Pricing the ‘Free Lunch’ of Meta-Evolution

Alexei V. Samsonovich
Krasnow Institute for Advanced Study
George Mason University
Fairfax, VA 22030-4444
703-993-4385
asamsono@gmu.edu

Kenneth A. De Jong
Krasnow Institute for Advanced Study and
Computer Science Department
George Mason University, Fairfax, VA
703-993-4398
kdejong@gmu.edu

ABSTRACT

A number of recent studies introduced meta-evolutionary strategies and successfully used them for solving problems in genetic programming. While individual results indicate possibilities of successes and failures (e.g., Kantschik, Dittrich et al., 1998, 1999), the emerging global picture suggests that the approach may have universal, domain-independent advantages over traditional methods. Trying to develop a general theoretical understanding of this concept, we use Price’s theorem to define fitness at a meta-level and show with two simple case studies (two-dimensional optimization and the Eight puzzle) that the ideology based on Price’s theorem can work at a meta-level in a similar manner for very different problems. Specifically, Pricean definition of fitness for reproductive operators appears to be practically useful and essential for performance and stability of a certain class of meta-evolutionary algorithms.

Categories and Subject Descriptors

I.1.2 [Computing Methodologies] Symbolic and Algebraic Manipulation – Algorithms. I.2 [Computing Methodologies] Artificial Intelligence – General.

General Terms

Algorithms, Measurement, Performance, Design, Theory, Verification.

Keywords

Evolvability, self-adaptive EAs, metaevolution, co-evolution, Price’s theorem.

1. INTRODUCTION

A traditional evolutionary computation scheme uses only a limited, fixed set of reproductive operators, such as mutation and crossover. This basic paradigm remains unchanged in the majority of studies that treat parameters of reproductive operators as dynamic, evolvable entities [3]. Self-adaptive parameters used in these studies included the mutation step [1, 11], mutation rates [2], crossover templates, or masks [10], and the like.

In contrast, the key idea of a meta-evolutionary approach is to consider a search for a best evolutionary algorithm as an optimization problem to be solved by evolutionary computation. In a globalistic interpretation of this idea [5], a set of base-level evolutionary algorithms, each with its own set of parameters and opereators, are treated as the evolving population of individuals at the meta-level. An alternative paradigm [7] is to treat reproductive operators and primitives on their own as individuals subject to evolution. E.g., in this case, the population of operators can evolve in parallel with the base-level population, thereby adapting themselves locally, to the current base-level population, rather than globally, to the base-level problem as a whole. The present work is focused on this “local” meta-evolutionary approach. In this framework, reproductive operators that modify the meta-level population may be the same set of operators (if they can modify their own kind) – or yet another set of operators. Further elaboration leads to a multi-level meta-evolutionary architecture [6] with a hierarchy of populations evolving in parallel. In this architecture, higher-level individuals are used as operators to perform reproduction of immediately-lower-level individuals and, possibly, their own kind as well (Figure 1).

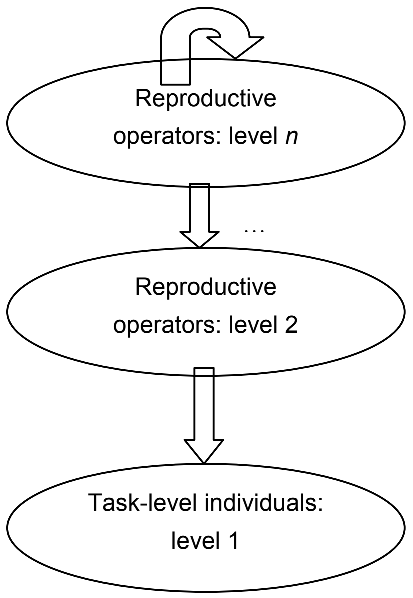

Reproductive operators: level *n*
…
Reproductive operators: level 2
Task-level individuals: level 1

Figure 1. General organization of a meta-evolutionary architecture ([6]).

This general framework was defined and proved successful for a set of specific problems in genetic programming [6, 13]. While individual results indicate possibilities of successes and failures (e.g., [6]), the emerging global picture suggests that, at least for some classes of problems in evolutionary computation, a meta-evolutionary approach has significant advantages over its more traditional counterparts. If this is the case, then a general theoretical analysis and an understanding of this phenomenon at a meta-level would be highly desirable from both theoretical and practical points of view. The present work addresses one of the key problems of this approach: the rational definition of fitness at a meta-level.

2. ANALYSIS OF AN ABSTRACT CASE

We start by considering a simple, abstract evolutionary scheme based on a fixed-size population of N individuals (first level), assuming that their fitness is given. We shall not assume a detailed knowledge of the entire fitness landscape: its smoothness, the top value, topology, etc. Our assumptions about the first-level evolutionary algorithm are the following. During each reproductive cycle, a set Y of N offsprings $\{y_i\}$ are produced that replace the entire parent population X of N parents $\{x_i\}$ (non-overlapping strategy). Each offspring y_i is produced by mutation of an individual $x_i \in X$ using a mutation operator g. At the beginning we shall assume that the operator g is defined by a fixed, deterministic function: $y=g(x)$, and there are no other reproductive operators. Let f be any real-valued function defined for every x and y, and $\langle f\rangle$ its population average. One can think of x and y as genotypes, and of $f(x)$ and $f(y)$ as the corresponding phenotypes. Let $w(x)$ be the number of children of x in Y. Then the following calculation leads to a generalized Price's formula [4, 8] for the difference in the average population phenotype between X and Y:

$$\begin{aligned} F(g) &\equiv \langle f(y)\rangle_Y - \langle f(x)\rangle_X \\ &= \langle w(x) f(g(x))\rangle_X - \langle w(x)\rangle_X \langle f(g(x))\rangle_X + \langle f(g(x)) - f(x)\rangle_X \\ &\equiv \mathrm{cov}(w, f') + \langle f' - f\rangle_X , \end{aligned} \tag{1}$$

where $f'(x) = f(g(x))$ is the value of f computed for a child of x, and we used the fact that $\langle w(x)\rangle = 1$ due to the fixed population size N. The covariance term in the right hand side reflects the effect of selection, and the second term characterizes the net individual phenotype alteration by mutation. The formula (1) holds in general, given the above assumptions. Moreover, it can be used in a case when g is stochastic, and when w is the expected rather than actual number of children, as long as the averaging over X provides an efficient averaging over realizations of g and the selection rule.

Now let f in (1) be the individual's fitness, and consider a fitness-proportional selection scheme, in which the range of fitness of X is linearly mapped onto the unit interval of the probability of selection, so that the minimal fitness value maps to zero, and the sum of all resultant probabilities is normalized to one. Then the expected number of children of x is

$$w(x) = \frac{f(x) - \min_X f}{\langle f\rangle_X - \min_X f}, \tag{2}$$

and it follows from (1), (2) that

$$F(g) = \frac{\mathrm{cov}(f', f)}{\langle f\rangle - \min_X f} + \langle f' - f\rangle. \tag{3}$$

Here all averages are taken over the parent population X. The quantity $F(g)$ is the rate of growth of the average population fitness $\langle f\rangle$ due to the action of the operator g. Therefore, if the immediate goal at the first level is to increase the average fitness of the population from X to Y, then (3) can be taken as a measure of fitness of the mutation operator g. This choice makes sense when g itself is subject to evolution, the goal of which is to optimize the evolution of X. Indeed, it is easy to check that when there is a population of mutation operators $G = \{g\}$, all of which with equal probability are used for mutation of X, then

$$\langle f(y)\rangle_Y - \langle f(x)\rangle_X = \langle F(g)\rangle_G ,$$

where $F(g)$ is given by (3). This result extends to a multi-level meta-evolutionary scheme. It is easy to check that when the rules of evolution of both populations X and G are the same, including the selection rule (2) with F substituted for f at the second level, then a similar to (3) formula (with F substituted for f) will provide a definition of fitness at the third level that is consistent with the objective at the first level, and so on. In the present work, however, we shall limit our consideration to a two-level scheme. Therefore, we do not make any restricting assumptions about the rules of evolution at the second level.

The idea that (3) is a good fitness definition for the second-level evolutionary algorithm implies one significant assumption that concerns the action of operators on the given landscape f. Specifically, we assume that the landscape f with respect to each operator g is "good" in the sense that all relevant statistical properties derived for $f(x)$ and $f(g(x))$, $x\in X$, remain robust with respect to the evolution of X, at least during a minimal number of generations required for relaxation of G to its optimum for the current X. In this sense, we assume that the evolution of G is 'fast' as compared to the evolution of X. This assumption implies certain restrictions on the landscape, as well as on the population dynamics (e.g., the relative frequencies of X and G updates). When this assumption holds for a meta-evolutionary scheme outlined above, then the fitness definition (3) can be expected to produce in general better results than its alternatives.

In contrast, in the literature (e.g., [6]), the fitness $F(g)$ of evolving (meta)operators is frequently taken in a different form, usually taking into account only the second term in the r.h.s. of (3). Therefore, as an example of an alternative to (3), we consider a "naïve" fitness definition that disregards the covariance term:

$$F_{naive}(g) = \langle f' - f\rangle_X , \tag{4}$$

where again $f'(x) = f(g(x))$. An interesting question is: how critical is the difference between (3) and (4) from a practical point of view? To address this question, we conduct a numerical study of the meta-evolutionary scheme defined above, using a two-dimensional optimization problem.

3. EXAMPLE I: 2-D OPTIMIZATION

3.1 Model and Its Implementation

In this relatively simple case study we used two populations $A = \{a\}$ and $B = \{b\}$ of 2-vectors (also viewed as complex numbers), with $N = 60$ individuals in each population. The first-level fitness was defined by the following function f of a complex argument z:

$$f(z) = \theta(|w| - |z|)\left[1 - 20\left|\frac{z-w}{z+w}\right| - \frac{|z+w|}{2}\right]_+, \tag{5}$$
$$\log_{2i/3} w = (\log_{2i/3} z)^*, \quad [x]_+ = x\theta(x),$$

where θ is the Heaviside theta function. This f defined by (5) is zero everywhere except a narrow strip that extends from 0 to 1 on the complex plane and is bounded by a logarithmic spiral. In a cross-section of this strip, the plot of f has a triangular profile with one vertical edge. This choice of f is motivated by the suitability of its features for a meta-evolutionary approach (due to its approximate scaling symmetry with shape parameters slowly changing along the spiral) and by the fact that most conventional optimization algorithms fail on this landscape (including all algorithms implemented in the optimization toolbox in Matlab6).

The second-level individuals $\{b\}$ are the mutation operators that act via addition on the first-level individuals $\{a\}$. The mutation of b's themselves was done via their multiplication by normally distributed complex numbers c that were sampled anew every time when they were used. This approach produced better results than additive Gaussian mutations. In summary, the updating rules were the following (primed entities belong to the next generation):

$$a'_j = a_j + b'_k, \quad b'_k = b_k c, \quad c = x + iy, \tag{6}$$
$$\langle x \rangle = 1, \quad \langle y \rangle = 0, \quad \langle x^2 \rangle = \langle y^2 \rangle = 0.01^2$$

The selection of parents a_j and b_k was fitness-proportional, given by (2), and the parent fitness at the second level was taken either in the form of Price's formula (3) or the 'naïve' formula (4): the latter was used in control runs. In both cases, the fitness of every $b \in B$ was computed by averaging over the entire population A. The selection of b'_k as a mutation operator for the first level was done with the uniform probability. At the end of each update cycle at each level, the N children replaced the entire parent population (non-overlapping scheme; see however below).

It follows from the general analysis in the previous section that the relaxation of the second level should be fast enough as compared to the first level, in order for the second level to be able to follow changes in the population at the first level. Therefore, two second-level updates were performed per each first-level update (we found this choice performing better than one-to-one ratio of the update frequencies).

The initial distributions of A and B were taken as symmetric Gaussians, with A centered at $a_0 = (2i/3)^{0.1}$ (marked by the star in Figure 3 A). B was centered at zero in one set of runs and at $b_0 = 0.01i(2i/3)^{0.1}$ in all the rest of runs. The standard deviation of the initial Gaussian distribution was set to 0.01 for A, 0.005 for B centered at zero, and 0.001 for B centered at b_0.

After initialization, the algorithm proceeds with updates according to (6), with two second-level updates per one first-level update, and is terminated when all a's have zero fitness (this event is considered as an escape). The best fitness achieved in the run is counted, together with the duration of the run.

3.2 Results

A substantial fraction of runs escaped very soon after start, typically within the first ten generations. Therefore, the escape probability was compared for the two versions of the algorithm (Price and 'naïve') on 100 runs for each version with B initially centered at zero and on 60 runs for Price and 100 runs for naïve formula with B centered at b_0. The results are given in Table 1.

Table 1. Percentage of successful starts in the four sets of runs

Total runs	B center	Price	'Naïve'	Wilcoxon *P*
100+100	0	0.75	0.30	2e-10
60+100	b_0	0.82	0.38	9e-8

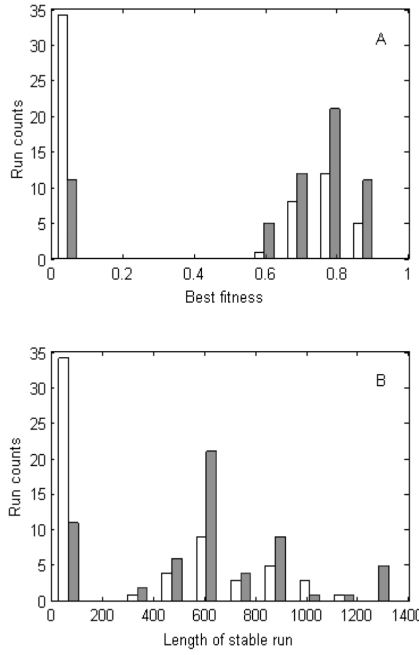


Figure 2. Statistics of 60 runs: Price (solid gray) vs. 'naïve' (clear) fitness formula used at the meta-level. **A:** Histogram of the best fitness achieved in a run. **B:** Histogram of the stable duration of a run measured in second-level generations. See also Table 2.

Continued comparative testing of the performance of the two versions (60 runs each) resulted in the histograms (Figure 2) and data (Table 2) showing that, as expected from the theoretical analysis, overall performance of Price's formula is significantly better than overall performance of the 'naïve' formula. At the same time, there is no significant difference between top ten runs of each version (Table 2), as well as between the sets of runs with successful starts. On the other hand, in this case none of the runs with successful starts provided an acceptable solution for the optimization problem (the maximal fitness was 0.915 achieved with Price's formula after 1346 updates, while the absolute maximum of the landscape is 1.0). Could it be that the significant advantage of Price's formula is limited to the initial phase, when a diffuse Gaussian cloud must focus on a narrow feature of the landscape?

Table 2. Overall comparison, original algorithm

All 60+60 runs	Price	'Naïve'	P-value
Mean best fitness	0.63	0.34	0.00005
Mean run duration	598	306	0.00002
Convergence rate x10^3	2.51	3.45	0.8
% Successful starts	0.82	0.43	0.00002
Top ten runs:			
Mean best fitness	0.86	0.83	0.08
Mean duration	746	740	0.5
Convergence rate x10^3	1.11	1.10	0.6

In order to address this question, tuning of the parameters of the algorithm was performed. The modification consisted in an incrased ratio of the update frequencies: six updates at the second level to one update at the first level. In addition, elitism was used at the second level that lasted during each first-level generation (i.e., in each update cycle limited to the second level only, best representatives were selected among the parent and the offspring populations). Results (Table 3) are similar to the previous set of results (Table 2), with two essential differences: (a) the average best fitness in top ten runs increased from 0.8 to 0.97, and (b) a phenomenon of spontaneous collapse of the populations to one point at both levels was observed, which prevented further convergence. The rate of collapse was significantly greater when 'naïve' formula was used (collapse was observed with Price's formula too, only with different values of parameters).

Table 3. Overall comparison, modified algorithm

All runs	Price	'Naïve'	P-value
Number of runs	19	42	-
Mean best fitness	0.85	0.36	0.001
Mean run duration	355	220	0.008
Convergence rate x10^3	3.13	8.74	0.6
% Successful starts	0.89	0.36	0.0001
Collapse probability, given a successful start	0.00	0.27	0.025
Top ten runs:			
Mean best fitness	0.97	0.97	0.45
Mean duration	416	581	0.47
Convergence rate x10^3	2.44	2.16	0.45

One further modification of the algorithm included variable ratio of the update rates, when the decision was made based on the second-level fitness. While the result was a significant advantage of Price's formula over the naïve formula, the comparison is difficult, because in this case the algorithm essentially depends on the definition of the second-level fitness. Needless to say that the same evolutionary scheme with its meta-level 'frozen' as the initially given Gaussian cloud is not capable of solving the given problem (typically its best-so-far fitness does not get above 0.4).

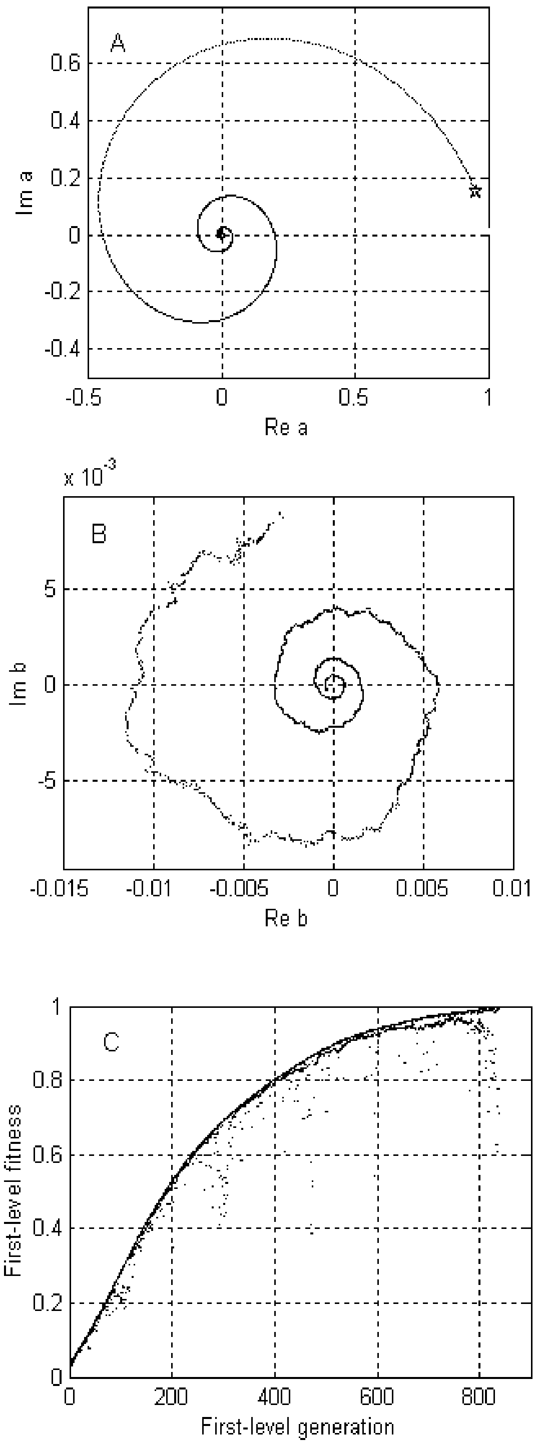


Figure 3. An example of a run with Price's formula solving the 2-D optimization problem. A: Trajectory of the first-level population following the spiral landscape. B: Trajectory of the second-level population: local adjustment of the direction and the size of mutation at the first level. C: Dots representing best-so-far and average fitness for each first-level generation (the best fitness for this run is 0.995, the global maximum is 1.0). A slightly modified algorithm was used in this case (see text).

The considered example is not unique among 2-D optimization problems that can be solved better with meta-evolution, and (we expect), in particular, with a Price-theorem-based meta-evolution. A possible alternative example may involve a quasi-periodic landscape, when the task for a meta-level at the initial stage would be to capture the period. Intuitively, a Price-based approach can be expected to have advantages in this case over a 'naïve' approach, in analogy with the case studied above.

4. EXAMPLE II: THE EIGHT PUZZLE

The Eight-Puzzle is traditionally used as a test bed for search algorithms. The puzzle can be solved exactly by mapping its entire state space [9]. Effective practical solutions are based on heuristics. Here this classical problem is used to test the generality of the above conclusions. The objective in evolution is to evolve a sequence of moves (script) that solves the puzzle. While it is not possible in this case to use exactly the above strategy (because most random mutations result in destruction of good results), the general ideology of the above approach still can be applied here in a more abstract sense.

4.1 Method

The initial configuration of the puzzle was one and the same for all experiments presented here (Figure 4 A). It was obtained by performing 1000 randomly generated moves, starting with the goal configuration (Figure 4 B). The best solution found automatically during the 100 runs (see below) consists of 24 moves.

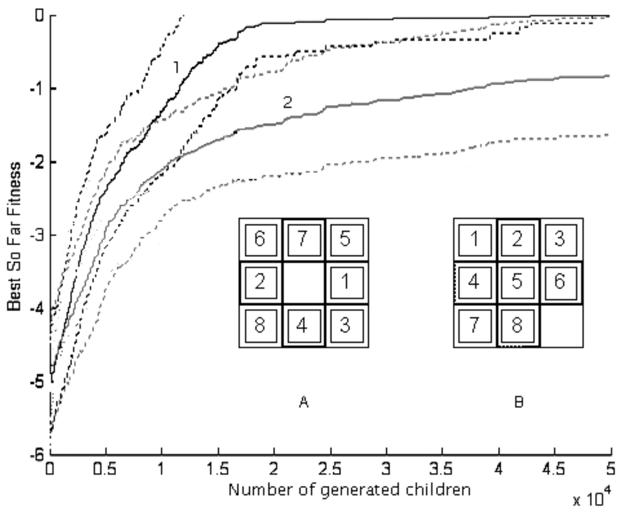
First-level individuals are defined as sequences of moves (scripts) that can be applied to the initial configuration. Each script is represented as a string of characters that encode elementary moves ('U', 'D', 'L', and 'R'). All scripts are legal as long as they produce trajectories confined within the square and have no kinks (i.e., 'U' followed by 'D', or 'L' followed by 'R', or vice versa: this syntactic constraint, although not necessary, is introduced in order to speed up the solution process by limiting the search space).

Fitness of a script is defined as the negative Hamming distance from the goal, plus a linear penalty for the length of the script. More precisely,

$$Fitness \quad = \quad -\begin{pmatrix} number\ of\ tiles \\ in\ wrong\ positions \end{pmatrix} - 0.0001 \cdot \begin{pmatrix} length\ of \\ the\ script \end{pmatrix}. \tag{7}$$

Here the "correct position" of a tile is its position in the goal configuration, and the number of "wrong" positions is counted after the script has been applied to the initial configuration. Therefore, fitness of a script of a sensible length that solves the puzzle must be close to zero.

Elementary modifications of scripts include insertion, attachment (appending at the end), and deletion of elementary moves. These elementary mutations are taken as building blocks for more complex reproductive operators (schemas). Like scripts, schemas are represented as sequences of moves. Three kinds of schemas are used in this study: (i) insertion schemas, (ii) attachment schemas, and (iii) deletion schemas. Insertion of a schema sequence is performed at a uniformly, randomly selected site in the script. Attachment is made always at the end of the script. In case of a deletion schema, a subsequence found in the script that exactly matches the schema is extracted from the script. If there are several matches, then one is selected randomly (again, with a uniform distribution). After a schema was applied to a script, all kinks are automatically removed from the result. A schema fails if there is no match (for deletions), or the resultant script is illegal (after removal of kinks).



. **Figure 4.** Solving the Eight Puzzle. The objective is to transform the initial configuration (A) into the goal configuration (B). The best out of 100 runs solution consists of 24 moves. 1: Main algorithm, the mean of 100 runs (solid line) plus-minus the standard deviation (dashed). 2: Control algorithm (fitness at meta-level is ignored). Again, solid line represents the mean over 100 runs, dashed lines show the standard deviation.

Available schemas are treated as the second-level population of individuals that co-evolve together with the population of scripts. New schemas are generated by applying schemas to schemas. The kind of the child schema (insertion, attachment or deletion) is inherited from the first parent (i.e., the one to which the second parent was applied as to a script).

Unlike in the previous example, in this case a new schema is accepted only when it succeeds in improving fitness of at least one script. Therefore, the derivation of a Price's formula (3) is not valid in this case. The 'naïve' formula (4) still makes sense and can be evaluated even based on one successful mutation, although (4) is already partially "taken into account" by the selection method (a new schema must improve at least one script). It is not practically feasible in this case to perform any fitness averaging over the entire population of available scripts in order to evaluate fitness of a schema (e.g., a sort of averaging is required for evaluation of the covariance term).

The idea used in this part of the study is to 'emulate' the Pricean correction to fitness, i.e., the addition of the covariance term to (4), by replacing (4) with its inverse. The rationale for doing this is that, the higher is the change in the individual fitness of a script due to mutation, the lower must be the covariance of the two values of fitness (before and after the mutation), because of the limited range of fitness. Therefore, here fitness F of a schema is defined in terms of its effect on the fitness of a script as follows:

$$F = \left\langle \frac{\theta(f' - f)}{f' - f} \right\rangle \quad (8)$$

where *f* and *f'* is fitness of a script before and after the mutation, and the average is taken over all explored successful mutations. The algorithm at both levels involved a tournament selection among children. In a control experiment, fitness of a schema was ignored (random tournament outcome).

In brief, the main loop of the algorithm was the following.

1. Randomly select a script and a schema.
2. Apply the schema to the script, producing a child script, or fail.
3. Eliminate kinks in the child, if any.
4. If the child is already in the population of scripts, fail.
5. Confirm that the child is a legal script, or fail.
6. If the child is fitter than its parent script, update fitness of the schema.
7. Compute best-so-far fitness in the population of scripts.
8. Randomly select an opponent among the population of scripts.
9. If the child is fitter than the opponent, replace the opponent with the child.
10. Randomly select two schemas.
11. Apply one to another, producing a child schema, or fail.
12. Remove all kinks in the child, if any.
13. Confirm novelty of the child, or fail.
14. Estimate fitness of the child by trying it on up to 20 randomly selected scripts (until one of them is improved), or fail.
15. Randomly select an opponent among schemas.
16. Replace the opponent by the child, iff the child is fitter.
17. Stop, if solution found.

Again, the control algorithm was different in that fitness of schemas was not estimated. In this case, instead of the tournament, the child schema was tried for its applicability on up to 20 randomly selected scripts, and if passed, it still could be rejected with a fixed probability.

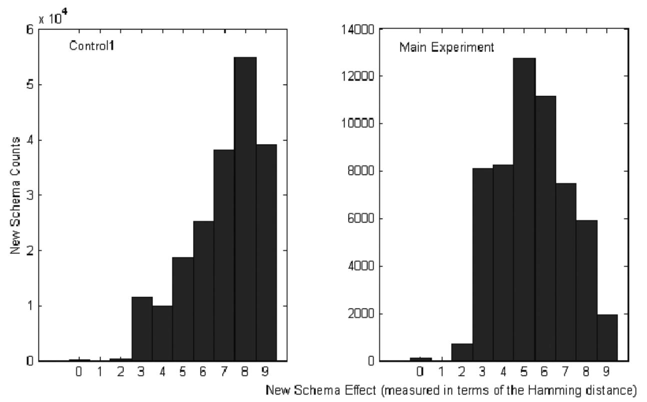


Figure 5. Histograms of the schema effect produced on the board configuration (measured as the Hamming distance) in two experiments represented in Figure 4.

4.2 Results

Results of the experiment and its control are represented in Figures 4, 5. All 100 runs of the main algorithm converged to a solution. Most (almost all) of 100 runs of the control algorithm did not. The nature of the difference between main algorithm and the control could be understood with the help of Figure 5 showing the hisograms of a measure of schema quality defined as the Hamming distance between the board configurations before and after the schema was applied (as a script). It is noticeable that a larger fraction of schemas generated during the main experiment, as compared to the control, produce smaller alterations of the board, and therefore, result on average in smaller changes of the fitness before and after mutation of a first-level individual, consistently with the objective function (8), and therefore, in the sense of an average over all possible scripts, those schemas result in a higher covariance of fitness before and after mutation. In conclusion, emulating the Pricean fitness (4) by (8) at a meta-level results in achieving a solution of the first-level problem, and may be vital for achieving a solution in this case.

The method used here can be further improved in several ways to solve the bloat problem observed in both numerical experiments. Specifically, not only kinks but also loops in the space of states can be eliminated automatically. The Manhattan distance can be used to define fitness at the first level instead of the Hamming distance. Finally, a better approximation of Price's formula specifically derived for this task can be used in the algorithm. Parameters of the meta-evolutionary algorithm used in this case can be optimized for better performance. These apparent "drawbacks", however, may not compromise the presented results, as the goal was to compare two general approaches on a testbed that is not specifically designed to achieve higher performance.

5. EXAMPLE III: BINARY DENDRITIC TREE RECONSTRUCTION

As the third example, we consider a simplified version of a problem that emerges as a subtask in automated reconstruction of neuronal dendritic trees from their microphotographs [14, 15]. One particular difficulty here is the ambiguity of apparent intersections of dendritic branches. Disregarding many details of the real dendritic reconstruction, here we simply assume the following. Each skeletonized two-dimensional projection of a dendritic tree is given as a finite, non-oriented, planar graph that may only include real branching points and false intersections. Two examples are shown in Figure 6 A. The specific goal is to reduce a given graph to a binary tree by resolving intersections using a selected schema[1]. Elementary schemas that may be used to resolve intersections are depicted in Figure 6 B, other schemas may be constructed from these by adding conditions of matching (i.e. by mutating these schemas: see below). The goal of graph reduction is considered achieved when the graph has no unresolved intersections and is simply connected. An attempt fails when there are no matching sites for the selected schema, and the graph is not reduced to a binary tree. The task is to evolve a

[1] Here by a *schema* we mean an abstract template that can be used to match and to alter task-level graphs as well as other schemas. This notion comes from a more general notion of a schema introduced in [16-18] and should not be confused with other usage of the same term.

schema that successfully reduces any of a given set of graphs to a binary tree.

The process of evolution starts with the population of individuals – action schemas depicted in Figure 6 B plus mutation schemas. The fitness of an action schema is computed as the success rate in reducing graphs to binary trees. Each attempt of reduction is performed with only one schema that is repeatedly applied to randomly selected sites where it matches the structure of the graph. Reproductive operators are mutation schemas that modify the original set by adding further conditions for matching. We consider two kinds of mutations (Figure 6 C, D). One of them adds a condition (dotted line in Figure 6 C) that there is a continuous path in the tree (which is to be reconstructed) that connects two terminals of the schema when they are bound to the graph. Another mutation adds a condition that there is no such path (crossed dotted line in Figure 6 D).

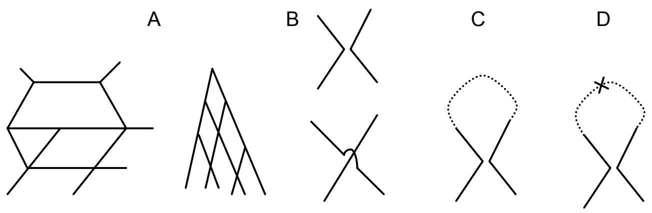


Figure 6. Problem space of the task of binary dendritic tree reconstruction from its skeletonized planar projection. A: sample tree projections; B: elementary schemas resolving intersections; C, D: mutated schemas.

The first mutation is useful, because it produces a schema (Figure 6 C) that successfully reduces the graphs in Figure 6 A and other similar graphs to binary trees with the rate 100%. In contrast, the result of the second mutation (Figure 6 D) does not guarantee a success: e.g., for the first graph in Figure 6 A its success rate is 6.9%.

Our preliminary numerical analysis based on this setup and a population of 6 action schemas shows that in order to select the right mutation (Figure 6 C over Figure 6 D), the mutation schema fitness should be evaluated based on (3) rather than based on (4). E.g., the value of (4) may not discriminate between the two mutation schemas, while the value of (3) would favor the first mutation. A more extensive study of this example is still ahead. While at this point we cannot make a general conclusion, the preliminary numerical result indicates that at least in some cases the definition (4) may be useful, if not vital, for this class of problems.

6. CONCLUSIONS

Results presented here support the general idea that meta-evolution understood as co-evolution of reproductive operators may improve the performance of an evolutionary algorithm, if the fitness is appropriately defined at the meta-level(s). A general approach in designing an efficient fitness definition for operators can be successful when it is based on facts of the highest generality applicable to evolution, one of which is the Price's theorem. Therefore, main steps in designing and using a meta-evolutionary algorithm can be the following.

(a) Define the space of main individuals, their representation, and their fitness.
(b) Define the space of reproductive schemas, their representation, and their fitness based on Price's formula.
(c) Continue building up meta-levels of reproductive schemas with Price's fitness for them, as long as this helps to improve the performance of the method.

In this work, trying to develop a general theoretical understanding of this concept, Price's theorem was used to define fitness at a meta-level and to show with three simple case studies that the ideology based on Price's theorem can work at a meta-level in a similar way for very different problems. Specifically, it is found that Pricean definition of fitness for reproductive operators matters for performance and stability of meta-evolutionary algorithms.

7. REFERENCES

[1] Back, T. and Hoffmeister, F. Basic Aspects of Evolution Strategies. *Statistics and Computing* 4 (2): 51-63, 1994.

[2] Bedau, M. A. and Packard, N. H. Evolution of evolvability via adaptation of mutation rates. *Biosystems* 69 (2-3): 143-162, 2003.

[3] De Jong, K. A. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ph.D. Thesis, University of Michigan, 1975.

[4] Frank, S. A. The Price equation, Fisher's fundamental theorem, and causal analysis. *Evolution* 51 (6): 1712-1729, 1997.

[5] Freisleben, B. Metaevolutionary approaches. In: *Handbook of Evolutionary Computation*, pp. C7.2:1-8. IOP Publishing Ltd and Oxford UP, 1997.

[6] Kantschik, W., Dittrich, P., Brameier, M., and Banzhaf, W. Meta-evolution in graph GP. In: Poli, R., Nordin, P., Langdon. W. B., and Fogarty, T. C. (Eds.). *Genetic Programming, Second European Workshop, Proceedings of EuroGP'99*, pp. 15-28. Berlin: Springer-Verlag, 1998.

[7] Koza, J. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: MIT Press, 1994/1998.

[8] Price, G. R. Selection and covariance. *Nature* 227: 520-521, 1970.

[9] Reinefeld, A. Complete solution of the eight-puzzle and the benefit of node ordering in IDA*. In Bajcsy, R. (Ed.). *IJCAI-93: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.* Vol. 1, pp. 248-253. Morgan Kaufmann: San Mateo, CA, 1993.

[10] Shinozawa, Y. and S. Okoma. The postprocessing of character recognition by genetic algorithms. *Transactions of*

the Information Processing Society of Japan 40 (3): 1106-1116, 1999.

[11] Schwefel, H.-P. *Evolutionsstrategie und Numerische Optimierung*. Dissertation, Technische Universität Berlin, 1975.

[12] Kantschik, W., and Banzhaf, W. Linear-graph GP: a new GP structure. In Foster, J. A., Lutton, E., Miller, J., Ryan, C., and Tettamanzi, A. G. B. (Eds.). *Genetic Programming: 5th European Conference, Proceedings of EuroGP 2002,* pp. 83-92. Berlin: Springer-Verlag, 2002.

[13] Kantschik, W., Dittrich, P., Brameier, M. and Banzhaf, W. Empirical analysis of different levels of meta-evolution. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99,* Washington, DC. Vol. 3, pp. 2086-2093. Piscataway, NJ: IEEE, 1999.

[14] Schmitt, S., Evers, J. F., Duch, C., Scholz, M. and Obermayer, K. New methods for the computer-assisted 3-D reconstruction of neurons from confocal image stacks. *NeuroImage* 23: 1283-1298, 2004.

[15] Evers, J. F., Schmitt, S., Sibila, M. and Duch, C. Progress in functional neuroanatomy: precise automatic geometric reconstruction of neuronal morphology from confocal image stacks. *Journal of Neurophysiology* 93: 2331–2342, 2005.

[16] Samsonovich, A. V. and De Jong, K. A. Meta-cognitive architecture for team agents. In Alterman, R., and Kirsh, D. (Eds.). *Proceedings of the 25th Annual Meeting of the Cognitive Science Society*, pp. 1029-1034. Boston, MA: Cognitive Science Society, 2003.

[17] Samsonovich, A. V. and De Jong, K. A. A general-purpose computational model of the conscious mind. In K. Forbus, D. Gentner, and T. Reigier (Eds.). *Proceedings of the Twenty-Sixth Annual Conference of the Cognitive Science Society,* pp. 382-383. Mahwah, NJ: Lawrence Erlbaum, 2004.

[18] Samsonovich, A. V. and Nadel, L. Fundamental principles and mechanisms of the conscious self. *Cortex* 42 (5), 2005, in press.