Developing a Market Timing System using Grammatical Evolution

Michael O'Neill
Dept. Of Computer Science And Information Systems,
University of Limerick, Ireland.

Tony Brabazon
Dept. Of Accountancy,
University College Dublin, Ireland.

Conor Ryan & J.J. Collins
Dept. Of Computer Science And Information Systems,
University of Limerick, Ireland.

Abstract

This study examines the potential of an evolutionary automatic programming methodology, Grammatical Evolution, to uncover a series of useful fuzzy technical trading rules for the ISEQ, the official equity index of the Irish Stock Exchange. Index values for the period 29/03/93 to 4/12/1997 are used to train and test the model. The preliminary findings indicate that the methodology has much potential.

1 Introduction

We have previously evolved trading rules for the UK FTSE 100 index [O'Neill et.al. 2001], and now wish to extend this approach to new markets, and through the inclusion of additional technical indicators.

1.1 Technical analysis

A market index is comprised of a weighted average measure of the price of individual shares which make up that market. The value of the index represents an aggregation of the balance of supply and demand for these shares. Some market traders, known as technical analysts, believe that prices move in trends and that price patterns repeat themselves [Murphy 1999]. If we accept this premise, that there are rules, although not necessarily static rules, underlying price behaviour, it follows that trading decisions could be enhanced through use of an appropriate rule induction methodology such as Grammatical Evolution (GE).

Although controversy exists amongst financial theorists regarding the veracity of the claim of technical analysts, recent evidence has suggested that it may indeed be possible to uncover patterns of predictability in price behaviour. Brock, Lakonishok and LeBaron [Brock, Lakonishok & LeBaron 1992] found that simple technical trading rules had predictive power and suggested that the conclusions of earlier studies that technical trading rules did not have such power were "premature". Other studies which indicated that there may be predictable patterns in share price movements include those which suggest that markets do not always impound new information instantaneously [Hong, Lim, & Stein 1999] [Chan, Jegadeesh & Lakonishok 1996], that stock markets can overreact as a result of excessive investor optimism or pessimism [Dissanaike 1997] and that returns on the market are related to the day of the week [Cross 1973] or the month of the year [DeBondt & Thaler 1987]. The continued existence of large technical analysis departments in international finance houses is consistent with the hypothesis that technical analysis has proven empirically useful.

1.2 Potential for application of evolutionary automatic programming

As noted by Iba and Nikolaev [Iba & Nikolaev 2000] there are a number of reasons to suppose that the use of an evolutionary automatic programming (EAP) approach can prove fruitful in the financial prediction domain. EAP can conduct an efficient exploration of the search space and can uncover dependencies between input variables, leading to the selection of a good subset for inclusion in the final model. Additionally, use of EAP facilitates the utilisation of complex fitness functions including discontinuous, non-differentiable functions. This is of particular importance in the financial domain as the fitness criterion may be complex, usually requiring a balancing of return and risk. EAP, unlike, for example, basic neural net approaches to financial prediction, does not require the ex-ante determination of optimal model inputs and their related transformations. Another useful feature of EAP is that it produces human-readable rules that have the potential to enhance understanding of the problem domain.

1.3 Motivation for study

This study was motivated by a number of factors. Much of the existing literature concerning the application of genetic algorithms (GA) or GP to the generation of technical trading rules [Allen & Karjalainen 1999] [Colin 1994] [Bauer 1994] [Neely, Weller & Dittmar 1997] [Deboeck 1994] concentrates on the US and to a lesser extent the Japanese stock markets. Published research on this area is both incomplete and scarce. To date, only a limited number of GA / GP methodologies and a limited range of technical indicators have been considered. This study addresses these limitations by examining index data drawn from the Irish stock market and by adopting a novel evolutionary automatic programming approach.

The paper is organised as follows. Section two discusses the background to the technical indicators utilised in this study. Section three describes the evolutionary algorithm adopted, Grammatical Evolution [O'Neill & Ryan 2001] [Ryan et.al. 1998]. Section four outlines the data and function sets used. The following sections provide the results of the study followed by a discussion of these results and finally a number of conclusions are derived.

2 Background

As with any modelling methodology, issues of data preprocessing need to be considered. Rather than attempting to uncover useful technical trading rules for the ISEQ index using raw current and historical price information, this information is initially pre-processed into technical indicators. The objective of these pre-processing techniques is to uncover possible useful trends and other information in the time series of the raw index data whilst simultaneously reducing the noise inherent in the series.

2.1 Technical Indicators

The development of trading rules based on current and historic market price information has a long history [Brown, Goetzmann & Kumar 1998]. The process entails the selection of one or more technical indicators and the development of a trading system based on these indicators. These indicators are formed from various combinations of current and historic price information. Although there are potentially an infinite number of such indicators, the financial literature suggests that certain indicators are widely used by investors [Brock, Lakonishok & LeBaron 1992][Murphy 1999] [Pring 1991].

Four groupings of indicators are given prominence in prior literature:

i. Moving average indicators

ii. Momentum indicators

iii. Trading range indicators

iv. Oscillators

Given the large search space, an evolutionary automatic programming methodology has promise to determine both a good quality combination of, and relevant parameters for, trading rules drawn from individual technical indicators.

We intend to use of each of these groupings as our model is developed. Our initial study on the FTSE dataset [O'Neill et.al. 2001] included only a moving average indicator. This study also includes momentum, and trading range volatility indicators.

2.1.1 Moving Average Indicators

The simplest moving average systems compare the current share price or index value with a moving average of the share price or index value over a lagged period, to determine how far the current price has moved from an underlying price trend. As they smooth out daily price fluctuations, moving averages can heighten the visibility of an underlying trend. A variation on simple moving average systems is to use a moving average convergence divergence (MACD) oscillator. This is calculated by taking the difference of a short run and a long run moving average. In a recursive fashion, more complex combinations of moving averages of values calculated from a MACD oscillator can themselves be used to generate trading rules. For example, a nine day moving average of a MACD oscillator could be plotted against the raw value of that indicator. A trading signal may be generated when the two plotted moving averages cross. Moving average indicators are trend following devices and work best in trending markets. They can have a slow response to changes in trends in markets, missing the beginning and end of each move. They tend to be unstable in sideways moving markets, generating repeated buy and sell signals (whipsaw) leading to unprofitable trading. Trading systems using moving averages trade-off volatility (risk of loss due to whipsaw) against sensitivity. The objective is to select the lag period which is sensitive enough to generate a useful early trading signal but which is insensitive to random noise.

2.2 Momentum

The momentum of a security is the ratio of a time-lagged price to the current price ($\frac{Price_t}{Price_{t-x}}$). The belief underlying this indicator is that strongly trending shares tend to continue to move in that direction for a period of time as more investors buy or sell the trending share. There is recent evidence that momentum trading strategies can work, particularly when investing in smaller firms [Hong, Lim, & Stein 1999]. Technical analysts consider that price momentum can foretell a price turning point as momentum will tend to peak before the price peaks.

2.3 Trading Range Breakout systems

It these systems, a signal is usually generated if a price breaks out of a defined range. A simple example of a trading rule would be to buy a share when it exceeds its previous high in the last four weeks and conversely to sell a share if it falls below its previous four week low. A more complex approach is to plot an envelope at $\pm$ 'x' standard deviations above and below a moving average. Penetration of the bands by the current day's price, indicates a possible price trend reversal.

A description of the evolutionary automatic programming system used to evolve trading rules now follows.

3 Grammatical Evolution

Grammatical Evolution (GE) is an evolutionary algorithm that can evolve computer programs in any language. Rather than representing the programs as parse trees, as in traditional GP [Koza 1992], a linear genome representation is adopted. A genotype-phenotype mapping process is used to generate the output program for each individual in the population. Each individual, a variable length binary string, contains in its codons (groups of 8 bits) the information to select production rules from a Backus Naur Form (BNF) grammar. The BNF is a plug-in component to the genotype-phenotype mapping process, that represents the output language in the form of production rules. It is comprised of a set of non-terminals that can be mapped to elements of the set of terminals, according to the production rules. An example excerpt from a BNF grammar is given below. These productions state that S can be replaced with either one of the non-terminals `expr`, `if-stmt`, or `loop`.

```
S ::= expr       (0)
    | if-stmt    (1)
    | loop       (2)
```

The grammar is used in a generative process to construct a program by applying production rules, selected by the genome, beginning from the start symbol of the grammar.

In order to select a rule in GE, the next codon value on the genome is generated and placed in the following formula:

$$Rule = Codon\ Value\ MOD\ Num.\ Rules$$

If the next codon integer value was 4, given that we have 3 rules to select from as in the above example, we get $4\ MOD\ 3 = 1$. S will therefore be replaced with the non-terminal `if-stmt`.

Beginning from the left hand side of the genome codon integer values are generated and used to select rules from the BNF grammar, until one of the following situations arise:

i. A complete program is generated. This occurs when all the non-terminals in the expression being mapped, are transformed into elements from the terminal set of the BNF grammar.

ii. The end of the genome is reached, in which case the *wrapping* operator is invoked. This results in the return of the genome reading frame to the left hand side of the genome once again. The reading of codons will then continue unless an upper threshold representing the maximum number of wrapping events has occurred during this individual's mapping process.

iii. In the event that a threshold on the number of wrapping events is exceeded and the individual is still incompletely mapped, the mapping process is halted, and the individual assigned the lowest possible fitness value.

GE uses a steady state replacement mechanism, such that, two parents produce two children the best of which replaces the worst individual in the current population if the child has a greater fitness. The standard genetic operators of point mutation, and crossover (one point) are adopted. It also employs a duplication operator that duplicates a random number of codons and inserts these into the penultimate codon position on the genome. A full description of GE can be found in [O'Neill & Ryan 2001] [Ryan et.al. 1998].

4 Problem Domain & Experimental Approach

We describe an approach to evolving trading rules using GE. This study uses daily data for the Irish ISEQ stock index drawn from the period 29/03/1993 to 4/12/1997. The training data set was comprised of 360 days from the first day (29/03/1993) with an additional 75 days at the beginning of this time to allow for the time lag introduced with technical indicators such as the moving average. The remaining data was divided into two hold out samples totaling 805 (i.e. two 365 day periods with a time lag of 75 days) trading days. The division of the hold out period into two segments was undertaken to allow comparison of the out of sample results across different market conditions, in order to assess the stability and degradation characteristics of the developed model's predictions.

The rules evolved by GE are used to generate one of three signals for each day of the training or test periods. The possible signals are *Buy*, *Sell*, or *Do Nothing*. Permitting the model to output a Do Nothing signal reduces the hard threshold problem associated with production of a binary output. This issue has not been considered in a number of prior studies. A variant on the trading methodology developed in Brock et al. [Brock, Lakonishok & LeBaron 1992] is then applied. If a buy signal is indicated, a fixed investment of \$1,000 (arbitrary) is made in the market index. This position is closed at the end of a ten day (arbitrary) period.

On the production of a sell signal, an investment of \$1,000 is sold short and again this position is closed out after a ten day period. This gives rise to a maximum potential investment of \$10,000 at any point in time (the potential loss on individual short sales is in theory infinite but in practice is unlikely to exceed \$1,000). The profit (or loss) on each transaction is calculated taking into account a one-way trading cost of 0.2% and allowing a further 0.3% for slippage. The total return generated by the developed trading system is a combination of its trading return and its risk free rate of return generated on uncommitted funds.

The rate adopted in this calculation is simplified to be the average interest rate over the entire data set (5.8%).

As well as the moving average, the momentum and trading range volatility technical indicators are adopted in these preliminary experiments.

In addition to the technical indicators the grammar also allows the use of the binary operators f_and, f_or, and the standard arithmetic operators, and the unary operator f_not [1].

The signals generated for each day, Buy, Sell, or Do Nothing, are post-processed using fuzzy logic. The trading rule, a fuzzy trading rule, returns values in the range 0 to 1. We use pre-determined membership functions, in this case, to determine what the meaning of this value is. The membership functions adopted were as follows:

$$Buy = Value < .33$$

$$Sell = .33 >= Value < .66$$

$$DoNothing = .66 >= Value$$

4.1 Data Preprocessing

The value of the ISEQ index increased substantially over the training and testing period, rising from 1337.44 to 3498.84. Before the trading rules were constructed, these values were normalised using a two phase preprocessing. Initially the daily values were transformed by dividing them by a 75 day lagged moving average. These transformed values are then normalised using linear scaling into the range 0 to 1. This procedure is a variant on that adopted by Allen and Karjalainen [Allen & Karjalainen 1999] and Iba and Nikolaev [Iba & Nikolaev 2000].

[1]The operations f_and, f_or, and f_not are fuzzy logic operators returning the minimum, maximum, of the arguments, and 1 - the argument, respectively.

4.2 Selection of Fitness Function

A key decision in applying a GP methodology to construct a technical trading system is to determine what fitness measure should be adopted. A simple fitness measure such as the profitability of the system both in and out of sample is inadequate as it fails to consider the risk associated with the developed trading system. The risk of the system can be estimated in a variety of ways. One possibility is to consider market risk, defined here as the risk of loss of funds due to a market movement. A measure of this risk is provided by the maximum drawdown (maximum cumulative loss) of the system during a training or test period. This measure of risk can be incorporated into the fitness function in a variety of formats including: (return / maximum drawdown) or return - 'x'(maximum drawdown), where 'x' is a pre-determined constant dependent on an investor's psychological risk profile. For a given rate of return, the system generating the lowest maximum drawdown is preferred.

This study incorporates drawdown in the fitness function by subtracting the maximum cumulative loss during the training period from the profit generated during that period. This is a conservative approach which will encourage the evolution of trading systems with good return to risk characteristics. This will provide a more stringent test of trading rule performance as high risk / high reward trading rules will be discriminated against.

5 Results

The results from our preliminary experiments are now given. Runs were conducted with a population size of 500 for 100 generations. Trading rules were evolved with a performance superior to that of a benchmark buy and hold strategy. Under this benchmark, an amount of \$10,000 is invested in the market at the beginning of each of the test periods. The gain on this investment to the end of each period is then calculated. The best individual (set of trading rules) found to date made a profit of US\$5777 over the training period.

When tested on the two out of sample periods following the training data set we find that this individual was consistently profitable. Plots of the index over each of the test periods and the training period can be seen in Figure 1.

To facilitate assessment of these results, they are compared with those of the benchmark buy and hold strategy. The results of this buy and hold strategy can be seen in Table 1.

In assessing these results, the market risk profile of each trading strategy should be considered. The buy and hold strategy maintains an investment of \$10,000 plus cumulative returns since the start of the investment period, in the market at all times whereas the maximum investment of the developed trading system, ignoring drawdown, is \$10,000. Looking

Trading Period (Days)	**Buy & Hold** Profit (US$)	**Best-of-run** Profit(US$)	**Best-of-run** Avg. Daily Investment
Train (1095 to 1460)	2087	5777	8233
Test 1 (1461 to 1826)	3848	1478	5055
Test 2 (1827 to 2192)	3439	2627	8808
Total	9374	9882	

Table 1: A comparison of benchmarks with the best of run individual.

at Table 1 we can see the average daily investment made by the best of run individual for each test period. Averaged over all the test periods the developed system has an investment of $6932 in the market.

There is no clear evidence that the trading system has higher market risk than the buy and hold strategy.

6 Discussion

In evaluating the performance of any market predictive system, a number of caveats must be borne in mind. Any trading model constructed and tested using historic data will tend to perform less well in a live environment than in a test period for a number of reasons. Live markets have attendant problems of delay in executing trades, illiquidity, interrupted / corrupted data and interrupted markets. The impact of these issues is to raise trading costs and consequently to reduce the profitability of trades generated by any system. An allowance for these costs ('slippage') has been included in this study but it is impossible to determine the scale of these costs ex-ante with complete accuracy. In addition to these costs, it must be remembered that the market is competitive. As new computational technologies spread, opportunities to utilise these technologies to earn excess risk-adjusted profits are eroded. As a result of this technological 'arms-race', estimates of trading performance based on historical data may not be replicated in live trading as other market participants will apply similar technology. This study ignores impact of dividends. Although a buy-and-hold strategy will generate higher levels of dividend income than an active trading strategy, the precise impact of this factor is not determinable ex-ante. It is notable that the dividend yield on most stock exchanges has fallen sharply in recent years and that the potential impact of this factor has lessened.

7 Conclusions & Future Work

The results clearly show that the model is suffering from over-fitting to the training data set, however, the risk involved with the adoption of the evolved trading rules is less than the benchmark buy and hold strategy.

The risk of the benchmark buy-and-hold portfolio exceeded that of the portfolio generated by the technical trading rules because, the benchmark buy and hold portfolio maintains a fully invested position at all times in the market, whereas the portfolio generated by the technical trading system averaged a capital investment of $7,000 over the test periods.

There is notable scope for further research utilising GE in this problem domain. Our preliminary methodology has included a number of simplifications, for example, we only considered a small set of primitive technical indicator. The incorporation of additional technical indicators may further improve the performance of our approach. Future work will also seek to remove the over-fitting that is occurring on the ISEQ data set in order to achieve the generalisation properties demonstrated by the evolved rules on the FTSE data set [O'Neill et.al. 2001].

References

[Allen & Karjalainen 1999] Allen, F., Karjalainen, R. (1999) Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, **51**, pp. 245-271, 1999.

[Bauer 1994] Bauer R. (1994). Genetic Algorithms and Investment Strategies, New York: John Wiley & Sons Inc.

[Brock, Lakonishok & LeBaron 1992] Brock, W., Lakonishok, J. and LeBaron B. (1992). 'Simple Technical Trading Rules and the Stochastic Properties of Stock Returns', Journal of Finance, 47(5):1731-1764.

[Brown, Goetzmann & Kumar 1998] Brown, S., Goetzmann W. and Kumar A. (1998). 'The Dow Theory: William Peter Hamilton's Track Record Reconsidered', Journal of Finance, 53(4):1311-1333.

[Chan, Jegadeesh & Lakonishok 1996] Chan, L. K. C., Jegadeesh, N. and Lakonishok, J. (1996). 'Momentum strategies', Journal of Finance, Vol. 51, No. 5, pp. 1681 - 1714.

[Colin 1994] Colin, A. (1994). 'Genetic Algorithms for Financial Modelling', in Guido Deboeck (Editor) (1994). Trading on the edge: neural, genetic and fuzzy systems for chaotic and financial markets, New York: John Wiley & Sons Inc.

[Cross 1973] Cross, F. (1973). 'The Behaviour of Stock prices on Friday and Monday', Financial Analysts' Journal, Vol. 29(6), pp.67-74.

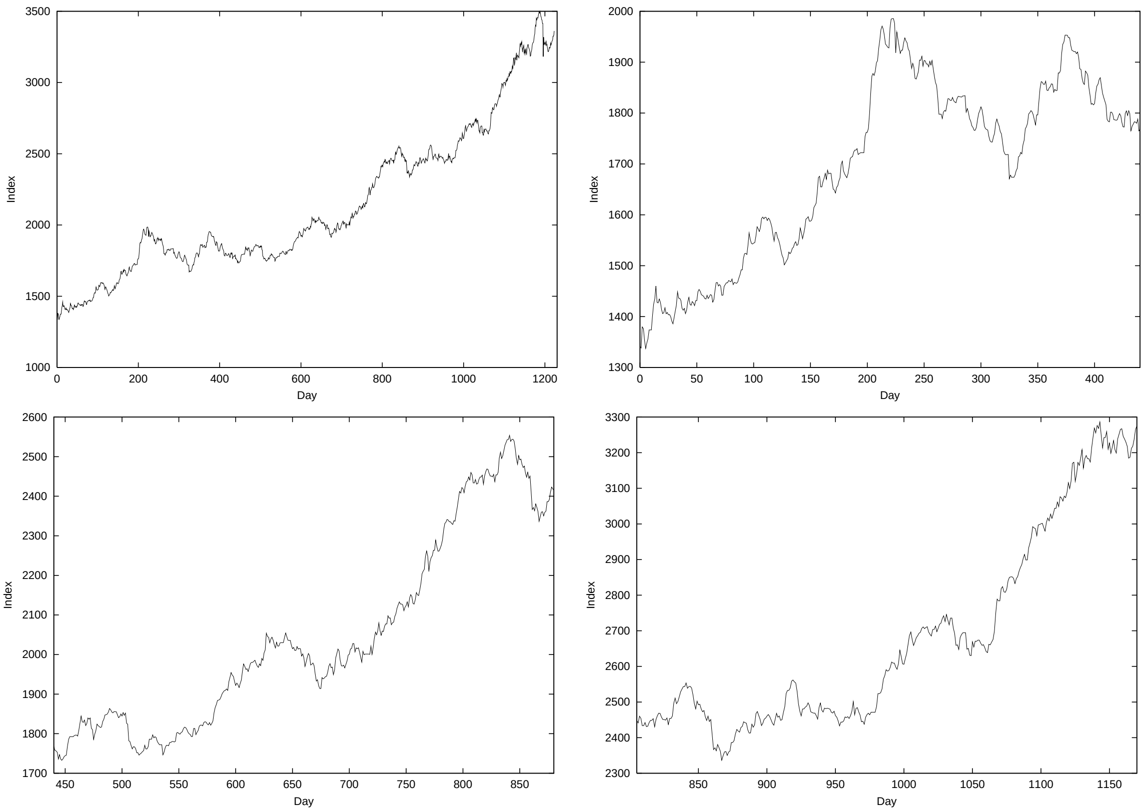

Figure 1: A plot of the ISEQ over the entire data set (top left), over the training period (top right), over the two test periods (bottom left and right respectively).

[Deboeck 1994] Deboeck G. (1994). 'Using GAs to optimise a trading system', in Guido Deboeck (Editor) (1994). Trading on the edge: neural, genetic and fuzzy systems for chaotic and financial markets, New York: John Wiley & Sons Inc.

[DeBondt & Thaler 1987] DeBondt, W. and Thaler, R. (1987). 'Further Evidence on Investor Overreaction and Stock Market Seasonality', Journal of Finance, Vol. 42(3):pp.557-581.

[Dissanaike 1997] Dissanaike, G. (1997). 'Do stock market investors overreact?', Journal of Business Finance & Accounting (UK), Vol. 24, No.1, pp. 27-50.

[Hong, Lim, & Stein 1999] Hong, H., Lim, T. and Stein, J. (1999). 'Bad News Travels Slowly: Size, Analyst Coverage and the Profitability of Momentum Strategies', Research Paper No. 1490, Graduate School of Business, Stanford University.

[Iba & Nikolaev 2000] Iba H. and Nikolaev N. (2000). 'Genetic Programming Polynomial Models of Financial Data Series', In *Proc. of CEC 2000*, pp. 1459-1466, IEEE Press.

[Koza 1992] Koza, J. (1992). *Genetic Programming*. MIT Press.

[Murphy 1999] Murphy, John J. (1999). Technical Analysis of the Financial Markets, New York: New York Institute of Finance.

[Neely, Weller & Dittmar 1997] Neely, C., Weller P. and Dittmar, R. (1997). 'Is technical analysis in the foreign exchange market profitable? A genetic programming approach", Journal of Financial and Quantitative Analysis, Vol. 32, No. 4, pp. 405 - 428.

[O'Neill et.al. 2001] O'Neill M., Brabazon, A., Ryan C., & Collins J.J. (2001). Evolving Market Index Trading Rules Using Grammatical Evolution. In Proceedings of EvoIASP 2001.

[O'Neill & Ryan 2001] O'Neill M., Ryan C. (2001). Grammatical Evolution. *IEEE Trans. Evolutionary Computation*.

[Pring 1991] Pring, M. (1991). Technical analysis explained: the successful investor's guide to spotting investment trends and turning points, New York: Mc Graw-Hill Inc.

[Ryan et.al. 1998] Ryan C., Collins J.J., O'Neill M. (1998). Grammatical Evolution: Evolving Programs for an Arbitrary Language. *Lecture Notes in Computer Science*

1391, Proceedings of the First European Workshop on Genetic Programming, pages 83-95. Springer-Verlag.

A Grammar used by GE to evolve trading rules

```
N={<code>,<expr>,<fopbi>,<fopun>,
   <matbi>,<relbi>,<var>,<int>}

T={p,=,(,),f_and,f_or,f_not,+,-,*,
   >,<,>=,<=,scale,ma,day,
   1,2,3,4,5,10}

S=<code>

P=

<code> ::=  p = <expr> ;

<expr> ::= <fopbi> (<expr>, <expr>)
         | <fopun> (<expr>)
         | <expr><matbi><expr>
         | <expr><relbi><expr>
         | <var>

<fopbi> ::= f_and
          | f_or

<fopun> ::= f_not

<matbi> ::= + | - | *

<relbi> ::= > | < | >= | <=

<var> ::= scale( <int> )
        | scale(ma( <int> , day ))
        | scale(day)
        | scale(momentum( <int> , day ))
        | scale(trb( <int> , day ))

<int> ::= 1 | 2 | 3 | 4 | 5 | 10
```