Pattern Discovery by Genetic Algorithms in Syntactic Pattern Based Chart Analysis for Stock Market[1)]

Hyun Soo Kim(Department of MIS Dong-A University)

Abstract

This paper presents a pattern generation scheme from financial charts. The patterns constitute knowledge which consists of patterns as the conditional part and the impact of the pattern as the conclusion part. The patterns in charts are represented in a syntactic approach. If the pattern elements and the impact of patterns are defined, the patterns are synthesized from simple to the more highly credible by evaluating each intermediate pattern from the instances. The overall process is divided into primitive discovery by Genetic Algorithms and pattern synthesis from the discovered primitives by the Syntactic Pattern-based Inductive Learning (SYNPLE) algorithm which we have developed. We have applied the scheme to a chart : the trend lines of stock price in daily base. The scheme can generate very credible patterns from training data sets.

1. Introduction

Chart Analysis are conveniently used by investors. As a tool to help investors, chart analysis is used to visualize the stock prices and trading volumes. From the charts, investors attempt to detect patterns which can help the prediction of future stock prices. Some investors might have their own ways of handling these issues based on their own experiences. The rules that can be generated from a pattern look like

"If a pattern occurs, then the price will increase (or decrease)."

이 논문은 1993년도 동아대학교 학술연구 조성비에 의하여 연구되었음

In (Lee, Kim & Trippi, 1992), we focused on the support of chart analysis. To automate chart analysis, syntactic pattern-based inductive learning scheme called SYNPLE was proposed, that adopted AI approaches such as syntactic pattern recognition and inductive machine learning. In this scheme following aspect of chart analysis can be supported.

(a) Formalization of pattern definition.
(b) Automatic recognition of patterns.
(c) Rule generation based on the patterns.
(d) Performance evaluation of the rules.

In this scheme, a pattern is represented by pattern primitives and compositional operators. A pattern primitive has several attributes that describe its characteristics. We have set two compositional operators : CONC (concurrent composition) and SEQ (sequent composition). They synthesize primitives to constitute patterns on the time horizon. To synthesize reliable patterns, SYNPLE generates a set of patterns from the set of primitives with the highest performance within the constraints of syntactical grammars.

However, in the scheme, the pattern primitives must be given beforehand to represent patterns in syntactic pattern recognition scheme. Determining the pattern primitives are likely to be somewhat intuitive work. The designed pattern primitives influence the performance of pattern of chart and recognition process of pattern very much. The search space of primitives are very huge and the shape can not be known. This is why we used Genetic Algorithms - a robust and heuristic search method. By the method, primitives are discovered without human intervention. We experimented the two step approach: primitive discovery by Genetic Algorithms and pattern synthesis by SYNPLE to predict the future stock price behavior. using a chart : trend line of stock price in daily base.

2. Syntactic approach to representation and recognition of patterns of charts

We adopt a syntactic approach to represent a pattern from charts(Lee, Kim & Trippi, 1992). This approach is particularly useful for patterns which cannot be described in the numerical measurements. The syntactic pattern recognition views patterns as complex of primitives and compositional operators. A pattern can be represented by the composition of simple subpatterns and these can be described by even simpler subpatterns. That is, a pattern is described by a hierarchical structure of subpatterns. The terminals of subpatterns in the structure are primitives. The compositional operators can combine the primitives into a pattern. The pattern composed of primitives by compositional operators can also be recognized by

parsing the pattern into the primitives and compositional operators.

A primitive is described by several attributes and their values. For example, the primitives of stock charts may have attributes such as slope, change of slope, gap, and relative position which describe the spacial characteristics. Moreover, the pattern of stock charts has a special feature that has dynamic nature. That is, two patterns having similar shape can have different meaning according to how long the pattern keeps its shape. Therefore the attributes that describe a primitive should contain the duration of primitive as well as the spacial characteristics in the two-dimensional plane. The dynamic characteristic is very peculiar feature of pattern of chart that is drawn on time horizon.

In summary, the composition process of pattern can be diagrammed as in Figure 1. The parsing process for the recognition of a pattern is the reverse direction against the composition process.

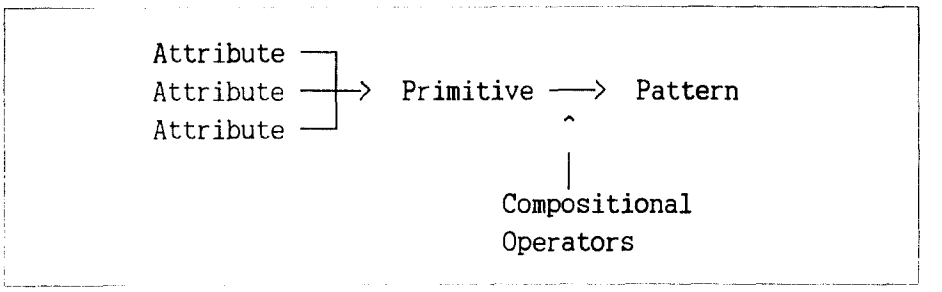


Figure 1. Composition process of a pattern

3. Definition of trend line of stock price(TLP) and its primitives

In this paper, we use a chart: trend line of stock price. Trend lines are piecewisely fitted lines that indicate the local trends of stock price. Three types of trend lines are central line, upper supporting line and lower supporting line. Figure 2 shows a central line. The central line is the same as the simple linear regression line. However, if the mean square error exceeds the predefined tolerance, new central line will be invoked. Thus the tolerance determines the degree of local fitness. Obviously, the larger the tolerance, the longer the central lines. On the other hand, the upper supporting line shown in Figure 3 is drawn so as to minimize the mean distances between the line and stock price maintaining the stock price below the line. The lower supporting line shown in Figure 4 is the opposite of upper supporting line.

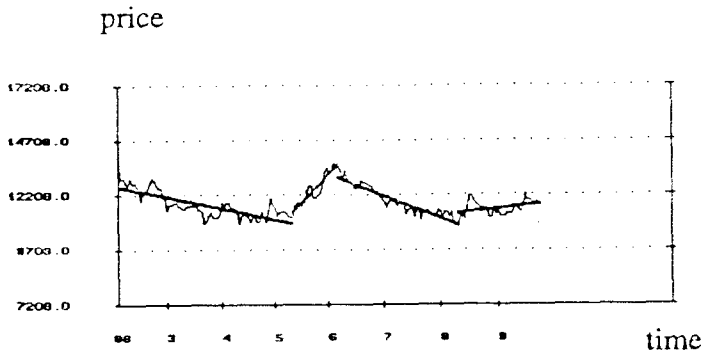


Figure 2. Illustrative stock price trend lines(central line)

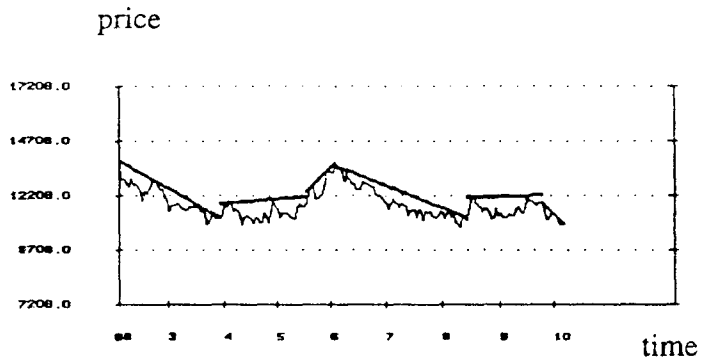


Figure 3. Illustrative stock price trend lines(upper supporting line)

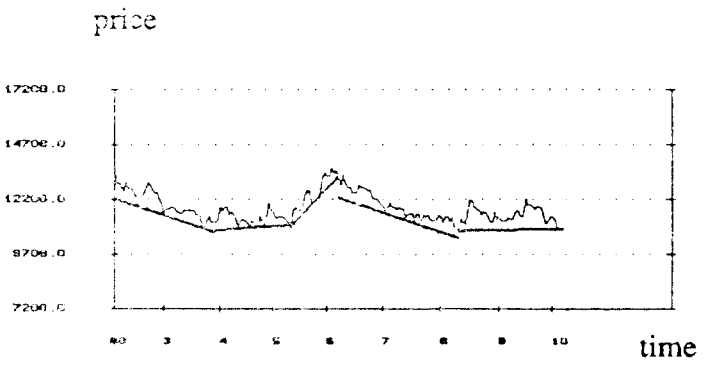


Figure 4. Illustrative stock price trend lines(lower supporting line)

To define the trend lines formally, let us use the following notations.

t : time point in the time horizon.
i : relative time point in a trend line i = 1,...,l.
l : length of a trend line.
s : starting time of the current line.
$\underline{l}$: minimally required length to be a trend line.
P(t) : stock price at the time t.
α, β : estimated parameters.
$\hat{P}(t)$: stock price estimate at time t by the trend line $\alpha + \beta t$
$e(t) = P(t) - \hat{P}(t)$: estimation error at time t.

(1) Central Line

The central line is represented in a linear form $\hat{P}(t) = \alpha + \beta t$. The parameter α and β are estimated according to the following steps.

Step 1. Set the tolerance in regards of the mean square error. Set s=1, and i= $\underline{l}$.

Step 2. Solve the model EQ(1) which minimizes the mean square error to obtain α and β with the given s and i.

$$\text{Minimize} \quad \sum_{t=s}^{s+i-1}((P(t)-\alpha-\beta t)^2 \div i) = \sum_{t=s}^{s+i-1}(e(t)^2 \div i) \qquad \text{EQ(1)}$$

Step 3. If the mean square error is less than the tolerance, increment i by 1 and go to step 2. Otherwise, go to step 4.

Step 4. To trim the errors at the tail of the line, decrement i backwardly up to the point whose mean square error is non-increasing. Freeze l=i. Then a line is constructed.

Step 5. Reset s=s+l and i=l, and go to step 2 until all data points are exhausted.

(2) Upper Supporting Line

The steps of drawing the upper supporting line is similar to those for the central line. The only difference is that the mean square error in EQ(1) is replaced by the term $\sum_{t=s}^{s+i-1}(-e(t) \div i)$ in EQ(2). The model can be solved by a linear programming algorithms.

$$\text{Minimize} \quad \sum_{t=s}^{s+i-1}((\alpha+\beta t-P(t)) \div i) = \sum_{t=s}^{s+i-1}(-e(t) \div i) \qquad \text{EQ(2)}$$

subject to $\alpha + \beta t \geq P(t)$.

(3) Lower Supporting Line

In the same way, the lower supporting line can be defined by replacing the formula EQ(1) by EQ(3).

$$\text{Minimize} \quad \sum_{t=s}^{s+i-1}((P(t)-\alpha-\beta t) \div i) = \sum_{t=s}^{s+i-1}(e(t) \div i) \qquad \text{EQ(3)}$$

subject to $P(t) \geq \alpha + \beta t$.

Key elementary attributes to specify the trend lines are slope of each line. shift direction of the starting point of following line at the tail of the line, and change of gaps between two different types of lines. However, in the paper we focus only slope of trend line. The slope can be classified by the intervals of angles. The primitives of trend line can be represented in frames as shown in Figure 5.

{{ TLP-primitive
LINE-TYPE : [*central-line upper-supporting-line lower-supporting-line*]
TOLERANCE : *value*
SLOPE : *lower-bound upper-bound*
DURATION : *lower-bound upper-bound* }}

Legend

Capital letters : reserved words
[ ] : one of the statements should be chosen

Figure 5. Syntax of primitives of trend line of stock price

For instance, the example elements are as follows.

{{ TLP-1
LINE-TYPE : upper-supporting-line
TOLERANCE : 14
SLOPE : 38 79
DURATION : 39 45 }}

The above primitive means that that the line type is upper supporting line, the tolerance is 14, the slope is between 38 and 79 degrees and the duration is between 39 and 45 days.

4. Compositional operators : CONC and SEQ

The primitives can be combined into the more complex patterns by the use of compositional operators. There are two types of compositional operators. One is concurrent combination and the other is sequential combination of the two subpatterns along the time horizon. If we abbreviate the concurrent combination to CONC, and the sequential combination to SEQ, then examples of the patterns are as follows:

PATTERN-A = CONC(TLP-1, TLP-2)

PATTERN-B = SEQ(TLP-3, TLP-4)

PATTERN-C = SEQ(PATTERN-A, PATTERN-B)
= SEQ(CONC(TLP-1, TLP-2), SEQ(TLP-3, TLP-4))

The range of PATTERN-A is the time when two primitives TLP-1 and TLP-2 occur concurrently. Likewise, the range of PATTERN-B is the time when two primitives TLP-3 and TLP-4 occur in sequence. In detail, SEQ(P1, P2) indicates that the primitive P1 starts to occur at least one day earlier than the occurrence of primitive P2, and the gap between the ending point of P1 and starting point of P2 does not exceed one day.

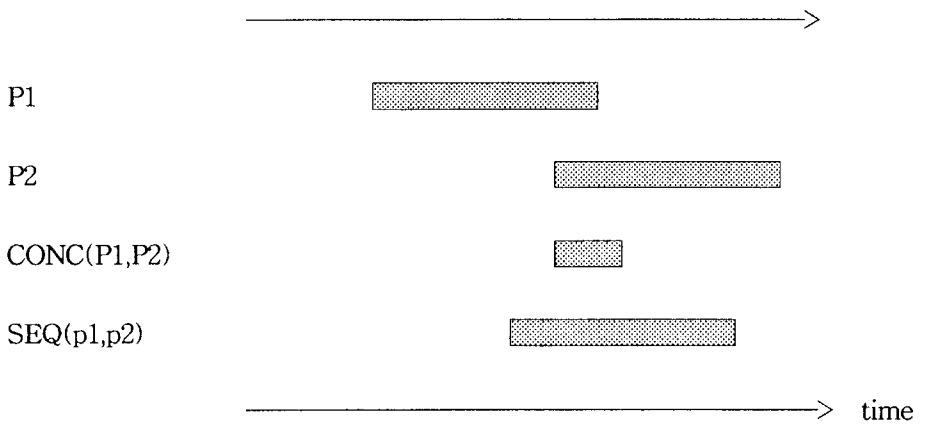


Figure 6. The interval where CONC(P1,P2) and SEQ(P1,P2) reside

5. Primitive discovery and pattern generation

The primitive itself should be designed in a syntactic approach. However, designing the attribute, that is, determining the value of each attribute in the primitive is very difficult for human. So we need some methodology to discover the primitives. The fundamentals that should be prepared by a designer in pattern description is the attribute of primitive. The overall scheme of pattern generation is the search of patterns from the examples in syntactic pattern description architecture. The process was illustrated as in Figure 1. The attributes may be different from chart to chart. The attributes should be defined properly. That is at the designer's discretion. The composition of attributes with their values becomes a primitive, which is the simplest pattern that can be recognized in the chart. We should determine proper initial set of values to make a primitive. That is a search problem in 5 or 6-dimensional search space. After having discovered the primitives, we can combine the primitives to make more highly performable patterns. The

search of primitives can be viewed as inductive learning. First, it is because that the target classes of primitives should be predefined according to the stock price change after the occurrence of the primitive. Second, the discovered primitives are evaluated from their instances.

However, the unanswered question now is how we can set the primitives. The full search or mathematical search method cannot be the solution because the search space is very huge and the shape of search space can not be known. We adopt Genetic Algorithms as the inductive inference method to generate the primitives. At the next section, we will overview GAs.

Based on the primitives discovered by GAs, the next step is the syntactical synthesis of primitives into more sophisticated patterns with the highest credibilities. To attain this goal, we used and algorithm named SYNPLE(SYNtactic Pattern-based LEarning). SYNPLE synthesizes good patterns syntactically from the primitives evaluating the past instances of each candidate pattern. See references (Lee, Kim & Trippi, 1992) for detail steps of the algorithm.

6. Genetic Algorithms

6.1. Overall concept

Genetic Algorithms(GAs) are a family of adaptive search procedures. The name GA is originated from the fact that the algorithms are conceptually based on models of genetic change in a population of individuals. These models consist of three basic elements : (1) a Darwinian notion of fitness; (2) a mating operator, which produces offspring for the next generation; and (3) genetic operators, which determine the genetic makeup of offspring from the genetic material of the parents (De Jong, 1988). Adaptation proceeds not by making incremental changes to a single structure but by maintaining a population of structures from which new structures are created using genetic operators such as crossover and mutation. Each structure in the population has an associated fitness (goal-oriented evaluation), and these scores are used in a competition to determine which structures are used to form new ones. GAs are able to exploit the accumulation of information about an initially unknown search space in order to bias subsequent search into useful subspace. The major advantage of GAs is that GAs can provide robust search in complex spaces. The basic execution cycle of GAs are as follows (Austin, 1990).

(a) From the set of existing structure, select candidates according to fitness for reproduction. Candidates with the highest fitness factor have a greater probability of contributing offspring.

(b) After reproduction, randomly choose pairs for mating through crossover (exchange of genetic material between two selected candidates) and select, again possibly at random, a site where the material will be exchanged, resulting in the

creation of children or offspring.
(c) Apply the secondary genetic-operator mutation
(d) Evaluate the performance of the new population
(e) Eliminate the weakest performances.

One cycle is called generation. There are following three primary genetic operators: reproduction, crossover and mutation. To implement GAs, following genetic parameters should be considered for the best performance : population size (N), crossover rate(C), mutation rate(M), selection strategy (S).

6.2. Niche and Species formation

In the optimization of multimodal functions, a simple GA cannot maintain controlled competition among the competing schemata corresponding to different peaks and the stochastic error associated with the genetic operators causes the population to converge to one alternative or other (Deb and Glodberg, 1989). Moreover, in dealing with multimodal functions with peaks of unequal value, a simple GA converges to the best peak; whereas, in addition to wanting to know the best solution, one may be interested in knowing the location of other optima. Interspecies mating in this manner tends to generate low-performance offspring. So some pressure should be maintained to cause similar individuals to mate with one another and restrict crosses between dissimilar near-optima. To overcome these limitations a natural remedy is tried. In nature, a niche is viewed as an organism's task in the environment, and a species is a collection of organisms with similar features. The subdivision of environment on the basis of an organism's role reduces inter-species competition for environmental resources, and this reduction in competition helps stable subpopulations to form around different niches in the environment. A number of methods are suggested to introduce this notion in GAs. Among them, crowding scheme by De Jong (1975) and sharing scheme by Goldberg and Richardson(1987) were suggested. In the discovery of primitives from a chart, these concepts should be adopted to prevent GA from converging to a single best primitive, because we want to discover as many primitives as possible.

7. Primitive discovery by Genetic Algorithms

As in Table 1, we set values of each attribute of trend line of stock price.

Attributes	Values
LINE-TYPE	central-line, upper-supporting-line, lower-supporting line
TOLERANCE	25,30,35,40,...,95 (for central line) 8,9,10,11, ..., 22 (for upper-supporting-line and lower-supporting-line)
SLOPE	-90 ≤ lower-bound ≤ 90, -90 ≤ upper-bound ≤ 90, lower-bound < upper-bound
DURATION	1, 2, 3, 4, ..., 45

Table 1 Boundary of values of each attribute

We used multiparameter coding. The value of an attribute is coded into a binary vector. A primitive can be represented by concatenating several vectors, and the concatenated one is called string which indicates a primitive. The string structure in genotype (the artificial chromosome or bit string) and their phenotype (the decoded parameter or parameter) are illustrated in Table 2.

```
(0 1 | 0 0 1 1 |0 1 1 1 0 1 0 1 |1 0 0 1 0 0 0 1 |0 0 0 1 1 0 |0 1 1 1 0 0)
 a1      a2          a3                  a4                a5          a6

a1 : line-type ( 0 : central line, 1:upper-supporting-line, 2:lower-supporting-line)
a2 : tolerance  ( tolerance = a2 x 5 + 25 for central line)
                ( tolerance = a2 + 8  for upper-supporting-line,
                                          lower-supporting-line)
a3 : lower-slope  ( lower-slope = a3 - 45 )
a4 : upper-slope  ( upper-slope = a4 - 45 )
a5 : lower-duration
a6 : upper-duration

(a1 a2 a3 a4 a5 a6) = (upper-supporting-line 10 22 50 6 24)
```

Table 2 Illustrative strings and their decoded values

A string consists of 0, 1 integer value in the binary scale. The attribute line-type and tolerance has categorical value. The attribute slope and duration have lower and upper bound. The range of slope represented in a string is from 0 to 180 which are converted to from -90 to 90 in the process of GAs. The example string in Table 2 represents a primitive of which line type is upper-supporting line; tolerance is 10; lower-slope is 22; upper-slope is 50; lower-duration is 6; upper-duration is 24.

The fitness of a string is the mean impact of a string that is measured from the instances of the string. The fitness can be adjusted for further purposes. Deb and Goldberg(1989) investigated several niche formation methods such as crowding scheme and sharing scheme to make the population not to converge to the one best

peak in the multimodal search space. They used a formula to measure the degree of sharing. The degree of sharing adjusts the fitness. Our approach to niche and species follows Deb and Goldberg's approach.

We derated the fitness of a string if the number of instances of the primitive is less than some threshold to acquire primitives which have as many instances as possible.

We used in-parameter crossover. That means that we do not swap all the characters in the string starting at the crossover point, but swap the characters which are only in an attribute. This situation is illustrated as in Figure 7 when attribute 2 is crossovered.

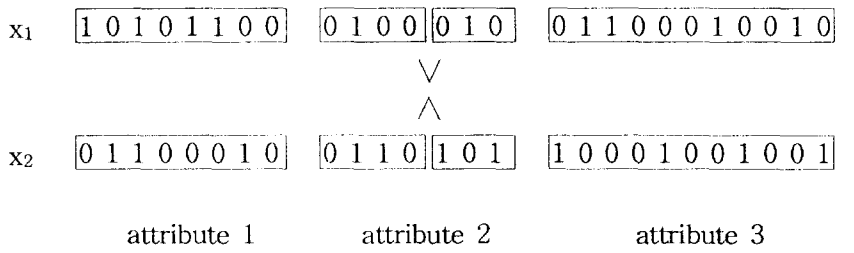


Figure 7. An illustrative in-parameter crossover

When two strings are to be crossovered, if the difference between two strings $|f(x_1) - f(x_2)|$ is greater than given threshold Δ_p, in-parameter crossovers only the attributes which have similar values between two strings. Otherwise, crossovers occur on the attributes which do not have similar values. This is because that when the difference $|f(x_1) - f(x_2)|$ is large, this difference is due to the different values of attributes between two strings. Therefore the attributes having different values should not be varied, but the attributes that have similar values between two strings can be crossovered to generate new strings. When the difference is not so different, that is, the difference $|f(x_1) - f(x_2)|$ is less than Δ_p, the attributes that have similar values should be kept, but the attributes which have the most different values between two strings can be crossovered. The similarity of two values is determined by some given threshold Δ_a.

The mutation is used only when the generated strings are identical to the pre-crossovered strings. After two strings z_i and z_j are generated from x_i and x_j after crossover and mutation if necessary, the best two strings among the x_i, x_j, z_i and z_j will remain in the new population. This is somewhat different from other GAs. This guarantees that the fitness of strings always increases generation after generation.

We used the elitist model (De Jong, 1975) to preserve the best string in the previous population.

The detailed steps of the algorithm are as follows:

Step 1. Provide initial population P(0) of size N by random sampling. Set n=0.

Step 2. Calculate the performance of each string $f(x_i)$ in population $P(n)$. Sort the performance and keep the best string x.

Step 3. Calculate the degree of sharing according to the distance $d(x_i, x_j)$ and calculate the derated performance $f_s(x_i)$ of each string.

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} s(d(x_i,x_j))}$$

Step 4. If $I(x_i)$, the number of instances of a string $x_i \leq I_{threshold}$, then derate the performance of the string:

$f_s(x_i) = \frac{f_s(x_i)}{F(I(x_i))}$ where F() is an decreasing function and $F(I_{threshold}) = 1$.

Step 5. Calculate $pselect_i = \frac{f_s(x_i)}{\sum f_s}$ and select two strings y_i and y_j using biased roulette wheel whose slot size is proportional to *pselect*.

Step 6. Calculate the difference of values of attributes between strings y_i and y_j after normalizing. Determine which attributes are similar or different from each other comparing given threshold Δ_a.

Step 7. Calculate the difference of performances y_i and y_j. If the difference is larger than given threshold Δ_p, call in-parameter crossover between similar attributes. If less than Δ_p, call in-parameter crossover between different attributes.

Step 8. Syntax check the crossovered strings z_i and z_j. If syntactically incorrect, return to step 7. However, if the repetitions are more than N_c times because of incorrect syntax, then go to step 5.

Step 9. If the newly generated string z_i or z_j is equal to one of the previous string y_i or y_j, then go to step 10; otherwise, go to step 12.

Step 10. Mutate z_i (z_j) until different $z_i(z_j)$ is generated.

Step 11. Syntax check the mutated strings. If syntactically incorrect, return to step 10. However, if the repetitions are more than N_g times because of incorrect syntax, go to step 5.

Step 12. Gather the instances of z_i and z_j. If there is no instance of z_i (or z_j), then go to step 10. However, if the repetitions are more than N_i times because of empty instance, then go to step 7.

Step 13. Calculate the performances $f(z_i)$ and $f(z_j)$.

Step 14. Insert best two strings among z_i, z_j, y_i and y_j into $P(n+1)$. If the number of $P(n+1) \leq N$ (population size), then go to step 5.

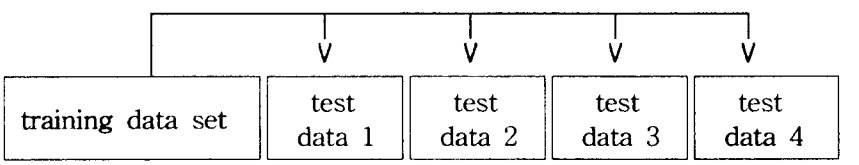
Step 15. If the best string x^* in the previous population P(n) is not in $P(n+1)$, then replace x^* with the worst string in the $P(n+1)$.

Step 16. If $n+1 \geq G_{max}$ (maximum number of generation) then stop. Otherwise, set n = n + 1 and go to step 2.

8. Experiment

8.1. Data Set

Data sets about stock prices are collected in daily basis for 61 Korean companies in variety of manufacturing industries. The data sets come from the period between January 5, 1987 and June 29, 1991, a total of 54 months (4.5 years). The front half of period is a season of bull market, and the last half period is a season of bear market. The data are divided into 17 data sets each of which has six-month interval and three-month shift between adjacent data sets. Patterns are generated from each of 15 data sets (except the last two data sets) and applied to the following 4 data sets which have three-months interval. The generated patterns from the last training data set can be applied only two three-months data sets.



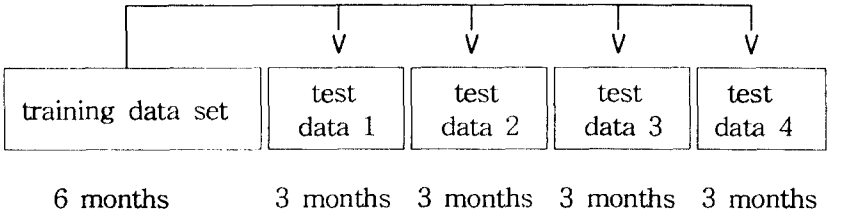


Figure 8. Diagram of training data sets and testing data sets

8.2. Definition of impact classes

We set our purpose as to predict future stock price after n days. The impact of patterns or primitives, that is, the price change after the occurrence of instances of a pattern is defined as in EQ(4).

$$\Delta P = \frac{(MAP_5(t+n) - MAP_5(t))}{MAP_5(t)} \qquad \text{EQ(4)}$$

where $MAP_n(t) = \sum_{j=t-n+1}^{t} \frac{stockprice(j)}{n}$. We used 5-days moving average in EQ(4) to smooth illegal fluctuations. We did experiment for each of value n = 10, 20, 30. The impact of patterns belongs to one of the following classes: Up, Sustained or Down. The class Up means that the price change of the individual stock is higher than the overall mean price change of the data set by 2.0%; and the class Down means the price change is lower than the overall mean price change of the data set by -2.0%; the other cases belong to the class Sustained.

8.3 Experiment I: Primitive discovery by GAs

We experimented for each two class Up and Down. The initial population size is set as 100. Initial population is set randomly, but strings were discarded that average ΔP of the instances is not greater than zero for class Up and not less than zero for class Down. The number of generations is 45. We experimented 3 prediction interval by setting n as 10, 20, 30 in ΔP defined in EQ(4). Figure 9 summarizes mean of normalized price change of discovered primitives at each generation, prediction interval and class. Here, the definition of normalized price

change is as in EQ(5)

$$\frac{\Delta P - P^*}{S^*} \qquad \text{EQ(5)}$$

where P^* is overall mean price change and S^* is overall standard deviation of price change of a data set. The one dot in Figure 9 represents that the averaged EQ(5) for all primitives in all 15 training data set at each generation. The mean of normalized price change is increasing for class Up and decreasing for class Down after generation and generation. But the slope becomes flat as generation goes.

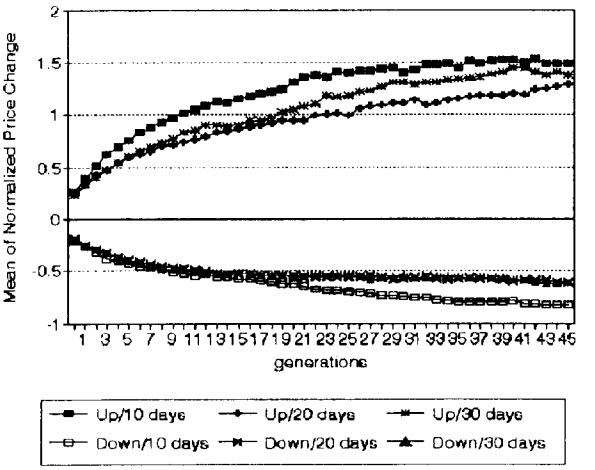


Figure 9. The change of mean of normalized price change at each generation

Figure 10 and 11 shows change of credibility(CR) as generation goes for each class and prediction interval. The CR of a primitive(or pattern) P that belong to class C is defined as EQ(6).

$$CR(C|P) = \frac{\text{number of instances that belong to the class C in the instances of the pattern P}}{\text{number of instances of the pattern P}} \qquad \text{EQ(6)}$$

In Figure 10 and 11, the CR is increasing and reaches to around 0.8.

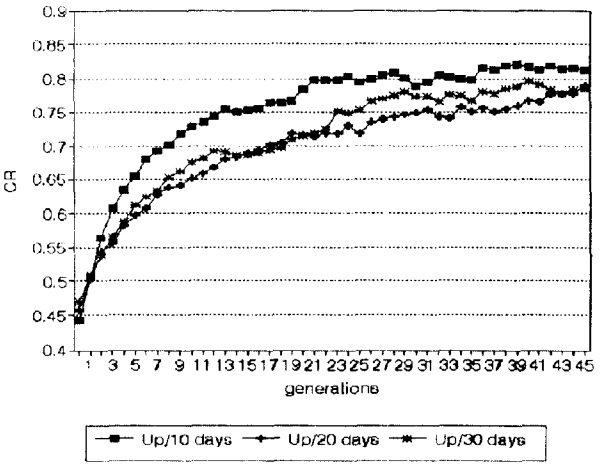


Figure 10. Average credibility for class Up at each generation

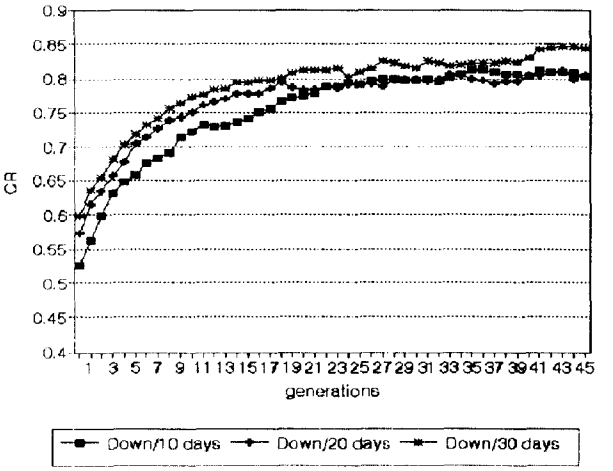
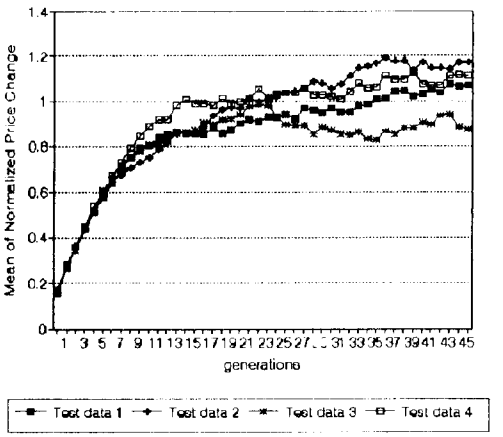
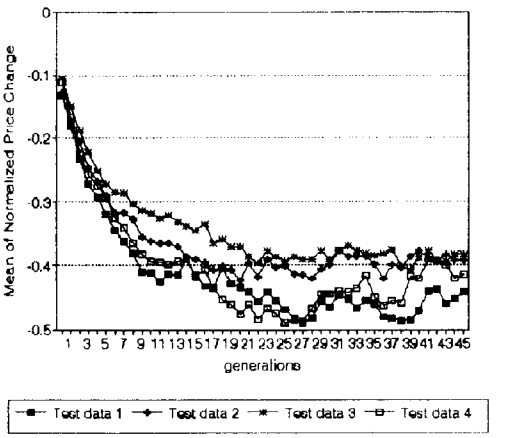


Figure 11. Average credibility for class Down at each generation

We applied the generated primitives (the simplest pattern) into following 4 data set. Each testing data set has 3-month interval. The discovered primitives are recognized in the testing data set if they occur, and the mean of normalized price change after n (n=10, 20, 30) days from the occurrence and average credibilities are calculated. Figure 12 shows mean of normalized price change of primitive set at

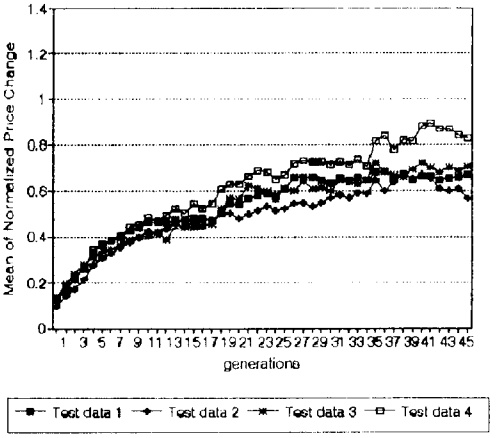
each generation. The performance becomes good as generation goes, but their marginal prediction performance enhancement disappears. 21 ~ 25 generations are enough in the point of predicting performance. The point we should focus is that prediction performances for 10 days interval is much greater than those of 20 or 30 days.

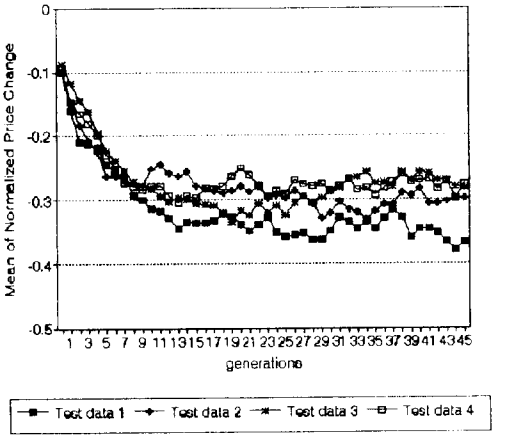




Class Up Class Down

Prediction interval = 10 days





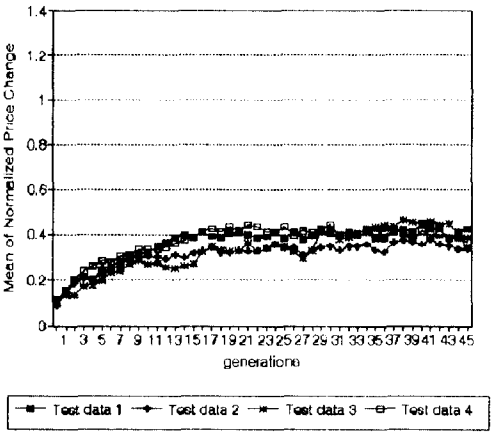
Class Up Class Down

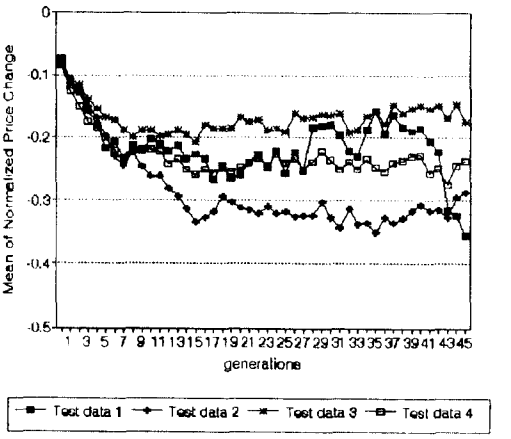
Prediction interval = 20 days

Class Up Class Down

Prediction interval = 30 days

Figure 12. Mean of normalized price change at 4 testing data sets





This means that the shorter the prediction interval, the better the performance is. However, there is no clear difference in performance between testing data set. This means that the time gap between training data set and testing data set is not important.

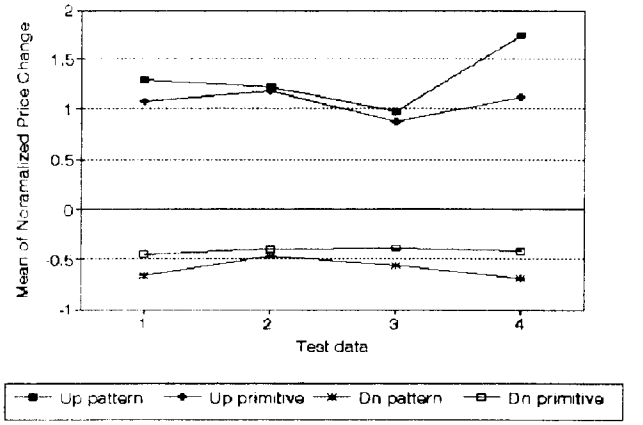
8.4. Experiment II: Pattern generation by SYNPLE

The primitives discovered at the last generation(45th generation) at each data set were selected for further synthesis. SYNPLE algorithm is applied and the credibility of generated patterns are enhanced as shown is Table 3. The credibility of Up and Down classes are all 1.0.

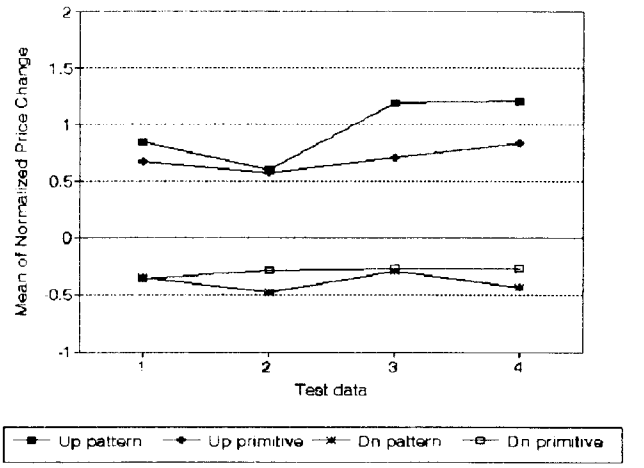
		Mean of Normalized Price Change	Average Credibiliry
Up	10 days	2.8348069	1.0
	20 days	2.4713760	1.0
	30 days	2.4180962	1.0
Down	10 days	-1.5638524	1.0
	20 days	-1.1376825	1.0
	30 days	-1.1913708	1.0

Table 3 Performance of generated patterns at training data sets

The generated patterns at each training data set were tested in the following 4 data sets. The performances are compared to the performance of the primitives as in Figure 13.



Prediction interval = 10 days



Prediction interval = 20 days

Prediction interval = 30 days

Figure 13. Mean of normalized price change at 4 testing data sets comparing the generated patterns to primitives

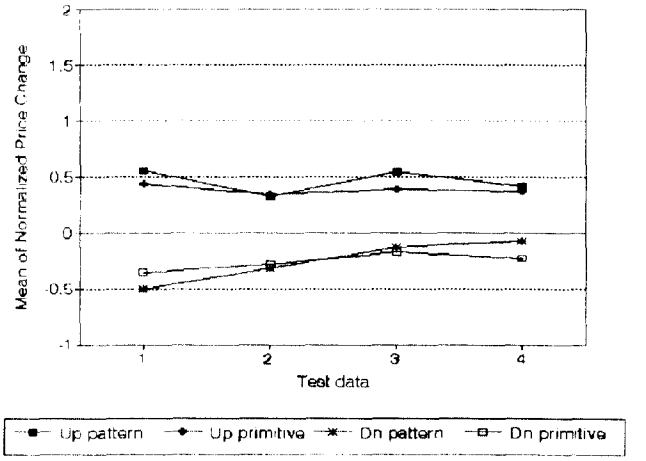


Figure 13 shows that the patterns generated by SYNPLE from the primitives are better in performance. However, in prediction interval 30, the performance difference is not so clear.

The summary of performance of generated patterns are as in Table 4. The maximum predicting power is 79% when prediction term is 10 days in class Up. The predicting power decreases as predicting interval becomes long.

			Data 1	Data 2	Data 3	Data 4	Average
Up	10	MPC	1.292812	1.214327	0.975668	1.733589	1.304099
		CR	0.780798	0.816948	0.763569	0.837257	0.799643
	20	MPC	0.840737	0.598807	1.182644	1.206410	0.957150
		CR	0.710394	0.794251	0.765568	0.718978	0.747298
	30	MPC	0.550017	0.324004	0.540666	0.407495	0.455546
		CR	0.668366	0.566821	0.507219	0.473639	0.554011
Down	10	MPC	-0.659320	-0.458730	-0.560900	-0.691960	-0.592730
		CR	0.711127	0.660827	0.701842	0.774276	0.712018
	20	MPC	-0.359950	-0.491540	-0.307440	-0.437880	-0.399200
		CR	0.641939	0.757704	0.596791	0.704898	0.675333
	30	MPC	-0.504780	-0.321560	-0.133820	-0.076510	-0.259170
		CR	0.776524	0.632607	0.618548	0.556980	0.646165

* MPC = Mean of normalized price change

Table 4 Performance of generated patterns

9. Concluding remarks

We have discovered primitives by GAs. The experimental results show that the primitives are meaningful for predicting future stock price. The synthesized patterns from the primitives can perform better than the primitives themselves in 10 and 20 days prediction. The performances are not influenced by the time gap between testing data sets. We can conclude that from our experiments GAs can discover credible primitive patterns in trend line of stock price for short term prediction. The further research points are that extending experiments to other charts and synathesizing primitives from them. Secondly, experimenting in homogeneous market phase or homogeneous industry stocks to know whether the performance can be increased if stocks are in one class of industry or in one phase of market cycle such as bull or bear market.

- REFERENCES -

Austin, S. (1990a). An introduction to genetic algorithms. *AI Expert* 5(3), 48-53.

Deb, Kalyanmoy, & Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 42-50). CA: Morgan Kaufmann.

De Jong, K. (1988). Learning with genetic algorithms : An overview. *Machine Learning,* 3, 121-138.

Fu, K. S. (1982). *Syntactic pattern recognition and application.* New jersey: Prentice Hall.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the Second International Conference on Genetic Algorithms.* New Jersey: Lawrence Erlbaum.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning.* Mass: Addison-Wesley.

Granville, J. E. (1976). *A strategy of daily stock market timing for maximum profit.* Englewood Cliffs: Prentice-Hall.

Grefenstette, J. J. (1986). Optimization of Control Parameters for Genetic Algorithms. IEEE Trasactions on Systems, Man, and Cybernetics, SMC-16(1), 122-128.

Kim, H.. S. (1992). Syntactic Pattern Based Inductive Learning and Primitive

Discovery for Chart Analysis, Doctoral dissertation, Department of Management Science, Korea Advanced Institute of Science and Technology.

Lee, J. K., & Kim, H. S. (1990). Syntactic pattern based inductive learning for chart analysis: K-INDUCE. The Third International Symposium on Expert Systems in Business, Finance and Accounting. CA: University of Southern California, School of Accounting.

Lee, J. K., Kim, H. S., & Trippi, R. (1992). Security Trading Rule Synthesis: A Syntactic Pattern-Based Learning Approach, forthcoming in *Heuristics*.

Schaffer, J. D., Caruana, R. A., Eshelman, L. J., & Das, Rajarshi. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. Proceedings of the Third International Conference on Genetic Algorithms (pp. 51-60). CA: Morgan Kaufmann.

You, K. C., & Fu, K. S. (1979). A syntactic approach to shape recognition using attributed grammars. IEEE Trans. Syst. Man Cybern., SMC-9(6), 334-345.