Three Ways to Grow Designs:
A Comparison of Embryogenies for an Evolutionary Design Problem

Peter Bentley **Sanjeev Kumar**
Department of Computer Science, University College London
Gower Street, London WC1E 6BT, UK
P.Bentley@cs.ucl.ac.uk, S.Kumar@cs.ucl.ac.uk

Abstract

This paper explores the use of growth processes, or embryogenies, to map genotypes to phenotypes within evolutionary systems. Following a summary of the significant features of embryogenies, the three main types of embryogenies in Evolutionary Computation are then identified and explained: external, explicit and implicit. An experimental comparison between these three different embryogenies and an evolutionary algorithm with no embryogeny is performed. The problem set to the four evolutionary systems is to evolve tessellating tiles. In order to assess the scalability of the embryogenies, the problem is increased in difficulty by enlarging the size of tiles to be evolved. The results are surprising, with the implicit embryogeny outperforming all other techniques by showing no significant increase in the size of the genotypes or decrease in accuracy of evolution as the scale of the problem is increased.

1 INTRODUCTION

The use of computers to evolve solutions to problems has seen a dramatic increase in popularity and success over the last decade. This is especially true for design problems, with designs of great variety now being evolved by computers (Bentley, 1999). However, as more difficult design problems are set, the complexity of the corresponding solutions is increasing. Unless great care is taken in the representation of the solutions, this increase in complexity can cause significant problems for evolution, with phenomena such as disruption of inheritance and premature convergence to local minima often preventing the generation of fit solutions altogether.

Yet natural evolution is no stranger to complexity. Life is clearly the most complex of all designs to have evolved, making our very best evolved designs look absurd in their simplicity. How does nature overcome the problems of evolving such intricate solutions?

The answer lies in the evolved embryogeny of living creatures. Unlike most evolutionary algorithms in computer science, nature does not use a one-to-one mapping from genotypes to phenotypes. Multicellular organisms are, of course, *grown* using the instructions provided in their genetic material. Living creatures grow as a result of a carefully choreographed dance of genes, amino acids, proteins, cells, chemical signals, electrical signals and movements. The unimaginable convolutions of interaction between all parts of the growing creature permit a small amount of information to generate immensely more complex forms. Because it implicitly encompasses concepts such as compression, iteration, recursion, adaptation and memory, a natural embryogeny is an inherently scalable process. Once evolved in nature, this growth process opened up the door to the evolution of every creature from fruit fly to blue whale.

Evolutionary computationists are now beginning to explore the new possibilities that embryogenies can provide. This paper first explains the concepts behind embryogenies, and then explores the three main types of embryogeny in use today, by comparing an implementation of each with an evolutionary algorithm which uses no growth process. A simple design problem is used to investigate the scalability of each of the types of embryogeny, with some unexpected and significant results.

2 EMBRYOGENY EXPLAINED

An embryogeny is the process of growth that defines how a genotype is mapped onto a phenotype. (It should be noted that the correct term is *embryogeny*, which refers to the process, rather than the oft-misused term *embryology*, which refers to the science of studying embryos and embryogenies.) Out of the four main types of evolutionary algorithm (EA) - the genetic algorithm (GA), evolutionary programming (EP), evolutionary strategies (ES), and genetic programming (GP) - only the GA treats genotypes separately from phenotypes, and so only the GA has made use of a mapping stage from the beginning of its development. Nevertheless, a mapping stage from genotypes (evolved parameter values) to phenotypes (solutions to problems) can be introduced into any of the evolutionary algorithms without difficulty.

Although a simple one-to-one mapping from genes to parameters could be regarded as an elementary form of embryogeny, this does not, strictly speaking, incorporate any of the ‘growth’ processes or polygeny with which embryogenies are associated. In other words, an embryogeny is a special kind of mapping process. It has the following features:

- **Indirect correspondence between alleles and phenotypic effects**. The genotype is now regarded a set of ‘growing instructions’ – a recipe which defines how the phenotype will develop.
- **Polygeny**. Phenotypic traits are produced by multiple genes acting in combination.

An embryogeny can provide the following benefits (Bentley, 1999):

- **Reduction of search space.** Embryogeny permits highly compact genotypes to define phenotypes. This compression (often recursive, hierarchical and multifunctional) results in genotypes with fewer parameters than their corresponding phenotypes, causing a reduction in the dimensionality of the search space, and hence a smaller search space for the EA.
- **Better enumeration of search space.** Mapping permits two very differently organised spaces to coexist, i.e. a search space designed to be easily searched can allow the EA to locate corresponding solutions within a hard-to-search solution space.
- **More complex solutions in solution space.** By using ‘growing instructions’ within genotypes to define how phenotypes should be generated, a genotype can define highly complex phenotypes.
- **Repetition.** Properly designed embryogenies can improve the ability of evolution to generate solutions with repeating structures such as symmetry, segmentation, and subroutines.
- **Adaptation.** It is possible to 'grow' phenotypes from genotypes adaptively, allowing constraints to be satisfied (Yu & Bentley, 1998), improvement to variable conditions, and correction of malfunctions in designs (Sipper, 1997).

Embryogenies also suffer from some drawbacks (Bentley, 1999):

- **Can be hard to design.** All types of embryogeny require careful design, and to date, only those researchers capable of performing this difficult art have demonstrated successful results.
- **Can be hard to evolve.** If care is not taken, embryogenies can introduce problems for evolutionary algorithms. Bloat, pleiotropy and disruption of child solutions is possible, resulting in the need for carefully designed genetic operators.

In nature, embryogenies are defined by the interactions between genes, their phenotypic effects and the environment in which the embryo develops. In EAs, we can define embryogenies in three main ways: *externally*, *explicitly*, and *implicitly* (Bentley, 1999).

2.1 EXTERNAL (NON-EVOLVED) EMBRYOGENIES

Embryogenies are, in a very real sense, complex designs in their own right. Most embryogenies are hand-designed and are defined globally and externally to genotypes. For example, Evolutionary Art systems often use embryogenies defined by fixed, non-evolveable structures which specify how phenotypes should be constructed using the genes in the genotypes (Bentley, 1998). The advantage with such external embryogenies is that the user retains more control of the final evolved forms, and can potentially improve the quality of evolved designs by careful embryogeny design. In addition, this type of embryogeny produces the fewest harmful effects for evolution, and requires no specialised genetic operators. The disadvantage of this approach is that these embryogenies are not evolved, so they remain static and unchanging during the evolution of genotypes. This does not necessarily imply that the evolved designs will be any less fit, but it does mean that the designer of the embryogeny must take care to ensure that this complex mapping process will always perform the desired function.

William Latham’s evolved art (Todd & Latham, 1992) provides an excellent example of an external embryogeny in use. Latham hand-designs embryogenies comprising tori, 'ribs' and other shapes and how they are placed and duplicated. For example, a simple embryogeny might be:

```
{ Spiral ( radius, curliness, depth )
  Rotate&Duplicate ( angle, duplicates ) }
```

with the values *radius, curliness, depth, angle, duplicates* being evolved in the separate genotypes, to generate different forms. Alternatively, Bentley’s generic evolutionary design system (Bentley and Wakefield, 1997) illustrates an adaptive external embryogeny process. Designs evolving in the system have various constraints satisfied by mapping illegal genotypes to legal phenotypes. In addition, the external embryogeny applies high-level functions such as reflection, to allow symmetrical designs to be generated.

2.2 EXPLICIT (EVOLVED) EMBRYOGENIES

If each step of an embryogeny is explicitly specified in a data structure, the embryogeny resembles a computer program. Designs are 'grown' by following the instructions in this program, and these instructions may contain conditional statements, iteration, and even subroutines.

Although it is possible to hand-design such 'programs', Genetic Programming allows us to evolve them. Typically, the genotype and embryogeny are combined, allowing the evolution of both simultaneously. Clearly, this approach avoids the need to hand-design embryogenies, and allows the emergence of adaptive mapping from genotype to phenotype (i.e., different initial conditions acting on conditional statements could trigger the growth of different phenotypes). There are some disadvantages, however. The creation of suitable representations can be difficult. Successfully evolving such representations can also be difficult (often specialised genetic operators are required to ensure disruption is minimised (Koza et al, 1999)).

Nevertheless, this type of embryogeny is used by many researchers. Koza and his team have made an extensive study of the evolution of analogue electronic circuits using Cellular Encoding as their explicit embryogeny (Koza et al, 1999). Coates (1997) evolves architectural forms using a Lindenmayer system as his embryogeny. Gero and Rosenman (1999) use a number of different types of grammar-based explicit embryogenies for their work on evolving architecture.

2.3 IMPLICIT (EVOLVED) EMBRYOGENIES

Natural evolution does not use externally defined embryogenies, nor does it explicitly represent embryogenies in our genes. Instead, natural evolution uses highly indirect chains of interacting 'rules' to generate complex embryogenies, which result in the development of living creatures. The flow of activation is not completely predetermined and pre-programmed, it is dynamic, parallel and adaptive.

To summarise in very simple terms, natural embryogenies use chemicals surrounding each cell to activate or suppress genes within the chromosomes of the cell, triggering patterns of cellular growth. Cellular death, differentiation, and the production of chemicals is also triggered by genes. Living creatures are grown in wombs or eggs with chemicals carefully placed to guide the early development of the embryo. As embryos develop, complex chains of gene activation occur, cells grow and die to form the appropriate shapes, and cells are differentiated to perform specialised functions. Even the movement of the developing muscles of the embryo affects the development and placement of cells (Slack, 1991).

Few researchers have explored the use of implicit embryogenies for Evolutionary Design, and yet the potential advantages of this approach are significant. Because of the way in which genes can be activated and suppressed many times during the development of phenotypes, because the same genes can be used to specify multiple functions, and because of the inherent parallelism of gene activation, such implicit embryogenies go far beyond today's Genetic Programming. Through emergence during evolution, these implicit embryogenies incorporate all concepts of conditional iteration, subroutines, and parallel processing which must be manually introduced into explicit GP embryogenies.

Implicit embryogenies are types of constrained generating procedures, which, as Holland describes, resemble neural nets, game theory and classifier systems (Holland, 1998). By evolving a set of simple rules which can then be iteratively applied to each element of the growing solution, many large-scale problems can be tackled. For example, Taura and Nagasaka (1999) describe the use of an implicit embryogeny to define patterns of cells on the surface of a sphere, which are then used via a second, external embryogeny to define the morphology of shapes. Alternatively, Mitchell evolves CA-based systems designed to perform computations (Mitchell et al, 1993), that can be regarded as types of implicit embryogeny. However, the best example of this type of approach is the work of Hugo de Garis, who for some years has been successfully evolving CA-based implicit embryogenies to grow artificial neural nets on an immense scale (de Garis, 1994).

3 COMPARING EMBRYOGENIES

Having explained the three main types of embryogeny in use today, the rest of the paper describes the first known experimental comparison between these techniques for a single problem. Four evolutionary systems were employed: a GA with no embryogeny, a GA with an external embryogeny, a GP system with explicit embryogeny, and a GA with an implicit embryogeny. The problem chosen was to evolve the morphology of a two-dimensional tile, which tessellates with itself at each corner.

3.1 EVOLVING TESSELLATING TILES

Although the four evolutionary systems used different genotypes and embryogenies, all used the same phenotype representation and the same fitness function to guide evolution. As shown on the left of figs 1 and 2, phenotypes (evolved tiles) were represented using a simple 2D grid, with each element either filled or empty. To calculate how well a phenotype tessellated with itself, four copies were overlaid onto each corner of the tile, see the right of figs. 1 and 2. For every empty element and for every overlapping element in the central tile, the fitness score was incremented by 1. To illustrate this, fig. 1 shows a tile which does not tessellate properly, and fig. 2 shows a perfectly tessellating tile.

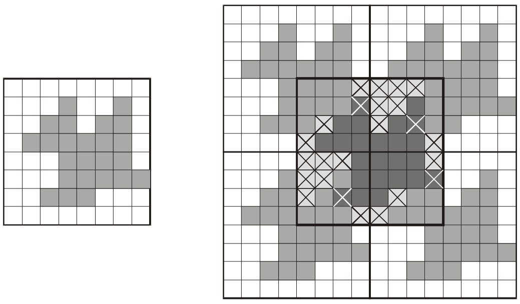


Figure 1: An example of an imperfectly tessellating 8x8 tile, with a fitness of 24

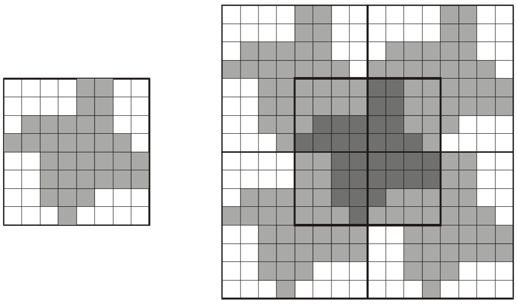


Figure 2: An example of a perfectly tessellating 8x8 tile, with a fitness of 0

Clearly, a tessellating tile can be constructed without search using a simple set of heuristics, but the purpose of this problem was to allow the comparison of embryogenies, and not to create the optimal tile-generating software. The advantage of this problem is that one aspect of its difficulty can be altered very simply. As will be shown, by increasing the size of the phenotype grid from 4x4, to 8x8, to 16x16 elements, the problem can be scaled up. This permits the study of how the genotypes and embryogenies change in efficiency (in terms of memory and fitness), i.e., the scalability of each type of embryogeny can be explored.

3.2 NO EMBRYOGENY

The first evolutionary system acted as the experimental control in the subsequent tests. A simple GA with binary strings as chromosomes, elitism and single-point crossover was used. The GA used a one-to-one mapping from genotype to phenotype, with each bit in the genotype defining whether a corresponding element in the phenotype grid was filled or not, see fig. 3.

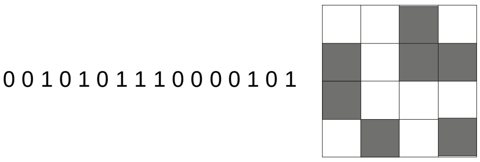


Figure 3: An example genotype without embryogeny and its corresponding 4x4 phenotype.

3.3 EXTERNAL EMBRYOGENY

The second evolutionary system again used a simple GA with elitism and single-point crossover to manipulate genotypes. However, instead of a direct bit-to-element mapping, this system employed an external embryogeny. Genotypes define the co-ordinates of a set of hand-designed shapes in the phenotype grid, with each gene pair corresponding to a predefined shape. Shapes were permitted to overlap with each other and the edges of the grid, see fig. 4.

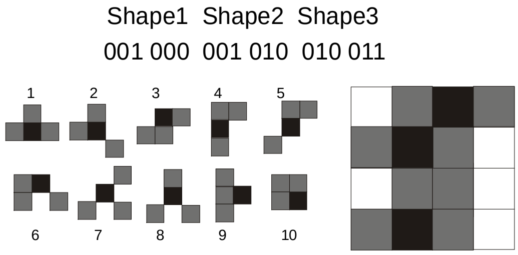


Figure 4: An example external embryogeny defined by a set of evolved co-ordinates for predefined shapes (in the binary genotype), and its corresponding 4x4 phenotype.

In total, ten different shapes, each composed of exactly four cells, were used to construct tiles. It can be calculated that tiles with fitnesses of zero always take up half the number of elements in the phenotype grid. From this, an appropriate number of shapes from the set of ten can be calculated. In the experiments, 4x4 tiles were constructed from three shapes, 8x8 tiles from ten, and 16x16 from 33 shapes (reusing the same ten shapes up to four times).

3.4 EXPLICIT EMBRYOGENY

The third evolutionary system used GP to evolve explicit embryogenies in the form of program trees. Beginning at a seed or zygote cell placed at the centre of the phenotype grid, the embryogeny defines the direction of growth at every point. Four functions were used: LEFT, RIGHT, UP and DOWN, with each node in the tree allowed up to four branches. Paths of growth were permitted to overlap. Fig. 5 shows an example genotype defining the explicit embryogeny.

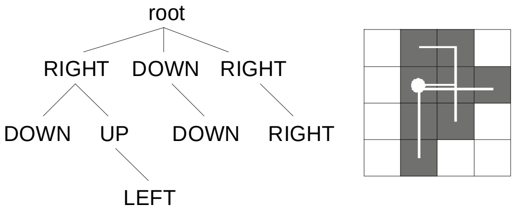


Figure 5: An example explicit embryogeny defined by a tree of nine nodes, and its corresponding 4x4 phenotype.

The GP system used steady-state selection and a crossover designed to minimise disruption by crossing parents at points of similarity in the two trees. Further details of this system and crossover operator can be found in (Mallinson & Bentley, 1999 and Bentley & Wakefield, 1996). As with all GP systems, bloat occurred, so an additional fitness function penalised genotypes with more nodes.

3.5 IMPLICIT EMBRYOGENY

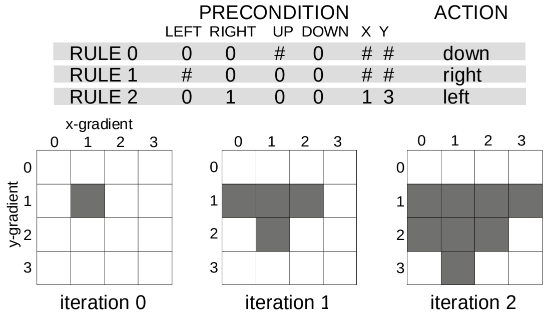
The fourth and final evolutionary system used an advanced GA to evolve implicit embryogenies. Each genotype comprised a variable number of rules (usually between four and eight). Each rule had a precondition and an action. Each precondition had six fields: *LEFT, RIGHT, UP, DOWN, X, Y.* A specific rule can take the following values for each precondition field (where # is *don't care,* 0 is *empty,* 1 is *filled,* **0**,**1**,**2**,**3** are gradient zones):

LEFT	*RIGHT*	*UP*	*DOWN*	*X*	*Y*
0,1,#	0,1,#	0,1,#	0,1,#	**0**,**1**,**2**,**3**,#	**0**,**1**,**2**,**3**,#

For a rule to be fired, values in at least four of the six fields in the precondition must be matched. (This provides the equivalent of disjunction for rule preconditions.) The action of a rule can be: DIE, UPDATE, or grow LEFT, RIGHT, UP or DOWN.

Growth takes place in a phenotype grid, which as usual can be 4x4, 8x8 or 16x16 elements. In order to permit evolution of specialised rules that can provide detail in specific areas of the phenotype, the grid has two 'gradients' - one in the x direction, one in the y direction. In a similar way to the gradients used to provide positional information in eggs and wombs of nature (Slack, 1991, Lawrence, 1995), the gradients divide the grid into 16 zones, regardless of the number of elements in the phenotype grid.

At iteration zero, a seed cell is placed in the centre of the phenotype grid. Following the example of biological cell growth, the rules are then applied for a fixed number of iterations to each *filled* element in the current embryonic phenotype grid. (This is unlike traditional cellular automata, where rules are applied to empty or filled grid elements.) Depending on whether the neighbouring elements of the current element exist or not, and on the strength of the two gradients at that point, the rules may be activated, causing growth or cell death in the phenotype. Rules are applied 'in parallel' so that the results of applying the rules to each filled element only take effect at the end of each iteration step. However, a rule which performs the UPDATE action causes all activated rules in the current iteration to be applied. By prematurely placing cells in the phenotype grid in this way, evolution can increase the number of rules applied in each iteration and provide extra growth where needed. This new type of rule action was added to the embryogeny because during the development of the system, the number of iterations was found to be overly critical. Fig. 6 provides an example of the growth process.



	PRECONDITION						ACTION
	LEFT	RIGHT	UP	DOWN	X	Y	
RULE 0	0	0	#	0	#	#	down
RULE 1	#	0	0	0	#	#	right
RULE 2	0	1	0	0	1	3	left

Figure 6: Example of a three-rule implicit embryogeny and its corresponding phenotype after two iterations.

A steady-state GA with a crossover designed to cope with variable numbers of rules in the genotype was used. Further details of this GA and its operators can be found in (Bentley, 1999, ch. 18).

4 EXPERIMENTS

4.1 OBJECTIVES AND PARAMETERS

There were two main objectives of the experiments: to investigate how the use of different embryogenies affected efficiency of search and to investigate the scalability of the different embryogenies for the tessellating tile problem. To this end, three experiments were performed for each of the four evolutionary systems described above, with the size of the phenotype grid being increased from 4x4 to 8x8 to 16x16 elements. At least twenty runs of each system were performed for all experiments. Population sizes in the systems were 100, each system evolved for up to 100 generations. Initialisation of genotypes was random. The explicit embryogeny system was given random trees with depth no larger than 3 for the first experiment, no larger than 4

Table 1: Results of experiments.

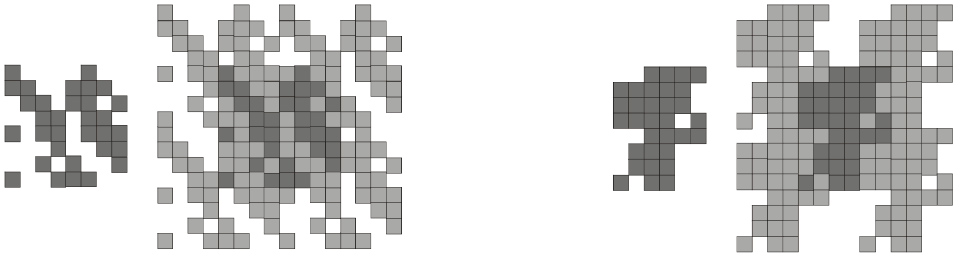
METHOD	4X4 GRID		8X8 GRID		16X16 GRID	
	mean soln. size	**mean fitness**	**mean soln. size**	**mean fitness**	**mean soln. size**	**mean fitness**
No embryogeny	16 bits	0	64 bits	0	256 bits	4.1
External Embryogeny	3 shapes	0	10 shapes	2.4	33 shapes	180.2
Explicit Embryogeny	9.5 nodes	0	64.8 nodes	0.769	845.7 nodes	23.2
Implicit Embryogeny	5.28 rules	0	6.05 rules	0	6.18 rules	0

for the second, and no larger than 5 for the third experiment. The implicit embryogeny system was initialised with between four and seven random rules per genotype. The number of iterations for which rules were applied in the implicit embryogeny was set at 3, 4 and 10 for the first second and third experiments, respectively. (Parameters were chosen to minimise genotype sizes and growth times.) Random crossover was used to generate all offspring, mutation was employed with a probability of approximately 0.001 per bit in each system.

4.2 RESULTS

Table 1 summarises the results from the experiments. For the first experiment, where tiles of 4x4 elements were evolved, all methods found perfect solutions, every run. For the second experiment, where tiles of 8x8 elements were evolved, only the evolutionary systems with no embryogeny and implicit embryogeny found perfect solutions every run. The explicit embryogeny still obtained good fitnesses with a mean of 0.769, with the external embryogeny fairing slightly worse with a mean fitness of 2.4. For the third experiment, where tiles of 16x16 elements were evolved, only the GA using implicit embryogeny was able to evolve perfect solutions every run, in less than 100 generations. Indeed, this method often found perfect solutions in fewer than ten generations. Given more than 100 generations, the system with no embryogeny was able to match this fitness average. However the EAs with external and explicit embryogenies were unable to find perfect solutions for this problem, with the external substantially worse than the explicit.

However, the most astonishing findings came from the change in solution size for the three different tile sizes. Not only did the implicit embryogeny outperform all other approaches in terms of fitness, but this embryogeny also showed no significant change in the evolved length of genotypes for all three experiments. This is in stark contrast to all other approaches which showed dramatic increases in genotype sizes - particularly the explicit embryogeny. It should also be noted that because the GA using implicit embryogeny did not suffer from bloat, there was no additional fitness criteria employed to reduce the number of rules. The lowest number of implicit rules


[A] No Embryogeny

[B] External embryogeny


[C] Explicit embryogeny

[D] Implicit Embryogeny

Figure 7: Examples of perfect tessellating tiles evolved using each embryogeny.

observed to generate perfect tessellating tiles in all experiments was three.

Fig. 7 provides examples of perfect 8x8 tiles evolved by each approach. The GA using no embryogeny generated the most diverse set of ’tiles’, however most were fragmented and full of holes. The GA using external embryogeny also provided diverse solutions, most of which were not very fragmented and contained few holes. The GP system with explicit embryogeny produced tiles which typically were all subtle variations of the ’central diamond’ design. Although diversity was reduced, no tiles were fragmented and all were without holes. The GA using implicit embryogeny evolved designs that were never fragmented or containing holes. However, for these experiments, very little diversity of output was shown, with the system always producing rectangles of one orientation or another. Nevertheless, as is shown in the next section, this lack of diversity is not inherent in the representation, which, with the right rules, can define any possible tile shape.

4.3 ANALYSIS OF RESULTS

The results of the experiments prompt a number of questions that can be addressed. These include: why did each approach favour different types of solution with different diversities of output, why did the approaches differ in their scalability, and how efficient