

# Commodity Trade Automation using Evolution and Pattern Recognition

Andrzej Ruta

AGH University of Science and Technology  
Institute of Computer Science

Al. Mickiewicza 30, 30-059 Krakow, Poland

Phone: +48-12-6173497, Fax: +48-12-6339406

email: [dimas2@poczta.onet.pl](mailto:dimas2@poczta.onet.pl)

**ABSTRACT:** This paper is to present a novel approach to market prediction in terms of commodity trade optimisation. It is based on the abstraction of evolving "market translators", understood as popular statistical indicators that can collectively adapt to the price data being described. Each possible subset of available indicators forms a subspace in which raw quotes can be seen from a unique viewpoint, perhaps suitable to identify profitable entry and exit points for the investment. The main idea behind this work is to: 1) treat market indicators as specimens that can evolve within population, 2) treat the return on investment made with the data projected to a given indicator subspace as a driving force for this evolutionary process. It is shown through the test results a prototype decision-support system has demonstrated, that following above principles directs the search towards finding the best possible subset to pick for real-time trading.

**KEYWORDS:** market trade automation, decision support, time-series analysis, evolution, feature subset optimisation, pattern recognition, classification

## INTRODUCTION

In the era of global trade the opportunities for investment are so broad that it is hard for the ordinary people not to find themselves lost in a multitude of offers. Yet, being constantly out of time, many of us are not prepared for managing owned capital ourselves. Hence, we are confined to various forms of active money-saving and portfolio management offered by banks, investment funds and other financial institutions. More aware and passionate individuals frequently take the risk of playing the market game individually, for example by trade on the equity or futures market. In this multiplicity there is an enormous potential for computer aid and automation, yet still too few real-life, efficient, trading systems. In this respect, we could pose a question, why it is so difficult to overcome machine's lack of intelligence and why the market analysts in the beginning of 21st century still so much rely on their expert knowledge and premonition, but in the same time cannot deliver that expertise to a computer system.

In pursue of improving predictability of financial time series many analysts and researchers have challenged the above question. Frequently, the approach taken is based on the assumption that the historical prices can be treated as a realisation of some stochastic process. Then, given the observed data points up to time  $T$  as random variables  $x_1, x_2, \dots, x_T$ , the prediction is defined in terms of estimation of future variables  $x_{T+1}, x_{T+2}, \dots$ . The drawback of the linear regression-based models like ARMA [2], [3] is their limited ability to capture complex nonlinearities in the price behaviour and, in turn, low reliability. Non-linear models, e.g. ARCH/GARCH [2], [3], although able to overcome this problem to some extent, exhibit another, somewhat common disadvantage – the lack of deeper, context-level association with the domain of market trade. Finally, there is a long way from the estimation of future price to yielding actual trading order, the latter needed to be done by the human in the course of further strenuous analysis.

The rule-based models [3], [4], [5] represent what is probably the current mainstream of automatic trading. They do not constitute the homogenous class of systems, but rather define a general methodology to deal with the problem: 1) find potentially favourable entry and exit points for the investment, 2) formulate criterions for their use in generating trading orders. Rule-based trading algorithms are widespread. They can be found in most of the existing decision-support applications, e.g. *Metastock*. Frequently they are very efficient, however the latter very much depends upon the degree to which the price behaves in a way considered typical for the particular set of rules. Requirement for user's domain-level expertise is among the main deficiencies.

More recently, different techniques were applied to financial prediction. The feed-forward neural networks have been used to implement a particular form of nonlinear, multiple regression by prediction of some target variable,

based on the input values. In [6] it is shown that the prediction of Radial Basis Function Neural Network model for forecasting currency exchange rates is significantly better than the prediction of a traditional autoregressive model in both directional change and estimation of the exchange rate itself. In [5] the training on the whole portfolio of tradables is proposed for prediction of the current or the near-future market behaviour. A suggested approach is to develop the neural network to predict both: where the market is in terms of its near-future range and whether tomorrow's open represents a turning point. Also the techniques based on genetic algorithms and Bayesian Networks have been applied to the problem. For example, in [7] a genetic algorithm is utilised to synthesise a Bayesian belief net from a stream of temporal currency data. Then the resulting network is used for forecasting its future trend.

While evolutionary algorithms and pattern recognition techniques have been successfully applied to solve a number of problems in a range of disciplines, very little has been done to popularise them in the world of finance. It seems that what lacks in particular is a self-contained decision-support system that would incorporate these techniques and not only predict the future state of the market, but most essentially convert its findings into a clear, present action suggestion, comprehensible to inexperienced users. In this article an effort has been made to fill in this gap. A novel approach to the automatic trading is presented. Inspired by some university lectures [8], [9], interest in economy and a little author's personal experience in share trade, a model for yielding current-day trading orders is built. Proposed decision-support system attempts to infer the market rules from the historical data, unbiased by hindsight, and use them to generate real-time, profitable trading orders without human judgement or intervention. The alternative concept, contrary to most of the existing designs, introduces intelligent, nature-inspired mechanisms that have already proved their excellence in many applications, from visual object recognition to fraud detection. The detailed discussion of the conceptual issues, together with a compilation of test results and a few words of closing comments are presented in the following paragraphs. The full version of this paper has been submitted as thesis [1] for the degree of Master of Science in the AGH University of Science and Technology in Krakow, Poland and was defended in September 2005.

## GENERAL IDEA

The sketch design of the proposed decision-support system for market traders is depicted in the following list of main goals, principles, assumptions and restrictions. It will give the reader an insight of how the system works and explain briefly its nature inspiration.

- The model is trained on the historical data sets containing the following end-of-day information: open and close price, daily minimum and maximum price and total transaction volume.
- It is assumed that each data point may fall into one of three categories: *BUY*, *SELL* or *WAIT*. This means that the labelled data point may be identified with its corresponding action: entering the market, exiting it, or making no decision.
- For the sake of simplicity each action is taken in accordance with the rule "all or nothing", which means that if we buy, we spend all the capital owned and if we sell, all owned securities are traded away. Moreover, the trade can involve just one selected commodity. Such principles are set in order to avoid confusing the main ideas of this paper with the quantitative optimisation issues.
- For relevant information extraction a number of features are generated from the raw price data representing market history in some specified past time period.
- Out of the whole pool of features a number of random subsets of arbitrary length are chosen to form a population. Each specimen, a feature subset, is subject to evolution driven by recombination, mutation and selection.
- Each specimen establishes some feature subspace for an arbitrarily chosen classifier which is first trained on the initial part of data, then tested on the following part, both projected into this subspace.
- The training set is optimally pre-labelled according to the rule "buy at minima, sell at maxima, wait otherwise", advantaging the knowledge from the known history.

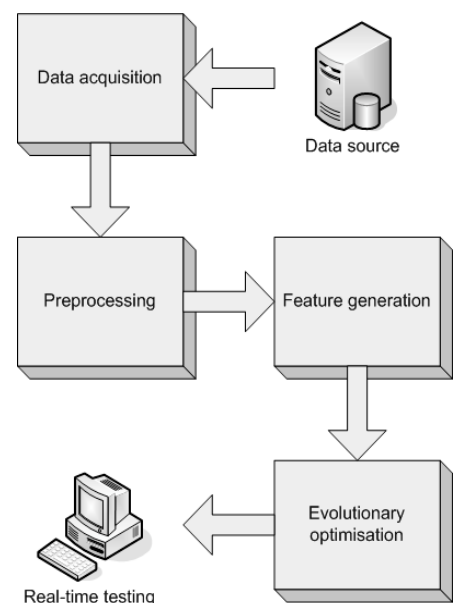


Figure 1: Block diagram of the decision-support system

- To the test set, which follows the training set, the trained classifier is applied to yield a sequence of actions: *BUY*, *SELL* or *WAIT*. This sequence may be understood as a record of a hypothetical past trade, easy to simulate. Although the optimal labelling for the test set is also known, it is not presented to the classifier.
- The criterion for evaluation of specimens' fitness is a total return on simulated investment mentioned above.
- Evolution is terminated after a fixed number of generations and it gives the best feature subset as outcome. "The best" means here the one for which the classifier yields the highest profits in the past test period.
- Real-time prediction incorporates the optimal feature space and trained classifier found in the model training stage. The appropriate features are computed for the current end-of-day price data and the resulting feature vector is then classified to suggest an action for the next day.

	open	low	high	close	Volume	
training set	day 1	po_1	pl_1	ph_1	pc_1	vol_1
	day 2	po_2	pl_2	ph_2	pc_2	vol_2
	...	...	...	...	...	...
	day N	po_N	pl_N	ph_N	pc_N	vol_N
	day N+1	po_N+1	pl_N+1	ph_N+1	pc_N+1	vol_N+1
	day N+2	po_N+2	pl_N+2	ph_N+2	pc_N+2	vol_N+2
test set	...	...	...	...	...	...
	day N+s	po_N+s	pl_N+s	ph_N+s	pc_N+s	vol_N+s
	day N+s+1	po_N+s+1	pl_N+s+1	ph_N+s+1	pc_N+s+1	vol_N+s+1
	day N+s+2	po_N+s+2	pl_N+s+2	ph_N+s+2	pc_N+s+2	vol_N+s+2
	...	...	...	...	...	...
	day N+s+t	po_N+s+t	pl_N+s+t	ph_N+s+t	pc_N+s+t	vol_N+s+t

Figure 2: Input data set divided into training and test part

The diagram on [figure 1](#) depicts main functional modules of the system. First, the raw quotes are imported from one of the available data sources, typically text files. Then, in pre-processing phase, the data is subject to format conversion and filtering. Any seasonal effects may also be eliminated, if applicable. Besides, logarithms of the price data points may be taken, as logarithmic scaling enables analysts to better spot the percentage changes in value when the market moves to the different price level. The next component is responsible for feature generation. At this point the technical analysis indicators as moving average, MACD or ROC are used as they are known to carry valuable information facilitating identification of correct entry and exit points for the

investment. On top of these classic TA indices more robust features are built, which will be discussed in the next paragraph. After that, a preliminary feature selection may be conducted. A few useful techniques for dealing with this task have been introduced in [\[3\]](#), however, in the system described below this stage is omitted. Here comes the main module of the system – evolutionary index set optimiser which yields a vector of the most informative dimensions found for the chosen classifier. These dimensions form a feature subspace to which the today's data is projected and passed to the classifier in order to obtain system's suggestion of what decision to make tomorrow.

## INPUT DATA AND FEATURES USED

The input data for the proposed system may come from any vendor, as long as it contains the necessary information. In this research historical stock quotes from Warsaw Stock Exchange and currency exchange rates have been chosen. To be considered useful, the data set must contain at least a single price value for each day but the true informative potential can be unlocked only from end-of-day data comprising the following values: open and close price, daily minimum (low) and maximum (high) price and daily volume. This data set is initially divided into training and test part. Both parts may be separated by a non-zero-length time lag as shown on [figure 2](#). The operational data for the system comprises a number of features that can be generated from the raw market quotes. For the numerical representation and processing technical analysis indices seem the most appropriate. They carry a great deal of information about the time series under consideration: the trends, their direction, strength and volatility as well as cash inflows and outflows. In contrast to fundamental analysis indices like ROE or P/E, or macroeconomic variables e.g. GDP, they are broadcast or can be computed on the spot, or at least on a daily basis, depending on the level of analysis granularity desired. Other data representations like chart shapes encoding have been disregarded as either inefficient or technically unfeasible. A number of articles and tutorials on technical analysis and its mathematical apparatus can be found in the World Wide Web, e.g. [\[13\]](#). The reader should also refer to the literature for details, e.g. [\[4\]](#), [\[5\]](#).

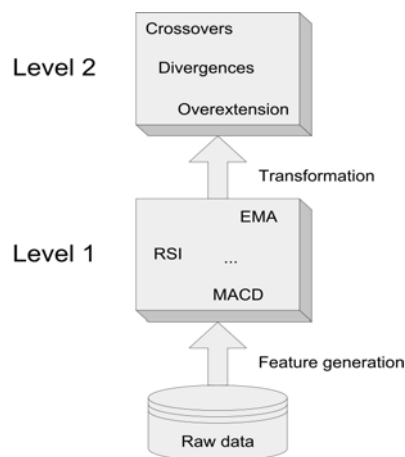


Figure 3: Feature generation scheme. The first level is built by the core technical analysis indices like EMA, MACD or RSI. Simple transformations applied to them yield more informative features of level 2.

It is vital to notice that in a raw form technical analysis indices are not highly descriptive, unless the way of trading signal inference is known. Normally, this inference, being a domain of expert market traders, is what lets ones outperform the market and makes others lose. In a computer system inference model can be constructed with a set of higher-level data features. In ideal settings the classifier is expected to find all optimal entry and exit points without any slippage i.e. assign BUY labels exactly to the days when the minima occur and SELL labels exactly to the days when the price reaches peaks. The goal is then to locate discrete entry and exit points by building a discriminant that could separate BUY and SELL clusters. To make both characteristic points form distinguishable clusters a more sophisticated data representation than simple index values is needed. For example a popular index, moving average, without a broader context gives no indication of whether the current trend is steady or is nearing a turning point. Yet, augmented with the information of its location relative to the price itself, MA offers a mechanism of trading signal issuing based on crossover analysis. The same rule governs feature generation model in the system hereby described. Basic TA indices are subject to various simple transformations made in order to guarantee a better separability of entry and exit points. They rest primarily on the following four principles:

- to locate price – average or index – average crossover points,
- to locate index – centreline crossover points for indices oscillating within a fixed range,
- to locate divergences i.e. the regions where an index and the price fail to show confirming trends,
- to locate the overextension (overbought or oversold) regions for oscillators.

This yields a two-level feature generation scheme illustrated on [figure 3](#).

## CLOSER LOOK INTO EVOLUTION

For our model each set of features generated from the raw data constitutes a feature subspace contained in the space of all available features. Each of such subsets may be evaluated with respect to how well separable it makes the data points representing different classes. It is an evolutionary search algorithm proposed in this paper that can deal with the optimisation of feature subset. Such an algorithm is supposed to differentiate population of random subsets and sift through the universe of admissible instances to promote the best ones. [Figure 4](#) illustrates the functions of evolution in pursuance of constructing an optimal feature subset.

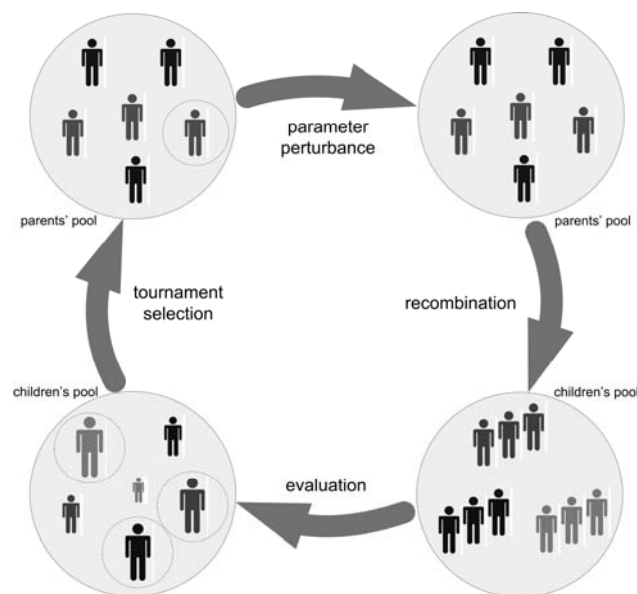


Figure 4: Conceptual model of evolution

## REPRESENTATION

Under the constraint of proposed scheme and format of data to process, the following representation of population is assumed:

- Population contains a fixed number of specimens.
- Each specimen consists of two sets of feature identifiers. One set points only to the features used for separation of *BUY* and *HOLD* points. Identifiers pointing to features used for separation of *SELL* and *HOLD* points are gathered in the second set<sup>1</sup>.
- Features themselves are objects with an appropriate, possibly parameterised computational formula built-in.

Figure 5 depicts structure of a single specimen.

BH_feat_1	BH_feat_2	BH_feat_3	...	BH_feat_N
SH_feat_1	SH_feat_2	SH_feat_3	...	SH_feat_M

Figure 5: Specimen structure

## ALGORITHM

The main loop of evolutionary algorithm used for model training can summarised in a following piece of pseudo-code.

```

generate initial population of length m;
while (not stop)
{
    mutate some specimens;
    perform recombination generating n children;
    evaluate each child;
    select m children to the next generation;
}

```

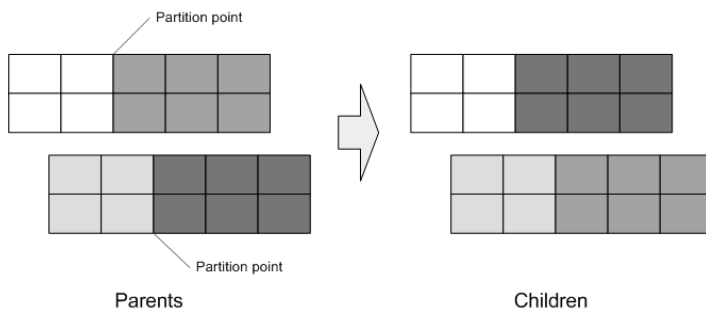


Figure 6: Recombination scheme. Two parents produce a pair of offspring by single-point crossover

Population is initialised semi-randomly. Out of the pool of available features a user-defined number is selected to represent each specimen. The same feature identifiers may occupy multiple slots in the array within a single specimen. It introduces yet no redundancy and is acceptable, because during the optimisation process the same features may evolve independently as a result of mutation. In the first step of a single generation processing a slot in the feature set of randomly chosen specimen is selected. If the feature referred to by the identifier occupying this slot is parameterised<sup>2</sup>, random parameter is selected and its current value is perturbed according to the normal distribution fixed number of times. Each

feature in each specimen's feature set is assigned a lookup table containing standard deviation values of normal distributions governing mutation of parameters of that feature over time. The key assumption on mutation in the proposed scheme is that mutation parameters are self-adaptive [9], [10]. This means in our case that the mentioned values of standard deviation are altered depending on the outcome of the series of feature perturbations according to the formula (1):

$$\sigma(i+1) = \begin{cases} \sigma(i)\delta_f & \text{if } r < 0.2 \\ \sigma(i)\delta_s & \text{otherwise} \end{cases} \quad (1)$$

<sup>1</sup> Test results proved that it would be reasonable to teach one classifier to distinguish between *BUY* points and *HOLD* noise and the second to separate *SELL* points from the *HOLD* points and to switch between the two according to the current investment state. This justifies maintenance of separate feature sets within the specimen.

<sup>2</sup> Each feature may be (and usually is) parameterised. For example features based on moving average are parameterised in terms of the averaging period. Features may contain multiple parameters as well.

where  $r$  is a ratio of the number of successful mutations to the total number of mutations,  $\delta_f$  is a coefficient by which the old value of standard deviation is multiplied, if there were no more than 20% successful mutations<sup>3</sup> in the cycle and  $\delta_s$  is a coefficient by which that value is multiplied otherwise. The initial values of  $\delta_f$  and  $\delta_s$  are set to 0.8 and 1/0.8 respectively.

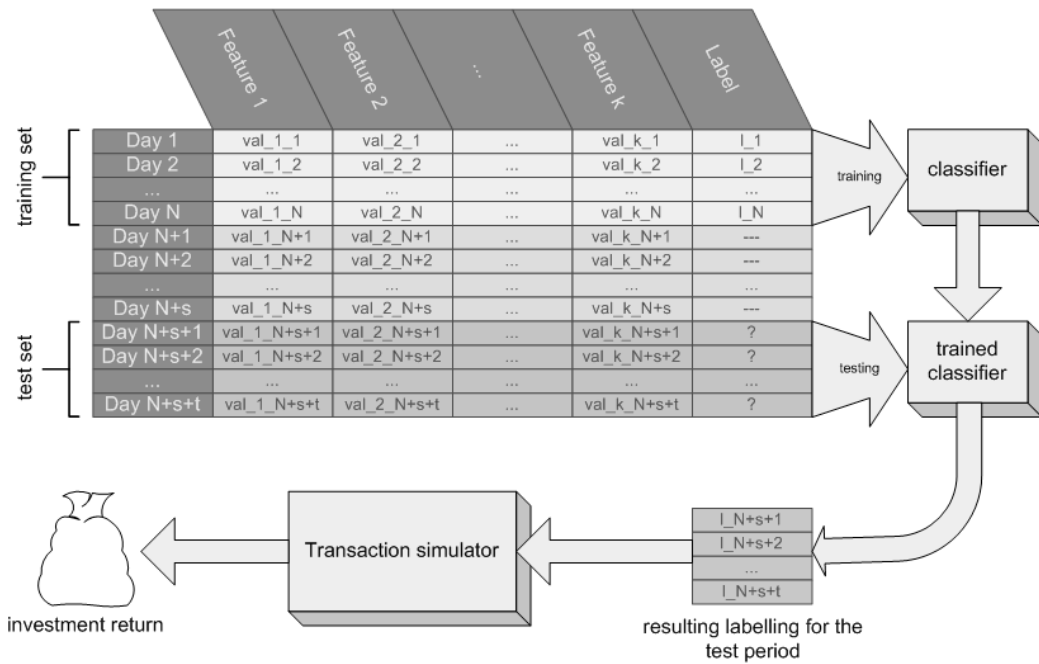


Figure 7: Specimen evaluation scheme

Recombination is arranged as a global crossover operation in which, assuming population of length  $m$ ,  $n$  children are spawned, where  $n = km$ . With  $m$  and  $k > 1$  defined by the user (usually value  $k = 7$  was used in the experiments), in each generation  $n/2$  times two random parent specimens are selected to produce two children using a simple single-point crossover scheme as depicted on figure 6. This simple mechanism guarantees global exchange of features within the population, contributing to efficient diversity maintenance. One should notice that crossover does not enforce feature re-computation as mutation does. Parent specimens merely exchange feature identifiers without altering the underlying formulas.

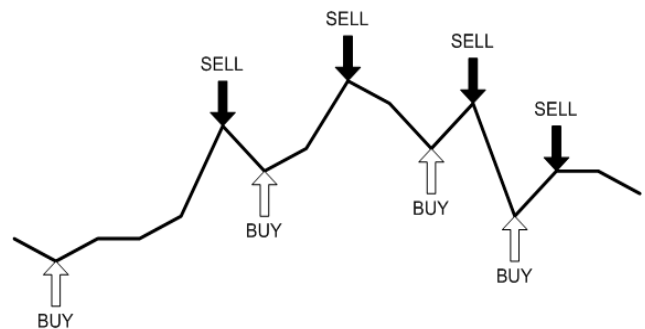


Figure 8: Optimal labelling

Recombination produces a large surplus of child specimens. Before the best ones can be selected, offspring is evaluated and each specimen is assigned fitness value. Evaluation scheme has been depicted on figure 7. It is assumed for simplicity that the optimal labels for the training points are a priori known. In practice they are automatically determined by a simple heuristic applied to the input set before the evolution process is started. The principle of this heuristic can be summarised as assigning *BUY* labels to the detected minima in the price signal and *SELL* labels to all found maxima, which yields a sequence of alternating entry and exit points for the trade, as on figure 8. Minor violations of this rule have been introduced in terms of two exceptions justified by the real market observations:

- No buy – sell transaction is made if the difference between corresponding low and high price levels is smaller than the transaction commission. In practice doubled 0.5% commission threshold is used.
- Too frequent opening and closing position is prevented.

<sup>3</sup> Mutation is considered successful when the specimen is evaluated better after than before.

The former principle simply prevents the system from minor losses and over-reactive behaviour. The latter plays a similar role, imposing a common-sense policy of restraint to avoid panic or greedy decisions and reminds about the real-market settings in which transaction orders are normally processed within a couple of days. For further details of training data labelling the reader should refer to [1].

In purpose of specimen evaluation the following steps are taken:

- The selected classifier is trained on the training set (days 1 to N).
- It is then tested on the test set to produce transaction labels for the days  $N+s+1$  to  $N+s+t$ .
- Resulting labels define a sequence of actions to perform in a transaction simulator accounting for a fixed transaction commission.
- Total simulated investment return is considered to be specimen's fitness value.

Above steps call for explanation. As already mentioned, the current pool of features possessed by given specimen is divided into two subspaces:  $\Omega_{BH}$  and  $\Omega_{SH}$ . Also two binary classifiers are used instead of one. The first,  $C_{BH}$ , is trained exclusively on *BUY* and *HOLD* points from the training data projected to the  $\Omega_{BH}$  space. The second classifier,  $C_{SH}$ , is trained only on *SELL* and *HOLD* points from the training set projected to  $\Omega_{SH}$  space. Preliminary tests indicated that it is a relatively easier task to separate entry and exit points individually from the overwhelming *HOLD* noise than to distinguish between all three classes at once. This in turn, resulted in the introduction of a transaction state and yielded the following test points classification scheme:

```

state = 'has_funds';
index_buy = 0;
index_sell = 0;
return = 0;
for i = 1 to length(data)
{
    if (state == 'has_funds')
    {
        label = BHClassifier.classify(BHdata(i));
        if (label == 'BUY')
        {
            index_buy = i;
            state = 'has_shares';
        }
    }
    else
    {
        label = SHClassifier.classify(SHdata(i));
        if (label == 'SELL')
        {
            index_sell = i;
            return = return + getReturn(index_buy, index_sell);
            state = 'has_funds';
        }
    }
},

```

where *data* stands for the test set, *BHdata* stands for its projection to the  $\Omega_{BH}$  space and *SHData* is its projection to the  $\Omega_{SH}$  space.

In the above procedure test points, regardless of their inherently known class membership, are equally treated at the time they are presented to the classifiers. The latter, trained in recognition of the right moments for the transaction, work in turn, dependent on the current trade status denoted by the *state* variable. It is initially set to value meaning that no shares are owned. If the current state is '*has\_funds*', the first point is taken from the *BHdata* set to pass to the *BH* classifier. If it is assigned label *BUY*, the index of this point is remembered. Otherwise, the next point is classified until the result is *BUY*. In that case the state is changed to denote that position has been opened. In '*has\_shares*' state the current point in *SHData* set is passed to the *SH* classifier. If it is classified as *SELL* point, the return from the just-closed investment is computed and cumulated based on the current index, the index of stored point of purchase and the pre-set transaction commission. Otherwise, the next point is processed until the result of classification is *SELL*. Then the state is again altered to denote closed position and regained funds. Value of *return* variable is successively updated to store cumulative profit or loss in the end. This value is finally used as a measure of specimen's fitness.

Processing of a single generation in the evolution loop is finalised in the selection step. Presented algorithm incorporates tournament selection scheme illustrated on [figure 9](#). Under the assumption that  $n$  children have been generated and the population counts  $m$  specimens,  $m$  times a fixed-size tournament pool is build from randomly chosen children. In each turn the best (i.e. the one with the highest fitness value) from the group is selected to the next generation. Once all  $m$  specimens have been selected, the new parent population is assembled and the remaining  $n-m$  specimens are disposed. At any time, regardless of which specimens have actually propagated, a reference to the best specimen registered so far is always kept, yet outside the population. This principle leads to the guaranteed convergence, typical to the elitist model [\[9\]](#), but has demonstrated superiority over the latter in the ability of maintaining population diversity effectively.

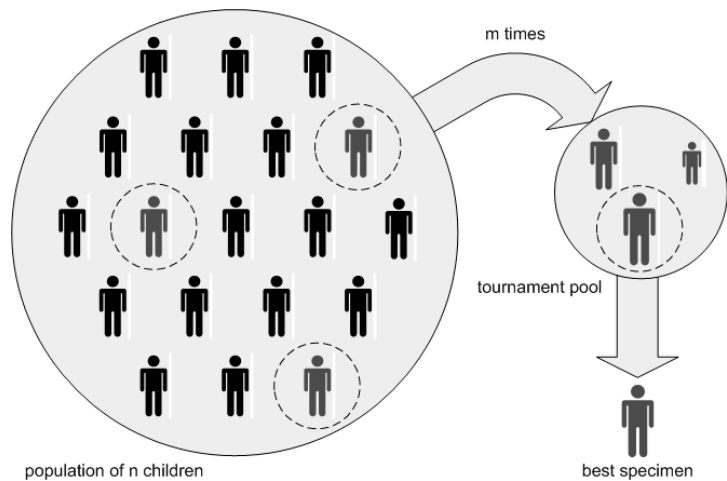
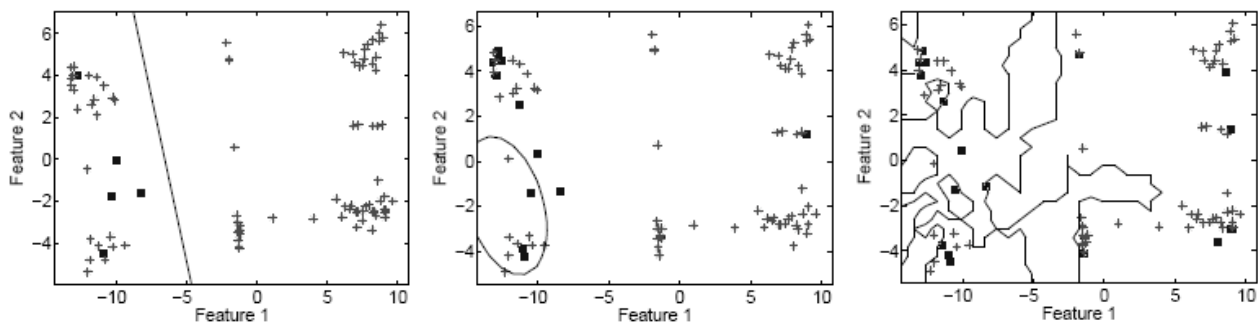


Figure 9: Tournament selection scheme

## CLASSIFIERS USED



A series of experiments have been conducted in order to check how various popular classifiers [\[8\]](#) could cope with the relevant transaction point separation. [Figure 10](#) illustrates the decision boundary produced by three different classifiers discriminating between *BUY* and *HOLD* points taken from a dataset of *Comarch SA*<sup>4</sup>. The dataset was projected to a 2-D space built by two sample, randomly chosen level 2 features. Although the space is normally higher and features themselves are more reasonably grouped, this example shows well that sparse clusters of interest points are neither linearly separable, nor can they be captured by a range of more sophisticated classifiers. In this situation, not surprisingly, the relatively best results were achieved by the nearest neighbour classifiers as from the theoretical point of view, the error rate of a  $k$ -NN classifier tends to the Bayes optimal as the sample size tends to infinity [\[11\]](#). The efficiency to complexity ratio obtained for NN classifiers was reasonable, provided the parameter  $k$  was set to an appropriately small value due to the quantitative imbalance between the number of *BUY* and *HOLD* points (typically, most of the time we wait for the favourable time to trade, but actually quite seldom take any action). Together with  $k$ -NN, also the support vector machine classifiers [\[8\]](#), [\[12\]](#) were picked for implementation. These performed satisfactorily too in a series of similar tests to the one presented, yielding quite well-fitted curved decision boundaries and showing considerable generalisation ability.

<sup>4</sup> IT business solutions provider listed in Warsaw Stock Exchange

## TEST RESULTS

Capabilities of the proposed decision support system have been verified in a complete working prototype system implemented in a *Microsoft Visual Studio .NET* platform. The tests were performed on contiguous past data sets divided into training and test parts as explained earlier. Several large companies listed in Warsaw Stock Exchange were selected to find the optimal set of features for investment decision making<sup>5</sup>. The system was additionally tested on several currency exchange rate data volumes to find out, whether the features dedicated primarily to equity market analysis could turn out useful for FX markets too. For all tests it was assumed that each transaction was charged commission of 0.5%. Details of additional transaction-making policy parameters can be referred to in [11]. Two parameterised classifiers implemented in the prototype application were used: nearest neighbour classifier and support vector machine classifier. Compiled test results have been summarised in tables I and II.

Equity	Best return	Number of transactions	Naive return
TPS	25.99%	2	15.88%
PEO	28.71%	6	10.48%
NET	16.09%	3	12.82%
ELE	26.97%	4	-16.67%

Equity	Best return	Number of transactions	Naive return
TPS	27.58%	4	15.88%
PEO	30.76%	6	10.48%
NET	15.31%	3	12.82%
ELE	33.50%	7	-16.67%

Table I: Results of test performed on equity data in the period between November 2003 and November 2004. The system was trained on the first 144 quotes and tested on the following 97 using two 3-nearest neighbour classifiers. The population consisted of 10 specimens of length 6 and generated 60 children in each crossover.

Table II: Results of the same test but using two SVM classifiers with radial basis function kernel and parameters:  $\sigma^2 = 2.0$ ,  $C = 100$ ,  $\nu = 0.5$ ,  $\varepsilon = 0.01$ .

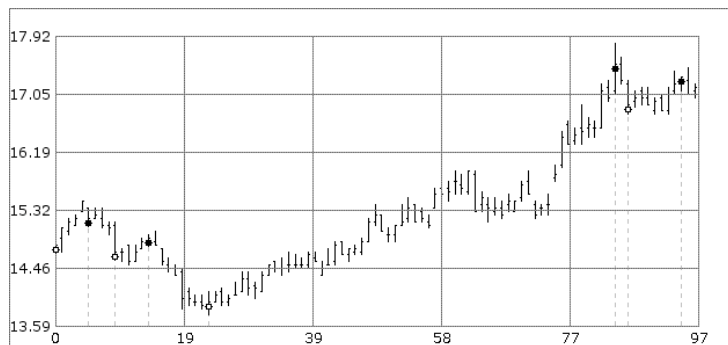
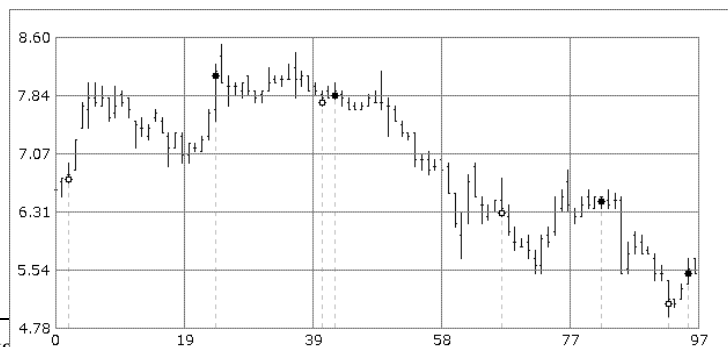


Figure 11: Visualisation of the best trade on Telekomunikacja Polska SA (TPS) equity summarised in table II



<sup>5</sup> Telekomunikacja Polska SA (TPS), Bank Polska SA (PEO), Netia SA (NET) and Elektrim SA (ELE)

In Figure 12: Visualisation of the best trade on Elektrim (ELE) equity summarised in [table I](#)

each table

the second column contains return from the best trade discovered, i.e. the one achieved with the data projected to the most adequate indicator set found. The third column gives the number of *BUY-SELL* transactions made on this occasion. Perhaps the most interesting piece of information comes from the comparison between the best obtained returns and the entries in the fourth column, somewhat confusingly called "naive return". The latter means here the return obtained from a single haphazard transaction: purchase at the beginning of the test period and sale at its end. It serves as a benchmark for the performance of our model. What the reader can learn from this comparison is simply to what degree the system can reveal the current price trend and possibly outperform it. Apparently, in each case it was capable of doing so, most spectacularly in the case of *ELEKTRIM* (ELE) equity, when it even did not fail to capitalise after the price had passed the long-term up-trend turnover point and started to drop. Figures [11](#) and [12](#) localise in time the exact entry (empty circles) and exit points (filled circles) for TPS and ELE share price.

In the case of currency exchange rates, available datasets contained only one relevant entry – daily rate. Despite the fact that fewer indicators could be generated from such data, comparably good results were achieved. With the population comprised of 10 specimens of length 5 and 4-NN classifiers trained on 847 quotes, during the period of the following 364 working days obtained profits on Zloty-Euro and Zloty-US Dollar exchange rates reached 12.61% and 17.04%. In the same period the real values dropped by 7.84% and 22.21% respectively.

An interesting property of the evolutionary system introduced, observed throughout the experiments, was its fast convergence. Due to quite lengthy computational process only up to 60-generations-long runs were triggered. Yet, frequently near to optimal solutions were unleashed after only 10 generations. [Table III](#) illustrates this limited time demand for convergence on an example of *PKN Orlen*<sup>6</sup> equity data. It should be emphasised, that the values of return in the second column do not correspond to the snapshots of the model at different time points of the same test. They were produced in the independent variable-length runs of the algorithm. Nevertheless, the tendency to converge fast can be spotted even under the influence of randomness occurring inherently in the course of evolution.

generations	return	number of transactions
10	67.00%	8
20	67.74%	4
30	67.90%	8
40	71.94%	4
50	66.36%	8

Table III: Relationship between the obtained return and algorithm running time for *PKN Orlen* share data. The dataset covering period between November 2000 and November 2004 was divided into training part (685 quotes) and test part (295 quotes). During the test period the price rose by 57.81%, compared to 8.07% rise during the training period.

## CONCLUSIONS

Decision support system for market traders presented in this paper is an alternative concept to most of the existing ones, usually either too simple or too complex, and in the same time inadequate or incomplete. Reasonable performance the prototype has demonstrated when tested on historical data is a decent reason to claim, that this approach has good prospects for competing successfully with the existing solutions, like popular rule-based systems.

Contrary to many other techniques, the approach based on the metaphor of evolving market descriptors deserves much to be called intelligent. In the beginning of the evolutionary process the model encapsulates no global knowledge about the market rules governing temporal behaviour of the price. It is yet incrementally gathered through the interaction of population specimens that, in accordance to Holland's principle, combine information building blocks together to form pieces of higher quality. On top of the evolutionary search engine, a good training data labelling algorithm and powerful classifiers are in place to ensure that the feature set being optimised evolves in the right direction – towards the one leading to the highest return on investment. This bottom-up methodology of knowledge extraction is set against the traditional approach in which the predefined market strategies are encoded in the model, which works for the datasets having expected properties, but frequently fails for the other. In addition, it should be emphasised that the proposed system, despite its built-in simplicities, can be thought of as a useful tool relieving investor from the burden of strenuous market analysis. Instead, it merely requires competency in providing representative training data and some set of indicators for operation. On output it produces a model that can immediately be applied to the real-life data and with a high probability yield a profit.

<sup>6</sup> Poland's largest refiner of crude oil and marketer of petroleum and related products listed in Warsaw Stock Exchange

Further work is planned, especially aimed at improving and extending the selection of input market indicators and massive, multicriterial testing. In order to convert the system into a complete software solution, all prototype-stage simplicity assumptions need to be eliminated and the model of operation carefully revised in terms of real-market requirements. Finally, the ultimate goal is to enable fully functional quantitative portfolio management.

## ACKNOWLEDGEMENTS

This work was supported by the Polish Committee of Research (KBN) grant 3T11C 05926/18.25.120.498. I wish to thank Dr. Witold Dzwiniel from the Institute of Computer Science, AGH University of Science and Technology, Krakow, Poland for his supervision and useful remarks.

## REFERENCES

- [1] Ruta, A. (2005) "Pattern Recognition Methods in Investment Decision Support". M.Sc. thesis
- [2] Tsay, R. S. (2001) "Analysis of Financial Time Series", Wiley-Interscience
- [3] Würtz, D. (2002) "S Plus for Financial Engineers", "Econophysics" lecture notes, Institut für Theoretische Physik, ETH Zürich
- [4] Wilder, J. W. (1978) "New Concepts in Technical Trading Systems", Trend Research
- [5] Katz, J. O., McCormick, D. L. (2000) "The encyclopedia of trading strategies", McGraw-Hill
- [6] Vojinovic, Z., Kecman, V, Seidel, R. (2002) "A data mining approach to financial time series modelling and forecasting", International Journal of Intelligent Systems in Accounting, Finance & Management, 10(4): 225-239(15)
- [7] Novobilski, A., Kamangar, F. (2002) "A Genetic Algorithm Based Approach for Discovering Temporal Trends Using Bayesian Networks", 6-th World Conference on Systemics, Cybernetics and Informatics
- [8] Dzwiniel, W. (2004) "Pattern Recognition Methods", lecture notes, AGH University of Science and Technology, Krakow, Poland
- [9] Kisiel-Dorohinicki, M. (2004) "Evolutionary Algorithms", lecture notes, AGH University of Science and Technology, Krakow, Poland
- [10] Back, T. (1996) "Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms", Oxford University Press
- [11] Cover, T. M. (1968) "Estimation by the Nearest Neighbour Rule", IEEE Transactions on Information Theory, 14(1):50-55
- [12] Cristianini, N., Shawe-Taylor, J. (2000) "An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods", Cambridge University Press
- [13] StockCharts.com. Education (2005). Available from: <http://stockcharts.com/education> [Accessed between January and July 2005]