

Optimization of a Trading System using Global Search Techniques and Local Optimization

Donald L. Iglehart and Siegfried Voessner
Department of Engineering Economic Systems & Operations Research
Stanford University
Stanford, CA 94305-4023, U.S.A.

Many problems in finance require the optimization of a mathematical model with respect to a set of parameters. Often the objective function for these models is a nonlinear, discontinuous, nonconvex, nondifferentiable function of both discrete and continuous variables. Objective functions of this type present a formidable challenge to classical methods of mathematical programming. In this paper we present a Hybrid Algorithm (HA) that combines a robust genetic algorithm (GA) with a local optimization technique (LOT). The LOT uses a quasi-Newton algorithm (QNA) for continuous variables and a hill-climbing algorithm (HCA) for discrete variables. HA is implemented using the software package GOAL/MINOS. HA is applied to a rule based system for trading the S&P500 Index using daily closing prices that was developed by Meyers [1997]. This trading system has 16 parameters and the objective function displays all the difficult features mentioned above. The HA is shown to improve the performance of this trading system in a reasonable amount of computer time without using any previous knowledge of good parameter values. The success HA showed in handling this problem holds out promise for tackling other difficult optimization problems in finance and other areas.

1. INTRODUCTION

Mechanical trading systems are extensively used in financial markets to create "buy" and "sell" signals. These systems require as input a collection of market generated data. Given this input data, a trading system determines an individual's position in the market: long, short, or out. Typically, these systems consist of a rule base which depends on a set of parameters. Therefore, it is possible for a given set of rules to optimize the performance of the trading system by selecting appropriate values of the parameters.

In this paper we focus on an algorithm for the optimization of this parameter set. Due to the nature of stock market data used as input and the complex rule base, the objective function is nonlinear, discontinuous, not everywhere differentiable nor convex, and also has many deceptive local optima. Furthermore, some of the parameters are discrete while others are continuous. Handling such problems requires a fairly robust optimization algorithm. There are a number of techniques that are designed to deal with several aspects of this problem. Their performance depends heavily on the structure of the problem. In general, it can be said that the more assumptions an algorithm makes about properties of the objective function (and if the assumptions are satisfied) the more efficient is the algorithm. Conversely, it is true that these

algorithms tend to fail easily if the assumptions are not satisfied. Algorithms that do not assume too much about the structure of the optimization problems are often inefficient, but pretty robust.

We present a Hybrid Algorithm (HA) that combines a robust genetic algorithm (GA) with a highly efficient local optimization technique (LOT). The LOT uses a quasi-Newton algorithm (QNA) for the continuous variables and a hill-climbing algorithm (HCA) for discrete variables. The GA is used to find the areas in search space that have a locally convex structure, where the local model is justified. Since the GA does not rely on any gradient information, smoothness or continuity properties, it is not affected by jumps in the objective function or by plateaus. By performing a cluster analysis on the populations of the GA, we use an efficient technique for detecting these subspaces and for giving information about the state of convergence of the GA. The GA decides based on this convergence when to switch to the local search. Being only applied to areas where the GA has detected a convex structure of the search space, it is possible for the LOT to avoid "bad neighborhoods".

Our experiments have shown that the HA is a robust technique for tackling this kind of problem. Compared to other merely gradient based algorithms the danger of getting caught in local optima and dead spots (zero-gradient) is significantly lower. Another advantage is that the algorithm is almost entirely independent of the trading system. This property makes it easy to apply changes to the rule base without having to change the optimization algorithm. To implement the HA we have combined two commercial optimization packages. We use GOAL (Voessner and Brauningl [1996]) for the GA and the HCA, and MINOS (Saunders and Voessner [1998]) for the continuous variable portion of the LOT. We shall refer to this combination as GOAL/MINOS.

In this paper we have selected a trading system designed to trade the Standard & Poors (S&P) 500 Index. This system was developed by Meyers [1997]. This index is a capital-weighted index of 500 large capitalization stocks which is frequently used as a benchmark for mutual fund managers in the U.S. There are mutual funds that mimic the S&P500 and an S&P500 futures contract that an individual can trade using this trading system. A financial newsletter, Hunter [1998], tracks over 40 such systems. The system we are studying is ranked among the best performing systems among these 40. It trades on a daily basis and requires as input data the following daily information: closing price of the S&P500 Index, the number of advancing and declining stocks on the New York Stock Exchange (NYSE), and the number of stocks on the NYSE making new 52 week highs and lows. It consists of a sophisticated rule base with a total of 16 bounded, mixed integer parameters. The parameters in the original Meyers system were also determined by optimization.

Finally, we show a comparison between the performance of the existing system and the system optimized with GOAL/MINOS on a trade by trade basis. The results show that we could improve the performance of the existing system from a "cold" start in a reasonable amount of computer time (2 hours on a PC with Intel 166MHz Pentium processor). To ensure local optimality, a sensitivity analysis was also carried out. The GOAL/MINOS solution was shown to be locally optimal, whereas the Meyers [1997] solution was not.

2. OPTIMIZATION ALGORITHM

The optimization problem that occurs in this context is an extremely difficult one: the objective function is not smooth, not everywhere differentiable and continuous, nonlinear and nonconvex. Furthermore, there are lots of local optima (multimodal) and fairly large flat regions with gradient zero. This situation is schematically shown in Figure 1. Assuming a minimization problem,

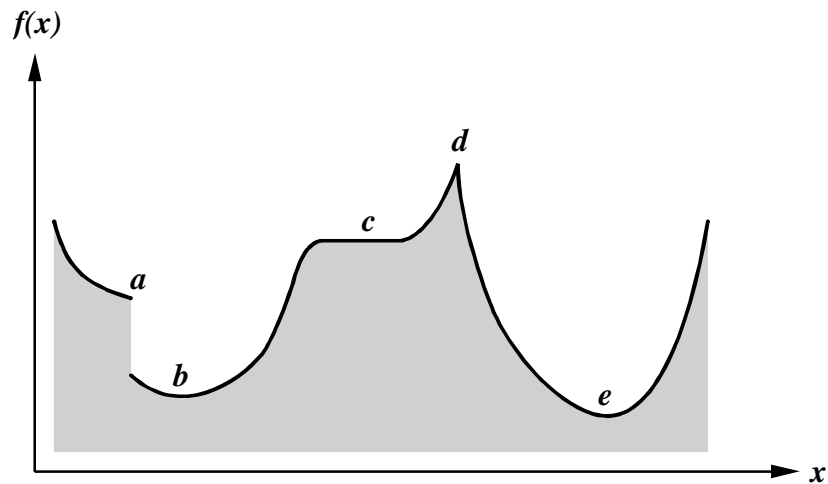


Figure 1: Some special properties of the objective function.

a marks a discontinuity, b a local optimum, c a plateau area (dead spot), d an area with no defined gradient and e finally labels the global optimum of $f(x)$ in the given interval. In addition to that some variables can only have discrete values which makes the optimization problem mixed integer. Figure 2 shows a schematic sketch of such a problem for two variables x_1 and x_2 , where x_2 is integer.

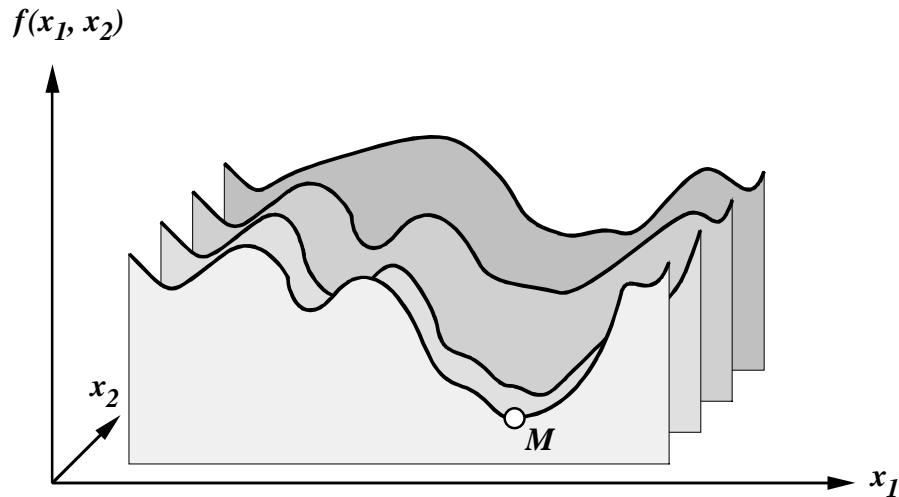


Figure 2: Mixed integer optimization problem.

For this kind of optimization problem there exists no algorithm that can guarantee finding the global optimum in finite time (Gill, Murray and Wright [1981]). Applying classical optimization techniques alone does not provide a decent solution. Nevertheless with modern optimization methods it is possible to get very close to the optimal solution, although there can be no guarantee of optimality. We will present a HA that combines a robust GA and a LOT. The continuous variable portion of the LOT is a QNA implemented in MINOS. MINOS is a highly efficient local optimization technique that can build a local quadratic model of the nonlinear objective function (Figure 3).

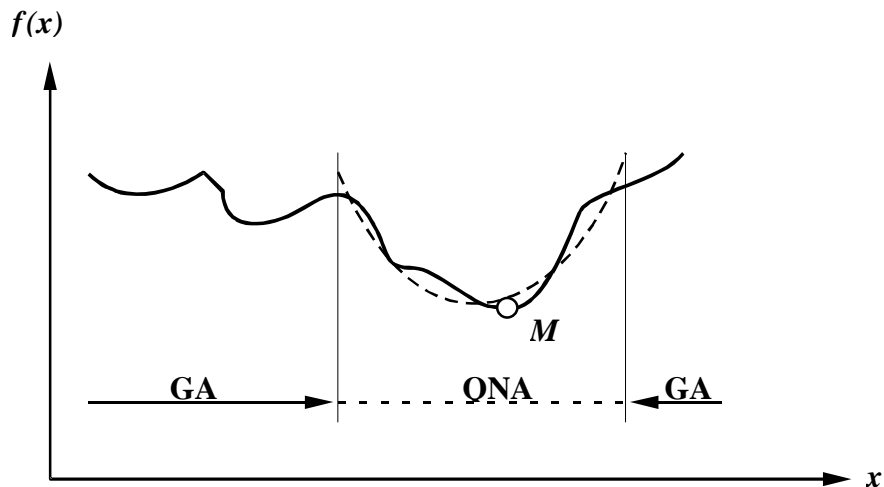


Figure 3: Global (GA) and local optimization (QNA).

The GA is used first to find the areas of the search space where the local, quadratic model is justified. Since the GA does not rely on any gradient information, smoothness- or continuity-properties, "bad neighborhoods" like *a*, *b*, *c* and *d* in Figure 1 have no influence. The more

sensitive local search, using a quadratic model (dotted parabola in figure 3), takes over in areas with a convex (speaking of minimization) substructure.

2.1 GLOBAL SEARCH -- GENETIC ALGORITHM

Global optimization is defined as the task of finding the absolutely best set of parameters to optimize an objective function. While simply stated, this type of search can become very complicated. In general, there can be many solutions that are locally, but not globally optimal. Consequently, global optimization problems are typically quite difficult to solve.

There are a number of techniques that are designed to deal with these kind of problems (like Branch and Bound methods, Simulated Annealing, Genetic Algorithms and many others). Their performance depends heavily on the structure of the problem.

We use a special, robust GA combined with Cluster Analysis to find concave areas in search space around local maxima. The algorithm is based on D. E. Goldberg's Simple Genetic Algorithm (SGA), with a population size of 100 and standard genetic parameter settings from his book, Goldberg, D. E. [1989]. GA's have (aside from their robustness) the property that their convergence slows down significantly close to a solution. After a certain number of generations, the initially randomly generated candidate solutions have converged close enough to regions around local optima that we can use them as starting points for the local search. From this point on, a continued search with the GA would be rather inefficient.

2.2 LOCAL OPTIMIZATION -- QUASI-NEWTON ALGORITHM

The global search provides, depending on the structure of the search space, several areas where a local search could possibly lead to further improvement. From the center of these areas, which are detected by cluster analysis, we start separate local search runs until they converge, compare the objective values and take the best one as the final result. Since there are usually many areas to investigate, we need a fast search method. Close to an optimum a continuous, smooth objective function can be approximated by a quadratic function, see Gill, Murray and Wright [1981]. We shall assume the objective function satisfies these properties locally. Therefore, we chose for optimizing the mixed integer problem locally a QNA method for the continuous variables and a simple steepest ascent (maximizing) method for the discrete variables (HCA).

3. TRADING SYSTEMS

The trading systems we consider in this paper are rule based using market generated information. Such systems depend on a set of parameters which are selected by the optimization of an objective function. Over 40 trading system of this type are tracked by Hunter [1998]. One of the consistently most profitable systems is the “Turbo A/D, NH, NL Market System” developed by Meyers [1997]. We have chosen this system, which we shall refer to as T87, to illustrate the effectiveness of our HA.

T87 is designed to trade the S&P 500 Index on a daily basis. The input data required by T87 are the following: closing price of the S&P 500, number of daily advancing issues on the New York Stock Exchange (NYSE), number of daily declining issues on the NYSE, daily advancing volume on the NYSE, daily declining volume on the NYSE, number of stocks on the NYSE making 52-week new highs, and the number of stocks on the NYSE making 52-week new lows. Given this data as shown in Table 1, T87 computes the defined constants, functions, and variables shown in Tables 2, 3, and 4. Based on the conditions given in Table 5, T87 will buy the S&P500 on the close if B0 and (B1 or B2) are true. Based on the conditions given in Table 6, T87 will sell on the close if S0 and (S1 or S2 or S3) are true.

This trading system contains 16 variables. As Meyers [1997] points out, “with only four steps per variable, this would create more than 4.3 billion cases to examine (4 to the 16th power). At one second per case, that’s about 136 years of computing!” Faced with this daunting task, Meyers elects to optimize the buy and sell variables independently. He suboptimizes within groups of up to three variables at a time, holding all the other variables fixed. After optimizing within the first group of variables, he uses those values as he optimizes within the second group of variables. He rotates in this fashion through all the variables. After making a first pass through all the variables, a second and third pass are made in the hope that this procedure will arrive at an optimal set of variable values. For initial values of the variables, Meyers uses values obtained in earlier related trading systems, and by visual inspection of the input data.

The data available for Meyer’s study was from January 1, 1978 to May 16, 1997. This data was divided into two out-of-sample segments and one training segment. The training segment was from January 1, 1987 to December 31, 1995. The two out-of-sample segments were from January 1, 1978 to February 28, 1987 and from January 1, 1996 to May 16, 1997. Using the optimization procedure outlined above, the values found for the variables are listed in Table 7. Also contained in Table 7 are the ranges and types of the variables used in our HA. Notice that the number of cases that we optimize over is greatly in excess to the 4.3 billion cases mentioned above. Using our HA, we developed our version of T87 which we call DNS. The objective function that we set out to maximize was the total net percentage profit. Optimizing over the training segment we obtained the values for the variables listed in Table 8.

Ai	=	Daily NYSE advancing issues
di	=	Daily NYSE declining issues
av	=	Daily NYSE advancing volume
dv	=	Daily NYSE declining volume
nl	=	Daily NYSE 52-week new lows
nh	=	Daily NYSE 52-week new highs
spx	=	Standard & Poor's 500 index

Table 1. Input Data for the Trading System T87.

$nl\alpha$	=	$2/(n+1)$ n-day exponential smoothing constant for smoothing nl-data (n=3)
$nh\alpha$	=	$2/(n+1)$ n-day exponential smoothing constant for smoothing nh-data (n=3)

Table 2. Defined Constants for T87.

Lowest (x, d)	=	lowest value of x in the last d days (including today)
Highest (x, d)	=	highest value of x in the last d days (including today)
Range (x, d)	=	Highest(x, d) - Lowest(x, d)
Strength (x, d)	=	$100 (x - \text{Lowest}(x, d)) / \text{Range}(x, d) - 50$

Table 3. Defined Functions for T87.

adr	=	ai/di advance/decline issue ratio
advr	=	av/dv advance/decline volume ratio
dar	=	di/ai decline/advance issue ratio
davr	=	dv/av decline/advance volume ratio
nlx	=	last $nlx + nl\alpha (nl - \text{last } nlx)$ exponential moving average of nl
nhx	=	last $nhx + nh\alpha (nh - \text{last } nhx)$ exponential moving average of nh
NLstr	=	Strength(nlx, Ldays) Ldays=lookback period in days
NHstr	=	Strength(nhx, Hdays) Hdays=lookback period in days

Table 4. Defined Variables for T87.

B0	if $(NLstr \leq NLbx)$ and $(NHstr \geq NHbx)$ and $(spx > \text{Lowest}(spx, \text{since last sell}) * pctud)$
B1	if $(adr \geq buyr)$ for $xbuy$ days out of $nbuy$ days and $(adr \geq 1)$ for all of the $nbuy$ days
B2	if $(advr \geq buyvolr)$

Table 5. Conditions Used for T87 Buy Signals.

S0	if (NLstr \geq NLsx) and (NLstr- last NLstr < 30) and (NHstr \leq NHsx) and (spx < Highest(spx, since last buy) * <i>pctud</i>)
S1	if (dar \geq <i>sellr</i> for <i>xsell</i> days out of <i>nsell</i> days)
S2	if (davr \geq <i>sellvolr</i>) and (last davr \geq <i>sellvolr2</i>)
S3	if (davr \geq <i>sellvolr2</i>) and (last davr \geq <i>sellvol</i>)

Table 6. Conditions Used for T87 Sell Signals.

Number	Name	Value	Range	Type
1	<i>pctud</i>	1.25	0 .. 10	Real
2	<i>NLbx</i>	0.0	-50 .. 50	Real
3	<i>NHbx</i>	-10.0	-50 .. 50	Real
4	<i>Hdays</i>	10	2 ..129	Integer
5	<i>buyr</i>	1.2	0 .. 5	Real
6	<i>xbuy</i>	2	0 .. 63	Integer
7	<i>nbuy</i>	4	0 .. 63	Integer
8	<i>buyvolr</i>	7.5	0 .. 20	Real
9	<i>NLsx</i>	30.0	-50 .. 50	Real
10	<i>NHsx</i>	0.0	-50 .. 50	Real
11	<i>Ldays</i>	50	2 ..129	Integer
12	<i>sellr</i>	1.9	0 .. 10	Real
13	<i>xsell</i>	3	1 .. 16	Integer
14	<i>nsell</i>	4	1 .. 16	Integer
15	<i>sellvr</i>	10.0	0 .. 20	Real
16	<i>sellvr2</i>	2.2	0 .. 20	Real

Table 7. Decision variables for T87 and ranges for our optimization.

3.1 RESULTS

The total time span of the data used in Meyers [1997] is from January 1, 1978 to May 16, 1997. Following the suggestions in that paper the data were first divided into three out-of-sample segments (OS1: January 1, 1978 to February 28, 1987, OS2: January 1, 1996 to May 16, 1997 and OS3: January 1, 1996 to June 19, 1998) and one training segment (TRN: January 1 1987 to December 31, 1995). This type of segmentation is standard procedure to help avoid the perils of curve fitting. The GA part of the optimization took about 2 hours on a PC with Intel 166MHz Pentium processor, the local search was finished in about 10 minutes on the same machine. We maximized the total net percentage profit v ($v_i = v_{i-1} (II_i/II_{i-1})$, $v_0 = 100$) trading the S&P500 by adjusting the following 16 variables: *pctud*, *Nlbx*, *Nhbx*, *Hdays*, *buyr*, *xbuy*, *nbuy*, *buyvolr*, *Nlsx*,

$Nhsx$, $Ldays$, $sellr$, $xsell$, $nsell$, $sellvr$, $sellvr2$. Here Π_i is the closing price for the S&P500 on day i . Furthermore the cumulated profit and loss pnl is reported ($pnl_i = pnl_{i-1} + (\Pi_i - \Pi_{i-1})$). In Table 8 are displayed the values of the optimized decision variables for T87 over the training segment and for DNS over the training segment and over the total data set. It is clear from the three columns in Table 8 that the values found represent three different local optima. Table 9 displays the values of the objective functions for both T87 and DNS over different segments of the data. Notice that the DNS values are roughly 4% better than the T87 values except for the Objective v over the total data set, where DNS is 15% better. In Table 10 the performance of the two trading systems are compared using results from the Ultra Systems software (Hunter [1998]). The abbreviation CAR denotes cumulative annual return. From this table we see that the trading performance of T87 and DNS are remarkably similar in spite of the fact that they represent different local optima of the objective function. Finally, in Figures 4-7 we have display the trades taken using T87 and DNS over the span of the total data set using the parameter values obtained when optimizing over only the training set. Clearly, the optimization carried out by Meyers was quite effective.

Number	Name	T87 Value Training Segment	DNS Value Training Segment	DNS Value Total Segment
1	<i>pctud</i>	1.25	-0.1587	0.0
2	<i>NLbx</i>	0.0	8.7302	0.7937
3	<i>NHbx</i>	-10.0	-37.3	-35.7
4	<i>Hdays</i>	10	34	3
5	<i>buyr</i>	1.2	5.0	0.2380
6	<i>xbuy</i>	2	1	3
7	<i>nbuy</i>	4	3	4
8	<i>buyvolr</i>	7.5	2.2222	7.6190
9	<i>NLsx</i>	30.0	2.3810	29.3651
10	<i>NHsx</i>	0.0	-3.8888	-11.9048
11	<i>Ldays</i>	50	38	43
12	<i>sellr</i>	1.9	4.7619	1.9048
13	<i>xsell</i>	3	1	3
14	<i>nsell</i>	4	16	4
15	<i>sellvr</i>	10.0	16.1905	10.1587
16	<i>sellvr2</i>	2.2	7.6190	2.2222

Table 8. Decision variables for T87 and DNS over training segment and total period.

Data Segment	Criterion	T87 Value	DNS Value
TRN	Profit & Loss <i>pnl</i>	572.14	595.70
(01/02/1987 - 12/31/1995)	Δpnl		4.12%
	Objective v	344.12	356.97
	Δv		3.73%
OS1+OS2+TRN	Profit & Loss (<i>pnl</i>)	1018.74	1069.52
(01/02/1978 - 5/16/1997)	Δpnl		4.98%
	Objective	1573.36	1819.28
	Δv		15.63%

Table 9. Comparison of optimization results from T87 and DNS.

	OS1 T87	OS1 DNS	TRN T87	TRN DNS	OS2 T87	OS2 DNS	TOT T87	TOT DNS	OS3 T87	OS3 DNS
Total Closed Trades	15	14	9	9	3	3	25	24	4	4
% Winning Trades	66.67	71.43	100.00	100.00	100.00	100.00	80.00	83.33	75.00	75.00
Largest Gain	48.01	48.01	44.86	44.86	19.20	19.20	48.01	48.01	42.85	42.85
Largest Loss	-6.37	-6.37	0.00	0.00	0.00	0.00	-6.37	-6.37	-4.02	-3.92
Average Gain/Trade	10.06	11.32	15.82	15.98	11.14	11.14	12.56	13.47	15.79	15.81
Maximum Drawdown	-22.74	-18.41	-13.94	-12.62	-6.18	-6.18	-22.74	-18.41	-10.80	-10.80
Total Gain	273.85	300.02	252.13	256.71	36.70	36.70	1472.6	1604.6	71.81	71.98
CAR	15.95	16.51	15.02	15.19	25.63	25.63	15.49	15.83	24.58	24.62
% Time Invested	82.06	78.71	86.68	87.91	89.37	89.37	84.66	83.63	92.30	92.46
CAR While Invested	19.77	21.43	17.53	17.46	29.15	29.15	18.55	19.21	26.91	26.89

Table 10. Results for T87 and DNS Trading.

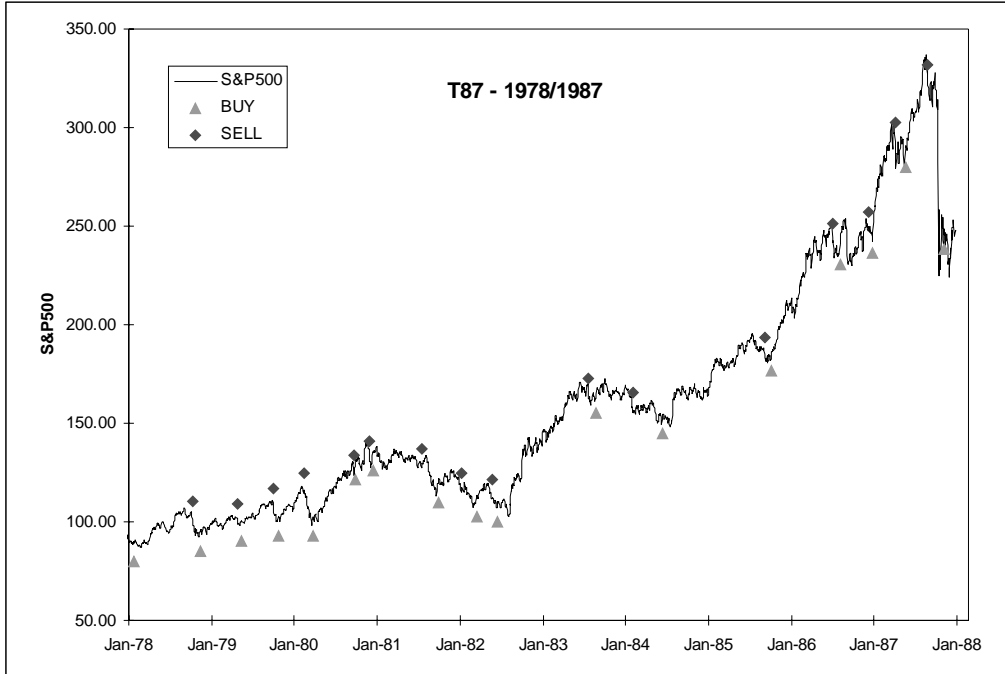


Figure 4. Trades T87 between 01/02/1978 and 12/31/1987.

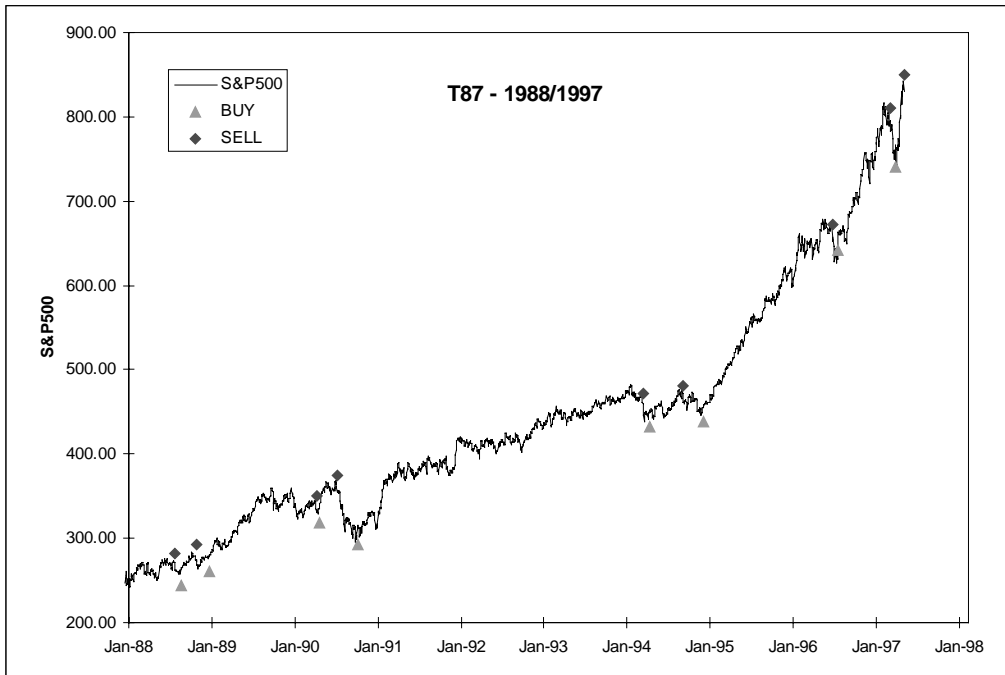


Figure 5. Trades T87 between 01/01/1988 and 5/16/1997.

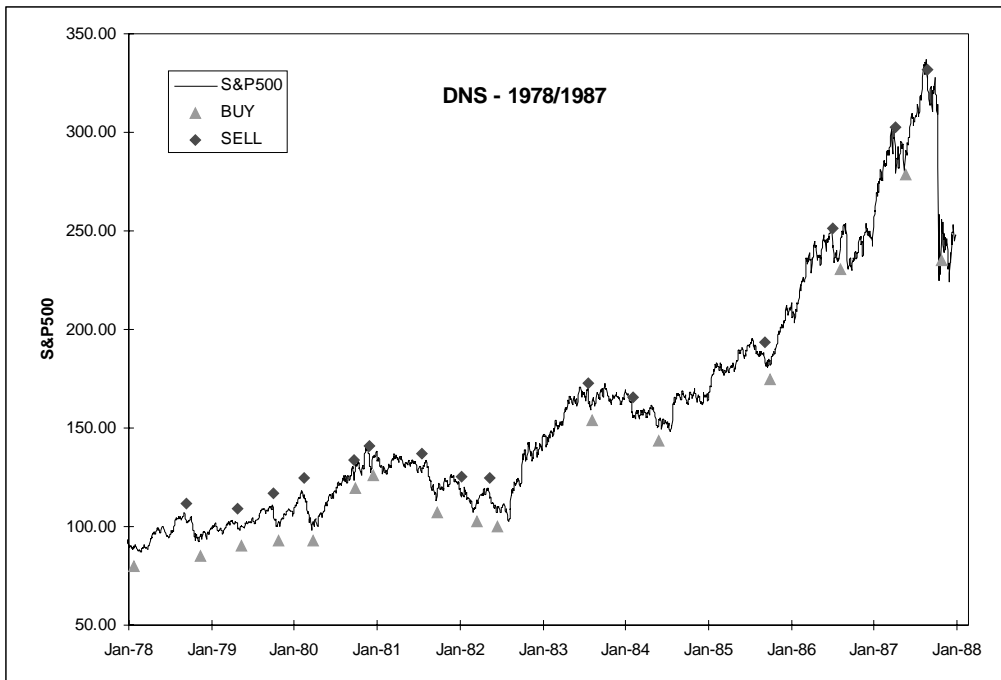


Figure 6. Trades DNS between 01/02/1978 and 12/31/1987.

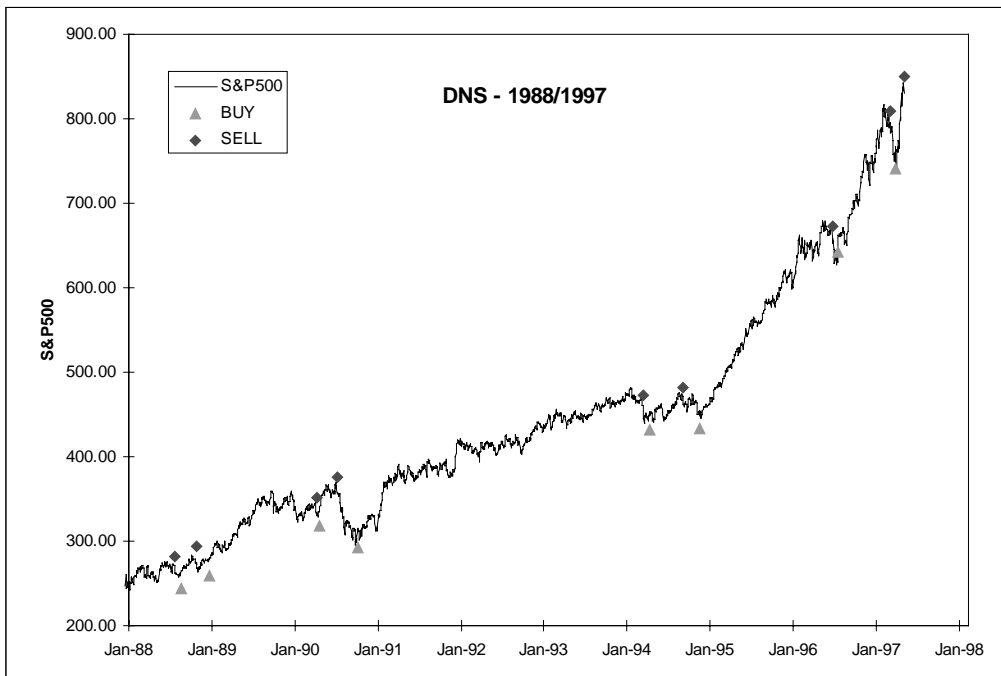


Figure 7. Trades DNS between 01/01/1988 and 5/16/1997.

3.2 SENSITIVITY ANALYSIS

To ensure that local optimality had been achieved, a sensitivity analysis was carried out. Starting from the given solution, each variable was changed in a range from -10% to +10% while keeping the others fixed. In case one variable has a value of zero, it is incremented in absolute steps of 1/10 (marked with (*) in the legend of the figures). With these sets we evaluated the performance of the trading system over the entire period (01/02/1978 - 5/16/1997). The necessary condition for a multivariate local maximum is that the gradient there is zero and the Hessian matrix is negative definite. Plotting the objective value versus the changing variables, the necessary conditions for a local optimum would be a set of concave functions with horizontal tangent at the original solution.

First we look at the given solution by Meyers [1997] and test it for local optimality.

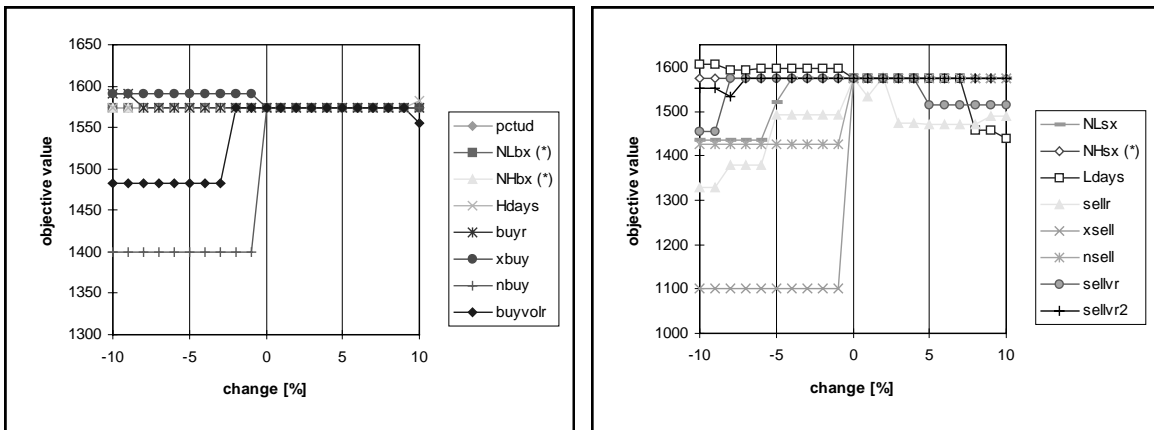


Figure 8. Sensitivity Analysis for T87.

Figure 8 shows clearly that the given solution is not a local optimum: By changing the variables *Hdays*, *buyr*, *xbuy* and *Ldays* it is possible to improve the objective value relative to the given solution. The sensitivity analysis for the system presented in this paper, DNS, is shown in Figure 9 to be at least locally optimal. Local changes in all variables either decreases the objective function or leaves it unchanged.

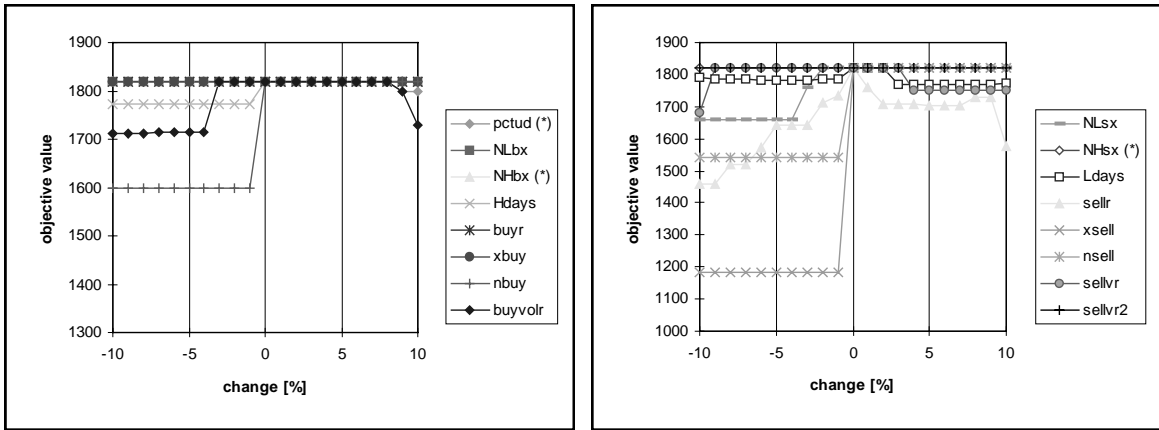


Figure 9. Sensitivity Analysis for DNS.

4. CONCLUSIONS.

In this paper we have shown that our Hybrid Algorithm is a very effective technique for optimizing a trading system containing 16 parameters with an extremely difficult objective function. This objective function can not be handled by classical methods of mathematical programming. The trading system optimized was developed by Meyers [1997]. His optimization used pre-existing knowledge from a previous system together with visual inspection of the data set. Our technique used no previous knowledge of parameter values, was much faster, and nevertheless obtained slightly better results.

5. References.

Gill, P. E., W. Murray, and M. Wright [1981]. Practical Optimization. Academic Press, New York.

Goldberg, D. E. [1989]. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Massachusetts, Addison-Wesley.

Hunter, S.M. [1998]. The Ultra Market Letter and Software. Ultra Financial Systems, Flower Mound, TX.

Meyers, D. [1997]. "The Turbo A/D, NH, NL Market System." Technical Analysis of Stocks & Commodities, August 1997, 16 - 29.

Saunders, M. A. and Voessner, S. [1998]. MINOS. Stanford, Systems Optimization Laboratory (SOL).

Voessner, S. and R. Braunstingl [1996]. G.O.A.L. (Genetic Optimization ALgorithm). Graz, Austria, Genetic Optimization Lab.