行政院國家科學委員會專題研究計畫　成果報告

對創新行為的代理人基計算經濟建模：功能性模組法的應用

計畫類別：個別型計畫
計畫編號：NSC92-2415-H-004-005-
執行期間：92 年 08 月 01 日至 93 年 07 月 31 日
執行單位：國立政治大學經濟學系

計畫主持人：陳樹衡

計畫參與人員：陳樹衡

報告類型：精簡報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 93 年 11 月 17 日

行政院國家科學委員會專題研究計畫成果報告

對創新行爲的代理人基計算經濟建模: 功能性模組法的應用

A Functional-Modularity Approach to the Agent-based Computational Economic Modeling of Innovation

計畫編號: NSC 92-2415-H-004-005

執行期限: 92年8月1日至93年7月31日

主持人: 陳樹衡　國立政治大學經濟學系

Abstract

No matter how commonly the term *innovation* has been used in economics, a concrete analytical or computational model of innovation is not yet available. This paper argues that a breakthrough can be made with *genetic programming*, and proposes a functional-modularity approach to an agent-based computational economic model of innovation.

Keywords: Agent-Based Computational Economics; Innovation; Functional Modularity; Genetic Programming.

Motivation and Introduction

No matter how commonly the term "*innovation*" or "*technological progress*" has been used in economics, or more generally, in the social sciences, a concrete analytical or computational model of innovation is not yet available. Studies addressing specific technology advancements in different scientific and engineering fields are, of course, not lacking; however, the *general representation* of technology, based on which innovation can be defined and its evolutionary process studied, does not exist.

While direct modeling of innovation is difficult, economists' dissatisfaction with the neo-classical economic research paradigm is increasing, partially due to its incompetence in terms of producing novelties (or the so-called *emergent property*). We cannot assume in advance that we know all new goods and new technology that will be invented in the future. Therefore, in our model, we must leave space to anticipate the unexpected. Recently, (2; 3) introduced Zabell's notion of *unanticipated knowledge* to economists (11). This notion is motivated by *population genetics*. In probability and statistics it is referred to as the *law of succession*, i.e. how to specify the conditional probability that the next sample is never seen, given available sets of observations up to now. However, the Ewens-Pitman-Zabell induction method proposed by Aoki is still rather limited. Basically, the nature of diversity of species and the nature of human creativity should not be treated equally (5).

This paper proposes *genetic programming* as a possible approach leading to simulating the evolution of technology. Our argument is based on two essential standpoints. First of all, as regards the innovation process, we consider it to be a *continuous process* (*evolution*), rather than a *discontinuous process* (*revolution*). According to the continuity hypothesis, novel artifacts can only arise from antecedent artifacts. Second, the evolution can be regarded as a *growing process* by combining low-level building blocks or features to achieve a certain kind of high-level functionality. In plain English, new ideas come from the use (the combination) of the old ideas (building blocks). New ideas, once invented, will become building blocks for other more advanced new ideas. This feature, known as *functional modularity*, can be demonstrated by GP, and that will be shown in this paper.

Background

The idea of functional modularity is not new to economists. For example, Paul Romer has already mentioned that "Our physical world presents us with a relatively small number of building blocks–the elements of the periodic table–that can be arranged in an inconceivably large number of ways." (Romer, 1998). That GP can deliver this feature has already been well evidenced in a series of promising applications to the scientific, engineering, and financial domains.

A decade ago, financial economists started to apply the functional-modularity approach with GP to discover *new* trading rules. (9) and (1) took *moving average rules* and *trading range break-out rules* as the building blocks (primitives). GP was employed to grow new trading rules from these primitives. Hence, GP has already demonstrated the evolution of trading technology: combining low-level building blocks (MA, filter, or break-out rules) to achieve a certain kind of high-level functionality (profitable performance).

John Koza's application of GP to *Kepler's law* is another striking example. Here, not only did GP redis-

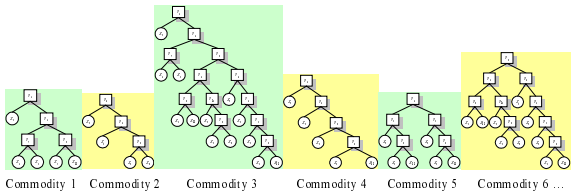

Figure 1: A Functional-Modularity Representation of Commodities.

Commodities are associated with their respective production processes which, when written in LIST programming language, can be depicted as parse trees as shown here.

cover the law, but also, as the system climbed up the fitness scale, one of its interim solutions corresponded to an earlier conjecture by Kepler, published ten years before the great mathematician finally perfected the equation (8; 4). A further application of GP by John Koza to analog circuits shows that GP-evolved solutions can actually compete with human ingenuity: the results have closely matched ideas contrived by humans. Koza's GP has produced circuit designs that infringe 21 patents in all, and duplicate the functionality of several others in novel ways (10).

Commodities and Production

Commodities in economic theory are essentially empty in terms of content. Little attention has been paid to their size, shape, topology, and inner structure. A general representation of commodities simply does not exist in current economic theory. In this paper, each commodity is associated with its *production process*. Each production process is described by a sequence of processors and the materials employed. In general, each sequence may be further divided into many parallel subsequences. Different sequences (or subsequences) define different commodities. The commodity with the associated processor itself is also a processor whose output (i.e. the commodity) can be taken as a material used by an even higher level of production. With this structure, we can ascertain the two major elements of GP, namely, the *function set* and the *terminal set*. The former naturally refers to a set of *primitive processors*, whereas the latter refers to a set of *raw materials*. They are denoted respectively by the following,

$$Function\ \ Set : \Xi = \{F_1, F_2, ..., F_k\}, \tag{1}$$

$$Terminal\ \ Set : \Sigma = \{X_1, X_2, ..., X_\kappa\}. \tag{2}$$

Each sequence (commodity, processor) can then be represented by a *LISP S-expression* or, simply, *a parse tree* (Fig. 1). The evolution of production processes (commodities) can then be simulated by using standard GP. The *knowledge capital* of the society at a point in time can then be measured by the complexity and the diversity of its existing production processes.

Commodity Space

Before introducing the functional-modularity approach to preferences, let us start with a brief review of the utility function used in conventional economic theory. The utility function $U(.)$ is generally a mapping from non-negative real space to real space $\mathcal{R}$.

$$U : \mathcal{R}_+^n \to \mathcal{R} \tag{3}$$

This above mapping is of little help to us when what we evaluate is a sequence of processors rather than just a quantity. In our economy, what matters to consumers is not the *quantity* they consumed, but the *quality* of what they consumed. Therefore, the conventional commodity space $\mathcal{R}_+^n$ is replaced by a new commodity space which is a collection of sequences of processors. We shall call the space $\mathcal{Y}$. The representation of the commodity space $\mathcal{Y}$ can be constructed by using the *theory of formal language*, for example, the *Backus-Nauer form* (**BNF**) of grammar. So $\mathcal{Y}$ is to be seen simply as the set of all expressions which can be produced from a start symbol Λ under an application of *substitution rules* (*grammar*) and a finite set of primitive processors (Σ) and materials (Ξ). That is, $\mathcal{Y}$ represents the set of all commodities which can be produced from the symbols Σ and Ξ.

$$\mathcal{Y} = \{Y \mid \Lambda \Rightarrow Y\} \tag{4}$$

While, as we saw in Fig. 1, each Y ($Y \in \mathcal{Y}$) can be represented by the language of expression trees (**ETs**), a more effective representation can be established by using *Gene Expression Programming* (**GEP**) developed by (7). In GEP, the individuals are encoded as *linear strings of fixed length* (the genome or set of chromosomes) which are afterwards expressed as nonlinear entities of different sizes and shapes, i.e. different expression trees. As (7) showed, the interplay of chromosomes and expression trees in GEP implies an unequivocal translation system for translating the language of chromosomes into the language of ETs. By using GEP, the commodity space can then be defined as a subset of the *Kleene star*, namely,

$$\mathcal{Y} = \{Y_n \mid Y_n \in (\Sigma \cup \Xi)^* \cap GEP\}, \tag{5}$$

where Y_n is a string of length n,

$$Y_n = y_1 y_2 ... y_n, \ \ y_i \in (\Sigma \cup \Xi), \forall i = 1, ..., n. \tag{6}$$

We have to emphasize that, in order to satisfy the syntactic validity, $\mathcal{Y}$ is only a subset of the Kleene star $(\Sigma \cup \Xi)^*$. To make this distinction, the $\mathcal{Y}$ described in (5) is referred to as the *strongly-typed Kleene star*. Each Y_n can then be translated into the familiar parse tree by using GEP. This ends our description of the commodity space.

Preferences

Unlike a commodity space, a preference space cannot be a collection of finite-length strings, since they are not satisfied by the *non-saturation* assumption. Economic theory assumes that consumers always prefer more to less, i.e. the marginal utility can never be negative. Even though we emphasize the *quality* dimension instead of the *quantity* dimension, a similar vein should

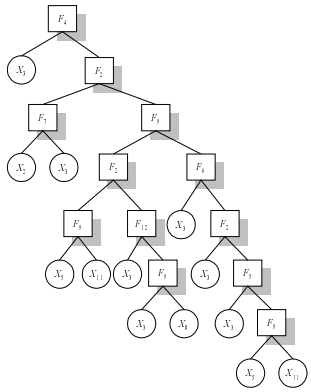

Figure 2: Preference: The Parse-Tree Representation
What is shown here is only part of the potentially infinitely large parse tree, i.e. only U^l of $[U^l]$.

equally hold: *you will never do enough to satisfy any consumer*. If consumers' preferences are represented by finite-length strings, then, at a point, they may come to a state of complete happiness, known as the *bliss point* in economic theory. From there, no matter how hard the producers try to upgrade their existing commodities, it is always impossible to make consumers feel happier. This is certainly not consistent with our observation of human behavior. As a result, the idea of a commodity space cannot be directly extended to a preference space.

To satisfy the non-saturation assumption, a preference must be a string of infinite length, something like

$$...u_1u_2...u_l... = ...U^l... \tag{7}$$

However, by introducing the symbol ∞, we can regain the finite-length representation of the preference, i.e.

$$\infty u_1u_2...u_l\infty = \infty U^l\infty = [U^l]. \tag{8}$$

First of all, as we mentioned earlier, consumers may not necessarily know what their preferences look like, and may not even care to know. However, from Samuelson's *revealed preference theory*, we know that consumers' preferences *implicitly* exist. Equation (8) is just another way of saying that consumers' preferences are *implicit*. It would be pointless to write down the consumers' preferences of the 30th century, even though we may know that these are much richer than what has been revealed today. To approximate the feedback relation between technology advancements and preferences, it would be good enough to work with *local-in-time* preferences (temporal preferences).

Secondly, Equation (8) enables us to see the possibility that preference is adaptive, evolving and growing. What will appear in those ∞ portions may crucially depend on the commodities available today, the commodities consumed by the consumer before, the consumption habits of other consumers, and other social, institutional and scientific considerations.

Utility Function

Given the preference $[U^l]$, let $U \mid [U^l]$ be the utility function derived from $[U^l]$. $U \mid [U^l]$ is a mapping from

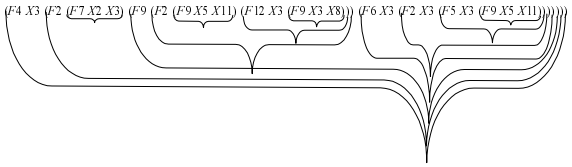

Figure 3: Modular Preference: The LISP Representation

Table 1: Modular Preferences Sorted by Depth

D	Subtrees or terminals	
1	$X_2, X_3, X_5, X_8, X_9, X_{11}$	1
2	$S_{2,1} = (F_7X_2X_3)$ $S_{2,2} = (F_9X_5X_{11})$ $S_{2,3} = (F_9X_3X_8)$ $S_{2,4} = (F_9X_5X_{11})$	2
3	$S_{3,1} = (F_{12}X_3(F_9X_3X_8))$ $S_{3,2} = (F_5X_3(F_9X_5X_{11}))$	4
4	$S_{4,1} = (F_2(F_9X_5X_{11})(F_{12}X_3$ $(F_9X_3X_8)))$ $S_{4,2} = (F_2X_3(F_5X_3(F_9X_5X_{11})))$	8
5	$S_5 = (F_6X_3(F_2X_3(F_5X_3$ $(F_9X_5X_{11}))))$	16
6	$S_6 = (F_9(F_2(F_9X_5X_{11})(F_{12}X_3(F_9X_3$ $X_8)))(F_6X_3(F_2X_3(F_5X_3$ $(F_9X_5X_{11})))))$	32
7	$S_7 = (F_2(F_7X_2X_3)(F_9(F_2(F_9X_5X_{11})$ $(F_{12}X_3(F_9X_3X_8)))(F_6X_3(F_2X_3$ $(F_5X_3(F_9X_5X_{11}))))))$	64
8	$S_8 = (F_4X_3(F_2(F_7X_2X_3)(F_9F_2(F_9X_5$ $X_{11})(F_{12}X_3(F_9X_3X_8)))(F_6X_3(F_2X_3$ $(F_5X_3(F9X_5X_{11})))))))$	128

the *strongly-typed Kleene Star* to $\mathcal{R}_+$.

$$U \mid [U^l] : \mathcal{Y} \to \mathcal{R}_+. \tag{9}$$

Hereafter, we shall simply use U instead of $U \mid [U^l]$ as long as it causes no confusion.

The modular approach to preference regards each preference as a hierarchy of modular preferences. Each of these modular preferences is characterized by a parse tree or the so-called building block. For example, the preference shown in Fig. 2 can be decomposed into modular preferences of different depths. They are all explicitly indicated in Fig. 3. Consider S_i to be the set of all modular preferences of *depth* i. Then Table 1 lists all modular preferences by means of these S_i. From both Fig. 3 and Table 1, it is clear that each subtree at a lower level, say S_j, can always find its parent tree, of which it is a part, at a higher level, say S_i where $i > j$. This subsequence relation can be represented as follows:

$$S_i \sqsupset S_j. \tag{10}$$

A commodity Y_n is said to *match* a modular preference S_i of U^l if they are exactly the same, i.e. they share

the same the LISP expression and the same tree representation. Now, we are ready to postulate the first regularity condition regarding a well-behaved utility function, which is referred to as the *monotonicity condition*.

Given a preference $[U^l]$, the associated utility function is said to satisfy the *monotonicity condition* iff

$$U(Y_{n_i}) > U(Y_{n_j}) \tag{11}$$

where Y_{n_i} and Y_{n_j} are the commodities matching the corresponding modular preferences S_i and S_j of U^l and S_i and S_j satisfy Equation (10).

The *monotonicity* condition can be restated in a more general way. Given a preference $[U^l]$ and by letting $\{h_1, h_2, ...h_j\}$ be an increasing subsequence of $\mathcal{N}_+$, then the associated utility function is said to satisfy the *monotonicity condition* iff

$$U(Y_{n_j}) > U(Y_{n_{j-1}}) > ... > U(Y_{n_2}) > U(Y_{n_1}) \tag{12}$$

where $Y_{n_1}, ..., Y_{n_j}$ are the commodities matching the corresponding modular preferences $S_{h_1}, ..., S_{h_j}$ of U^l, and

$$S_{h_i} \sqsupset S_{h_{i-1}} \sqsupset ... \sqsupset S_{h_2} \sqsupset S_{h_1}. \tag{13}$$

If S_k is a subtree of S_i as in Equation (10), then S_k is called the *largest subtree* of S_i if S_k is a *branch* (descendant) of S_i. We shall use "$S_i \lhd S_k$" to indicate this largest-member relation. Depending on the grammar which we use, the largest subtree of S_i may not be unique. For example, each modular preference in Fig. 2 has two largest subtrees. In general, let $S_{h_1}, S_{h_2}, ...S_{h_j}$ be all the largest subtrees of S_i, denoted as follows:

$$S_i = \sqcup_{h_1}^{h_j} S_k \lhd \{S_{h_1}, S_{h_2}, ...S_{h_j}\}, \tag{14}$$

where $\{h_1, h_2, ...h_j\}$ is a non-decreasing subsequence of $\mathcal{N}_+$. Notice these largest trees may not have sub-relationships (10) among each other. However, they may have different depths, and the sequence $\{h_1, h_2, ...h_j\}$ ranks them by depth in an ascending order so that S_{h_1} is the largest subtree with minimum depth, and S_{h_j} is the one with maximum depth.

The second postulate of the well-behaved utility function is the property known as *synergy*. Given a preference $[U^l]$, the associated utility function is said to satisfy the *synergy condition* iff

$$U(Y_{n_i}) \geq \sum_{k=1}^{j} U(Y_{n_k}), \tag{15}$$

where Y_{n_i} and $\{Y_{n_k}; k = 1, ..., j\}$ are the commodities matching the corresponding modular preferences S_i and $\{S_{h_k}; k = 1, ..., j\}$ of $[U^l]$, and S_i and $\{S_{h_k}; k = 1, ..., j\}$ satisfy Equation (14).

For convenience, we shall also use the notation $\sqcup_{k=1}^{j} Y_{n_k}$ as the synergy of the set of commodities $\{Y_{n_k}; k = 1, ..., j\}$. Based on the *New Oxford Dictionary of English*, synergy is defined as "the interaction or cooperation of two or more organizations, substances, or other agents to produce a combined effect greater than the sum of their separate effects." "*The whole is greater than the sum of the parts*" is the fundamental source for *business value creation*. Successful business value creation depends on two things: *modules* and the *platform* to combine these modules. Consider the consumer characterized by Fig. 2 as an example. To satisfy him, what is needed are all of the modules listed in Table 1. Even though the technology has already advanced to the level S_7, knowing the use of processor F_4 to combine X_3 and S_7 can still satisfy the consumer to a higher degree, and hence create a greater business value.

A modular preference may appear many times in a preference. For example, $S_{2,4}$ in Table 1 appears twice in Fig. 2. In this case, it can simultaneously be the largest subtree of more than one modular preference. For example, $S_{2,4}$ is the largest subtree of both $S_{3,2}$ and $S_{4,1}$. Let S_k be the largest subtree of S_{h_1}, S_{h_2},..., and S_{h_j}. Denote this relation as

$$S_k = \sqcap_1^j S_{h_i} \rhd \{S_{h_1}, S_{h_2}, ...S_{h_j}\}. \tag{16}$$

Given a preference $[U^l]$, the associated utility function is said to satisfy the *consistent condition* iff

$$U(Y_{n_i} \mid S_k \rhd S_{h_1}) = ... = U(Y_{n_i} \mid S_k \rhd S_{h_j}), \tag{17}$$

where $Y_{n_i} \mid S_k \rhd S_{h_1}$ is the commodity which matches the corresponding modular preference S_k in the designated position, $S_k \rhd S_{h_i}$. The consistency condition reiterates the synergy effect. No matter how intensively the commodity Y_{n_i} may significantly contribute to the value creation of a synergy commodity, its value will remain identical and lower when it is served *alone*.

Given a preference $[U^l]$, the associated utility function U is said to be *well-behaved* iff *it satisfies the monotone, synergy and consistency condition*. It generates a sequence of numbers $\{U(Y_{n_i})\}_{i=1}^{h}$ where Y_{n_i} matches the respective modular preference $S_{d,j}$. $S_{d,j}$ is the jth modular preference with depth d. The utility assigned in Table 1 is an illustration of a well-behaved utility function derived from the preference shown in Fig. 2. In fact, this specific utility function is generated by the following exponential function with base 2.

$$U(S_{d,j}) = 2^{d-1} \tag{18}$$

Utility function (18) sheds great light on the synergy effect. Thus, primitive materials or rudimentary commodities may only satisfy the consumer to a rather limited extent. However, once suitable processing or integration takes place, their value can become increasingly large to the consumer. The exponential function with base 2 simply shows how fast the utility may be scaled up, and hence may provide a great potential incentive for producers to innovate. Of course, to be a well-behaved utility function, U can have many different functional forms.

Firms and Production

On the production side, the economy is composed of n_f producers, each of which is initially assigned an equal operating capital, K_0.

$$K_{1,0} = K_{2,0} = ... = K_{n_f,0} = K_0. \tag{19}$$

With this initial capital, the producers are able to buy materials and processors from the input markets up to the amount that they can afford. There are two types of input markets at the initial stage, namely, the *raw-material market* and the *rudimentary processor market*. For simplicity, we assume that the supply curves of the two markets are infinitely elastic with a fixed unit cost (c) for each raw material and for each rudimentary processor:

$$C_{X_1} = C_{X_2} = ... = C_{X_\kappa} \\ = C_{F_1} = C_{F_2} = ... = C_{F_k} = c. \tag{20}$$

With the materials and the rudimentary processors purchased from the input market, the producer can produce a variety of commodities, defined by the associated sequence of processors. The cost of each commodity is then simply its total amount of materials and the number of processors, or, in terms of GP, the *node complexity* of the parse tree. However, to allow for the *scale effect*, each additional unit of the same commodity produced by the producer should be less costly. This can be done by introducing a monotonically decreasing function $\tau(q)$ $(0 \leq \tau(q) \leq 1)$, where q is the qth unit of the same commodity produced. The cost of each additional unit produced is simply the cost of the previous unit pre-multiplied by $\tau(q)$. With this description, the *capacity constraint* for a *fully-specialized producer* i ($i \in [1, ...n_f]$), i.e. the producer who supplies only one commodity, should be

$$K_0 \geq \sum_{q=1}^{\bar{q}} C_q, \tag{21}$$

where $C_q = \tau(q)C_1$ is the unit cost of the qth unit and $\tau(1) = 1$. For a *fully-diversified producer*, i.e. the producer who produces a variety of commodities and one for each, the capacity constraint is

$$K_0 \geq \sum_{m=1}^{\bar{m}} C_{m,1}, \tag{22}$$

where $C_{m,1}$ is the cost of the first unit of commodity m. In general, the capacity constraint for the producer i is

$$K_0 \geq \sum_{m=1}^{\bar{m}} \sum_{q=1}^{\bar{q}_m} C_{m,q}, \tag{23}$$

where $C_{m,q} = \tau_m(q)C_{m,1}$.

In Equation (23), the strategic parameters are $\bar{m}$, $\bar{q}$ and C_m. To survive well, producers have to learn how to optimize them. $\bar{m}$ can be be taken as a measure of the degree of *diversification*, whereas $\bar{q}$ can be taken as the degree of *specialization*. C_m, i.e. the node complexity of the commodity m, is also a behavioral variable. Given the capacity constraint, the producer can choose to supply a large amount of primitive commodities (a quantity-oriented strategy), or a limited amount of highly delicate commodities (a quality-oriented strategy). Therefore, the choice of C_m can be regarded as a choice of the level of *quality*.

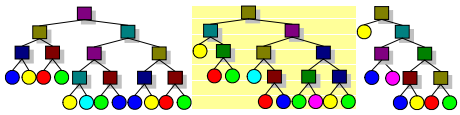

Figure 4: EvolTech: Preferences Initialization

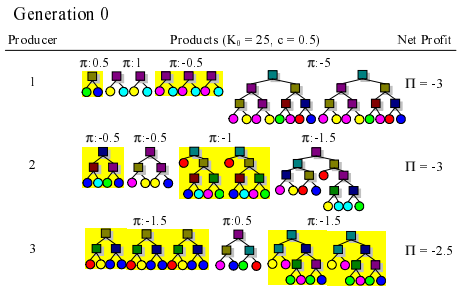

Figure 5: EvolTech: Generation 0

Demonstration

The idea presented above has been written into a computer program called **EvolTech**, which stands for "Evolution of Technology." In this section, we demonstrate a *vanilla* version of EvolTech. What we mean by *vanilla* will become clear as we give the demo.

First, as in all computational economic models, we start with a simple description of *initialization*. The initialization in EvolTech includes the generation of preferences. Based on the formulation given in Sections and , the preferences of three consumers are randomly generated, as shown in Figure 4. The complexity of the preference has been severely restricted to a depth of 5 and is fixed throughout the entire evolution.[1] Notice that these preferences are characterized by colorful nodes. Each different color is refereed to a different primitive, sampled from a given primitive set.

In addition to the three consumers, there are three producers in the economy. Based on the same given primitive set, commodities are also randomly generated by these producers, as shown in Figure 5. Notice that the three dimensions of production behavior, i.e. quantity, quality, and diversity, are all randomly determined as long as they together satisfy the capacity constraint (23). The initial capital capacity K_0 is set to 25, and the unit cost c is set to 0.5. The scale effect is ignored. Each of the commodities is then served to the consumers whose preferences are displayed in Figure 4.

Without the details about how trade actually proceeds, it is not easy to describe how the final price and hence the profit is determined. Therefore, in this vanilla version of EvolTech we simply take the highest reservation price as the market price.[2]

[1] In other words, we do not consider the general preferences as discussed in Equation (8), neither the evolution of preferences in this vanilla version.

[2] We have proposed an algorithm to compute Equation (9) for a well-behaved utility function U, as defined in Section . This algorithm, called the *module-matching algorithm*, is very intuitive. It looks for the *projection* of the commodity Y_i to $[U^l]$, i.e. a measure of distance between a commodity Y_i and

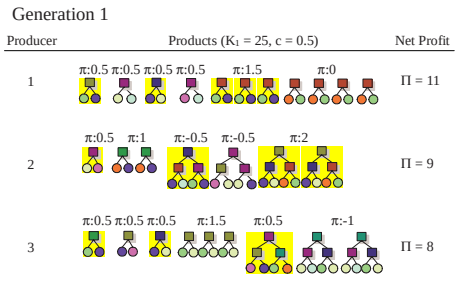

Figure 6: EvolTech: Generation 1

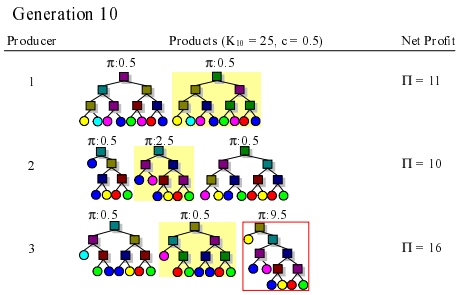

Figure 7: EvolTech: Generation 10

This simplification will facilitate our calculation of profits, π. In Figure 5, π is shown on the top of each commodity. We can see that most commodities have negative profits. This is not surprising because at this initial stage all commodities are randomly designed, and the chance of meeting consumers' needs to any significant degree is naturally low with the given combinatoric complexity.

Sophisticated commodities may be even worse than those simple designs because they induce higher production costs, and can only satisfy consumers to a very limited degree. So, as evidenced in Figure 5, commodities that suffer great economic losses tend to be those with sophisticated designs, i.e. trees with a large degree of node complexity. By summing the profits over all commodities, we get aggregate profits for each firm, which are shown on the right-hand side of the figure. In this specific case, all three firms make a loss. This finishes our description of the initial generation.

While moving to the next market period (next generation), firms start to learn from *experience*. Their profit profiles provide them with fundamental clues on how to re-design their products for the next generation. What is shown in Figure 6 is the result of their adaptation. It is interesting to notice that these new-generation commodities seem to become simpler compared with those of the previous generation (Figure 5). This is mainly because sophisticated designs do not contribute to profits but losses. Therefore, firms tend to replace those sophisticated designs with simpler ones. The economy as a whole can be described as a *quantity-based economy*, since all firms choose to produce a large number of rudimentary commodities (i.e. they repeat doing simple standard things).

However, this strategy turns out to work well. While each simple commodity can earn a firm a tiny profit, summing them together is still quite noticeable. So, in the end, the profits of all three firms improve quite significantly. This process is then further reinforced, and in the coming generations, more resources are devoted to rudimentary commodities. Sophisticated designs are almost entirely given up. However, since there are not too many rudimentary commodities to develop in the market, when all firms concentrate on producing rudimentary commodities, the limited number of rudimentary-commodity markets become highly competitive, and the profits from producing these commodities decline as a result of the keen competition. At this point, the economy actually moves toward *an era of zero profit*.

Once producing primitive commodities is no longer profitable, the selection bias towards it also becomes weaker. Some sophisticated designs occasionally coming out of the crossover and mutation operators may find it easier to survive. That improves the chances of satisfying consumers to a higher degree. When that indeed happens, not only do firms make a breakthrough by successfully having a sophisticated (delicate) design, but the lucrative profits also attract more resources that can be devoted to quality products. While this does not always happen and the process is not always smooth, the process may be reinforcing. So, commodities with more delicate designs and higher profits may come one after the other. In the end, the economy is gradually transformed into a quality-based economy, as shown in the 10th generation of our simulation (Figure 7).[3]

Concluding Remarks and Future Work

In this paper, commodities, production and preference, those fundamentals of economic theory, have been reformulated in light of functional modularity. We believe that this re-formulation work is original and productive. It lays the foundation upon which one can build and simulate the evolution of technology, more specifically, within the context of agent-based computational economic (ACE) models. A full picture of this ACE model has not been presented in this paper, partially due to the limitations of size imposed on the paper. We, therefore, can only give a sketch of some other essential ingredients, and leave a more detailed account to a separate paper.

References

[1] Allen, F. and R. Karjalainen, (1999). "Using Genetic Algorithms to Find Technical Trading

the preference $[U^l]$. Once the operational meaning of Equation (9) becomes clear, it is possible to infer the *reservation price* which a consumer would like to pay for commodity Y_i from $[U^l]$.

[3] The progress may not be smooth. Severe fluctuations can happen. The progress may not be sustained long enough either. The economy may stagnate after a short but fast take-off, and consumers are only supplied with some "basic needs." For a more detailed demonstration of the complex variety, see Chen and Chie (2004).

Rules," *Journal of Financial Economics*, **51(2)** 245–271.

[2] Aoki, M., (2002a). "Open Models of Share Markets with Two Dominant Types of Participants," *Journal of Economic Behavior and Organization*, **49** 199–216.

[3] Aoki, M., (2002b). "Applications of Ewens–Pitman–Zabell Inductive Methods in New Economic Dynamics," *Proceedings of the Sixth International Conference on Complex Systems*, 29–35.

[4] Banville, J., (1993). *Kepler: A Novel (Vintage International)*. Vintage Books.

[5] Basalla, G., (1988). *The Evolution of Technology*. Cambridge University Press.

[6] Chen, S.-H. and B.-T. Chie, (2004) "Functional Modularity in the Test Bed of Economic Theory–Using Genetic Programming," in R. Poli, et al. (eds.), *GECCO-2004: Proceedings of the Genetic and Evolutionary Computation Conference*, June 26-30, 2004, Seattle, Washington, USA.

[7] Ferreira, C., (2001). "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems," *Complex Systems*, **13 (2)**, 87-129.

[8] Levy, S., (1992). *Artificial Life: A Report from the Frontier Where Computers Meet Biology*, Vintage, New York.

[9] Neely, C., P. Weller, and R. Dittmar, (1997). "Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach," *Journal of Financial and Quantitative Analysis*, **32(4)** 405–426.

[10] Willihnganz, A., (1999). "Software that Writes Software," *salon.com*, WWW Article.

[11] Zabell, S., (1992). "Predicting the Unpredictable," *Synthese*, **90** 205–232.