Implications of Incorporating Learning Probabilistic Context-sensitive Grammar in Genetic Programming on Evolvability of Adaptive Locomotion Gaits of Snakebot

Ivan Tanev

Department of Information Systems Design,
Faculty of Engineering, 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

i_tanev@atr.jp

Abstract. In this work we propose an approach of incorporating learning context-sensitive grammar in strongly typed genetic programming (GP) employed for evolution and adaptation of locomotion gaits of simulated snake-like robot (Snakebot). In our approach the probabilistic context-sensitive grammar is derived from the originally defined context-free grammar (which usually expresses the syntax of genetic programs in strongly typed GP), using aggregated reward values obtained from the evolved best-of-run healthy, undamaged Snakebots. The probabilities of applying each of particular production rules with multiple right-hand side alternatives in derived probabilistic context-sensitive grammar depend on the context, and these probabilities are “learned” from the aggregated reward values. Empirically obtained results indicate that employing probabilistic context-sensitive grammar contributes to the improving the ability of Snakebot to adapt to partial damage by gradually improving its velocity characteristics. Snakebot recovers completely from single damage and recovers a major extent of its original velocity when more significant damage is inflicted. In all considered cases of inflicted partial damage of 1, 2, 4, and 8 out of 15 morphological segments, the incorporation of learning context sensitive grammar in GP improves the evolvability of adaptive locomotion gaits in that the recovery of partially damaged Snakebot is (i) faster and to (ii) higher values of velocity of locomotion.
Keywords: adaptation, locomotion, snake-like robot, strongly typed genetic programming, probabilistic 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 [1,3,12]. 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) smaller payload, (ii) more difficult thermal control, (iii) more difficult control of locomotion gaits and (iv) inferior speed characteristics. Considering the first two drawbacks as beyond the scope of our work, and focusing on the drawbacks of control and speed, we intend to address the following challenge: how to develop control sequences of Snakebot's actuators, which allow for achieving the fastest possible speed of locomotion.

Although for many tasks, handcrafting the robot locomotion control code can be seen as a natural approach, it might not be feasible for developing the control code of Snakebot due to its morphological complexity. While the overall locomotion gait of Snakebot might emerge from relatively simply defined motion patterns of morphological segments of Snakebot, neither the degree of optimality of the developed code nor the way to incrementally improve the code is evident to the human designer [7]. Thus, an automated mechanism for solution evaluation and corresponding rules for incremental optimization of the intermediate solution(s) are needed [6]. The proposed approach of employing genetic programming (GP) implies that the code, which governs the locomotion of Snakebot is automatically designed by a computer system via simulated evolution through selection and survival of the fittest in a way similar to the evolution of species in the nature. The use of an automated process to design the control code opens the possibility of creating a solution that would be better than one designed by a human. Additional motivation for applying such an automated process to design the control code of Snakebot is that the anticipated application areas of Snakebot (exploration, reconnaissance, medicine, inspection etc.) feature challenging environments in which a partial damage to the Snakebot might have been inflicted. In order to successfully accomplish its mission is such environments the Snakebot should be able quickly, automatically, and autonomously adapt to these damages.

The *objectives* of our work are (i) to explore the feasibility of applying GP for automatic design of the fastest possible locomotion of realistically simulated Snakebot and (ii) to investigate the adaptation of such locomotion to degraded abilities (due to partial damage) of Snakebot. We are particularly interested in the implications of incorporating a learning context-sensitive grammar in strongly typed GP (employed for automatic design of the fastest possible locomotion of Snakebot) on the evolvability of adaptive locomotion gaits of partially damaged Snakebot.

The discussed approach is related to employing learning Bayesian optimization algorithms and reinforcement learning in evolutionary computations [2, 4, 8, 9, 10]. In neither of these methods however the incorporation of learning context-sensitive grammar in strongly typed GP has been explored and the curiosity about 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 the GP proposed for evolution of locomotion of simulated Snakebot. The same section presents empirical results of evolving fast locomotion gaits of Snakebots. Section 3 introduces the proposed approach of incorporating learning context-sensitive grammar in the strongly typed GP and its implications on the evolvability of adaptive locomotion gaits of partially damaged Snakebot. Finally, 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 (Figure 1). 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.

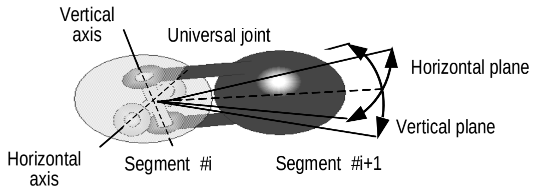


Fig.1. Morphological segments of Snakebot linked via universal joint. Horizontal and vertical actuators attached to the joint perform rotation of the segment `#i+1` in vertical and horizontal planes respectively.

2.2 Algorithmic Paradigm

GP. GP [5] 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. The major attributes of GP - function set, terminal set, fitness evaluation, genetic representation, and genetic operations are elaborated in the remaining of this Section.

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. Since locomotion gaits are periodical, we include the trigonometric functions `sin` and `cos` in the GP function set in addition to the basic algebraic functions. The choice of these trigonometric functions reflects our intention to verify the hypothesis (first expressed by Petr Miturich in 1920's) that undulative motion mechanisms could yield efficient gaits of snake-like artifacts operating in air, land, or water. Terminal symbols include the variables `time`, `index` of morphological segment of Snakebot, and two constants: `Pi`, and `random` constant within the range [0, 2]. The main parameters of the GP are summarised in Table 1.

Table 1. Main parameters of GP

Category	Value
Function set	{sin, cos, +, -, *, /}
Terminal set	{time, segment_ID, Pi, random constant, ADF}
Population size	200 individuals
Selection	Binary tournament, ratio 0.1
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>30) *or* (no improvement of fitness for 16 generations)

The rationale of employing automatically defined function (ADF) is based on empirical observation that the evolvability of straightforward, independent encoding of desired turning angles of both horizontal and vertical actuators is poor, although it allows GP to adequately explore the search space and ultimately, to discover the areas which correspond to fast locomotion gaits in solution space. We discovered that (i) the motion patterns of horizontal and vertical actuators of each segment in fast locomotion gaits are highly correlated (e.g. by frequency, direction, etc.) and that (ii) discovering and preserving such correlation by GP is associated with enormous computational effort. ADF, as a way of introducing modularity and reuse of code in GP [5] is employed in our approach to allow GP to explicitly evolve the correlation between motion patterns of horizontal and vertical actuators as shared fragments in algebraic expressions of desired turning angles of actuators. Moreover, the best result was obtained by (i) allowing the use of ADF as a terminal symbol in algebraic expression of desired turning angle of vertical actuator only, and (ii) by evaluating the value of ADF by equalizing it to the value of currently evaluated algebraic expression of desired turning angle of horizontal actuator.

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. The real values of the raw fitness, which are usually within the range (0, 2) are multiplied by a normalizing coefficient in order to deal with integer fitness values within the range (0, 200). A normalized fitness of 100 (one of the termination criteria shown in Table 1) is equivalent to a velocity which displaced Snakebot a distance equal to twice its length.

Genetic Operations. Binary tournament selection is employed – a robust, commonly used selection mechanism, which has proved to be efficient and simple to code. Crossover operation 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 randomly chosen rule from the set of applicable rewriting rules as defined in the context-free grammar of strongly typed GP.

ODE. We have chosen Open Dynamics Engine (ODE) [11] to provide a realistic simulation of physics in applying forces to phenotypic segments of Snakebot, for simulation of Snakebot locomotion. 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.

2.3 Automatic Design of Fastest Locomotion Gaits of Healthy Snakebot: Empirical Results

Figure 2 shows the fitness convergence characteristics of 10 independent runs of GP (Figure 2a) and sample snapshots of evolved best-of-run locomotion gaits (Figure 2b and Figure 2c) of healthy Snakebot when fitness is measured in *any* direction in an unconstrained environment. 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 Snakebot morphology.

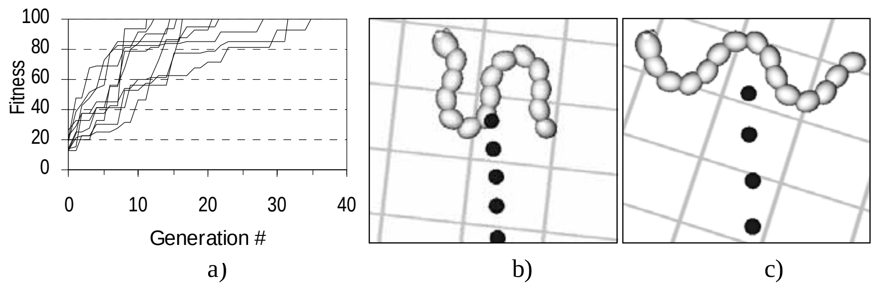


Fig. 2. Fitness convergence characteristics of 10 independent runs of GP for cases where fitness is measured as velocity in any direction (a) and snapshots of sample evolved best-of-run sidewinding locomotion gaits of simulated Snakebot (b, c), viewed from above. The dark trailing circles depict the trajectory of the center of the mass of Snakebot. Timestamp interval between each of these circles is fixed and it is the same (10 time steps) for both snapshots.

The dynamics of evolved turning angles of actuators in sidewinding locomotion result in characteristic circular motion pattern of segments around the center of the mass as shown in Figure 3a. The circular motion pattern of segments and the characteristic track on the ground as a series of diagonal lines (Figure 3b) suggest that during sidewinding the shape of Snakebot takes the form of a rolling helix. Figure 3 demonstrates that the simulated evolution of locomotion via GP is able to invent the improvised "wheel" of the sidewinding Snakebot to achieve fast locomotion.

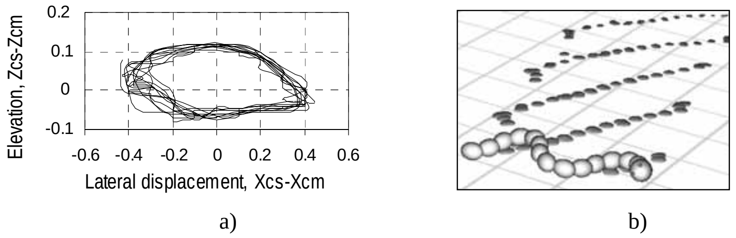


Fig. 3. Trajectory of the central segment (cs) around the center of mass (cm) of Snakebot for a sample evolved best-of-run sidewinding locomotion (a) and traces of ground contacts (b).

3 Incorporating Learning Context-sensitive Grammar in GP

3.1 Context-free Grammar of Strongly-typed GP Employed for Automatic Design of Fastest Locomotion Gaits of Healthy Snakebot

Context free grammar *G* is defined as (*N*, Σ, *P*, *S*) where *N* is a finite set of nonterminal symbols, *Σ* 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

$V \rightarrow w$

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. For the considered case of context free grammar of strongly typed GP, employed for automatic design of locomotion gaits of healthy Snakebot (as elaborated before in Section 2), the set of non-terminal symbols is defined as follow:

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

where `STM` is a Statement, `STM1` – Unary statement, `STM2` – Binary (dyadic) statement, `VAR` – variable, `OP1` – unary operation, `OP2` – 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 terminals is:

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

The start (nonterminal) symbol is `GP`, and the set of production rules expressed in Backus-Naur form (BNF) is as shown in Figure 4.

```
(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
```

Fig. 4. BNF of production rules of the context free grammar G of strongly typed GP, employed for automatic design of locomotion gaits of healthy Snakebot. The following abbreviations are used: `STM` – statement, `STM1` – unary statement, `STM2` – binary (dyadic) statement, `VAR` – variable, `OP1` – unary operation, `OP2` – binary (dyadic) operation

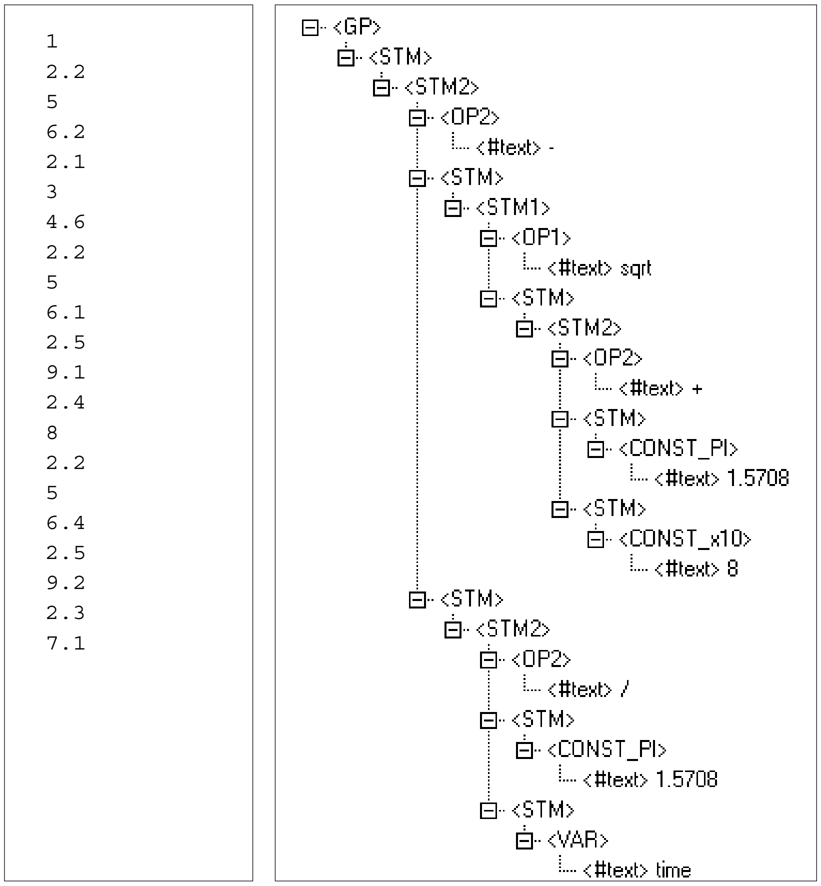
The algebraic expression of horizontal (and vertical) desired angle of actuators of Snakebot evolved through GP can be obtained from described context-free grammar *G* and starting symbol `GP` applying a corresponding production rule for the currently leftmost non-terminal symbol in the derivative expression. The production rules with multiple alternative right-hand sides (such as rules 2, 4, 6, 7 and 9) are chosen randomly. GP uses the defined production rules of the grammar during the creation of initial population and during mutation of genetic programs. Because crossover is implemented in strongly typed way, the syntax of resulting offspring complies with the allowed syntax of genetic programs as defined by the context-free grammar *G*. For example, applying the sample sequence of production rules, shown in Figure 5a on the stage of creation of initial population of GP yields a genetic program (e.g. for defining the desired horizontal angle of actuators of Snakebot) as shown in Figure 5b.

3.2 Learning Probabilistic Context-sensitive Grammar of Strongly-typed GP

In our approach, the probabilistic context-sensitive grammar *G** is introduced as a set of the same attributes (*N**, *Σ**, *P**, *S**) as for the context-free grammar *G* defined in 3.1 before. While the attributes *N**, *Σ**, and *S** are identical to *N*, *Σ*, and *S* of *G*, the set of production rules *P** of *G** are derived from *P* of *G* as follows: (i) The production rules of *P* which does not have right-hand side alternatives are defined in the same way in *P** as in *P*, and (ii) production rules in *P*, which do have multiple right-hand side alternatives $V \rightarrow w_1 / w_2 / \ldots / w_N$ are re-defined for each instance of the context (i.e. $context_i$) as follows:

$context_i\ V \rightarrow context_i\ w_1\ (p^i{}_1)$
$context_i\ V \rightarrow context_i\ w_2\ (p^i{}_2)$
...
$context_i\ V \rightarrow context_i\ w_N\ (p^i{}_N)$

where p^i_1, p^i_2, $\ldots p^i_N$ are probabilities (frequencies) of applying each alternative rule in the given $context_i$. For each set of production rules featuring multiple right-hand side alternatives, and for given $context_i$, $\sum p^i_n = 1$, $n=1,2..N$.



```
1
2.2
5
6.2
2.1
3
4.6
2.2
5
6.1
2.5
9.1
2.4
8
2.2
5
6.4
2.5
9.2
2.3
7.1
```

```
<GP>
  <STM>
    <STM2>
      <OP2>
        <#text> -
      <STM>
        <STM1>
          <OP1>
            <#text> sqrt
          <STM>
            <STM2>
              <OP2>
                <#text> +
              <STM>
                <CONST_PI>
                  <#text> 1.5708
              <STM>
                <CONST_x10>
                  <#text> 8
      <STM>
        <STM2>
          <OP2>
            <#text> /
          <STM>
            <CONST_PI>
              <#text> 1.5708
          <STM>
            <VAR>
              <#text> time
```

a) b)

Fig. 5. Sample sequence of applied production rules (a) and resulting genetic program (b)

The proposed approach is based on the idea of introducing bias in applying the most preferable rule from the set of rules with multiple, alternative right-hand sides. We assume that (i) such preferences of applying certain rule can be defined as probabilities (preferred frequencies) of applying the rule and (ii) the preferences of applying certain production rules would depend on the context, i.e. on which rules have been applied before. In the proposed approach, the initial distribution of probabilities $p_1 \ldots p_N$ is even. The distribution of probabilities is learned (adjusted) from the best performing evolved healthy Snakebots and then used in adaptation of Snakebot via

GP to partial damage.

Sample sequence of applying the production rules (as shown in Figure 5a) immediately after applying the production rule 9.1, and the corresponding genetic program in prefix notation and as a parsing tree are shown in Figure 6a, 6b and 6c respectively. The current leftmost non-terminal, as shown in Figure 6b and 6c is `STM`, which requires applying one of production rules 2.1-2.5 (Figure 4). Assuming that for the considered context, the "learned" preferences of applying rules 2.1-2.5 indicates highest probability, and consequently, preferential bias towards the rule 2.4: `STM` $\longrightarrow$ `CONST_x10`, then this production rule will be most likely applied yielding the genetic program shown Figure 5b.

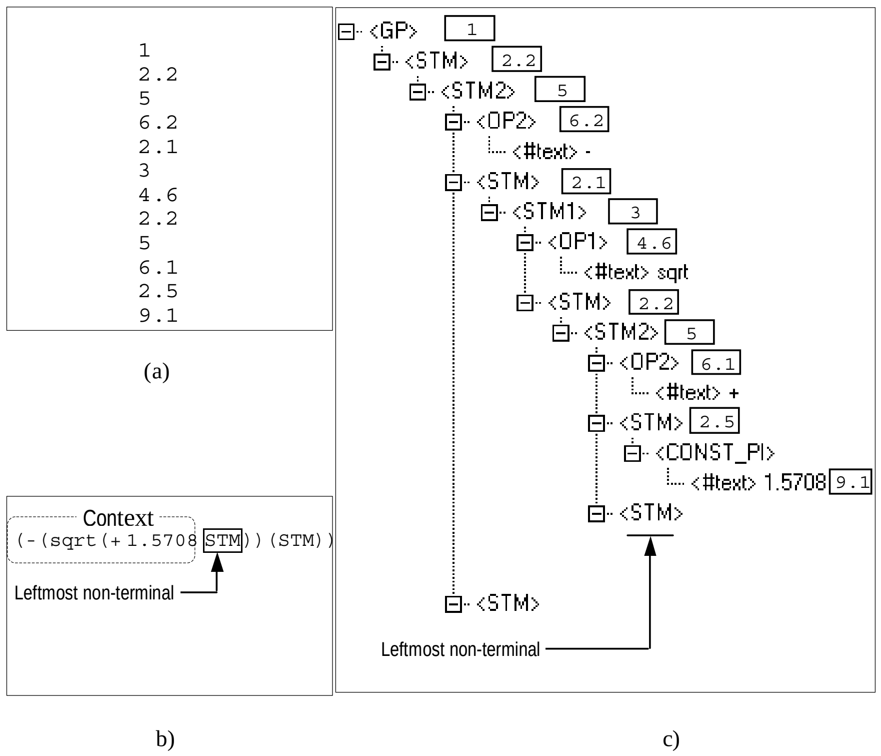
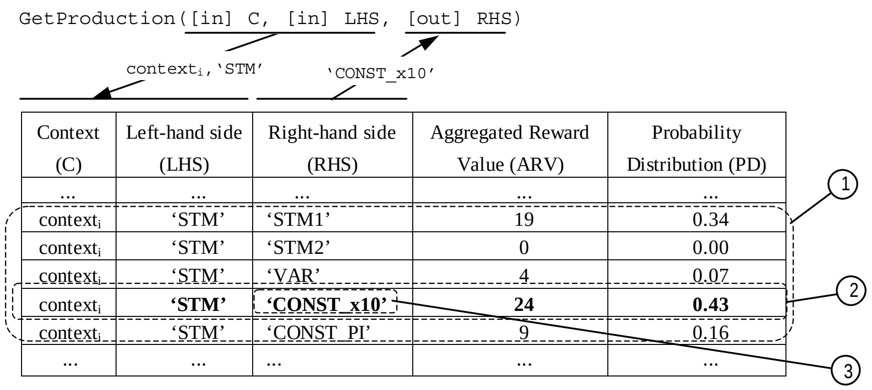


Fig. 6. Sample sequence of applied production rules (a) and resulting genetic program in prefix notation (b) and parsing tree (c)

Obtaining the probabilities of applying certain production rule with multiple right-hand side alternatives implies maintaining a probability distribution table. Each entry in the table stores the probability of applying certain production rule for given context. The probability is proportionally calculated from the reward values, aggregated over 10 best of run genetic programs obtained from experiments with evolving healthy Snakebot as elaborated in Section 2. The string of symbols of the right-hand side `RHS` of concrete production rule that should currently replace the leftmost nonterminal (i.e. the corresponding left-hand symbol in production rule, `LHS`) for given context `C` is obtained by function `GetProduction([in] C, [in] LHS, [out] RHS)` which operates on probability distribution table as illustrated in Figure 7.



Context (C)	Left-hand side (LHS)	Right-hand side (RHS)	Aggregated Reward Value (ARV)	Probability Distribution (PD)
...	...	...	...	...
context$_i$	'STM'	'STM1'	19	0.34
context$_i$	'STM'	'STM2'	0	0.00
context$_i$	'STM'	'VAR'	4	0.07
context$_i$	**'STM'**	**'CONST_x10'**	**24**	**0.43**
context$_i$	'STM'	'CONST_PI'	9	0.16
...	...	...	...	...

Fig. 7. Obtaining the string of symbols of the right-hand side `RHS` of production rule that should currently replace the left-most non-terminal (i.e. left-hand symbol in production rule, `LHS`), and the context `C`: 1) Selecting the set of entries associated with rules featuring the considered left-hand side `LHS` and context `C`, 2) Selecting a certain production rule (from the set of entries featuring considered `LHS` and `C`) with probability of selection, proportional to the learned probability distribution, and 3) returning the `RHS` of selected production rule

3.3 Empirical Results

The adaptation of Snakebot to partial damage is implemented via GP, where the latter is initialized with a population comprising the 10 best-of-run genetic programs, obtained from the experiments as described in Section 2.3, plus 190 individuals created applying the probabilistic context sensitive grammar. The genetic operations and the value of parameters of GP employed for adaptation of the damaged Snakebot are virtually the same as used for evolution of healthy Snakebot (as elaborated in Section 3.1), with the only difference that (i) the mutation ratio is increased from 0.01 to 0.1 and (ii) the altering the genetic programs during mutation operation is performed using the preferential application of certain grammar rules of the probabilistic context-sensitive grammar. The ability of sidewinding Snakebot to adapt to partial damage to 1, 2, 4 and 8 (out of 15) segments by gradually improving its velocity by simulated evolution via GP is shown in Figure 8. Demonstrated results are averaged over 20 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. As Figure 8 illustrates, Snakebot completely recovers from damage to single segment attaining its previous velocity in 25 generations with "canonical" strongly typed GP employing context-free grammar *G*, and only in 11 generations with GP employing probabilistic learning context sensitive grammar *G**. Snakebots recovers to average of 94% (with *G*) and 96% (*G**) of its previous velocity in the case where 2 (13% of total amount of 15) segments are damaged. With 4 (27%) and 8 (53%) damaged segments the degree of recovery is 77% (*G*) and 80% (*G**), and 64% (*G*) and 75% (*G**) respectively. In all considered cases of partial damage incorporating learning context sensitive grammar contributes to faster adaptation of Snakebot, and in all cases the Snakebot recovers to higher values of velocity of locomotion.

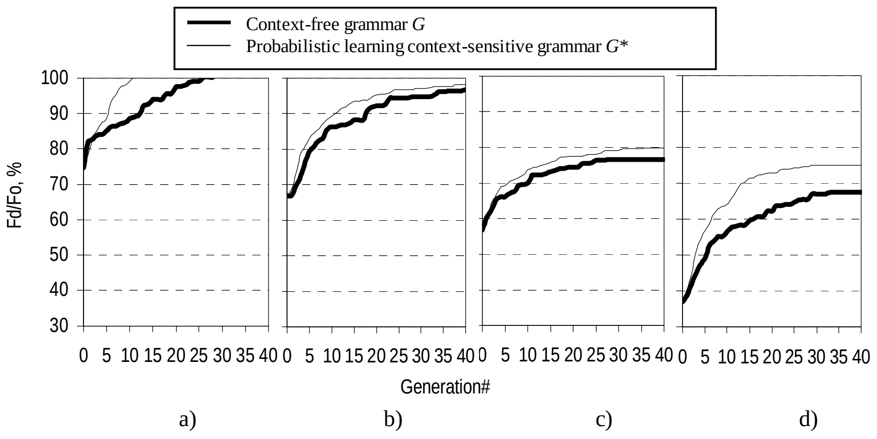


Fig. 8. Adaptation of sidewinding Snakebot to damage of 1 (a), 2 (b), 4 (c) and 8 (d) segments using context-free grammar and learning probabilistic context-sensitive grammar. Fd is the best fitness in evolved population of damaged snakebots, and Fh is the best fitness of 10 best-of-run healthy sidewinding Snakebots.

4 Conclusion

In this work we propose an approach of incorporating learning context-sensitive grammar in strongly typed genetic programming (GP) employed for evolution and adaptation of locomotion gaits of simulated snake-like robot (Snakebot). In our approach the probabilistic context-sensitive grammar is derived from the originally defined context-free grammar (which usually expresses the syntax of genetic programs in strongly typed genetic programming), using aggregated reward values obtained from the evolved best-of-run healthy, undamaged Snakebots. The probabilities of applying each of particular production rules with multiple right-hand side alternatives in derived probabilistic context-sensitive grammar depend on the context, and these probabilities are “learned” from the aggregated reward values. Empirically obtained results indicate that employing probabilistic context-sensitive grammar contributes to the improving the ability of Snakebot to adapt to partial damage by gradually improving its velocity characteristics. Snakebot recovers completely from single damage and recovers a major extent of its original velocity when more significant damage is inflicted. In all considered cases of inflicted partial damage of 1, 2, 4, and 8 out of 15 morphological segments, the incorporation of learning context sensitive grammar in GP improves the evolvability of adaptive locomotion gaits in that the recovery of partially damaged Snakebot is (i) faster and to (ii) higher values of velocity of locomotion.

Acknowledments. The author thanks Katsunori Shimohara, Thomas Ray, Hideaki Suzuki and Hidefumi Sawai for the stimulating discussions, suggestions and support. This research was supported in part by the National Institute of Information and Communications Technology of Japan.

References

1. 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)
2. Downing, K., L.: Adaptive Genetic Programs via Reinforcement Learning, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), (2001), 19-26.
3. Hirose, S.: Biologically Inspired Robots: Snake-like Locomotors and Manipulators, Oxford University Press (1993)
4. Kamio, S., Mitsuhashi, H. and Iba, H., Integration of Genetic Programming and Reinforcement Learning for Real Robots, Proceedings of the Genetic and Evolutionary Computation (GECCO-2003), (2003) 470-482.
5. Koza, J.R.: Genetic Programming 2: Automatic Discovery of Reusable Programs, The MIT Press, Cambridge, MA (1994)
6. Mahdavi, S., Bentley, P.J.: Evolving Motion of Robots with Muscles. In Proc. of EvoROB2003, the 2nd European Workshop on Evolutionary Robotics, EuroGP 2003 (2003) 655-664
7. Morowitz, H.J.: The Emergence of Everything: How the World Became Complex, Oxford University Press, New York (2002)
8. Pelikan M., Goldberg D. E., and Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), I, (1999) 525-532
9. Salustowicz, R. and Schmidhuber, J.: Probabilistic incremental program evolution. Evolutionary Computation, Vol.5 No.2, (1997) 123-141
10. Sawai, H. and Adachi, S.: Adaptive Strategy for GAs Inspired by Gene-Duplication, Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL-2002), Singapore, Vol.2, (2002) 592-599
11. Smith, R.: Open Dynamics Engine (2001-2003) http://q12.org/ode/
12. Zhang, Y., Yim, M. H., Eldershaw, C., Duff, D. G., Roufas, K. D.: Phase automata: a programming model of locomotion gaits for scalable chain-type modular robots. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003); October 27 - 31; Las Vegas, NV (2003)