Modularity, Product Innovation, and Consumer Satisfaction: An Agent-Based Approach

Shu-Heng Chen and Bin-Tzong Chie

AI-ECON Research Center, Department of Economics
National Chengchi University, Taipei, Taiwan 11623
chchen@nccu.edu.tw,g0258501@nccu.edu.tw

Abstract. The importance of modularity in product innovation is analyzed in this paper. Through simulations with an agent-based modular economic model, we examine the significance of the use of a modular structure in new product designs in terms of its impacts upon customer satisfaction and firms' competitiveness. To achieve the above purpose, the automatically defined terminal is proposed and is used to modify the simple genetic programming.

1 Motivation and Introduction

This work is a continuation of [1, 2], which provide an agent-based model that simulates the evolution of product innovation by growing it from the bottom up. The earlier work is not just to provide an agent-based approach, but to introduce a new representation of commodities, production, and preference, via the use of *genetic programming* (GP). However, [1, 2] only consider simple genetic programming [3]. The end result is that in many of their early simulations, only primitive desires are satisfied, and the economy can rarely advance to a mature state where consumers' desires can be met a sophisticated degree. One cause of this problem is that simple GP is not an appropriate tool to work with the idea of *functional modularity* (to be detailed in Sect. 3). This limitation has been long realized by GP researchers, e.g., [4]. In this paper, we remedy this problem by replacing the simple GP with automatically defined terminals (ADTs), which are very similar in spirit to automatically defined functions (ADFs), invented by John Koza [4]. As Koza pointed out, devices of this kind can provide some hierarchical mechanism to exploit modularities inherent in problem environments.

With this modified version of GP, two experiments are carried out. The first experiment examines the contribution of functional modularity to consumers' satisfaction. The second series of experiments then examines the importance of modularity in competition among firms. We simulate an agent-based economy to allow the firm that designs new products using a modular structure to compete with the firm that does not. In a sense, this is equivalent to replicating the well-known story regarding the competition between Hora and Tempus, two imaginary watchmakers offered by Herbert Simon in his celebrated work on *the architecture of complexity* [5].

The rest of the paper is organized as follows. Sect. 2 provides a brief review of the agent-based modular economy introduced in [1, 2]. Sect. 3 proposes the automatically defined terminal and motivates this idea with its connection to hierarchical modularity. The two experiments are conducted in Sect. 4 and 5. Sect. 6 gives the concluding remarks.

2 The Agent-Based Modular Economy

The economic model on which our simulation of product innovation is based is largely the same as [1, 2], which we shall briefly review here. Chen and Chie [1, 2] considered an economy composed of a number of firms and consumers with the usual objectives. To maximize profits, firms have the incentive to search for the products which can satisfy consumers to the highest degree. In the meantime, consumers allocate their limited budget to commodities which provide them with the greatest degree of enjoyment (measured in terms of consumer's surplus). The interaction between consumers and producers drives the evolution of a series of new products (the innovation process), as shown in Fig. 1.

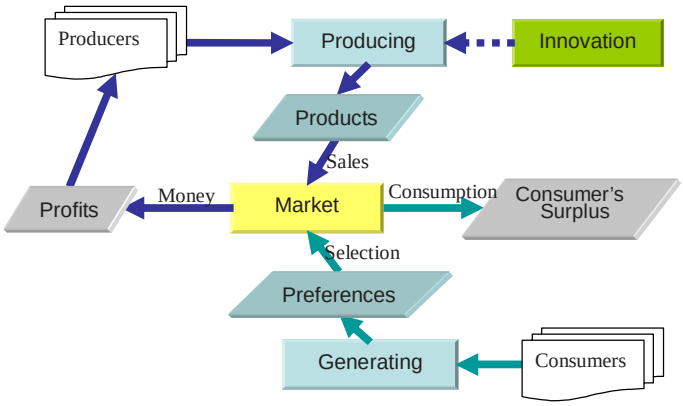

Fig. 1. The agent-based modular economy.

The commodity in this economy is represented as a *parse tree* as shown in the first row of Fig. 2. Each parse tree corresponds to a LISP program. The very bottom of the tree, i.e., the leaves, corresponds to the raw inputs (materials) $X1$, $X2$,..., whereas the root and all intermediate nodes represent the processors, $F1$, $F2$,..., applied to these raw materials in a bottom-up order, as the usual behavior of a LISP program. The whole parse tree can, therefore, be interpreted as a production process associated with the commodity. The unit cost of the commodity is a positive function of the number of processors and the number of raw inputs, i.e., it is a positive function of the (node) complexity of the

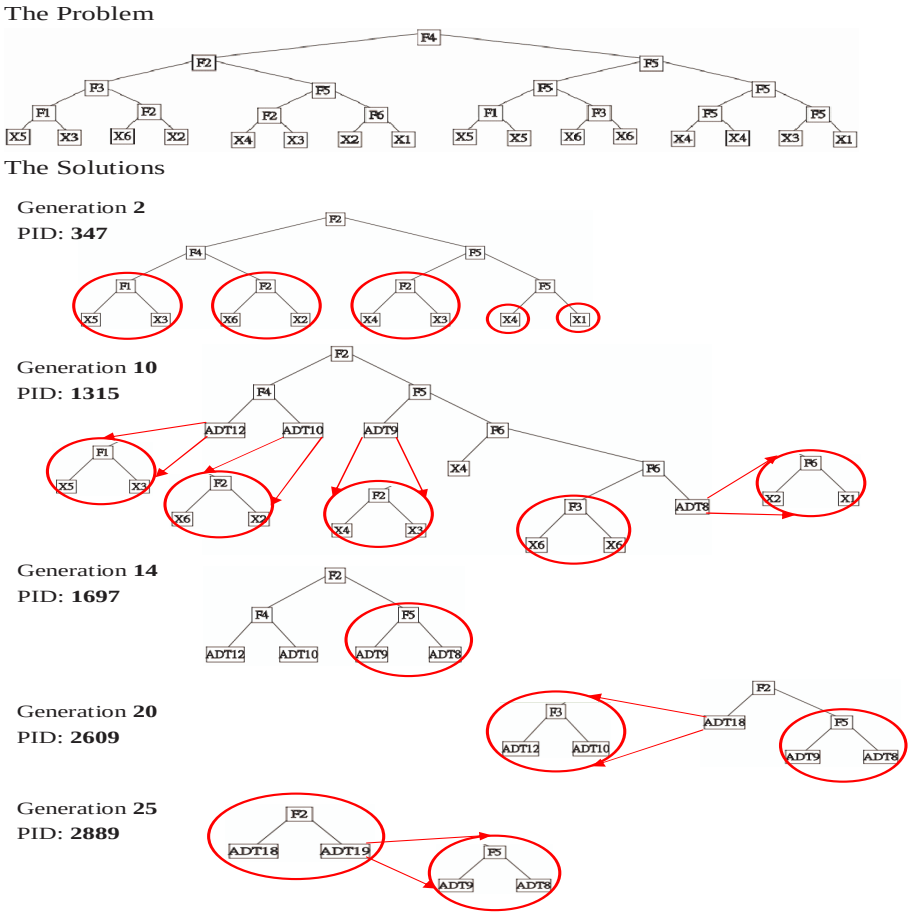

Fig. 2. An illustration of a process of product innovation.

commodity. In a simple way, we assume that the unit cost is a linear function of the node complexity.

In each market period, the firm has to decide how to allocate her limited working capital (budget) to R&D (designs of new products), to the production of existing commodities with different quantities and to reserves. R&D is the sole source of new products and is implemented by genetic programming with or without automatically defined terminals (to be detailed in Sect. 3), as shown in Fig. 3.

The preference of the consumers in the economy is also represented by a parse tree. To make the preference tree economically meaningful, three assumptions have been made [2], namely, the monotone, synergy, and consistency condition. The utility from consuming a commodity is based on the module-matching al-

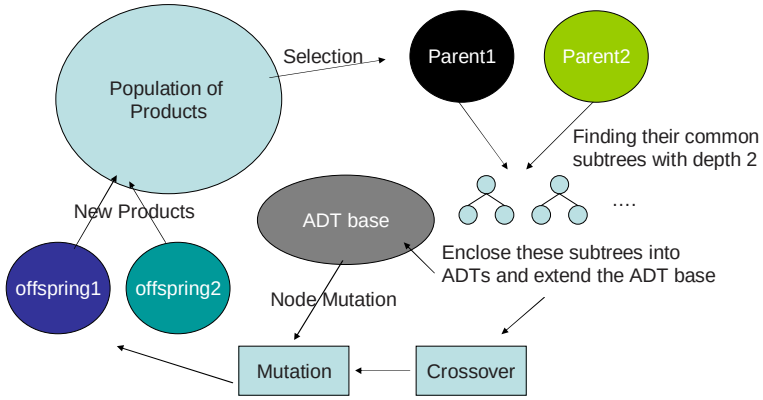

Fig. 3. Genetic programming as an engine for R&D.

gorithm proposed in [2]. The idea is to match each possible module (subtree) of the commodity with each possible module of the preference in descending order in relation to the depth of the tree. So, the big modules are matched first; if one succeeds, we sop, and if one fails, we move o the next biggest one. To satisfy the synergy condition and hence the idea of added-value, [2] assumes a power utility function for the preference tree as shown in Fig. 4. As a result, the utility is increasing at a faster rate when higher and higher levels of modular preferences are satisfied.

3 Modularity and Automatically Defined Terminals

Simple genetic programming is designed to find the solution to a problem, which is represented by a *parse tree*. In our case, a solution is analogous to a product, and whose corresponding problem is the ideal product which can bring the highest enjoyment to a target consumer (Fig. 2). The parse tree, from the bottom to the top, can be read as how the solution (product) can be constructed in *parallel*, as well as *incrementally* and *progressively*. What is accomplished at each incremental and parallel step is a minimum or marginal effort to combine what has been established in the previous steps.

As an illustration, Fig. 2 traces an artificial history of product innovation. Consider a target consumer whose preference is depicted in the first row of Fig. 2, which can be regarded as the solution to the problem. Firms do not know this design, and have to figure out the best design by themselves. The five products listed below are the designs discovered in generations 2, 10, 14, 20, and 25. These products match the consumer's needs to a higher and higher level. For example, the product PID 2889, i.e., the 2,889th new product designed by the firm, has completely answered the target's need to the entire first half at level four. Nonetheless, this product does not come out all of a sudden; all it has

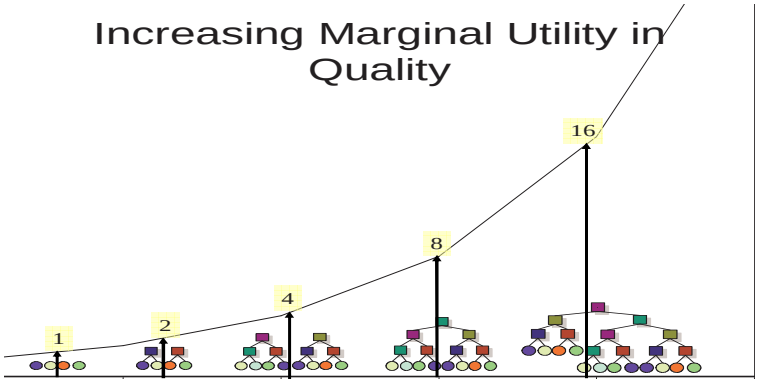

Fig. 4. The power utility function of a preference tree.

done is to combine two commodities which were already known before, namely, commodities ADT 18 and ADT 19, both of which were already known to the firm before generation 25. The "marginal" effort here is to assemble them in a right way, i.e., using processor F2.[1]

The results obtained in each step then become the bases or the *building blocks* for the subsequent steps. For example, as shown in Fig. 2, ADT 18 and ADT 19 serve as building blocks for all designs after generation 20. The entire process can then be viewed as a growing but convergent process from leaves to small trees, then to bigger and bigger trees, and finally to the target tree (the solution).

The description above enables us to see how genetic programming can be related to the Simonian notion of complexity [5], i.e., *hierarchy*. Herbert Simon viewed hierarchy as a general principle of complex structures. Hierarchy, he argued, emerges almost inevitably through a wide variety of evolutionary processes, for the simple reason that hierarchical structures are *stable*. To demonstrate the importance of a *hierarchical structure* or *modular structure* in production, Simon offered his well-known story about a competition between Hora and Tempus, two imaginary watchmakers. In this story, Hora prospered because he used the modular structure in his design of watches, whereas Tempus failed to prosper because his design was not modular. Therefore, the story is mainly about a lesson: the advantage of using a modular structure in production.

While using parse tree as the representation, simple genetic programming is not good at using a modular structure. The standard crossover and mutation can easily destroy the already established structure, which may cause the whole discovery or learning process to be non-incremental and non-progressive, and hence very inefficient. This problem is well-known in the GP literature, and has

[1] Of course, from an *ex ante* view, knowing what to combine and in which way is not trivial. In fact, in this example, it took the firm five generations to learn this. In this sense, the contribution is not entirely *marginal*. However, from an *ex post* view, it is just a combination of what we already known.

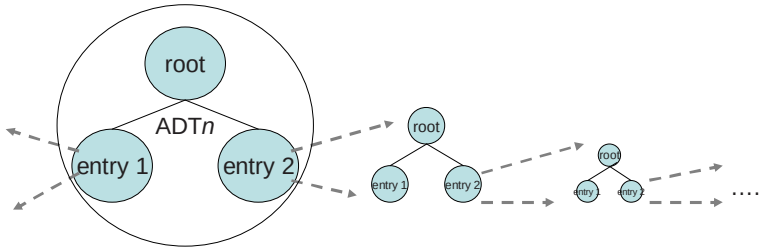

Fig. 5. Automatically defined terminals.

been extensively studied with various treatments [6, 7, 4, 8]. Motivated by these earlier studies, we propose automatically defined terminals (ADTs) as a way to enhance GP to find structured solutions.

An ADT, as shown in Fig. 5, is very similar to the automatically defined function (ADF) [4]. It itself has a fixed structure, in this case, a tree with a depth of two. The root of an ADT can be any function from the primitives (function set), while its leaf can be either a terminal from the primitives (terminal set) or can be any existing ADTs. In this way, it shares the same spirit as an ADT, namely, simplification, reuse, and encapsulation. The last item is particularly important because it means that whatever is inside an ADT will not be further interrupted by crossover and mutation. In this way, ADTs can be considered to be the part of learning in which we have great confidence, and which leaves no room for doubt. Through ADTs we distinguish what is considered to be *knowledge* from what is still in a trial-and-error process. Only the former can then be taken as the building blocks (modules), but not the latter. Without ADTs or equivalents, Simple genetic programming is essentially not designed to develop building blocks; therefore, it is not very good at finding the modular structure inherent in the problem.

4 Modularity and Consumer Satisfaction

Simple genetic programming can also detect a modular structure, but it does so only by chance, and hence it may be very difficult to detect complex modules.[2] To see this, in this section, we simulate how well consumers are served when the firm designs new products with modular GP (standard GP plus ADTs), and then compare the result with that of standard GP.

In this simulation, there are 100 consumers in the market. Each consumer has a preference tree with a depth of six. Viewed from the topmost level (the root level), the preference tree is composed of two modules. The one on the left,

[2] To define and measure complexity, Simon [5] advocated the use of a hierarchical measure – the number of successive levels of hierarchical structuring in a system or, in our case, the depth of the parse tree.

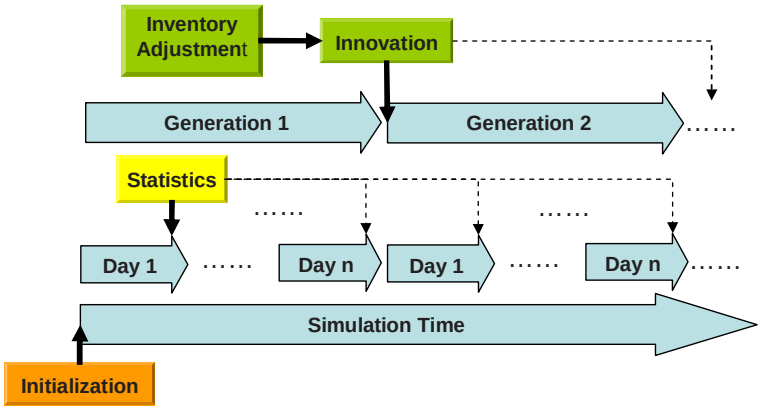

Fig. 6. Market days and learning cycles.

having a depth of five as shown in the first row of Fig. 2, is identical among all consumers, whereas the one on the right, having a depth of five or less, is heterogeneous, and is randomly generated by the ramped half-and-half method, an initialization method frequently used in GP. In this way, consumers' preferences have a common part as well as an idiosyncratic part. For the idiosyncratic part, the complexity is also different.

A profit-maximizing monopoly firm will try to serve the needs of this group of consumers by producing different products with different quantities and also with different degrees of specialization or diversification (customization).[3] The firm has to learn the consumers' preferences and hence, through R&D (driven by GP), design better products. The entire market process is summarized in Fig. 6. The learning cycle (GP cycle) is run with a number of generations (in our case, 5,000). Each generation is composed of a number of trading days (in our case, five). After each learning cycle, the firm has to decide what to produce, including some new products developed via production innovation, how many to produce, and how much to charge. The decision regarding production and R&D is based on the sales and profits statistics collected on the previous market days. The firm then supplies what has been produced, including those new items, during the next few market days.

For further analysis, in each generation, statistics regarding consumer satisfaction are reported. Consumer satisfaction is measured by the actual utility the consumer received from consumption divided by the maximum potential utility the consumer can possibly gain given his preference. The ratio is then multiplied by 1,000, and the measure lies in [0, 1,000]. By averaging the consumer satisfaction over all 100 consumers, we then derive the aggregate consumer satisfaction, which also lies in the same interval. The result is shown in Fig. 7. What Fig. 7 shows is not the result based on a single run, but on fifty runs. Accordingly,

[3] See [2] for details.

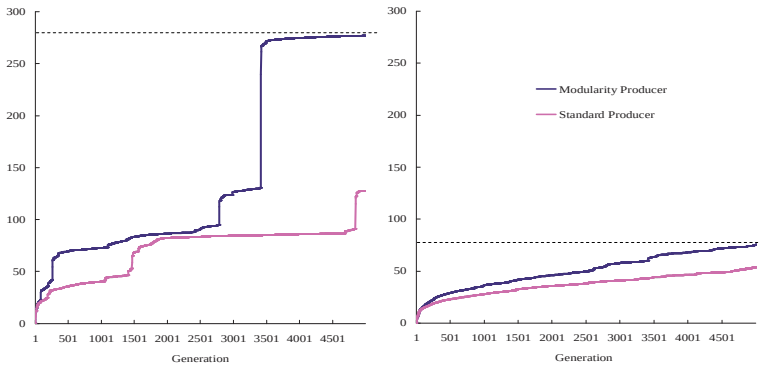

Fig. 7. Modularity and consumer satisfaction.

what is shown in the left panel of Fig. 7 is the average of the 50 runs, whereas what is shown in the right panel is the maximum of the 50 runs. It can be seen quite easily that the firm whose product design uses a modular structure can satisfy the consumers to a higher degree than the firm whose product design uses a non-modular structure.

5 Modular Structure and Competitiveness

In the previous section, under the assumption of a monopoly firm, we have seen the positive impact of using a modular structure on consumer satisfaction. In this section, we shall pursue this matter further by inquiring about the implications of a modular structure for the competitiveness of firms. In a sense, we attempt to re-examine the story given by Herbert Simon on the competition between two watchmakers: one using a modular structure and one not. For that purpose, we consider a duopolistic competition in which one firm uses a modular structure in her R&D (new product designs) and the other firm does not.

The two duopolistic firms compete with each other in a market composed of 100 consumers whose preferences are partially identical and partially idiosyncratic (see Sect. 4). We then watch their market share, i.e., the total sales of each firm divided by the total sales of the market, and the results are displayed in Fig. 8.[4] The results presented here are not based on a single run, but on one hundred runs. The one shown in the left panel of Fig. 8 is the mean of the 100 runs, whereas the one shown in the right panel is the median of the 100 runs. Below the separation line is the market share owned by the non-modular firm, and above the line is the market share owned by the modular firm. Clearly, their sum equals 100%.

[4] Notice that firms generally produce more than one product and can be very different from each other. Therefore, it is meaningless to measure the market share based on a single product.

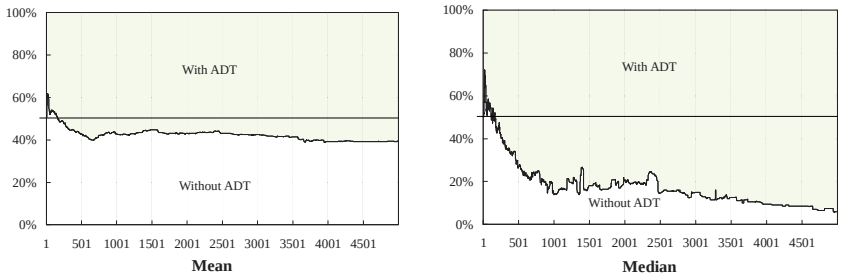

Fig. 8. Modularity and competitiveness.

Due to the existence of outliers, the time series behavior of the mean and that of the median is not quite the same, but the eventual dominance of the modular firm is evident. In the first few hundreds of generations, the non-modular firm, however, did have some competitive advantage over the modular firm. This is because establishing modules is costly. The idea of encapsulation associated with ADTs implies a fixed cost and hence a lower degree of mobility, depending on the degree of encapsulation or the complexity of ADTs.[5] Hence, the modular products will generally be more expensive. Unless these products can fit the consumers' needs to a higher degree, these high-priced products will have an adverse influence on marketing. Therefore, there is no guarantee (a probability of one) that the modular firm will always beat the non-modular firm. In fact, in 39 out of our 100 runs, the non-modular firm achieved a higher market share than the modular firm in the last generation (the 5,000th generation).

Finally, as one may expect, competition does bring a better quality to consumers. This is reflected in Fig. 9.

6 Concluding Remarks

Consumers are not random and their behavior can be studied and patterns can be extracted. On the other hand, innovation normally is not a random jump, but follows a gradually changing process. These two together suggest that the economy can be constructed in a modular way, or that the entire economy has a modular structure. In other words, Herbert Simon's notion of the architecture of complexity has the potential to be applied to the whole economy. In this paper, we study the significance of modularity in product innovation. We find that both consumers and producers can benefit from the use of a modular structure in product design. However, modularity may imply a higher cost and less mobility; therefore, its dominance is not certain. Using Simon's story, there is a chance that Tempus prospers and Hora fails.

[5] See footnote 2 for the measure of complexity.

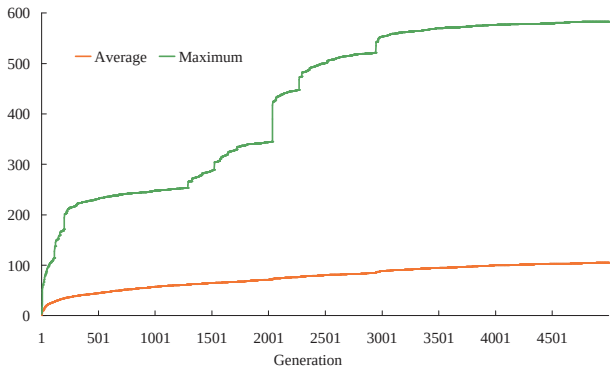

Fig. 9. Consumer satisfaction under competition.

Acknowledgement. The authors are grateful to Herbert Dawid, Christophre Georges, Massimo Ricottilli, and two anonymous referees for their helpful suggestions. NSC research grant No. 95-2415-H-004-002-MY3 as well as the computer time and facilities of National Center for High-performance Computing are gratefully acknowledged.

References

1. Chen, S.-H., Chie, B.-T.: Agent-Based Economic Modeling of the Evolution of Technology: The Relevance of Functional Modularity and Genetic Programming. International Journal of Modern Physics B, **18**, 17-19 (2004) 2376–2386
2. Chen, S.-H., Chie, B.-T.: A Functional Modularity Approach to Agent-based Modeling of the Evolution of Technology. In: A. Namatame, T. Kaizouji, and Y. Aruka. (eds.): The Complex Networks of Economic Interactions: Essays in Agent-Based Economics and Econophysics, Lecture Notes in Economics and Mathematical Systems Vol. 567, Springer-Verlag (2005) 165–178
3. Koza, J.R.: Genetic Programming: On the Programming of Computers by the Means of Natural Selection. The MIT Press, Cambridge (1992)
4. Koza, J.R.: Genetic Programming II–Automatic Discovery of Reusable Programs, The MIT Press, Cambridge (1994)
5. Simon, H.A.: The Architecture of Complexity. General systems. **10** (1965) 63–76
6. Angeline, P., Pollack, J.: Evolutionary Module Acquisition. Proceedings of the 2nd Annual Conference on Evolutionary Programming. MIT Press, Cambridge (1993) 154–163
7. Hoang, T.-H., Daryl, E., McKay, R., Nguyen, X.H.: Developmental Evaluation in Genetic Programming: The TAG-Based Framework. forthcoming in Knowledge Engineering Review (2007)
8. Rosca, J., Ballard, D.: Hierarchical Self-Organizaion in Genetic Programming. In: C. Rouveirol and M. Sebag (eds.): Proceedings of the Eleventh International Conference on Machine Learning. Morgan Kaufmann (1994)