the percentage error gaps achieved by a cGA(100) are {9.34, 9.58, 9.41} and by a GA(50,100) are {6.03, 5.83, 6.54}.

## 9. Conclusions

We have studied varying mutations applied parallel to crossover in generational deterministic and self-adaptive varying mutation GAs and compared them with the conventional generational model of varying mutations that apply mutation mostly serial to crossover. Experiments were conducted with several subclasses of 0/1 multiple knapsacks problems. We found that varying mutation parallel to crossover can be a more effective and efficient framework than the conventional model of varying mutations in both deterministic and self-adaptive GAs. In the case of deterministic mutation GAs, a GA with varying mutation parallel to crossover showed faster convergence and higher robustness to initial settings of mutation rate than a conventional varying mutation GA. Also, an ANOVA gave some indication of higher convergence reliability by the parallel application of deterministic varying mutation. In the case of self-adaptive GAs, the convergence velocity of a parallel self-adaptive mutation GA was matched by a conventional self-adaptive mutation GA only when initial diversity of parameters was allowed. Convergence reliability was higher for the parallel varying self-adaptive mutation GA with or without initial diversity of parameters. An ANOVA gave a strong indication in this direction. We also found that the conventional model of varying mutations affects negatively the self-adaptive mutation rate control. This strongly suggests that the conventional model of varying mutation GAs may not be appropriate for combining forms of control.

Deterministic, adaptive, and self-adaptive mutation schedules were also compared within the parallel varying mutation model. Best overall performance was achieved by the parallel varying mutation self-adaptive GA.

In the future, the effectiveness of varying mutation parallel to crossover should be further studied with other important classes of problems. Also, it is worth pursuing co-adaptation of parameters within the parallel varying mutation model.

## References


1) Eiben, A.E., Hinterding, R. and Michalewicz, Z.: Parameter Control in Evolutionary Algorithms, *IEEE Trans. on Evolutionary Algorithms*, Vol.3, No.2, pp.124–141 (1996).
2) Fogarty, T.: Varying the Probability of Mutation in the Genetic Algorithm, *Proc. 3rd Int'l Conf. on Genetic Algorithms*, Morgan Kaufmann, pp.104–109 (1989).
3) Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 3rd ed. (1996).
4) Bäck, T. and Schutz, M.: Intelligent Mutation Rate Control in Canonical Genetic Algorithms, *Proc. 9th Int. Symp. ISMIS'96*, Lecture Notes on Artificial Intelligence, Vol.1079, pp.158–167, Springer (1996).
5) Rechenberg, I.: *Cybernetic Solution Path of an Experimental Problem*, Royal Aircraft Establishment Library (1965).
6) Fogel, L.J., Owens, A.J. and Walsh, M.J.: *Artificial Intelligence through Simulated Evolution*, Wiley (1966).
7) Smith, J. and Fogarty, T.C.: Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm, *Proc. IEEE Int'l. Conf. on Evolutionary Computation*, pp.318–326 (1996).
8) Bäck, T.: Self-adaptation, *Handbook of Evolutionary Computation*, pp.C7.1:1–13, IOP and Oxford University Press (1997).
9) Holland, J.H.: *Adaptation in Natural and Artificial Systems*, Univ.of Michigan Press (1975).
10) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading (1989).
11) Aguirre, H., Tanaka, K. and Sugimura, T.: Cooperative Model for Genetic Operators to Improve GAs, *Proc. IEEE Int'l Conf. on Information, Intelligence and Systems*, pp.98–106 (1999).
12) Aguirre, H., Tanaka, K., Sugimura, T. and Oshita, S.: Cooperative-Competitive Model for Genetic Operators: Contributions of Extinctive Selection and Parallel Genetic Operators, *Proc.Late Breaking Papers at 2000 Genetic and Evolutionary Computation Conference*, pp.6–14 (2000).
13) Aguirre, H., Tanaka, K. and Sugimura, T.: Empirical Model with Cooperative-Competitive Genetic Operators to Improve GAs: Performance Investigation with 0/1 Multiple Knapsack Problems, *IPSJ Journal*, Vol.41, No.10,

pp.2837–2851 (2000).
14) Aguirre, H., Tanaka, K. and Oshita, S.: Increasing the Robustness of Distributed Genetic Algorithms by Parallel Cooperative-Competitive Genetic Operators, *Proc. 2001 Genetic and Evolutionary Computation Conference*, pp.195–202, Morgan Kaufmann (2001).
15) Chu, P.C. and Beasley, J.E.: A Genetic Algorithm for the Multidimensional Knapsack Problem, *Journal of Heuristics*, Vol.4, pp.63–86 (1998).
16) Aguirre, H. and Tanaka, K.: Parallel Varying Mutation Genetic Algorithms, *Proc. IEEE Int'l. Conf. on Evolutionary Computation*, pp.795–800 (2002).
17) Aguirre, H. and Tanaka, K.: Parallel Varying Mutation in Deterministic and Self-Adaptive GAs, *Proc. Int'l. Conf. on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, Vol.2439, pp.111–121, Springer (2002).
18) Bäck, T.: *Evolutionary Algorithms in Theory and Practice*, Oxford Univ. Press (1996).
19) Martello, S. and Toth, P.: *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester, West Sussex, England (1990).
20) Garey, M.R. and Johnson, S.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco (1979).
21) Lokketangen, A. and Glover, F.: Surrogate Constraint Analysis — New Heuristics and Learning Schemes for Satisfiability Problems, *Satisfiability Problem: Theory and Applications*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol.35, pp.537–572 (1997).
22) Gavish, B. and Pirkul, H.: Allocation of Databases and Processors in a Distributed Computing System, *Management of Distributed Data Processing*, pp.215–231 (1982).
23) Shih, W.: A Branch and Bound Method for the Multiconstraint Zero-One Knapsack Problem, *Journal of the Operational Research Society*, Vol.30, pp.369–378 (1979).
24) Gilmore, P.C. and Gomory, R.E.: The Theory and Computation of Knapsack Functions, *Operations Research*, Vol.14, pp.1045–1075 (1966).
25) Bertet, K., Chaudet, C., Guérin, L.I. and Viennot, L.: Impact of Interferences on Bandwidth Reservation for Ad Hoc Networks: A First Theoretical Study, *Proc. IEEE Symposium on Ad-Hoc Wireless Networks* (2001).
26) De Jong, K.A., Potter, M.A. and Spears, W.M.: Using Problem Generators to Explore the Effects of Epistasis, *Proc. 7th Int'l Conf. Genetic Algorithms*, pp.338–345, Morgan Kaufmann (1997).
27) Freville, A. and Plateau, G.: An Efficient Preprocessing Procedure for the Multidimensional 0-1 Knapsack Problem, *Discrete Applied Mathematics*, Vol.49, pp.189–212 (1994).
28) Glenberg, A.: Learning from DATA: An Introduction to Statistical Reasoning, Harcourt Brace Javanovich Publishers, Orlando (1988).

(Received April 12, 2003)
(Revised May 30, 2003)
(Accepted June 13, 2003)

**Hernán Aguirre** received his Engineer degree in computer systems from Escuela Politécnica Nacional, Quito, Ecuador in 1992. From 1997 to 2003 he was a research scholar sponsored by the Japanese Ministry of Education, Culture, Sports, Science and Technology. He received the M.S. and Ph.D. degrees from Shinshu University, Japan, in 2000 and 2003, respectively. Currently, he is a postdoctoral research fellow at Shinshu University. His research interests include evolutionary computation, machine intelligence, bioinformatics, and their applications. He is a member of IEEE, IEICE, and ISGEC.

**Kiyoshi Tanaka** received B.S. and M.S. degrees from National Defense Academy, Yokosuka, Japan in 1984 and 1989, respectively. In 1992, he received Dr. Eng. degree from Keio University, Tokyo, Japan. In 1995, he joined the Department of Electrical and Electronic Engineering, Faculty of Engineering, Shinshu University, Nagano, Japan, where he is currently an associate professor. His research interests include image processing, digital watermarking, information security and coding, chaos & fractals, evolutionary computation, and their applications. He is a member of IEEE, SPIE, EURASIP, ISGEC, IEICE, IPSJ, IIEEJ and SITA.