On the Cumulative Effect of Bloat and Genetic Transposition on the Efficiency of Incremental Evolution of Snake-like Robot

Ivan Tanev, Tüze Kuyucu, Katsunori Shimohara
Department of Information Systems Design
Doshisha University, Kyotanabe, Japan
{itanev, tkuyucu, kshimoha}@mail.doshisha.ac.jp

ABSTRACT

We present a study on the cumulative effect of the bloat and the seeding of the initial population, inspired by genetic transposition (GT), on the efficiency of incremental evolution of simulated snake-like robot (Snakebot). In the proposed incremental genetic programming (IGP), the task of coevolving the locomotion gaits and sensing of the bot in a challenging environment is decomposed into two subtasks, implemented as two consecutive evolutionary stages. First, we use genetic programming (GP) with two ways of bloat management, (i) parsimony pressure which penalizes the bloat and (ii) no bloat control, to evolve two pools of sensor-less Snakebots. During the second stage of IGP, we use these pools to seed the initial population of Snakebots applying two methods of seeding: canonical seeding and GT-inspired seeding. The empirical results indicate that the efficiency of the first stage of IGP for both bloat control techniques is similar. However, the bloated bots contribute to a much more efficient second stage of evolution. Compared to the canonical seeding with parsimony bots, the GT-inspired seeding with bloated Snakebots yields about five times higher probability of success and similar decrease of computational effort of the second stage of IGP.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Algorithms, Design, Experimentation

Keywords

Snakebot, genetic programming, bloat, genetic transposition, incremental GP

GECCO'12, July 7–11, 2012, Philadelphia, Pennsylvania, USA.

1. INTRODUCTION

The insufficient efficiency of the genetic programming (GP), together with its non-determinism are among the most important drawbacks that still hinder the wide adoption of the evolutionary paradigm for solving challenging real-world problems. The overall efficiency of GP depends on the cumulative effect of two major, relatively independent factors: (i) the computational effort, i.e., the number of genetic programs that should be evaluated in order to achieve a given probability of success, and (ii) the computational performance, i.e., the average run-time, required to evaluate a single genetic program. Therefore, most of the efforts of researchers and practitioners in evolutionary computation (EC) community are aligned along the two orthogonal directions–improving the computational effort and computational performance of GP.

The computational effort of GP could be improved in several ways, such as, incorporating a domain-specific knowledge into the key attributes of GP (e.g., genetic representation, genetic operations, etc.), imposing problem-specific syntax constrains (i.e., grammar) on the evolved genetic programs, employing probability-distribution models, etc. These approaches are usually intended to steer the simulated evolution towards the most promising areas in the explored (presumably rugged) fitness landscapes.

Another approach of improving computational effort of GP stems from the assumption that the main genetic operations (crossover and mutation), due to their randomness, are often damaging the partially constructed building blocks of the solution. Thus, the destructive effects of these operations could be limited if they are occasionally allowed to operate on the neutral genetic code, i.e., the code, that is irrelevant to the quality (fitness) of the corresponding genetic program. Moreover, such a neutral code might provide the simulated evolution with a "playground" where the latter can experiment with developing either novel-, or better-than-existing genotypic traits without the risk of damaging the already evolved ones.

In biology, the constructive role of neutrality has been well recognized, and noted to be a conducive mechanism in the evolution of many successful traits in biological organisms [8, 22]. The presence of neutral genes in Nature is often associated with smoother fitness landscapes, more robust genotypes and a support mechanism for discovering new phenotypes [8, 21, 22]. Similarly, the neutrality has also been a topic of interest and discussion in EC. In a recent article, Galvan-lopez et al. [6], provide a good overview of the

role of neutrality in EC. The authors conclude their work with the open issues in this area and with their opinion on the studies that would be beneficial to a better understanding of the role of neutrality in EC. Due to the complex and, to some extent, unpredictable nature of the latter, it is often difficult to achieve an in-depth theoretical analysis of the effects of mechanisms and parameters on the performance of these algorithms.

The work of Ebner [5] suggests that the neutrality induced by the "junk code" provides better performance for the evolution of genetic programs. It was also reported that the neutrality contributes to a better efficiency of evolution in Cartesian Genetic Programming (CGP) [20, 23]. This conclusion remains to be a controversial one, however, as in a follow-up study Collins reported that the latter work is being flawed and that the effects of neutrality could in fact be degrading the overall performance of evolution [3]. This ongoing discussion on the possible beneficial effect of neutrality on the overall efficiency of GP is also stated by Galvan-lopez et al. [6], who note that "one needs to find ways of predicting when the addition of neutrality can be beneficial in practical situations." Although neutrality has been shown to be beneficial in complex, rugged landscapes with multiple optima [1, 4, 12], it also has adverse effects on the evolution of simple problems with a small number of optima and a smooth landscape.

Despite the beneficial effects of neutrality on the computational effort of GP, its implications on the overall performance of GP are highly controversial. One of the most important drawbacks of neutral code is that it often causes a degradation of computational performance of GP. Indeed, the neutral code in GP is often associated with the resulting bloat [2], or, with a sharp increase of the size (and complexity) of genetic programs in due course of the simulated evolution. Usually, the increased size of the evolved genetic programs does not correlate well with the convergence of the respective fitness values. Moreover, due to the cache memory effects, the runtime overhead of interpreting the genetic programs (represented in the computer memory as highly fragmented parse trees) grows faster than linear with the increase of their size. Therefore, applying parsimony pressure in order to limit the growth of the size of genetic programs is often seen as a natural way to alleviate bloat-induced degradation of computational performance of GP [7, 14].

In our previous work we developed an incremental approach in order to improve the efficiency of evolution of complex robotic artifacts in challenging environment. In the proposed incremental genetic programming (IGP), the task of coevolving the locomotion gaits and sensing of a simulated Snake-like robot (Snakebot) in a challenging environment is decomposed into two subtasks, implemented as two consecutive evolutionary stages. First, we use genetic programming to evolve a pool of sensorless Snakebots that move fast in a smooth, open terrain. Then, during the second stage, we use these pools to seed the initial population of Snakebots that are further subjected to coevolution of their locomotion control and sensing morphology in a challenging environment. Moreover, we also demonstrated a mechanism, inspired by the genetic transposition (GT) seen in nature, to create redundant, neutral genetic spaces by artificially bloating the bots at the stage of seeding the second stage of IGP. This mechanism was shown to provide better performance in terms of the computational effort required to evolve high quality Snakebots [11]. The resulting approach provided a valuable gain in the overall performance due to the negligible decrease in the computational performance–since the main contributor to the computational overhead is the simulation of the Snakebots during the evaluation phase. However, the effect of neutral code in the bots, evolved during the first stage of IGP, on the efficiency of the second stage was considered beyond the scope of the considered work.

The objective of this work is to investigate the cumulative effects of both the bloat and GT on the efficiency of IGP used for incremental evolution of locomotion of sensing Snakebot in a challenging environment with obstacles. The successful bots should feature the evolved (emergent) know-how about how to clear a narrow corridor by (i) moving fast, (ii) following the walls of the corridor, (iii) overcoming a number of randomly scattered small boxes, and (iv) circumnavigating large obstacles. In the proposed incremental GP (IGP), the task of coevolving the locomotion and the sensing of Snakebot in a challenging environment is decomposed into two sub-tasks, implemented as two consecutive evolutionary stages. First we employ GP to evolve a pool of simple, sensor-less bots that are able to move fast in a smooth, open terrain. Then, during the second stage, we use this pool to seed the initial population of the bots that are further subjected to coevolution of their locomotion control and sensing in the challenging environment. We are especially interested on how the degree of bloat (and, consequently, the associated neutrality), introduced during the fist stage of the incremental evolution, effects the overall performance of IGP.

Our choice of the application of GP is motivated by two arguments that, as we believe, are in favour of the neutrality. First, the considered problem is rather challenging, as it features a large and highly rugged fitness landscape [11]. Shipman demonstrated that neutrality helps the discovery of multiple phenotypes, but reduces the evolutionary performance for achieving faster solutions in simpler problems [15]. Within this context, we would like to investigate if, during the second stage of IGP, the neutrality would decrease the computational effort of evolving novel traits (e.g., the sensory abilities of the bot) in addition to the already evolved locomotion of the bots. And second, as in the most of the tasks in evolutionary robotics, it is the realistic simulation of the physics of moving complex robotic artefact, rather than the parsing of the genetic programs that consumes the most of the run-time of GP. Therefore, we anticipate no major bloat-induced degradation of the computational performance of GP, as experienced with GT.

The remaining of this document is organized as follows. Section 2 introduces the morphology and the moving abilities of the Snakebot. In Section 3 we discuss the key attributes of the proposed evolutionary framework. Section 4 presents the empirical result on the effect of bloat on the efficiency of incremental evolution of the bot in a challenging environment. Section 5 draws a conclusion.

2. SIDEWINDING AND SENSING SNAKE-LIKE MODULAR ROBOT

Snake-like robots feature potential robustness characteristics beyond the capabilities of most wheeled and legged vehicles, such as: the ability to traverse challenging terrain and insignificant performance degradation when par-

tial damage is inflicted. Some useful features of snake-like robots include smaller size of the cross-sectional areas, stability, ability to operate in difficult terrain, good traction, and complete sealing of the internal mechanisms. Moreover, due to the modularity of their design, the snake-like robots feature high redundancy and fault tolerance [17]. Robots with such properties can be valuable for applications that involve exploration, reconnaissance, medicine and inspection. Designing a controller that can achieve optimal locomotion of a modular Snakebot is a challenging task due to the large number of degrees of freedom in the movement of segments of a Snakebot. The locomotion gait of such bots is often seen as an emergent property; observed at a higher level of consideration of complex, nonlinear, hierarchically organized systems, comprising many relatively simply-defined entities (morphological segments). In such complex systems the higher-level properties of the system and the lower-level properties of comprising entities cannot be directly induced from each other [13]. Therefore even if an effective incorporation of sensing information in fast and robust locomotion gaits might emerge from intuitively defined sensing morphology and simple motion patterns of morphological segments, neither the degree of optimality of the developed code nor the way of how to incrementally improve this code is evident to the human designer [10].

The previous research demonstrates that the control for a fast moving modular robotic organism could be automatically developed through various nature-inspired paradigms, based on models of learning and evolution. The work, presented in [17] demonstrates the use of GP for evolution of sensor-less sidewinding Snakebots in various environmental conditions. Furthermore, the coevolution of active sensing and the control of the locomotion gaits was demonstrated to be achievable, albeit difficult [11]. The control of a modular Snakebot with sensors for locomotion through a maze with obstacles was shown to be a challenging task to achieve via canonical GP even when ADFs were used. The use of incremental GP was shown to be a better approach, where initially the locomotion of the Snakebot in an obstacle free environment was achieved before evolving these Snakebots for a second time to utilize sensors. Furthermore, the use of a GT inspired incremental GP, which utilizes the addition of neutral code into the genotype of the seeding individuals for the second stage of the incremental GP runs had higher success rates as well as more robust solutions [11].

The morphology of the sensors, attached to each of the segments of the bot, coevolves with the way to incorporate the sensory readings into the control of locomotion of the bot. The genetically optimized morphological traits of the bot include the initial orientation, the timing of switching on, and the range of the simulated proximity sensors (e.g., laser range finders, LRF) attached to each of the segments of the bot. The emergent features of the evolved gaits include both the contact and contact-less wall-following navigation accomplished via adaptive, sensory-controlled differential steering of the fast moving sidewinding bot.

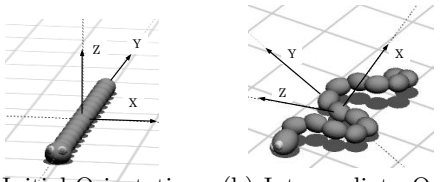
In this paper we investigate the coevolution of the active sensing and locomotion control of sidewinding Snakebot in the same environment presented in [11], which features a narrow corridor with several large obstacles and many randomly placed small obstacles constituting a rugged terrain. The sensors on the Snakebot used in this paper follow the same model as proposed in [18]: each segment of the Snakebot is provided with a fixed, immobile LRF with evolvable initial orientation, range and timing of firing. The most efficient locomotion gaits of Snakebot are not necessarily associated with the forward, rectilinear motions (and sidewinding might emerge as a fast and robust locomotion). Therefore, the eventual fusion of the readings of many sensors mounted in all the segments of the bot would provide Snakebot with the capability to perceive the features of surrounding environment along its whole body. In addition to the widening of the area of the perceived surroundings, multiple sensors offer the potential advantages of robustness to damage of some of them, dependability of the sensory information, and an ability to perceive the spatial features of the surrounding environment due to the motion parallax.

Scalable approaches that can handle multiple tasks are important in the evolution of Snakebot, as the complexity of evolving a controller for the described set up can become an issue. The size of evolutionary search space can be seen as a multiplication of the sizes of the search spaces of the following interdependent evolutionary sub-tasks:

- Evolution of the *control of locomotion*: the time patterns of turning angles of actuators that result in a fast locomotion of the bot.
- Evolution of the *morphology of the active sensing*: initial orientation of the sensors, their range, and timing of their activation.
- Evolution of the *incorporation of the sensor signals* into the control of locomotion of the bot.

The evolution of both the morphology and the incorporation of the signals from many sensors face the challenge of dealing with the uncertain sensor readings as they move synchronously with the coupled segments of the snake. Figure 1 illustrates how the initial orientation of the axes of the internal coordination systems of the segments of a bot dramatically differs from a sample instant orientation of these axes in a moving bot. A sensor fixed to the segment of a moving Snakebot would constantly change its spatial orientation, and consequently it might alternatively perceive no signal, a signal from the ground surface or from another segment of the snake (in both cases the sensory reading should be ignored), or eventually from an obstacle. Moreover, in the targeted environment the obstacle could be either a wall (to be followed), a large box (to be circumnavigated), or a small box (to be overcome).

The large search space of the evolution of the considered Snakebot results in an intractable computational effort, and as it was demonstrated in [11], canonical GP with Automat-



(a) Initial Orientation. (b) Intermediate Orientation.

Figure 1: Orientation of the axes of the internal coordination systems of the central segment at two different Snakebot positions.

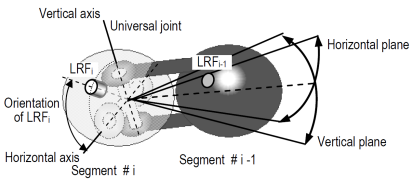


Figure 2: Morphological segments of Snakebot are linked via universal joint. Horizontal and vertical actuators attached to the joint perform rotation of the segment #i-1 in vertical and horizontal planes respectively. A single LRF is attached to each of the segments in the plane of the axes of the universal joint.

ically Defined Functions (ADF) is unable to effectively explore the emerging search space in a reasonable time frame.

3. EVOLUTIONARY FRAMEWORK AND THE SIMULATION ENVIRONMENT

For the experiments presented in this work we employ open dynamics engine (ODE) as a simulation platform for the Snakebot. ODE is a free, industrial quality software library for simulating articulated rigid body dynamics [16]. It is fast, flexible and robust, and it has built-in collision detection. Therefore, ODE is suitable for a realistic simulation of the physics of an entire Snakebot when applying actuating forces to its segments. The main ODE related parameters of the simulated Snakebot are same as elaborated in [17].

Snakebot is simulated in ODE as a set of 15 identical spherical morphological segments ("vertebrae"), linked together via universal (Cardan) joints (Figure 2). All joints feature identical angle limits and each joint has two attached actuators ("muscles"). A single LRF sensor, with a limited range is rigidly attached to each of the segments.

The functionality of the LRF can be defined by the values of the following set of parameters: (i) orientation, measured as an angle between the longitudinal axis of the sensor and the horizontal axis of the joint, (ii) range of the sensor (in cm), and (iii) the timing of activation, expressed as a threshold value of the turning angle of the horizontal actuator. The reading of LRF is a scalar value which corresponds inversely to the distance between the sensor and an object (if any within the sensor's range), measured along the longitudinal axis of the LRF. In the initial standstill position of Snakebot (as depicted in Figure 1(a)) 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.

For the evolution of the Snakebot, the genotype is represented as a triple consisting of a linear chromosome containing the encoded values of the three relevant parameters of LRF, and two parse trees corresponding to the algebraic

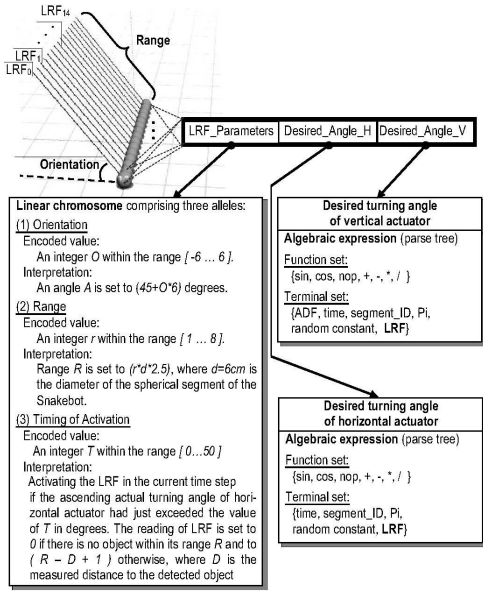


Figure 3: Genotype of the Snakebot consist of a triple containing the values of the parameters of LRF and two algebraic expressions of the temporal patterns of the desired turning angles of horizontal and vertical actuators, respectively. The genotype of Snakebot is homogeneous: therefore all segments feature the same triple.

expressions of the temporal patterns of the desired turning angles of both the horizontal and vertical actuators, respectively (Figure 3).

The Snakebot is genotypically homogeneous in that the same triple is applied for the setup of the LRF and for the control of actuators of all morphological segments. The encoding of the parameters of LRF is as elaborated in Figure 3. The same figure also illustrates the function set and the terminal set of the GP, employed to evolve the control sequences of both actuators. Since the locomotion gaits by definition are periodical, the periodic functions sine and cosine are included in the function set of GP in addition to the basic algebraic functions. Terminal symbols include the variables time, segment_ID, an ADF, the reading of the sensor (LRF), and two constants: pi, and a random constant within the range [0, 2]. The incorporation of the terminal symbol segment_ID (a unique index of morphological segments of Snakebot) provides GP with an effective way to specialize (by phase, amplitude, frequency etc.) the genetically identical motion patterns of actuators of each of the morphological segments of the Snakebot.

The rationale of employing ADFs is based on the observation that the evolvability of straightforward, independent encoding of desired turning angles of both horizontal and vertical actuators is rather poor. Even without ADFs, GP is able to adequately explore the potentially large search space and ultimately discover the areas that correspond to fast locomotion gaits in the solution space. However, it was observed in the previous work of Tanev et al. [17] that not only the motion patterns of adjacent segments are correlated, but the motion patterns of horizontal and vertical actuators of each segment in fast locomotion gaits are highly correlated

too. Moreover, discovering and preserving such correlation by GP is associated with enormous computational effort. ADFs, which provide a way of introducing modularity and reuse of code in GP [9], are 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 both actuators. Furthermore, we observed that the best results are 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) evaluating the value of ADF by equalizing it to the value of currently evaluated desired turning angle of horizontal actuator.

Table 1: Main parameters of GP.

Category	Value
Genotype	LRF parameters (linear chromosome) Horizontal actuator control (parse tree) Vertical actuator control(parse tree)
Population Size	200 individuals
Selection	Binary Selection ratio: 0.1 Reproduction ratio: 0.9
Elitism	4 individuals
Mutation Rate	0.01
Trial Interval	16s (400 time steps of 40ms per step)
Termination Criteria	(Fitness=120) or (Tot. No. of generations=80)

The main GP (hence the EA) parameters are summarized in Table 1. We use a DOM/XML-based implementation of GP [19], with binary tournament selection and a single point crossover. The crossover point is randomly selected between the three components of the genotype (as shown in Figure 3), unless stated otherwise. The mutation randomly alters either a value of an allele in the linear chromosome representing the parameters of LRF, or a sub-tree in one of the two parse tress that correspond to the temporal patterns of the control sequences of actuators.

4. EXPERIMENTAL SETUP

The experimental setup of employing IGP for evolution of Snakebot is illustrated in Figure 4. During the first stage of IGP, the locomotion of sensor-less bot is evolved in a smooth terrain with two cases of bloat control: parsimony pressure penalizing bloat (case 1a, pBC) and no bloat control (case 1b, noBC), respectively. In the former case, the fitness value is penalized by $\frac{1}{10}$ times the number of tree nodes in the genotype of the bot. In the case of no bloat control, the fitness value is not altered with respect to the complexity of the genotype of the bot. Each of these two cases are executed for 40 independent evolutionary runs. The target fitness value is 100, which corresponds to the velocity of locomotion that displace the Snakebot during the simulated trial of 16s a distance equal to twice its length.

The best six bots from each of these two cases are then used as seeding bots for the stage 2 of IGP, initialized by the following two seeding mechanisms: canonical seeding (stage 2a) and seeding via GT (stage 2b) respectively. In canonical seeding, 6 individuals from stage 1 are used "as is", while the remaining 194 individuals in the population are created randomly. In the seeding via GT all 200 individuals in population are created by incorporating the entire genomes of the 6 best bots from stage 1: 6 individuals being used "as is", and the remaining 194 individuals created using the GT inspired method described in Figure 4.

For the second stage of IGP, there are two experimental cases as well (shown in Figure 4 as stage 2a and 2b, respectively). These two stages differ in the mechanism of seen only, and feature identical evolutionary conditions. The experimental cases of both stages 2a and 2b feature the same parsimony pressure conditions - a penalty to the fitness of by $\frac{1}{10}$ times the number of tree nodes in the genotype. By applying bloat control in the second stage of IGP we could obtain good, yet genetically simple bots (i.e., Occam razor). These simple genotypes could be better comprehended and modified (if needed) by human, and efficiently implemented in the controller of the bot.

The difference in stages 2a and 2b is in the used two mechanisms of seeding: canonical seeding (stage 2a) and seeding via GT (stage 2b), respectively (Figure 4). In canonical seeding, 6 individuals from stage 1 are used "as is", while the remaining 194 individuals in the population are created randomly. In the seeding via GT all 200 individuals in population are created by incorporating the entire genomes of the 6 best bots from stage 1 (shown as Transposon A) in the randomly initialized genotype (Transposons B and C).

The fitness function is based on distance the Snakebot travels during the trial. The normalized fitness of 120 corresponds to the distance required to be traveled during the trial in order to clear a narrow corridor covered with obstacles of various sizes. The evolution is terminated if the bot clears the corridor (fitness is 120) or if the maximum number of generations is reached.

4.1 Stage 1: Evolution of Locomotion of Sensorless Snakebot

We executed 40 independent runs for both the Stage 1a (pBC) and 1b (noBC) of IGP, respectively. The aggregated fitness convergence characteristics are shown in Figure 5. As the figure illustrates, the use of a simple parsimony pressure, as described earlier, has no implications significant on the fitness convergence characteristics, and therefore, on the overall computational effort of the first stage of IGP. We also visually inspected the locomotion gaits of the successful Snakebots from each of these two cases and confirmed that

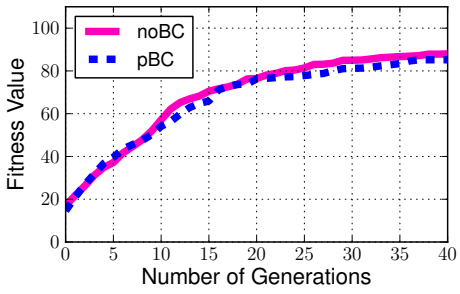


Figure 5: The convergence of average fitness during the first stage of IGP with two cases of bloat control - parsimony pressure (pBC) and no bloat control (noBC), respectively. The results are aggregated over 40 independent runs.

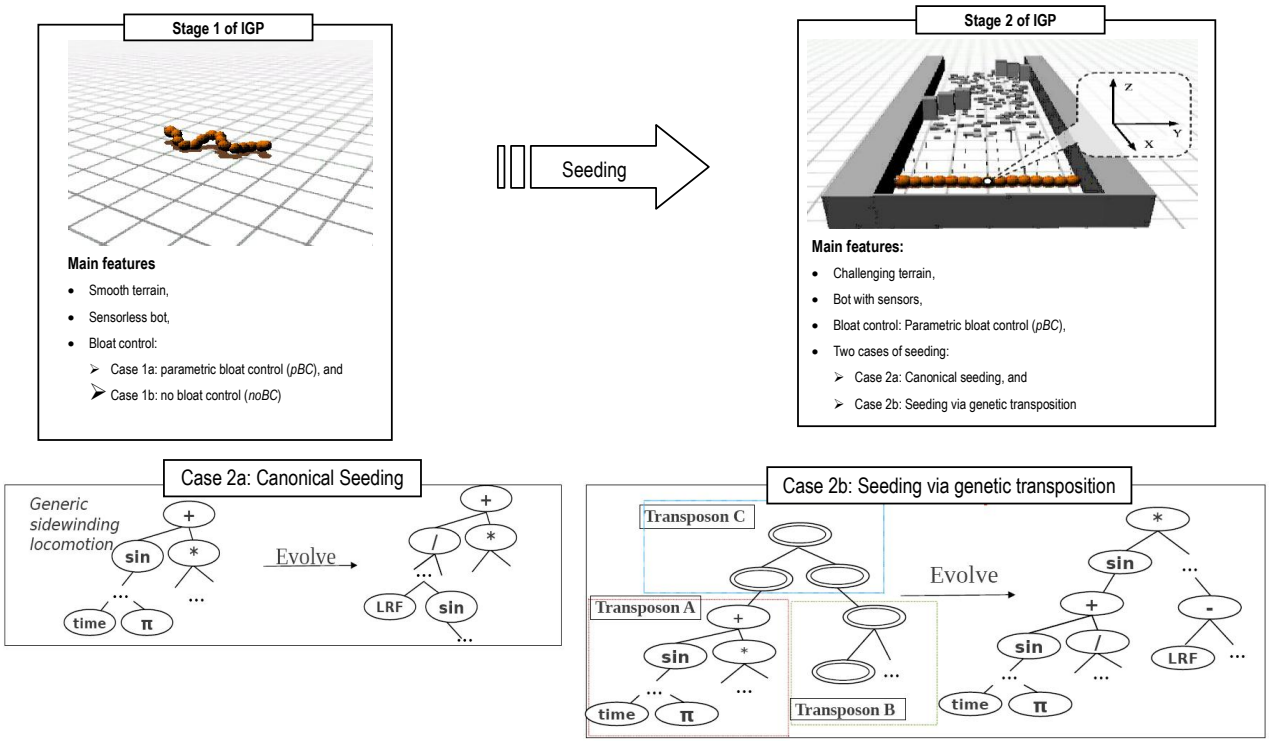


Figure 4: The two stages of IGP. In stage 1, the locomotion of senssorless bot is evolved in a smooth terrain with two cases of bloat control: parsimony pressure penalizing bloat (case 1a, pBC) and no bloat control (case 1b, noBC), respectively. The best six bots from each of these two cases are then used as seeding bots for the stage 2 of IGP, initialized by the following two seeding mechanisms: canonical seeding (stage 2a) and seeding via GT (stage 2b) respectively. In canonical seeding, 6 individuals from stage 1 are used "as is", while the remaining 194 individuals in the population are created randomly. In the seeding via GT all 200 individuals in population are created by incorporating the entire genomes of the 6 best bots from stage 1 (shown as Transposon A) in the randomly initialized genotype (Transposons B and C).

they are similar. The use of parsimony pressure did exactly what it is supposed to: the only significant difference between the two experimental cases was in the average tree sizes, and the use of parsimony pressure provided solutions with smallest tree sizes. The statistical results are shown in Table 2.

4.2 Stage 2a: Coevolution of Sensing and Locomotion via Canonical Seeding

We executed 40 independent evolutionary runs using canonical seeding with two different pools (of six best Snakebots each) of seeding bots, obtained from stages 1a (pBC) and 1b (noBC), respectively (Figure 4). The six bots from these pools are used as elite individuals and the remaining 194 individuals of the initial population are generated randomly. The average fitness convergence characteristics are shown in Figure 6(a). Unlike the runs from the first stage, there is a significant difference in the performance of the evolutionary runs using different seeds. The best evolutionary performance is obtained when seeds that were previously evolved with no bloat control are used, and the worst performance is observed when seeds with parsimony pressure are used. In average, the seeds with no bloat control reaches the best fitness values of the parsimony seeds about five times faster.

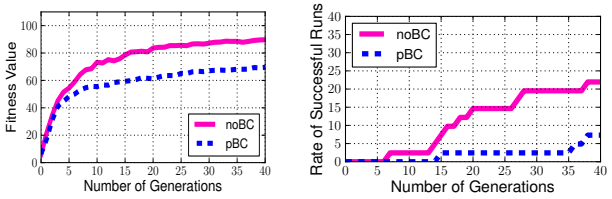
Table 2: Results of evaluation of IGP. Stages 2a and 2b are initialized via canonical and GT-inspired seeding, respectively. F_{avr} is the average fitness of population in the final generation, R_{succ} is the number of successful runs of the second stage (of 40), S_{avr} is the average size of genetic programs (number of tree nodes) in the final generation, and $SSeed_{avr}$ is the average size of genetic programs in the pool of the six seeding bots, respectively.

Stage of IGP		F_{avr}	R_{succ}	S_{avr}	$SSeed_{avr}$
Stage 1	noBC	101	NA	105.3	NA
	pBC	100.5	NA	82.4	NA
Stage 2a	noBC	87.3	8	197	131
	pBC	68	3	122	85
Stage 2b	noBC	95	16	220	131
	pBC	89.5	6	135	85

The probability of successful runs (i.e., the bot clears the narrow corridor) is shown in Figure 6(b) for the two cases of IGP runs. As the figure illustrates, a clearer distinction

between the runs using no bloat control (noBC) and the parsimony pressure (pBC) can be seen; the former case feature both a better convergence of the fitness and higher probability of success. Moreover, because the realistic simulation of the physics of the bot (rather than the parsing of the genotype) consumes most of the run-time of GP, no major bloat-induced degradation of the computational performance of IGP is observed.

The statistical information from these experiments is summarized in Table 2.



(a) (b)

Figure 6: The convergence of average fitness (a), and the probability of successful runs (b) of stage 2a (with canonical seeding) of IGP. pBC and noBC denote the results, obtained from the pool of seeding bots evolved during stages 1a and 1b, respectively. The results are aggregated over 40 independent runs.

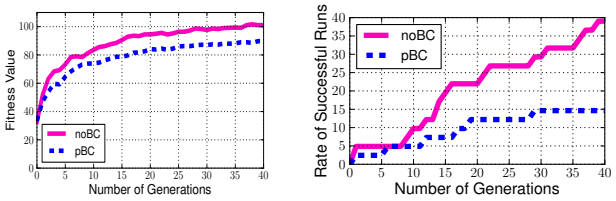
4.3 Stage 2b: Sensing and locomotion via IGP with GT

We conducted 40 independent evolutionary runs using GT-inspired seeding from two different pools of bots (of six best Snakebots each), obtained from stages 1a (pBC) and 1b (noBC) of IGP, respectively (Figure 4). Analogically to the experimental setup of stage 2a, six bots from the two pools are used as elite individuals. However, the remaining 194 individuals of the population are created via the GT-inspired seeding mechanism as detailed in [11] and illustrated in Figure 4. The genome of the seeding bot is used to form only a part of the new individual, where the rest of the genetic structure is generated randomly. As shown in Figure 7(a), the average fitness convergence of the seeds, evolved without bloat control (noBC) during stage 1b of IGP, is much faster than the seed with parsimony pressure (pBC). The runs with seeds evolved with no bloat control reach the best fitness values of the parsimony seeds almost three times faster (fitness of 90 is achieved in 15 vs 40 generations).

As Figure 7(b) illustrates, the probability of success of the runs using seeds evolved under parsimony pressure have success chances of 20% after 40 generations. The same probability of success is reached in 13 generations (i.e., three times faster) by runs using seeds evolved with no bloat control. The best probability of success of 40% after 40 generations is achieved by seeds evolved with no parsimony pressure.

5. CONCLUSIONS

In this work we studied the cumulative effect of bloat and seeding inspired by genetic transposition (GT) on the efficiency of incremental evolution of simulated sensing snake-like robot in a challenging environment. As the experimental results suggest, the use of a simple parsimony pressure has



(a) (b)

Figure 7: The convergence of average fitness (a), and the probability of successful runs (b) of stage 2b (with GT-inspired seeding) of IGP. pBC and noBC denote the results, obtained from the pool of seeding bots evolved during stages 1a and 1b, respectively. The results are aggregated over 40 independent runs.

no immediate effect on the efficiency of evolution of the first stage of incremental genetic programming (IGP) - evolution of fast moving sensor-less bots in a smooth terrain. However, when, during the second stage of IGP, evolved genetic programs are reused for further development under different conditions than the first stage, the neutrality caused by the bloated genetic programs are beneficial for the more efficient evolution of the sensing abilities of the bot. We assume that this is due to the presence of a neutral code (i.e., a code which has no immediate effect on the fitness), that could be utilized by IGP as an evolutionary playground to develop the needed sensory abilities without the risk of damaging the already evolved, fast locomotion.

Furthermore, we demonstrate that the best performance is achieved when bloated seeds are used with GT-inspired seeding. In average, the GT-inspired seed with bloated bots features about five times higher probability of success and similar decrease of computational effort of the second stage of IGP than canonical seed with parsimony bots. GT aims to introduce additional neutrality to the seeding genome in order to provide a safe playground for the genetic operators. The success of GT was demonstrated to significantly surpass the performance shown by canonical seeding in an earlier work [11]. In the experiments presented here, the neutrality introduced by GT is shown to complement the neutrality from bloated seeds, providing the best evolutionary performance as a result. The experimental results suggest that evolution benefits from the presence of different forms of neutrality introduced by both GT and bloated seeds when a two-stage IGP is used. Conversely, the explicit parsimony pressure negatively affects the computational effort of evolution.

The presented findings could be applied for the domains where the implementation of the side effects, rather than the parsing of genetic representation, is by far the most computationally expensive aspect of fitness evaluation. However, the overall effect of the interplay between (i) the bloat-induced degradation of the computational performance, and (ii) the possible improvement of the computational effort should be carefully considered before applying the proposed approach in domains where the parsing and evaluating the very genetic programs dominate the run-time overhead of the evolutionary system.

6. ACKNOWLEDGMENTS

The presented work is part of a project funded by Japan Society for the Promotion of Science (JSPS).

7. REFERENCES

[1] W. Beaudoin, S. Verel, P. Collard, and C. Escazu. Deceptiveness and neutrality: The nd family of fitness landscapes. In *GECCO 2006: Proceedings of the 2006 conference on genetic and evolutionary computation*, pages 507–514, 2006.

[2] M. Brameier and W. Banzhaf. Neutral variations cause bloat in linear gp. In *Proceedings of the 6th European conference on Genetic programming*, EuroGP'03, pages 286–296. Springer-Verlag, 2003.

[3] M. Collins. Finding needles in haystacks is harder with neutrality. In *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1613–1618, 2005.

[4] B. Doerr, M. Gnewuch, N. Hebbinghaus, and F. Neumann. A rigorous view on neutrality. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 2591 –2597, sept. 2007.

[5] M. Ebner. On the search space of genetic programming and its relation to nature's search space. In *Proceedings of the 1999 Congress on Evolutionary Computation. CEC 99*, pages 1357–1361, 1999.

[6] E. GalvÃąn-LÃşpez, R. Poli, A. Kattan, M. OâĂŹNeill, and A. Brabazon. Neutrality in evolutionary algorithmsâĂę what do we know? *Evolving Systems*, 2:145–163, 2011.

[7] S. Gelly, O. Teytaud, N. Bredeche, and M. Schoenauer. Universal Consistency and Bloat in GP. *Revue d'Intelligence Artificielle*, 20:805–827, 2006.

[8] M.A. Huynen, P.F. Stadler, and W. Fontana. Smoothness within ruggedness: the role of neutrality in adaptation. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 93, pages 397–401, 1996.

[9] J.R. Koza. *Genetic programming II: automatic discovery of reusable programs*. MIT Press, Cambridge, MA, USA, 1994.

[10] J.R. Koza, M.A. Keane, J. Yu, F.H. Bennett, and W. Mydlowec. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines*, 1:121–164, 2000.

[11] T. Kuyucu, I.T. Tanev, and K. Shimohara. Incremental genetic programming via genetic transpositions for efficient coevolution of locomotion and sensing of simulated snake-like robot. In *European Conference on Artificial Life*, pages 439–446, 2011.

[12] J. Lobo, J. H. Miller, and W. Fontana. Neutrality in technological landscapes. *Santa Fe Working Paper*, 2004.

[13] H.J. Morowitz. *The Emergence of Everything: How the World Became Complex*. Oxford Uni. Press, 2002.

[14] R. Poli and N.F. McPhee. Covariant parsimony pressure for genetic programming. Technical Report CES-480, Department of Computing and Electronic Systems, University of Essex, UK, 2008.

[15] R. Shipman. Genetic redundancy: Desirable or problematic for evolutionary adaptation. In *4th International Conf. on Artificial Neural Networks and Genetic Algorithms (ICANNGA'99)*, pages 1–11, 1999.

[16] R. Smith. *Open Dynamics Engine*, 2004.

[17] I. Tanev, T. Ray, and A. Buller. Automated evolutionary design, robustness and adaptation of sidewinding locomotion of simulated snake-like robot. *IEEE Transactions on Robotics*, 21:632–645, 2005.

[18] I. Tanev and K. Shimohara. Co-evolution of active sensing and locomotion gaits of simulated snake-like robot. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 257–264, New York, NY, USA, 2008. ACM.

[19] I.T. Tanev. Dom/xml-based portable genetic representation of the morphology, behavior and communication abilities of evolvable agents. *Artificial Life and Robotics*, 8:52–56, 2004.

[20] V.K. Vassilev and J.F. Miller. The advantages of landscape neutrality in digital circuit evolution. In *Proceedings of the 3rd International Conference on Evolvable Systems: From Biology to Hardware*, pages 252–26. Springer, 2000.

[21] A. Wagner. Robustness, evolvability, and neutrality. *FEBS Letters*, 579(8):1772–1778, 2005.

[22] C.O. Wilke, J.L. Wang, C. Ofria, R.E. Lenski, and C. Adami. Evolution of digital organisms at high mutation rates leads to survival of the flattest. *Nature*, 412:331–333, 2001.

[23] T. Yu and J.F. Miller. The role of neutral and adaptive mutation in an evolutionary search on the onemax problem. In *GECCO Late Breaking Papers'02*, pages 512–519, 2002.