# Learned Mutation Strategies in Genetic Programming for Evolution and Adaptation of Simulated Snakebot

Ivan Tanev
Department of Information Systems Design, Doshisha University,
1-3 Miyakodani, Tatara, Kyotanabe, Kyoto 610-0321, Japan
ATR Network Informatics Laboratories,
2-2-2 Hikaridai, "Keihanna Science City", Kyoto 619-0288, Japan
itanev@mail.doshisha.ac.jp

## ABSTRACT
In this work we propose an approach of incorporating learned mutation strategies (LMS) in genetic programming (GP) employed for evolution and adaptation of locomotion gaits of simulated snake-like robot (Snakebot). In our approach the LMS are implemented via learned probabilistic context-sensitive grammar (LPCSG). The LPCSG is derived from the originally defined context-free grammar, which usually expresses the syntax of genetic programs in canonical GP. Applying LMS implies that the probabilities of applying each of particular production rules in LPCGS during the mutation depend on the context. These probabilities are learned from the aggregated reward values obtained from the parsed syntax of the evolved best-of-generation Snakebots. Empirically obtained results verify that LMS contributes to the improvement of computational effort of both (i) the evolution of the fastest possible locomotion gaits for various fitness conditions and (ii) the adaptation of these locomotion gaits to challenging environment and degraded mechanical abilities of Snakebot. In all of the cases considered in this study, the locomotion gaits, evolved and adapted employing GP with LMS feature higher velocity and are obtained faster than with canonical GP.

## Categories and Subject Descriptors
G.1.6–Global Optimization; J.2-Physics

## General Terms: Algorithms, design

## Keywords: Mutation strategies, genetic programming, locomotion, Snakebot, context-sensitive grammar.

## 1. INTRODUCTION
Wheelless, limbless snake-like robots (Snakebots) feature potential robustness characteristics beyond the capabilities of most wheeled and legged vehicles – ability to traverse terrain that would pose problems for traditional wheeled or legged robots, and insignificant performance degradation when partial damage is inflicted. Some useful features of Snakebots include smaller size of the cross-sectional areas, stability, ability to operate in difficult terrain, good traction, high redundancy, and complete sealing of the internal mechanisms [4, 6].

Robots with these properties open up several critical applications in exploration, reconnaissance, medicine and inspection. However, compared to the wheeled and legged vehicles, Snakebots feature (i) more difficult control of locomotion gaits and (ii) inferior speed characteristics. In this work we intend to address the following challenge: how to automatically develop control sequences of Snakebot's actuators, which allow for achieving the fastest possible speed of locomotion.

In principle, the task of designing the controlling code of robots could be formalized and the formal mathematical models incorporated into direct programmable control strategies [6, 15, 17]. However, the eventual models of the Snakebot would feature enormous complexity and such models are not recognized to have a known, analytically obtained exact optimal solution. The complexity of the model stems from the considerable amount of degrees of freedom of the Snakebot, which cannot be treated independently of each other. The locomotion of the Snakebot is viewed as an emergent property at higher level of consideration of a complex hierarchical system, comprising many relatively simply defined entities (morphological segments). In such systems the higher-level properties of the system and the lower-level properties of comprising entities cannot be induced from each other.

The automated mechanisms for prompt generation of near-optimal solutions to such complex, ill-posed problems are usually based on various models of learning (ontogenesis) or evolution (phylogenesis) of species in the Nature [7, 9, 16]. The proposed approach of employing genetic programming (GP) implies that the code, which controls the locomotion of the Snakebot is automatically designed by a computer system via simulated evolution through selection and survival of the fittest in a way similar to the natural evolution of species [8]. GP (and evolutionary algorithms in general) is considered as an efficient way to tackle such difficult problems due to the ability to find a near-optimal solution in a reasonable runtime.

The *objectives* of our work are (i) to explore the feasibility of applying GP for efficient automatic design of the fastest possible locomotion of realistically simulated Snakebot and (ii) to investigate the adaptation of such locomotion to challenging environment and degraded abilities (due to partial damage) of simulated Snakebot. We are especially interested in the implications of the proposed incorporation of learned mutation strategies (LMS) in GP on the efficiency of evolution and adaptation of Snakebot.

Presented approach of incorporating LMS is implemented via learning probabilistic context-sensitive grammar (LPCSG), employed to express the preferable syntactical bias of mutation operation in GP. The proposed approach is related to the approach of grammatical evolution (GE) [10] in which the evolved genotype encodes the sequence of grammar rules, which should be applied during the simulated gene expression phase in order to generate the phenotype. Our work is also related to the incorporation of estimation of distribution algorithms (EDA) for biased mutations in evolutionary computations, mainly – in GA [5, 11, 12]. Motivated by the demonstrated advantages of both the GE and the EDA in GA, our work could be viewed as an attempt to fuse these two approaches in a way which allows for the biased mutation in GP (rather than GA, as in EDA) to be implemented via adjustable, learned preferences (rather than "hard coded" in the chromosome, as in GE) in applying the corresponding alternative grammar rules. Although a few grammar-based EDAs have been recently proposed [2, 13], in neither of these methods the incorporation of LPCSG in GP has been explored. Our interest in the feasibility of such approach additionally motivated us in this work.

The remainder of this document is organized as follows. Section 2 emphasizes the main features of GP proposed for evolution of locomotion of the Snakebot. Section 3 introduces the proposed approach of incorporating LPCSG in GP and discusses the empirically obtained results of efficiency of evolution and adaptation of Snakebot to challenging environment and partial damage. Section 4 draws a conclusion.

## 2. GP FOR AUTOMATIC DESIGN OF LOCOMOTION GAITS OF SNAKEBOT

### 2.1. Representation of Snakebot

Snakebot is simulated as a set of identical spherical morphological segments ("vertebrae"), linked together via universal joints. All joints feature identical (finite) angle limits and each joint has two attached actuators ("muscles"). In the initial, standstill position of Snakebot the rotation axes of the actuators are oriented vertically (vertical actuator) and horizontally (horizontal actuator) and perform rotation of the joint in the horizontal and vertical planes respectively. Considering the representation of Snakebot, the task of designing the fastest locomotion can be rephrased as developing temporal patterns of desired turning angles of horizontal and vertical actuators of each segment, that result in fastest overall locomotion of Snakebot. The proposed representation of Snakebot as a homogeneous system comprising identical morphological segments is intended to significantly reduce the size of the search space of the GP. Moreover, because the size of the search space does not necessarily increase with the increase of the complexity of Snakebot (i.e. the number of morphological segment), the proposed approach allows achievement of favorable scalability characteristics of GP.

### 2.2 Algorithmic Paradigm

#### *2.2.1 GP*

GP [8] is a domain-independent problem-solving approach in which a population of computer programs (individuals' genotypes) is evolved to solve problems. The simulated evolution in GP is based on the Darwinian principle of reproduction and survival of the fittest. The fitness of each individual is based on the quality with which the phenotype of the simulated individual is performing in a given environment.

#### *2.2.2 Function Set and Terminal Set*

In applying GP to evolution of Snakebot, the genotype is associated with two algebraic expressions, which represent the temporal patterns of desired turning angles of both the horizontal and vertical actuators of each morphological segment. Because locomotion gaits, by definition, are periodical, we include the periodic functions `sin` and `cos` in the function set of GP in addition to the basic algebraic functions. Terminal symbols include the variables `time`, `index` of the segment of Snakebot, and two constants: `Pi`, and `random` constant within the range [0, 2]. The main parameters of the GP are shown in Table 1.

#### *2.2.3 Context-free Grammar for Canonical GP*

The context-free grammar (CFG) *G,* usually employed to define the allowed syntax of individuals in GP consists of ($N$, $\Sigma$, $P$, $S$) where $N$ is a finite set of nonterminal symbols, $\Sigma$ is a finite set of terminal symbols that is disjoint from $N$, $S$ is a symbol in $N$ that is indicated as the start symbol, and $P$ is a set of production rules, where a rule is of the form

```
V -> w
```

where $V$ is a non-terminal symbol and $w$ is a string consisting of terminals and/or non-terminals. The term "context-free" comes from the feature that the variable $V$ can always be replaced by $w$, in no matter what context it occurs. The set of non-terminal symbols of $G$ of GP, is employed to develop the temporal patterns of desired turning angles of horizontal and vertical actuators of segments, that result in fastest overall locomotion of Snakebot, is defined as follows:

```
N = {GP, STM, STM1, STM2, VAR, CONST_x10,
CONST_PI, OP1, OP2}
```

where `STM` is a generic algebraic statement, `STM1` – a generic unary (e.g., `sin`, `cos`, `nop`) algebraic statement, `STM2` – a generic binary (dyadic, e.g. `+`, `-`, `*`, and `/`) algebraic statement, `VAR` – a variable, `OP1` – an unary operation, `OP2` – a binary (dyadic) operation, `CONST_x10` is a random constant within the range [0..20], and `CONST_PI` equals either `3.1416` or `1.5708`. The set of terminal symbols is defined as:

```
Σ = {sin,cos,nop,+,-, *,/,time,segment_id}
```

where `sin`, `cos`, `nop`, `+`, `-`, `*` and `/` are terminals which specify the functions in the generic algebraic statements. The start symbol is `GP`, and the set of production rules expressed in Backus-Naur form (BNF) are as shown in Figure 1. GP uses the defined production rules of $G$ to create the initial population and to mutate genetic programs. In the canonical GP the production rules with multiple alternative right-hand sides (such as rules 2, 4, 6, 7 and 9, shown in Figure 1) are usually chosen *randomly* during these operations.

### 2.2.4 Fitness Evaluation

The fitness function is based on the velocity of Snakebot, estimated from the distance, which the center of the mass of Snakebot travels during the trial. Fitness of 100 (the one of termination criteria shown in Table 1) is equivalent to a velocity, which displaced Snakebot a distance equal to twice its length.

**Table 1. Main parameters of GP**

| Category | Value |
|---|---|
| Function set | {sin, cos, nop, +, -, *, /} |
| Terminal set | {time, segment_ID, Pi, random constant} |
| Population size | 200 individuals |
| Selection | Binary tournament, selection ratio 0.1, reproduction ratio 0.9 |
| Elitism | Best 4 individuals |
| Mutation | Random subtree mutation, ratio 0.01 |
| Fitness | Velocity of simulated Snakebot during the trial |
| Trial interval | 180 time steps, each time step account for 50ms of "real" time |
| Termination criterion | (Fitness >100) *or* (Generations>40) |

```
(1)          GP ⟶ STM
(2.1-2.5)    STM ⟶ STM1|STM2|VAR|CONST_x10|CONST_PI
(3)          STM1 ⟶ OP1 STM
(4.1-4.6)     OP1 ⟶ sin|cos|nop|-|sqr|sqrt
(5)          STM2 ⟶ OP2 STM STM
(6.1-6.4)     OP2 ⟶ +|-|*|/
(7.1-7.2)    VAR  ⟶ time|segment_id
(8)          CONST_x10 ⟶ 0..20
(9.1-9.2)    CONST_PI  ⟶ 3.1416|1.5708
```

**Figure 1. BNF of production rules of the context free grammar G of GP, employed for automatic design of locomotion gaits of Snakebot. The following abbreviations are used: STM – generic algebraic statement, STM1 – unary algebraic statement, STM2 – binary (dyadic) algebraic statement, VAR – variable, OP1 – unary operation, and OP2 – binary operation**

### 2.2.5 Representation of Genotype

Inspired by its flexibility, and the recently emerged widespread adoption of document object model (DOM) and extensible markup language (XML) [18], we represent the evolved genotypes of the Snakebot as DOM-parse trees featuring equivalent flat XML-text. Both (i) the calculation of the desired turning angles during fitness evaluation and (ii) the genetic operations are performed on DOM-parse trees via API of the off-the shelf DOM-parser.

### 2.2.6 Genetic Operations

Selection is a binary tournament. Crossover is defined in a strongly typed way in that only the DOM-nodes (and corresponding DOM-subtrees) of the same data type (i.e. labeled with the same tag) from parents can be swapped. The sub-tree mutation is allowed in strongly typed way in that a random node in genetic program is replaced by syntactically correct sub-tree. The mutation routine refers to the data type of currently altered node and applies the chosen rule from the set of applicable rewriting rules as defined in the grammar of GP. The selection of the grammar rule, which should be applied to the currently altered tree node during the mutation is *random* in the canonical implementation of GP; and *biased* in the proposed approach of applying LMS as shall be elaborated in the following Section 3.

### 2.2.7 Open Dynamics Engine

We have chosen Open Dynamics Engine (ODE) [14] to provide a realistic simulation of physics in applying forces to phenotypic segments of the Snakebot. ODE is a free, industrial quality software library for simulating articulated rigid body dynamics. It is fast, flexible and robust, and it has built-in collision detection.

## 3. INCORPORATING LMS IN GP

## 3.1 Learning Probabilistic Context-Sensitive Grammar

The proposed approach is based on the idea of introducing bias in applying the most preferable rule from the grammar rules with multiple, alternative right-hand sides (RHS). We presume that the preferences of applying certain production rules depend on the surrounding grammatical context, defining which rules have been applied before. The initial probability distributions (PD) $p^i_1, p^i_2, \dots p^i_N$ for each $context_i$ for each grammar rule with multiple is RHS is even (equal) and then learned (tuned) incrementally at each generation from the subset of the best performing Snakebots. The learned PD is then used as a bias to steer the mutation of Snakebots.

In the proposed approach, the learning probabilistic context-sensitive grammar (LPCSG) *G** is proposed as a formal model describing such mutation. *G** is introduced as a set of the same attributes ($N^*$, $\sum^*$, $P^*$, $S^*$) as the CFG *G* defined in Section 2.2. The attributes $N^*$, $\sum^*$, and $S^*$ are identical to the corresponding attributes $N$, $\sum$, and $S$ of $G$. The set of production rules $P^*$ of $G^*$ are derived from $P$ of $G$ as follows:

(i) Production rules of $P_S$ ($P_S \subset P$) of $G$ which have a single right-hand side are defined in the same way in $P^*$ as in $P$, and
(ii) Production rules in $P_M$ ($P_M \subset P$) of $G$, which feature multiple right-hand side alternatives $V \rightarrow w_1 | w_2 | \dots | w_N$ are re-defined for each instance $i$ of the context as follows:

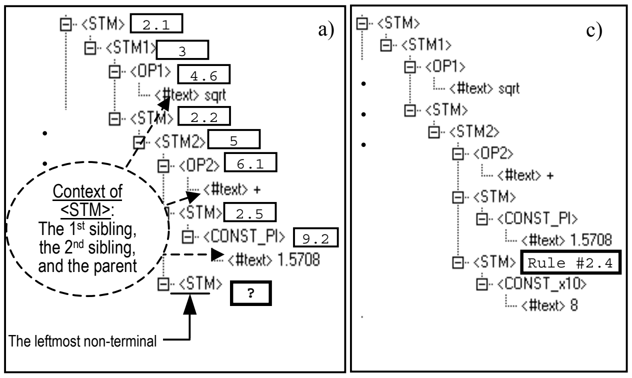
$$context_i\ V \rightarrow context_i\ w_1\ (p^i_1)$$
$$context_i\ V \rightarrow context_i\ w_2\ (p^i_2)$$
$$\dots$$
$$context_i\ V \rightarrow context_i\ w_N\ (p^i_N)$$

where $p^i_1, p^i_2, \dots p^i_N$ ($\sum p^i_n = 1,\ n=1,2..N.$) are the probabilities of applying each alternative rule with the left-hand side non-terminal $V$ for the given $context_i$.

Applying the `IF-THEN` stimulus-response paradigm, which usually expresses the reactive behavioral strategies of intelligent entities in AI (e.g., software agents, robots, etc.) to such biased mutation operations in GP, and viewing the evolved genotype not only as an evolving, but also as a learning intelligent entity, the above considered sample rule of *G** could be modeled by the following behavioral `IF-THEN` statement:

```
IF (Context_of_[V] is [context_i ]))
THEN        Apply_Rules_With_Probabilities(p^i_1,p^i_2,
…p^i_N)
```

The LMS strategy in our approach comprises the dynamic set of `IF-THEN` rules created and tuned by parsing the syntax of the best performing Snakebots of the current generation. A sample of biased application of production rules of *G** according to the learned PD and the corresponding `IF-THEN` rule of LMS for the considered leftmost non-terminal and the context are shown in Figure 2.



```
IF-THEN Rule of the LMS                                              b)
-------------------------------------------
IF (Context_of_[STM] is [1.57, +, sqrt])
THEN Apply_Rules_With_Probabilities [0.34, 0, 0.07, 0.43, 0.16]

#   Rule in LPCSG     Probability   For the Context
--- ----------------- ------------- ----------------
2.1 STM ——> STM1      0.34          [1.57, +, sqrt]
2.2 STM ——> STM2      0             [1.57, +, sqrt]
2.3 STM ——> VAR       0.07          [1.57, +, sqrt]
2.4 STM ——> CONST_x10 0.43          [1.57, +, sqrt]
2.5 STM ——> CONST_PI  0.16          [1.57, +, sqrt]
```

**Figure 2. Sample of biased application of production rules of *G**: the current leftmost non-terminal, as shown in (a) is `STM`, which requires applying one of the production rules 2.1-2.5 (refer to Figure 2). For the considered context (a), the LMS of applying rules 2.1-2.5 (b) suggests a highest probability for applying the production rule 2.4, yielding the genetic program as shown in (c).**

## 3.2 Algorithm of GP Incorporating LMS

The principal steps of algorithm of GP incorporating LMS via LPCSG are shown in Figure 3. As figure illustrates, additional `Steps 6` and `9` are introduced in the canonical algorithm of GP. The LMS is updated on `Step 6`, and the new offspring, created applying the proposed biased mutation via LPCSG on `Step 9` are inserted into already reproduced via canonical crossover (`Step 7`) and mutation (`Step 8`), growing new population of Snakebots. The parameter $K_{LMS}$ defines the ratio of the number of offspring $\#N_{LMS}$ created via biased mutation using LMS and the number of offspring $\#N_{CO}$ created via canonical crossover. $K_{LMS}$ is dynamically tuned on `Step 6` based on the stagnation counter $C_S$, which maintains the number of most recent generations without improvement of the fitness value. In our implementation, $K_{LMS}$ is kept within the range `[0, 5]`. It is defined according to the following rule:

$$K_{LMS} = 5 - \texttt{smaller\_of}(5, C_S)$$

The lower values of $K_{LMS}$ in stagnated population (i.e., for $C_S>0$) favor the reproduction via canonical random genetic operations over the reproduction using biased mutation via LMS. As we empirically investigated, the low values of $K_{LMS}$ facilitate avoiding premature convergence by increasing the diversity of population and consequently, accelerating the escape from the (most likely) local optimal solutions, discovered by the steering bias of the current LMS. Conversely, replacing the usually random genetic operations of canonical GP with the proposed biased mutation when $K_{LMS}$ is close to its maximum value (i.e., for $C_S=0$) can be viewed as a mechanism for growing and preserving the proven to be beneficial building blocks in evolved solutions rather than destroying them by usually random crossover and mutation.

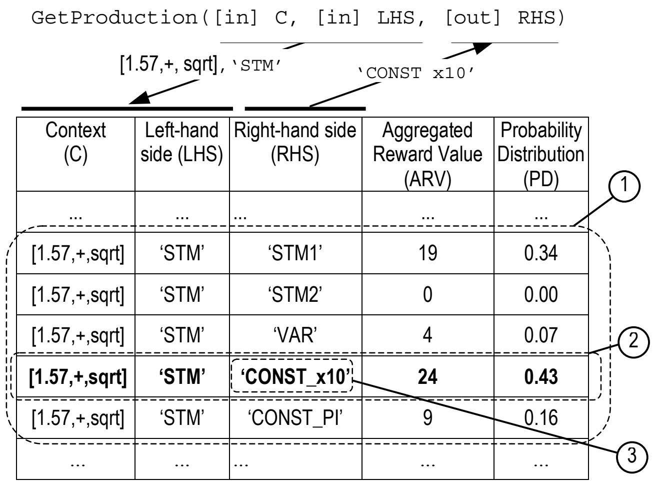
```
Step 0:Creating Initial Population and Clearing PDD;
Step 1:While (true) do begin
Step 2: Evaluating Population;
Step 3: Ranking Population;
Step 4: if TerminationCriteria then Go to Step 10;
Step 5: Selecting the Mating Pool;
Step 6: Updating LMS and K_LMS;
Step 7: Creating #N_CO offspring via canonical crossover;
Step 8: Mutating current population via canonical
        mutation;
Step 9: Creating #N_LMS offspring via mutation
        of mating pool using LMS;
Step10:end;
```

**Figure 3. Algorithm of GP incorporating LMS. `Steps 6` and `9` are specific for the proposed approach. `Steps 0`, `2-5`, `7` and `8` are common principal steps of canonical GP.**

Updating (Figure 3, `Step 6`) and applying LMS during the biased mutation (Figure 3, `Step 9`) implies maintaining a table, which represents the set of learned `IF-THEN` rules. Each entry in the table stores the context, the left-hand side non-terminal, the list of right-hand side symbols, the aggregated reward values and the calculated probability of applying the given production rule for the given context. A new entry is added or the aggregated reward value of existing entry is updated by extracting the syntactic features of the best performing genetic programs (the mating pool) of the current generation. The outdated entries, added 4 or more generations before are deleted, keeping the total number of entries in the table between 300 and 500. The string of characters, comprising the right-hand side `RHS` of given production rule that should be applied to the current leftmost non-terminal (i.e. the corresponding left-hand symbol in production rule, `LHS`) for the given context `C` is obtained by the function `GetProduction([in] C, [in] LHS, [out] RHS)` which operates on LMS table as shown in Figure 4.

## 4. EMPIRICAL RESULTS

This section discusses empirically obtained results verifying the effects of incorporating LMS on the efficiency of GP applied for the following two tasks: (i) *evolution* of the fastest possible locomotion gaits of Snakebot for various fitness conditions and (ii) *adaptation* of these locomotion gaits to challenging environment and degraded mechanical abilities of Snakebot. These tasks, considered as relevant for successful accomplishment of anticipated exploration, reconnaissance, medicine or inspection missions, feature different fitness landscapes. Therefore, the experiments discussed in this section are intended to verify the versatility and the scope of applicability of the proposed approach.



| Context (C) | Left-hand side (LHS) | Right-hand side (RHS) | Aggregated Reward Value (ARV) | Probability Distribution (PD) |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| [1.57,+,sqrt] | 'STM' | 'STM1' | 19 | 0.34 |
| [1.57,+,sqrt] | 'STM' | 'STM2' | 0 | 0.00 |
| [1.57,+,sqrt] | 'STM' | 'VAR' | 4 | 0.07 |
| **[1.57,+,sqrt]** | **'STM'** | **'CONST_x10'** | **24** | **0.43** |
| [1.57,+,sqrt] | 'STM' | 'CONST_PI' | 9 | 0.16 |
| ... | ... | ... | ... | ... |

**Figure 4. Obtaining the most preferable right-hand side (`RHS`) of production rule of LPCSG that should be applied to the left-most non-terminal (i.e. left-hand symbol, `LHS`), and the context (`C`) according to a sample `IF-THEN` rule of the current LMS: (1) Selecting the set of entries associated with the entries featuring the given `LHS` and `C`, (2) Choosing an entry from the obtained result set with a probability, proportional to the learned PD, and (3) returning the `RHS` of the chosen production rule. The sample `IF-THEN` rule of the LMS, shown in this Figure is the same as in Figure 2.**

In all of the cases considered, the fitness of Snakebot reflects the low-level objective (i.e. *what* is required to be achieved) of Snakebot in these missions, namely, to be able to move fast regardless of environmental challenges or degraded abilities. The experiments discussed illustrate the ability of the evolving Snakebot to learn *how* (e.g. by discovering beneficial locomotion traits) to accomplish the required objective without being explicitly taught about the means to do so. Such *know-how* acquired by Snakebot automatically and autonomously can be viewed as a demonstration of emergent intelligence [1], in that the task-specific knowledge of *how* to accomplish the task emerges in the Snakebot from the interaction of the problem solver and the fitness function.

## 4.1 Evolution of Fastest Locomotion Gaits

Figure 5 shows the results of evolution of locomotion gaits for cases where fitness is measured as velocity in any direction. Despite the fact that fitness is unconstrained and measured as velocity in *any* direction, *sidewinding* locomotion (defined as locomotion predominantly perpendicular to the long axis of Snakebot) emerged in all 10 independent runs of GP, suggesting that it provides superior speed characteristics for considered morphology of Snakebot. As Figure 5c illustrates, incorporating LMS in GP is associated with computational effort (required to achieve probability of success 0.9) of about 20 generations, which is about 1.6 times faster than canonical GP with CFG. Sample snapshots of evolved best-of-run sidewinding locomotion gaits are shown in Figures 5d-5g.

In order to verify the superiority of velocity characteristics of sidewinding we compared the fitness convergence characteristics of evolution in unconstrained environment for the following two cases: (i) unconstrained fitness measured as velocity in any direction (as discussed above and illustrated in Figure 5), and (ii) fitness, measured as velocity in forward direction only. The results of evolution of forward (rectilinear) locomotion, shown in Figure 6 indicate that non-sidewinding motion, compared to sidewinding, features much inferior velocity characteristics. The results also demonstrate that GP with LMS in average converges almost 4 times faster and to higher values than canonical GP. Snapshots taken during the motion of a sample evolved best-of-run sidewinding Snakebot are shown in Figures 6c and 6d.

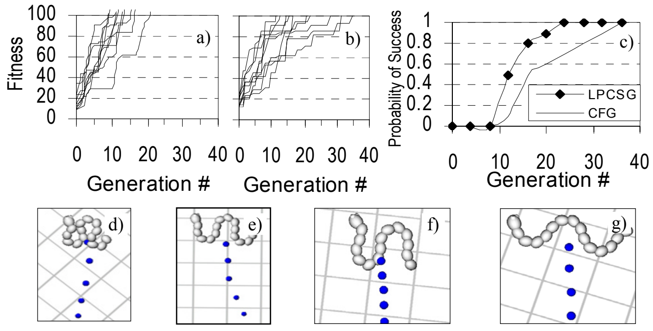


**Figure 5. Evolution of locomotion gaits for cases where fitness is measured as velocity in any direction: fitness convergence characteristics of 10 independent runs of GP with LMS (a), canonical GP (b), probability of success (c), and snapshots of sample evolved via GP with LMS best-of-run sidewinding Snakebots (d), (e), (f) and (g). The dark trailing circles in (d), (e), (f) and (g) depict the trajectory of the center of the mass of Snakebot.**

The results of evolution of rectilinear locomotion of simulated Snakebot confined in narrow "tunnel" are shown in Figure 7. As the fitness convergence characteristics of 10 independent runs (Figure 7a and Figure 7b) illustrate, GP with LMS is almost twice faster than canonical GP. Compared to forward locomotion in unconstrained environment (Figure 6), the velocity in this experiment is superior, and even comparable to the velocity of sidewinding (Figure 5). This, seemingly anomalous phenomenon demonstrates a case of emergent intelligence – i.e. an ability of evolution to discover a way to utilize the walls of "tunnel" as (i) a source of extra grip and as (ii) an additional mechanical support for fast yet unbalanced locomotion gaits (e.g., vertical undulation) in an eventual unconstrained environment.

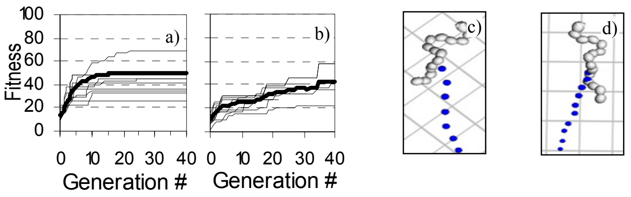


**Figure 6. Evolution of locomotion gaits for cases where fitness is measured as velocity in forward direction only. Fitness convergence characteristics of 10 independent runs of GP with LMS (a), canonical GP (b), and snapshots of sample evolved via GP with LMS best-of-run forward locomotion (c and d).**

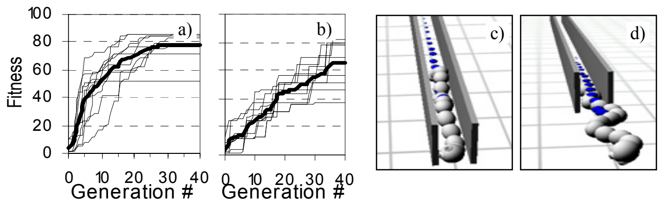


**Figure 7. Evolution of locomotion gaits of Snakebot confined in narrow "tunnel": fitness convergence characteristics of 10 independent runs of GP with LMS (a), canonical GP (b), and snapshots of sample evolved best-of-run gaits at the intermediate (c) and final stages of the trial (d)**

## 4.2 Adaptation of Sidewinding to Challenging Environment. Generality of Adapted Gaits

Adaptation in Nature is viewed as an ability of species to discover the best phenotypic (i.e. pertaining to biochemistry, morphology, physiology, and behavior) traits for survival in continuously changing fitness landscape. The adaptive phenotypic traits are result of beneficial genetic changes occurred during the course of evolution (phylogenesis) and/or phenotypic plasticity (ontogenesis – learning, polymorphism, polyphenism, immune response, adaptive metabolism, etc.) occurring during the lifetime of the individuals. In our approach we employ GP with LMS for adaptation of Snakebot to changes in the fitness landscape caused by (i) challenging environment and (ii) partial damage to 1, 2, 4 and 8 (out of 15) morphological segments. In all of the cases of adaptation, GP is initialized with a population comprising 20 best-of-run genetic programs, obtained from 10 independent runs of evolution of Snakebot in unconstrained environment, plus additional 180 randomly created individuals.

The challenging environment is modeled by the introduction of immobile obstacles comprising 40 small, randomly scattered boxes, a wall with height equal to the 0.5 diameters of the cross-section of Snakebot, and a flight of 3 stairs, each with height equal to the 0.33 diameters of the cross-section of Snakebot. The results of adaptation of Snakebot, shown in Figure 8 demonstrate that the computational effort (required to reach fitness values of 100 with probability of success 0.9) of GP with LMS is about 20 generations. Conversely, only half of all runs of canonical GP achieve the targeted fitness value, implying that the corresponding probability of success converges to the value of 0.5. Snapshots illustrating the performance of Snakebot initially evolved in unconstrained environment, before and after the adaptation (via GP with LMS) to challenging environment are shown in Figure 9. The additional elevation of the body, required to faster negotiate the obstacles represents the emergent know-how in the adapting Snakebot. As Figure 10 illustrates, the trajectory of the central segment around the center of the mass of sample adapted Snakebot (Figure 10b) is twice higher than before the adaptation (Figure 10a).

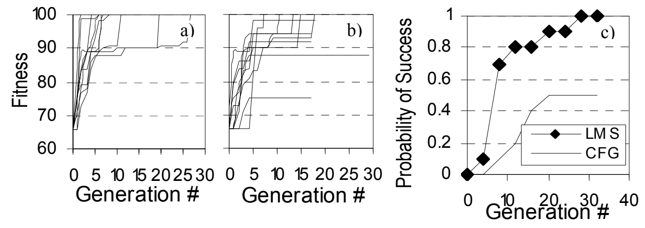


**Figure 8. Adaptation of sidewinding locomotion to challenging environment: fitness convergence characteristics of 10 independent runs of GP with LMS (a), canonical GP (b), and probability of success (c).**

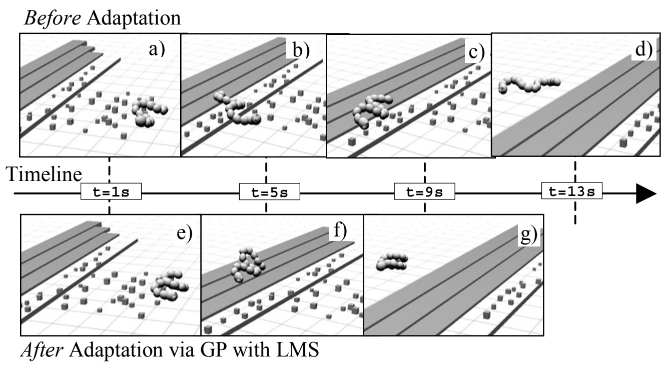


**Figure 9. Snapshots illustrating the sidewinding Snakebot, initially evolved in unconstrained environment, before the adaptation – initial (a), intermediate (b and c) and final stages of the trial (d), and after the adaptation to challenging environment via GP with LMS - initial (e), intermediate (f) and final stages of the trial (g).**

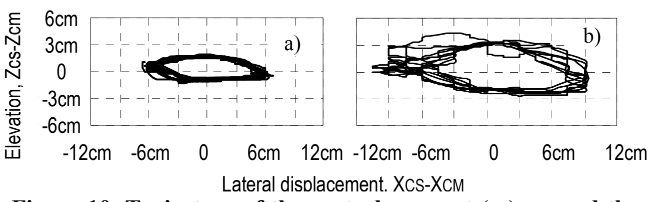


**Figure 10. Trajectory of the central segment (cs) around the center of mass (cm) of Snakebot for sample best-of-run sidewinding locomotion before (a) and after the adaptation (b) to challenging environment.**

The generality of the evolved via GP with LMS robust sidewinding gaits is demonstrated by the ease with which Snakebot, evolved in known challenging terrain overcomes various types of unanticipated obstacles such as a pile of boxes, a burial under boxes, and small walls, as illustrated in Figures 11, 12, and 13.

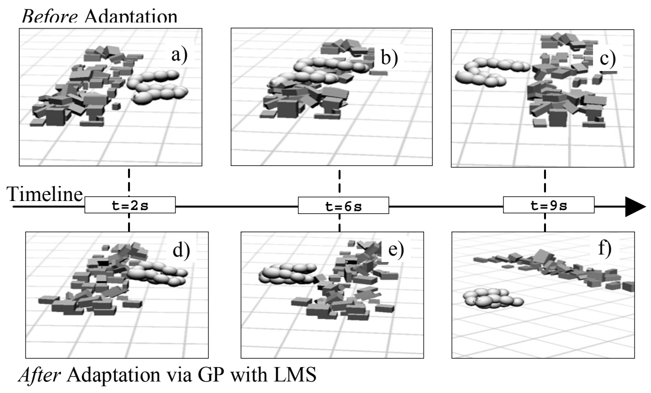


**Figure 11. Snapshots illustrating the generality of sidewinding Snakebot adapted to the known challenging environment as depicted in Figure 9. Before the adaptation to the known challenging environment the Snakebot overcomes an unanticipated pile of boxes slower (a, b and c) than after the adaptation (d, e, and f) via GP with LMS.**

## 4.3 Adaptation to Partial Damage

The adaptation of sidewinding Snakebot to partial damage to 1, 2, 4 and 8 (out of 15) segments by gradually improving its velocity is shown in Figure 14. Demonstrated results are averaged over 10 independent runs for each case of partial damage to 1, 2, 4 and 8 segments. The damaged segments are evenly distributed along the body of Snakebot. Damage inflicted to a particular segment implies a complete loss of functionality of both horizontal and vertical actuators of the corresponding joint.

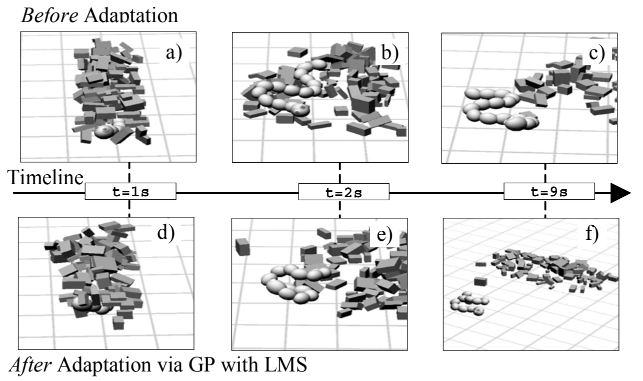


**Figure 12. Snapshots illustrating the generality of sidewinding Snakebot adapted to the known challenging environment as depicted in Figure 9. Before the adaptation to the known challenging environment the Snakebot emerges from an unanticipated burial under pile of boxes slower (a, b and c) than after the adaptation (d, e, and f) via GP with LMS.**

As Figure 14 depicts, Snakebot recovers completely from the damage to single segment attaining its previous velocity in 25 generations with canonical GP, and only in 7 generations with GP with LMS, resulting in a mean real-time of adaptation of a few hours of runtime on PC featuring Intel® 3GHz Pentium® 4 microprocessor and 2GB RAM under Microsoft Windows NT OS. Snakebots recovers to average of 94% (Canonical GP) and 100% (GP with LMS) of its previous velocity in the case where 2 segments are damaged. With 4 and 8 damaged segments the degree of recovery is 77% (Canonical GP) and 92% (GP with LMS), and 68% (Canonical GP) and 72% (GP with LMS) respectively. In all of the cases considered incorporating LMS contributes to faster adaptation of Snakebot, and in all cases the Snakebot recovers to higher values of velocity of locomotion. The snapshots of sidewinding Snakebot immediately after damage, and after having recovered from the damage of 1, 2 , 4 and 8 segments are shown in Figure 15.

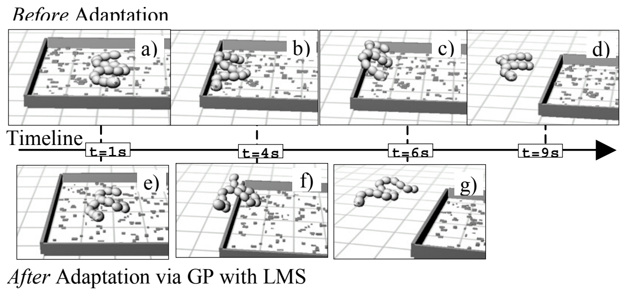


**Figure 13. Snapshots illustrating the generality of sidewinding Snakebot adapted to the known challenging environment as depicted in Figure 9. Before the adaptation to the known challenging environment the Snakebot clears an unanticipated walls forming pen slower (a, b, c and d) than after the adaptation (e, f, and g). The walls are twice higher than in the know challenging terrain, and their height is equal to the diameter of the cross-section of Snakebot.**

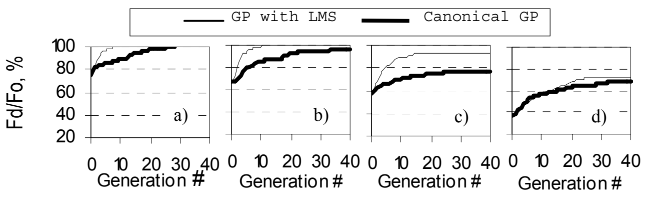


**Figure 14. Adaptation of Snakebot to damage of 1 (a), 2 (b), 4 (c) and 8 (d) segments. Fd is the best fitness in evolving population of damaged snakebots, and Fh is the best fitness of 20 best-of-run healthy sidewinding Snakebots.**

## 5. CONCLUSION

In this work we propose an approach of incorporating LMS implemented via LPCSG in GP and verified it on the efficiency of evolution and adaptation of locomotion gaits of simulated Snakebot. We introduced a biased mutation in which the probabilities of applying each of particular production rules with multiple right-hand side alternatives in the LPCSG depend on the context, and these probabilities are “learned” from the aggregated reward values obtained from the evolved best-of-generation Snakebots. Empirically obtained results verify that employing LMS contributes to the improvement of computational effort of both (i) the evolution of the fastest possible locomotion gaits for various fitness conditions and (ii) adaptation of these locomotion gaits to challenging environment and degraded mechanical abilities of Snakebot.

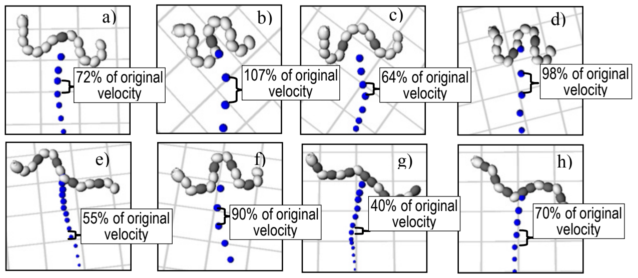


**Figure 15. Snapshots of the sidewinding Snakebot, immediately after damage to 1 (a), 2 (c), 4 (e), and 8 (g) segments, and after having recovered from the damage (b, d, f, and h) by adaptation via GP with LMS.**

The recent discoveries in molecular biology and genetics suggest that mutations do not happen randomly in the Nature. Instead, some fragments of DNA tend to repel the mutations away, while other fragments seem to attract it [3]. It is assumed that the former fragments are related to the very basics of life, (and therefore, any mutation within them can be potentially fatal to the species), while the latter fragments are relevant for the adaptability (and consequently, for the survival) of the organisms. The proposed approach of LMS incorporated in GP implies *focusing* the mutation operation toward the proven beneficial mutation points (i.e. the points with *even* distribution of the learned probabilities for the right-hand side alternatives of rules in LPCSG). In addition, the approach of LMS facilitates keeping the mutation *away* from the genotypic points which do not have a proven beneficial effect on the performance of genetic programs by always choosing the same right-hand side alternative (in case of highly *uneven* distribution of learned probabilities) and thus following with fidelity the syntactical trends which prevail in the best performing individuals. Within the context considered, the proposed incorporation of LMS in GP can be viewed as a biologically plausible attempt (i) to mimic the Natural mechanisms of genomic control over the mutation operations and (ii) to investigate the computational implication of these mechanisms on the efficiency of the simulated evolution and adaptation of engineering artifacts.

## REFERENCES

[1] Angeline, P. J. *Genetic Programming and Emergent Intelligence*. In Kinnear, K.E. Jr., editor, Advances in Genetic Programming, MIT Press, 1994, 75-98

[2] Bosman, P. and de Jong, E., Learning Probabilistic Tree Grammars for Genetic Programming, In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature PPSN-04*, 2004, 192-201.

[3] Caporale, L. H. *Darwin In the Genome: Molecular Strategies in Biological Evolution*, 2002, McGraw-Hill/Contemporary Books

[4] Dowling, K. *Limbless Locomotion: Learning to Crawl with a Snake Robot*, doctoral dissertation, tech. report CMU-RI-TR-97-48, Robotics Institute, Carnegie Mellon University, 1997.

[5] Goldberg D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addision-Wesley, 1989

[6] Hirose, S. *Biologically Inspired Robots: Snake-like Locomotors and Manipulators*, Oxford University Press, 1993.

[7] Kimura, H., Yamashita, T., and Kobayashi, S. Reinforcement Learning of Walking Behavior for a Four-Legged Robot, In *Proceedings of 40th IEEE Conference on Decision and Control*, 2 001, 411-416

[8] Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA, MIT Press, 1992.

[9] Mahdavi, S., Bentley, P.J. Evolving Motion of Robots with Muscles. In *Proc. of EvoROB2003, the 2nd European Workshop on Evolutionary Robotic (EuroGP 2003),* 2003 655-664.

[10] O'Neill, M. and Ryan, C. Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language, Series: Genetic Programming, Vol. 4, Springer, 2003.

[11] Pelikan M., Goldberg D. E., and Cantú-Paz, E. BOA: The Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO-99), I, 1999, 525-532.

[12] Salustowicz, R. and Schmidhuber, J Probabilistic Incremental Program Evolution*. Evolutionary Computation*, Vol.5 No.2, 1997, 123-141.

[13] Shan Y., McKay, R.I, and Baxter R. Grammar Model-based Program Evolution, In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, 20-23 June, 2004, Portland, Oregon, 478-485.

[14] Smith, R. *Open Dynamics Engine*, 2001-2003, URL: http://q12.org/od

[15] Stoy, K., Shen, W.-M. and Will, P.M. A Simple Approach to the Control of Locomotion in Self-reconfigurable Robots, *Robotics and Autonomous Systems*, Vol.44 , No.3, 2003, pp.191-200.

[16] Takamura, S., Hornby, G. S., Yamamoto, T. , Yokono, J. and Fujita, M. Evolution of Dynamic Gaits for a Robot, In *Proceedings of the IEEE International Conference on Consumer Electronics*, 2000, 192-193.

[17] Zhang, Y., Yim, M. H., Eldershaw, C., Duff, D. G. and Roufas, K. D. Phase automata: a programming model of locomotion gaits for scalable chain-type modular robots, In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, October 27 - 31; Las Vegas, NV, 2003.

[18] W3C, *Extensible Markup Language (XML) 1.0*, Second Edition, W3C Recommendation (2000), URL: http://www.w3.org/TR/REC-xml/