

# **THE GRAIL GENETIC OPTIMIZER**

## **Tutorial v. 1.2**

### **Copyright Technovest (Pty) Ltd, 2005**

By John J. Fanning, Executive VP, Product Marketing

#### ***GGO Rationale***

When most system traders refer to the “Holy Grail,” they are picturing in their minds the Ultimate Strategy that makes money in all markets in all timeframes. Such a strategy does not exist.

Calling ourselves “The Grail” in the context of system trading implies that we have a truly substantial set of statistical tools to offer the system trading market. Our name implies that we are implementing methods and techniques which if applied rigorously and consistently, will work.

The Grail—our software implementation, not a mythical trading strategy that works all the time in all markets in all timeframes, is simply a set of tools to help you find strategies that match markets or markets that match strategies. When you stop to think about it, that’s just common sense!

Now when a strategy truly matches a market, in other words when that strategy resonates with some non-noise aspect of the personality of that market, the resulting systems will walk forward in real, live trading on previously unseen data. Walking forward on unseen data is the only kind of trading you can do in real life whether you automate your trading or consider the charts and indicators and place your trades manually.

In order to find systems that will walk forward in real life, you must test those systems accordingly. In other words, you must subject those systems to a statistically rigorous series of tests on hitherto unseen data. In plainer terms, you have to be brutally honest with yourself and with your systems if you expect them to walk forward with any degree of reliability or robustness.

Unless a system has been tested on previously unseen data, preferably involving a reasonable amount of unprecedented (and therefore, unpredictable) price movement, it remains for all practical purposes, an untested system. If you want systems to walk forward in real live trading, you must design, build and test them to walk-forward standards.

This involves doing not one, but a series of tests on what statisticians call out-of-sample (OOS) data. In other words, you have to observe and quantify how your system will behave when you optimize it to a time span of known data and then turn it loose on a time span of unknown data. This is known as walk-forward testing.

Incidentally, let me clarify right here how I will use the terms Market, Strategy Components, Strategy, and System. A market is any eligible tradable you plan to buy or sell, be it a stock, option, futures contract, currency pair, etc.

Strategy components are the entries like moving average crossovers, the scaling components like trailing stops, or whatever other basic rules and components comprise your overall trading strategy. A strategy is a computer algorithm that you can apply to a market in TradeStation that generates buy and sell signals. A system is a market with a strategy applied to it.

All of your testing is ultimately done on systems. Testing a strategy by itself is...well, impossible! But even if you could do it, it would be like trying to determine whether an eyeglasses prescription is right or wrong without having the patient in the examination room.

The entire Grail suite is conceived, designed and built around the concept of performing statistically defensible walk-forward testing. The Grail is nothing more or less than the software implementation of a proven path. This is exactly what we mean when we call our product The Grail.

### ***Can't I Already Test Systems in TradeStation?***

Of course you can. In fact, if you're both an experienced discretionary trader and a system builder with some fluency in Easy Language and the TradeStation Strategy Builder, you can carefully study a market in a selected time frame, determine what basic entry strategy components will probably work, what scaling (including stop loss) components will probably work, and build a candidate system in less than a day with a minimum of brute-force combinatorial searches with most of those being for the fine-tuning of parameters.

I say candidate system, because unless and until that system has passed a series of walk-forward tests, you have no idea whether the system will do well on unseen data or not. All you know at that point is that given a range of known data, you have matched entry and exit points to the market that appear to make money. But for the mathematically inclined, you can do the same thing with an  $n$ th order polynomial approximation that represents something approaching a perfect curve fit but that has *no predictive value whatsoever*.

To see how your system will do on unseen or OOS (out-of-sample) data, you can optimize your system on a section of known data by manipulating the date properties on your chart and run your optimization then "unmask" the remaining data and see how your

system performs. Typically, performance on the hitherto unseen data segment is extremely poor.

This is due to the natural, statistical tendency of an optimized system to be over-optimized. It makes good money when you hand it tomorrow's Wall Street Journal because all the high, low and sideways price action is "visible" to it. It captures anomalous highs or lows right along with moves that are due to the underlying "personality" of the market.

The problem is that when you turn an over-optimized system loose on unseen data, the moves due to the underlying personality of the market tend to persist while the anomalous moves don't. Try to trade a system optimized to a mix of signal (underlying personality) and noise (anomalous price movement) and you have a good operational definition of "drawdown."

### ***So How Do Successful System Traders Test?***

Successful system traders carry out a rigorous series of tests before they ever consider deploying a system in real live trading. All of these tests can be implemented in TradeStation, but they take lots of computer and operator time to accomplish. Your author has been there and done that.

A careful system builder typically runs an optimization on an in-sample time section, imports the logged results into a spreadsheet (yes, this could be thousands of records), applies sorts and filters to that spreadsheet and then starts to make decisions about which parameter sets might be considered...for further testing.

One of the most common and effective techniques used to filter optimization results for potential future robustness is to take the very best net profits, profit factors, t-tests, or whatever performance criteria are being used to judge the parameter set, *and automatically throw them out*. Throw them out? Yes, because if a result is too good to be true, it probably is. The very best-looking results of an optimization are almost always (but in very rare instances, not always), hopelessly curve-fit.

Think of it this way. You have handed your system tomorrow's Wall Street Journal and it has faithfully modeled the entry and exit points it could "see." These inevitably include some fairly extreme and highly atypical, but at that moment, highly profitable signals. Will those special moments ever occur again? Probably not! So if you want really honest results, try taking away tomorrow's Wall Street Journal (or Barron's or Investors Business Daily or The Financial Times) and then see how well your system does!

If your testing is thorough, you will repeat multiple tests over different time segments. After a while, if the market you're testing has a good underlying "susceptibility" to the strategy you're testing, you will begin to see a pattern of parameters that seem to do fairly

well most of the time on previously unseen data. These are the parameters that are most likely to walk forward profitably in real life trading.

If you're already an experienced systems trader, you know how labor-intensive and time-consuming this kind of testing is. You have to reset the time limits on your chart, change your strategy properties, run your optimizations, import log files into your spreadsheet, sort, filter and narrow down the results. Many of these steps involve decisions that are more art than science. Then you get to do it again and again, perhaps 10 or 20 times before you can be confident in your results.

Any reports or graphs, outside of what you can run on an individual optimization in TradeStation, are up to you and your helper applications. You have few if any standard reports or criteria sets that you could consistently apply unless you created your own and then maintained very strict statistical discipline over your results.

I've been there and done that, and it isn't easy, but it is worth the results. But the system trading community has been begging for an easier and better and less expensive way to do this outside of expensive, custom-coded environments. Enter The Grail.

## ***So What Is The Grail?***

We can answer that question on two levels. By way of review, the generic, non-urban-myth grail is a well-matched strategy and market that work well together as a system. That system is tapping into some robust, underlying "personality" trait of the market, such that when that system is turned loose on hitherto unseen data, the system continues to do well.

The "other" Grail of course, is the trade name we gave to a software suite that automates to the greatest degree possible, the complex, multi-step task of carrying out the statistically sound and defensible testing of trading systems. The core or heart or engine of The Grail is a GA (genetic algorithm) enabled optimizer called GGO or the Grail Genetic Optimizer. The GGO makes easier and less time-consuming the complex process of conducting the rigorous and statistically defensible testing of trading systems.

It's well worth the effort to get to know the GGO because it's the engine that drives the GWFO (the Grail Walk-Forward Optimizer) and the CASB (Computer-Assisted Strategy Builder). If you achieve a good understanding of how the GGO works, it's very easy to use the GWFO and the CASB.

## **How The Grail Uses Genetic Algorithms**

With The Grail, you use a sophisticated code generator to enhance the Easy Language code of your strategy to make it GGO (Grail Genetic Optimizer) compatible without changing its underlying logic. Once this is done, you can perform complex combinatorial

searches canonically, that is to say, simultaneously on multiple parameters. The system then automatically creates entire directories of files and reports that permit further walk-forward testing.

In a few hours, a properly set up Grail run can perform the goal-directed, genetic optimization of several simultaneous parameters that would take years of computer time if done by brute force exhaustive sequential combinatorial searching. But as we'll see shortly, there are things that brute force combinatorial searching can't accomplish, even if you could tolerate decades long optimization runs.

If you'll pardon the play on words, genetic algorithms are not *generic* algorithms. In other words, there are about as many ways to implement genetic algorithms as there are ways to implement a spreadsheet. Sure, your spreadsheet will have rows and columns just as a genetic algorithm will be implementing some way of systematically concentrating on what's hot and just as systematically ignoring what's not. But there are lots of variations on that theme.

The GAs in The Grail are specifically chosen and implemented to detect underlying patterns in noisy financial data. Let's look at the ways we're implementing GAs in The Grail to maximize the likelihood of creating robust systems.

OK, I mentioned that genetic algorithms can do some things that brute-force sequential combinatorial searching with untransformed numeric parameters simply cannot do. GA searches work with encodings of parameters instead of straight parameters. This is handy when you don't know the exact real-world range of numbers you should be looking for on a particular market. After all, if you have a good idea of what range that floor amount or that number of ATRs should be, then why are you conducting an exhaustive search for it?

Using encoded or transformed parameters also tends to blur peaky, multi-modal results that can occur when you're searching sequentially on untransformed parameters. The GA doesn't "know" the actual value it's looking at, it only knows that value works. The GA working with transformed parameter values is far less likely to see a peak in net profit due to a really freak price move that will never happen again.

The Grail GA implementation uses pseudo-random numbers to choose multiple, simultaneous search points. The result is that the search is over a population of points and not single-increment next steps of a particular parameter. This also tends to blur and ignore modal performance peaks that can occur if you're incrementing a single parameter by single steps. You do NOT want your system to think it's "good" because it caught that Really Freak Price Move because the Percent Trailing Stop Floor Amount was set to 572.

After all, how often do you expect that Really Freak Price Move to ever occur in the future? By testing multiple variations of all the parameters in parallel (mathematically, testing a population of strings), metaphorically by searching for many points on a high

plateau instead of a single high peak, your probability of finding great looking but brittle results is greatly reduced.

The GAs implemented in The Grail use trading system-oriented fitness functions (typically NOT total net profit, although a net profit goal is an option) to guide an effective search towards better and better parameter sets. This is goal-directed searching for a realistic, calculated, statistically appropriate goal.

Because it's using a canonical search, The Grail GA looks for the best parameter values together, yet separately. Each individual parameter gets to look for its own optimal goal-seeking value. But the final result of each iteration of the optimization must be due to each of the individual parameters making its best contribution to the results of the entire set. In the language of mathematics and set theory, this is called a canonical search scheme, since like the chapters of a book, each parameter contributes its information to the whole body (canon) of knowledge.

This eliminates a Really Great Looking Result occurring because of a Really Improbable Combination of Parameters. In other words, in a brute-force, sequential combinatorial search, you might just get unlucky enough to achieve a magnificent-looking result that happens because you're looking at a combination of parameters interacting with your market data sample in a way that will never be found again in Nature. This kind of result is not something you want to walk forward with your trading capital.

This process can be described in technical terms like "avoiding local optima." But for those of us who struggled with huge spreadsheets and 3-dimensional graphs and subjective choices, it means that instead of avoiding parameter peaks and choosing parameters near the middle of some graphical high plateau, we're implementing a statistical system that does all this by the nature of its design and implementation!

Finally, GAs use probabilistic transition rules, not deterministic rules. The deterministic rule that a brute-force sequential combinatorial search uses is that each parameter is incremented to the next step one parameter at a time and one increment at a time. This approach is the most likely to match and curve-fit any and all atypical or anomalous price movements in your sample. The GAs implemented in The Grail use random (well, technically, pseudorandom) choices across the sampling range to make sure that the entire range of potential values gets sampled *without having to conduct an exhaustive search*. The GA then records and concentrates only on those regions of the search space that yield promising results.

As a bonus, this sampling method reduces the number of necessary iterations by a factor of a thousand or more since the GA learns where are the "dead" parameter ranges in your search and systematically ignores them. Concentrating only on the productive parameter ranges allows the GA to accomplish in an overnight run what could take years if done as a brute-force combinatorial search. And of course as described above, the GA search accomplishes several things that the brute-force combinatorial search simply cannot.

Taken together, these differences contribute to a genetic algorithm's robustness and resulting advantage over other more commonly used search techniques. You can find more information on the mechanics of Genetic Algorithms in the following book:

D.E.Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

## ***Procedural Summary***

So, here's what we're going to do. First we will choose our market (in this case, EURUSD) and pick a timeframe (in this case, 60 minute bars).

Then we will begin to test strategy components against that market. We'll start with entries (e.g., Bollinger, MACD, Keltner). Then we'll add scaling/exit strategy components, both profit-taking (e.g., Percent Trailing, Channel Trailing) and capital protection (Break Even Stop, Stop Loss).

We will do this until we have arrived at a group of higher-level strategy components that seems to do well with this market. Bear in mind that this is as much an art as it is a science. Once we have arrived at a set of strategy components we think will work, we will create a combined strategy out of it in Easy Language.

This is necessary because we will then be using the GGO Smart Editor, to convert this Easy Language strategy into a strategy specially enhanced to use the Grail Genetic Optimizer. This conversion does not change the underlying logic of the strategy, but it does had all the hooks into the Grail Genetic Optimizer.

Then we will go through the regular GGO optimization with stress testing and out-of-sample testing that will take us up to the point where we're ready to run a walk-forward analysis and cluster analysis of the resulting system using the GGO/GWFO environment. Once we determine a robust set of parameters for our strategy, we can plug those in to the original Easy Language strategy, double check our results by using the standard TradeStation performance reports and even deploy our system if we choose.

Bear in mind that even at this stage, we will have subjected our system to far more rigorous testing than most system traders ever do. We will be able to compare hundreds, even thousands (if you have the patience) of equity curves and observe the OOS (out-of-sample) performance of all of them.

## **Further Testing You Can Do**

More typically, we will subject our system to a comprehensive walk-forward analysis, using The Grail Walk Forward Optimizer (GWFO), but regardless of whether we do this,

this initial GGO testing is always the first step. And the GGO writes all the reports and files that the GWFO requires to carry out its more comprehensive walk-forward analysis.

At the end of this initial GGO process, you will have accomplished one of two crucial goals. Either you will demonstrate that your new system should never be let out of its cage, or that it is robust, profitable, has a nice linear equity curve and has demonstrated robustness even with out-of-sample data. If as we recommend, you perform further analysis on your data with the GWFO, will even get a quantitative recommendation for a re-optimization interval based on the observed “shelf life” of your market/strategy combination (i.e., your system).

Or you will discover that on that market in that time frame combined with that strategy cannot create a stable robust system. Like the little veterinary/taxidermy shop back in Grand Rapids, “Either Way, You Get Your Dog Back.” In other words, if you think it’s frustrating to watch your system disintegrate and create huge drawdowns in testing, just imagine how frustrating it would be to deploy it and watch it dissipate away your trading capital in real life!

Even if you don’t do a Grail walk-forward optimization, you will have done far more realistic preparation and testing than very many traders end up performing on their systems. Technically, the GGO can be used very well as a stand-alone product, although many of its best features are geared towards the further comprehensive testing of your systems.

## **Did Anyone Say This Was Simple and Easy?**

Nobody here at The Grail said that. System building is a challenging endeavor. But it’s not impossible. The purpose of this tutorial is not to discourage you, but rather to show you how, given a powerful enough tool set and the knowledge to use it properly, you can actually build systems that have a statistically defensible chance of success on hitherto unseen data (the only kind you can actually trade with in real life!).

System building in The Grail is a computer-assisted process. Think of The Grail as a set of really nice, purpose-built power tools with a considerable level of automation. A toolset might not automatically make you a carpenter, but it’s pretty tough to be a modern carpenter without a good tool set.

## ***Step One: Choosing Your Market***

To avoid lengthy discussions that are beyond the scope of what is intended to be primarily a procedural tutorial, I will from time to time refer to a book called [Design, Testing and Optimization of Trading Systems](#) by Robert Pardo. This tutorial is intended to help you figure out how to use the GGO, so I will try to keep system development theory, computer preparation and any other issues to a relevant minimum.

The highest abstraction level decision you will make as a system (or for that matter, discretionary) trader is what markets you want to test and trade. This is anything but a trivial decision. Markets vary radically in leverage, liquidity, volatility, short-ability (especially in the case of stocks) and in overall “system susceptibility.”

There are decision matrices available for those traders who want to add some structure and objectivity when weighing things like leverage, liquidity, volatility and potential risk of various types of markets. Different markets (equities, futures, FOREX, derivatives) have comparative advantages and disadvantages. They also trade differently just as different symbols among equities for example, trade differently.

You must decide what markets and timeframes are right for you and whether or not you want to accept the risk of overnight positions. As much as anything, your own personal trading style and psychology comes into play here. We’re back to our eyeglasses metaphor again. Glasses aren’t “good” or “bad.” They’re either properly or improperly fit to an individual.

If your trading system doesn’t fit you psychologically, you won’t allow it to trade even if it’s a good system. Hence, the saying, “Trading systems work, system traders don’t.”

### ***The Importance of Statistical Integrity to Your Systems***

However, let me add a real key point here regarding system trading and psychology. If you have created a statistically sound, extensively walk-forward tested and quantitatively robust system, you will gain confidence in your system as it walks forward with predictable profitability.

You will also gain confidence in your ability to build and test systems. Even more importantly, you’ll gain confidence in your ability to reject systems that don’t perform well on hitherto unseen data before you attempt to trade them and lose money. Since you’ll probably build and test ten systems for every one you eventually deploy, you’d better get good at rejecting bad systems.

By building, testing and deploying systems that trade profitably, you’ll gain true, reality-based confidence in your systems. You’ll be able to ride out the occasional non-statistical-outlier drawdown and not turn off your system just as it’s poised to make money.

No trading seminar can build in you the kind of substantial confidence that comes from building, testing and deploying a successful system. Seminar leaders can shine motivation at you until you get a sunburn, but unless you’ve lost contact with reality, you’ll never confidently trade a system **THAT DOESN’T WORK.**

No one in their right mind should ever trade a system that has no statistically defensible chance of making money. To attempt it with confidence or optimism just adds insult to injury! If anyone ever tries to teach you to system trade with confidence without teaching you how to create winning systems first, you should exercise your refund privileges (if they offer them) and leave, or just leave and save what's left of your weekend.

## ***Let's Get Started***

For this tutorial, we will use a FOREX symbol EURUSD on 60 minute bars. Using a currency pair avoids issues with data transforms on expiring futures contracts as discussed by Pardo in the Price Data subchapter in Chapter 4 of his book. We will use four years of historical data for our tests. The Size of the Test Window subchapter of Chapter 4 of Pardo (pp. 46-52) discusses sample size in system testing.

For the record, I personally trade equities, futures and currencies and occasionally, options on equities (a first-order derivative). Just be aware that each of these market areas has its own challenges and advantages. TradeStation for example, still requires auxiliary software (DynaOrder) to place automated FOREX orders through a third-party broker/clearinghouse. As of this writing, TradeStation is officially announcing single-source responsibility end-to-end automated FOREX trading.

Using a moderately long time interval accomplishes several things. Very few real time discretionary traders have the patience to day trade 60 minute bars. But the longer time intervals do filter out much of the psychological noise you see in shorter timeframes. You should of course experiment with different bar intervals, but I recommend initial trials in the “forgotten middle” range time frames of 20, 30, or 60 minutes.

Remember that I am not ruling out n-tick, 3 minute or daily bars. I'm simply suggesting a starting point with regard to timeframe.

## ***Some Critical Notes about Sample Size***

If you are using an inadequate sample size, your system is MUCH more likely to get “blindsided” by never before observed price patterns when you implement it. Adequate sample size is the cheapest insurance a system trader can get. Unless a system is tested on an adequate sample of historical data, it is statistically impossible for it to walk forward reliably no matter how good the test results (yes, even the Grail's GGO/GWFO/Cluster Analysis reports) look.

No amount of post-processing, analysis, walk-forward runs, out-of-sample testing or anything else can compensate for an inadequate historical data sample. If you sampled half a year of hourly wind speed data in Florida during a season when no hurricanes (or

even any severe local thunderstorms) hit, you might come away with the impression that you had gathered a representative sample of Florida wind speeds.

Now imagine that you run genetically sampled GGO and GWFO walk-forward tests with different OOS (out-of-sample) exclusion percentages and stress test coefficients on that same short sample of wind data. Then to get a quantitative idea of the number of walk-forward run and out-of-sample exclusion percentages that work best when walk-forward analyzing your system, and to secure multiple confirmations of your results, you perform a cluster analysis. And let's also assume that you're unlucky enough that everything you run passes all the logical tests for reliability above the accept threshold and comes out looking OK.

Unlucky, you say? Sure, because you've just set yourself a bear trap with nicer camouflage than most. You can slice and dice your data until your computer goes obsolete. You'll still never "see" the occasional hurricane that strikes Florida. That's because you're attempting to draw a set of complex statistical inferences out of a pathetically inadequate sample of data. The longer your sample, the higher the probability that the sample contains most of the conditions it's likely to encounter in the future.

It's human nature to want to curve-fit a "beautiful" system to a short sample of data. The Grail GGO contains several built-in and user-configurable features to help you avoid the creation of a curve-fit system that will disintegrate when it "sees" out-of-sample, true blind walk-forward data. Nevertheless, you can still spoof it pretty well if you throw it a short and unrepresentative enough data sample!

Permit me to acknowledge that sample sizing is a complex topic. Yes, there are high-risk but valid systems that trade very frequently, that may have a "shelf life" (the interval between re-optimizations) of a day or two and that might look at relatively short histories of (typically futures) data in n-tick or minute range time frames.

To a limited degree, data bars exhibit fractal properties. In other words, 25 years of daily data consists of about 6,000 bars. A year of 15 minute cash session bars also consists of about 6,000 bars, and both charts will exhibit similar patterns of trends, support and resistance, consolidations, double-bottoms, channel breakouts and other manifestations of the dynamic interaction of the group psychology of bulls and bears (and of the systems they create and trade!!!).

But does this mean that a year's worth of 15 minute data is equivalent to 25 years of daily data? No, it doesn't, any more than our six months' wind speeds in Florida data would record a hurricane any better if its sampling interval were 3 minutes instead of 60 minutes. So be careful, very careful with short data samples regardless of timeframe or trading frequency or even a short anticipated shelf life.

You may want to test on shorter data samples when you're doing very preliminary prospecting for markets or bar intervals just to save some computer time. But let me

volunteer that with very few exceptions, if you're planning to build, test and implement a any system on, arbitrarily, less than 3 or preferably 4 years of historical data, right here is a great place to stop!

## ***Preparation and Prerequisites***

This tutorial assumes that you have installed or imported all of the necessary components of GGO and GWFO and that you are running a current build of TradeStation. (I always leave the last stable build of TradeStation installed when upgrading to a new version, just in case you discover a bug in the new version that's a showstopper.) We're assuming that you have a reasonable navigational familiarity with the GGO, the GWFO, with TradeStation in general and with the Easy Language editor.

If you have not yet been through the mechanics of setting up or registering GGO or GWFO, you'll want to go to the Help -> Getting Started -> Where do I start option for all the steps required to set up the applications. The first two steps will take you to the point where you can perform the rest of the steps in this tutorial.

The Grail software suite is of necessity very, no, extremely computer and file intensive. I considered leaving much of the following material out of this tutorial, but when you've watched powerful, new, well-configured PCs drag to a stop like a mule knee-deep in mud, it's a good reminder that all the procedures and parameter settings in the world get pretty academic when the computer you're running them on locks up.

## **Performance Considerations**

There are some very simple things you can do to improve the performance of your computer. This could include deleting obsolete or un-needed directories of files from previous GGO runs. You should implement a generously sized (start with 2x the amount of RAM installed in your PC) static paging file in Windows. A static paging file is much more efficient than a Windows managed dynamic paging file.

You should defragment the files on your hard drive. You should not run any unnecessary applications during long optimization runs. It's a good practice to re-boot your PC before invoking long optimization runs.

I have observed that the repeated creation and deleting of gigabyte sized directories with tens of thousands of files can eventually lead to rather extreme entropy (scattered randomness) of data on the hard drive. The free utility supplied by Microsoft to perform disk defragmentation eventually stopped working for me, even on a 60GB disk with over 40% free space.

Raxco makes a business-grade disk defragmenter that does a good job of organizing badly fragmented disks. Raxco is the only defrag utility vendor with an official

endorsement from Microsoft. Raxco is also capable of organizing disk data by access frequency to help minimize further fragmentation. I have observed significant performance improvements (by significant, I mean from not grinding to a halt to measurably better) in large Grail optimizations by maintaining a well-organized hard disk. (No one at The Grail has any business or personal interest in Raxco, by the way).

A category of application that can really impact the performance of various Grail applications are virus checkers. You might want to conduct a brief, timed test run with your anti-virus software turned on and off to see if it is having an adverse impact on run times.

If, and only if, you're very comfortable with making low-level configuration changes to your computer, you can try turning hyperthreading (HT) off. This will probably improve your computer's performance on a highly scalar ("one track") task like a TradeStation/Grail optimization. Note that turning the HT feature off will probably deteriorate your computer's performance in complex, mixed task environments like running TradeStation with several open workspaces in real time.

You might try starting a GGO optimization and check the estimated completion time with HT on. Then re-boot, re-configure your PC with HT off and start the same optimization. If you then observe a significantly faster estimated completion time, you could save several hours or even half a day or more by configuring HT off before doing long optimization runs. Just remember that the computed estimates of completion times are based on extrapolations of your computer's performance in its current state.

I do NOT recommend writing different parameters to your PC's ROM (read-only memory) on a daily or any kind of routine basis. If you have multiple PCs and want to set one up as an optimization platform, you might want to configure it with HT off and leave it off. Most ROMs will not take regular re-programming without failing.

Some motherboards and chipsets or BIOS versions support hyperthreading better than others. I will simply remind the reader that computer performance and benchmarking is a complex, multi-faceted topic. There are no real "silver bullets" in computer performance other than the good engineering and integration of system components and a good match between a computer system and its application load.

RAM data get fragmented just like your hard disk data. Your disk data will be more fragmented after creating 15,000+ new files than it is now. The condition of your virtual memory paging file, and unsuspected bottlenecks like video RAM reserves, background processes, etc., can slow down your PC. Just realize that the lower completion time estimate that you might see with HT turned off is a theoretical best-case scenario.

Computer performance benchmarking and tuning is an extremely complex topic. Tweaking a computer is like peeling an onion. In other words, fixing one performance bottleneck will simply take you immediately to the next bottleneck. If a particular system attribute like CPU clock speed or RAM doesn't happen to be the performance bottleneck

for a particular real-world application on your system, then over-clocking your CPU or adding RAM will accomplish absolutely nothing.

Both TradeStation and the Grail suite seem to like larger L2 cache sizes, but this is a hardware attribute that's part of your computer. Remember that using genetic algorithms is cutting your optimization times by a factor of thousands. If you're shopping for a faster computer (the best way I know of to improve performance!), look for one that does well in floating-point and disk read/write "synthetic" benchmarks. While you're at it, remind your kids to do well in reading and math!

With that said, just remember that ethical and honest benchmark vendors openly acknowledge that their benchmark categories are in fact, synthetic. In the final analysis, demonstrated performance with the actual application(s) is the ultimate "benchmark." To accurately assess this, you need personal integrity and a stopwatch.

Bear in mind that I routinely complete the largest Grail optimizations (with the exception of a cluster analysis that does 30 walk-forward optimizations) in less than 24 and sometimes in less than 1 or 2 hours on PCs I would describe as "ordinary high-end" machines. One is an IBM T41p laptop with the 1.7GHz Mobile Pentium/Centrino configuration with a 1MB L2 cache, 2GB of memory and a 7,200 RPM hard drive. Various architectural features like a generous L2 cache, a fast hard drive, a well-engineered chipset and generous video RAM make this slim, quiet laptop a surprisingly fast "benchmark racer" and real-life number cruncher.

My other Grail machine (my wife is the primary user) is a dual-core (i.e., hyperthreading) 3.6 GHz Pentium with 1MB L2 cache, 2GB memory, and a 10,000 RPM SATA hard drive with an 8MB cache. My Asus P4P-800 motherboard supports hyperthreading nicely and for the record, I do NOT bother with turning off hyperthreading on this machine. Both of my machines turn in satisfying to impressive synthetic benchmark performance numbers from SiSoftware Sandra, but neither one of these machines is a \$10,000 dual Xeon server/workstation with a 15,000 RPM SCSI RAID array and cooling fans that sound like a beehive on steroids.

Of the two machines on which I commonly run The Grail Genetic Optimizer, my Centrino laptop machine is *slightly faster on optimization runs* even though it doesn't support multiple open TradeStation workspaces and charts as well in real time as the desktop system described above. Go figure.

If you are really into Exotic multi-CPU machines are cost-efficient on a highly specialized subset of applications, they're expensive to buy and they use more electricity than a refrigerator. My point is that you can run The Grail very successfully on an ordinary, cost-efficient (and quiet!) mid-to-high-end PC as long as you do the necessary "housekeeping" to keep it reasonably well-tuned.

## Security Considerations

Sadly, another factor that can decrease your PC's performance (or eliminate it entirely!) are Trojan horses, worms, spyware, malware, scumware, viruses, keyloggers and other pieces of code designed to sabotage, hijack, spy on or otherwise compromise the integrity of your computer. In a best case scenario, a piece of malware might end up soaking up billions of CPU cycles performing worthless tasks and slowing down your optimizations.

In a worst case, a malicious worm hidden in a Trojan horse e-mail attachment that gets carelessly opened behind your firewall could in milliseconds, cause damage that would cost more to fix than the cost of your computer, or it could infect and compromise other networked computers behind your firewall. Microsoft Help and Support and links available on MS Help and support can guide you to effective means of configuring, updating and protecting your PC against malware. Make sure you have Service Pack 2 installed and updated. Make sure that its protective features are turned on. The best defense against malware of all kinds is prevention.

If you want to exercise some aggressive preventive maintenance on your PC, go to <http://www.microsoft.com/windows/IE/community/columns/bugbusting.msp> This link will lead you to some of the best *legitimate* procedures and products for keeping your PC clean. Sadly, *the majority* of products that a Google search for "anti-spyware" turns up are for products that may turn out to be *just as malicious as the malware you're trying to get rid of!* Don't trade one brand of "free" malware for another that you pay for!

If you're operating a private hedge fund or even trading very actively with a larger account, your computer is sending and receiving information that is of great interest to unscrupulous individuals or organizations. Check out this ComputerWorld article: <http://computerworld.co.nz/news.nsf/0/333248A4D5914E0CCC25702600127044?OpenDocument&pub=Computerworld>

Hackers are now targeting financial computer installations the same way daylight armed robbers target banks and armored trucks. Or the way pirates used to target gold carrying Spanish galleons. Is any of this really a surprise to anyone?

Today, computer security is just as necessary as standby power and surge protectors. The good news is that deploying inexpensive firewall/routers and spending a few minutes a week spent on security administration can keep you system safe against anything but a physical break-in to your office. Don't leave your expensive SUV in the parking lot with the keys in the ignition switch, and don't neglect your computer security updates.

## Don't Uninstall Your Previous Release of TradeStation

While we're on the topic of practical considerations regarding your computer environment, notice that we're not advising you to immediately un-install the previous, stable, tried-and-true version of TradeStation the moment the next build comes out! Not

only can new releases have bugs, but if you have highly customized Easy Language code in any of your strategies, customized code is the most vulnerable to any changes made to the underlying logic of the next release of Easy Language simply because TradeStation has no way to test your custom code against the new release.

This is one reason why I prefer to use mostly unaltered TradeStation-published Easy Language strategy components in my strategies. And it's a reason why Grail Systems always tries to use Easy Language in its applications in the most standard and straightforward ways. That way we suffer relatively few release update problems that are related to Easy Language.

You may also encounter other problems like slow optimization performance with a new release. If you checked out all your strategies, indicators and other Easy Language applications to your satisfaction and everything appeared to work without requiring any Easy Language changes or re-verifications, do not take this as the green light to go ahead and un-install that last tried-and-true release.

## **A Few Notes about Event-Driven Fourth-Generation Languages**

By the time you're finished with this brief chapter and unless you're already highly familiar and skilled with Easy Language, hopefully you will think of it (and handle it) in ways you hadn't thought about before.

Easy Language is an interesting type of very high-level language called an event-driven 4GL or fourth-generation language. Easy Language isn't really all that easy. But in relation to what you'd have to do in a 3GL like C or C++ to achieve the same results, it's extremely easy. I suppose you could call it "A Whole Lot Easier Language"! LOL!

Easy Language also represents a very elegant solution to creating trading strategies. Developers like us consider it a good "environment" in the context of a complex, integrated trading platform like TradeStation. For the record, as traders and as software developers, we have tremendous respect for both TradeStation and Easy Language.

Unlike 3GLs like COBOL or C, each command in Easy Language may represent several hundred lines of underlying lower-level code. Also, because it is event-driven, the lines of code in Easy Language do not necessarily execute in any particular order, but rather when a relevant event occurs when the strategy containing that code is active (Status "On") for that market. Like a platoon of soldiers waiting for orders, Easy Language commands just sit tight until their name is called.

This property of the language is what makes it possible for you to stack strategy components in the strategy builder and have them act cumulatively to achieve the desired result. In other words, you might have a Bollinger Band long entry, a Bollinger Band short entry, together with long and short Percent Trailing profit stops and long and short Stop Loss safety stops stacked up in the Strategy Builder.

As long as these strategy components are present and their status is “on,” they all act as if they were part of one longer Easy Language program. Remember too that for the most part, the order of the Easy Language commands in your strategy is relatively unimportant. Easy Language doesn’t “execute” in any particular order. The Strategy Builder doesn’t “care” what order your strategies are in, for example.

The most common “Event” in Easy Language is a bar. Thus, if you apply a strategy to a chart with 1,000 bars, the first thing that’s going to happen is that TradeStation will start calculating Entries and Exits on the first bar and then for each subsequent bar. When you run an optimization, that event-driven process will occur for each and every combination of parameters (the number of iterations) that you run.

You might want to take this into consideration as you prepare strategies for testing, particularly if you’re planning to test very many parameters across very short time frame bars across a multi-year data sample. Remember, even genetic algorithms “only” improve your search performance by a factor of a thousand or so. And very short time frame searches begin to lose meaning across several years of historical data.

Now, the fact that Easy Language is event-driven does NOT mean you should ignore order and structure in your Easy Language strategies. Variable declarations, inputs and commands should be cleanly structured. Easy Language also supports loops and other structured methods that ARE position-dependent and order-dependent in your code. Just remember that event-driven or not, someone (look in the mirror) has to work with and maintain this code.

In fact, your ability as an Easy Language (or any other language) developer will be largely limited by your ability to maintain simple discipline and structure in your code. Is this really news? Compare the phrases “Applied symbolic logic” and “Applied symbolic spaghetti.”

I will sometimes move strategy components up or down in the Strategy Builder list just to organize them by function and hierarchy. I like to leave my long and short entry components at the top, profit stops in the middle and safety or capital protection stops at the bottom. I follow the same hierarchy when doing preliminary optimizations. But this is strictly a convenience for me. It has no effect on how these event-driven components behave.

Most of the strategy components you’ll ever need are available for free through TradeStation. Remember that technical analysis works because the markets are traded either by groups of traders, or by the systems that groups of traders create. Fear and greed are NOT complicated emotions. Most patterns of price movement are NOT complicated either.

Additionally, TradeStation publishes and offers some excellent courses in Easy Language programming. Unless you purchase protected-code systems from a third-party vendor or

implement third-party code without any customization, you will have to gain at least a working familiarity with Easy Language in order to system trade in TradeStation.

## **A Brief Reality Check Regarding Certain Markets**

Having said that, it would be naïve not to acknowledge that certain markets, particularly highly liquid and leveraged ones like the E-mini futures, often exhibit patterns of price movement that are challenging to match to any simple strategy. I like to call this type of market “system-saturated.”

Trading is a zero-sum activity, and you’re playing in a pretty battle-hardened arena when you choose to trade certain markets. Just don’t be too disappointed if your 98 tick E-mini futures contract doesn’t trade robustly on a nicely optimized single line moving average crossover.

The Grail does address such markets with some highly sophisticated products, including this one. Remember, GGO is a strategy optimizer, not a strategy builder. If you already have a strategy sophisticated enough to trade the most challenging markets, the GGO can help you get the most out of it.

We also offer a highly sophisticated product, namely The Grail CASB or Computer-Assisted Strategy Builder that allows the serious strategy builder to put on and walk in the league boots of heuristic algorithm assisted strategy building. And that’s the topic of a whole other tutorial.

## ***Set up the Strategy and your Market***

Markets are constantly trying to talk to you if you’re listening to them on the right frequency and if you’re using the right message decoders. I always try to let the market tell me which entries, profit stops and protective stops are going to work with it. In fact, the whole concept of a genetic algorithm enabled goal-directed search is based on the question, “please tell me what works,” implemented as applied symbolic logic.

This is how The Grail (Our Software Suite) is designed and implemented. Genetic algorithms achieve their efficiency by heuristically learning to ignore what’s cold and focus on what’s hot. And what’s hot is what achieves good results with a well-defined performance criterion.

Since we’re building a real time strategy for trading the US Dollar and the Euro currency pair (EURUSD), we first need to create a chart in TradeStation with this symbol with 60 minute bars. Set the first date in Symbol Attributes four years back. Depending on what you might already have cached on your computer for this symbol, it may take a while to retrieve the data, and TradeStation may download the history in repeated steps.

In this timeframe, the trading frequency will be what most traders would call swing trading. In other words, we can expect the system to open, reverse or close a position every day or two.

## **Discovering Your Easy Language Strategy**

The Grail Genetic Optimizer is appropriately named. In other words, it's an optimizer and to optimize a strategy, you must HAVE a strategy! It is not a strategy discovery or development tool. If you do not have a strategy for a particular market and you would like to model a new strategy from scratch, then you should use our sophisticated CASB (Computer-Assisted Strategy Builder).

Once again, since this tutorial is primarily procedural and not theoretical, I will not attempt to fully explain the complete rationale for building this strategy. Different strategies work on different markets for different reasons. In addition to Robert Pardo's definitive [Design, Testing, and Optimization of Trading Systems](#), the serious student of system design and testing will probably enjoy [The Encyclopedia of Trading Strategies](#) by Jeffrey Owen Katz, Ph.D. and Donna L. McCormick.

When used as it was intended, the TradeStation Strategy Builder can be very powerful. With a chart of your market open, you can substitute either public domain or proprietary strategy components in or out of the stack (or switch their status on or off) very quickly. Don't be afraid to throw some pasta at the wall and just see what sticks. Used in this manner, the TradeStation Strategy Builder works beautifully as a prototyping tool. You can work at the strategy components level to quickly determine what works with your market.

I am not implying that every market will respond to some combination of simple strategy components. But sometimes very simple systems will yield good results if they're appropriately matched and optimized to your market. This tutorial illustrates a very simple strategy on a major, highly liquid market that works very well in real life.

In general, you will find that markets that are trading in a range sometimes respond to oscillators like CCI (commodity channel index) or RSI (relative strength index). Trending markets or markets that reverse direction slowly and predictably do well with moving average crossovers. Markets that break in one direction and keep going for a while sometimes do well on channel breakouts or Keltner channels. Markets that look chaotic and downright schizophrenic often do amazingly well with Bollinger Bands.

There is no way to know before testing which strategy components or combinations will work well with a particular market, although most system building experts can look at a chart and make an educated guess as to what will work. Some theorists have attempted to create nice, clean classifications of what kinds of markets there are and what strategies will work with them in theory.

There are in fact, different market “personalities” that often do well with certain types of indicators. But I prefer to take the Fujita approach to markets and strategies. Fujita won the Nobel prize for creating a scale of tornado strength based entirely on inference. In other words, you look at the damage after the fact and then infer how strong the tornado had to be to “do that.”

If the tornado stripped off some siding and broke a few windows, you call it an F1 and extrapolate within a fairly narrow range what the wind speeds had to be. If the tornado erased all evidence of the former presence of a strong building, drove 10 inch steel spikes through utility poles and sucked most of the concrete pavement out of the parking lot, you call it an F5 and extrapolate its wind speeds accordingly.

## **Search for Your Strategy at an Appropriate Level of Abstraction**

Modern Doppler radar evidence has provided some remarkable confirmation to Fujita’s theory and the F-scale continues to be the way you measure a tornado’s strength. Even when radar can measure wind speeds in real time, tornados are classified after the fact by the damage they left behind.

I prefer to think of markets as “Bollinger markets” or “Keltner markets” or “stochastic markets” or “MACD markets” or whatever markets based on what fundamental long and short entries work with them. I’ll call a market truly chaotic not because it appears to have some arbitrary look of randomness, (which often but not always strongly hints at a Bollinger market, incidentally), but rather if nothing seems to work with it.

Now, go check out the strategy components that you get for free with TradeStation. You will notice that all of the major entry and scaling (profit and protective exit) strategies are right there in the library of strategy components just waiting to be stacked into the TradeStation Strategy Builder.

Unless you want to become an Easy Language developer, LOC (lines of code) is not the proper abstraction level for a systems trader to be working with Easy Language. If you have a natural talent for coding or if you really love to translate concepts into code, that’s one thing. But read the comments in the TradeStation supplied code, consider the underlying complexities and subtleties of Easy Language and ask yourself if you really want to develop strategies in Easy Language.

Ask yourself if you want to confront a potential maintenance challenge with each release update of TradeStation. Ask yourself if you have the talents and skills and cognitive style to be a competitive Easy Language developer. Then consider the fact that most markets will either respond to some relatively fundamental strategies or not. Then remember the fact that those strategies are sitting there waiting for you to try them in the Strategy Builder on your market.

Not every clever trader has the concurrent ability to develop strategies in Easy Language. If your intellect and cognitive styles encompass both disparate skill areas, congratulations. You have in your possession the most powerful optimization environment in existence. Your best hypotheses are but a test away from rejection or confirmation.

But if you're not an Easy Language developer on top of everything else, don't worry. Your basic strategic ideas can still be rapidly assembled from higher-level components, and your hypotheses are also just a test away from rejection or confirmation.

Even if you are a skilled and fluent Easy Language developer, you might still save tremendous amounts of time by discovering the fundamental strategic components that seem to strike a chord with your market, and then develop your strategy accordingly. Or you can use strategy component code as "boilerplate" to put together a unified strategy that you can enhance with the GGO Smart Editor since the GGO must use a single Easy Language document to prepare a GA-enabled strategy. (We'll do this shortly).

Some markets don't seem to do very well with anything. In cases like that, you have a couple of choices. You can leave that market alone unless you're prepared to do some really serious (and typically expensive) custom development. Or you can use a product like the Grail Computer-Assisted Strategy Builder (CASB) that's capable of testing multiple combinations of multiple strategy components or even the neural hybrids of multiple strategy components, even test multiple data streams or multiple timeframes, against your market.

Remember that discovering that a particular market and timeframe aren't very "system-susceptible" is frustrating and cheap. Attempting to trade a market with a curve-fit strategy or a strategy that doesn't work very well is frustrating...and very expensive.

## **Assembling Your Strategy**

Since you now have a good (or at least better than before) understanding of how Easy Language works, you will find it a relatively straightforward task to combine the strategies you stacked up in the Strategy Builder into a one-piece Easy Language Document. By experimenting with different strategy components, your author discovered that EURUSD is basically a "Keltner market." In other words, by applying standard, generic Keltner Channel long and short entries (and don't forget real-world commission and slippage properties!), a choppy but fairly profitable and consistent equity curve resulted.

Further experimentation with higher level strategy components demonstrated that both Percent Trailing and ATR long and short profit stops (but only after a "rough-cut" optimization) enhanced the equity curve. Now let's edit these strategy components into one single Easy Language text document. Start by creating a new Easy Language document in the editor and naming it GGO\_EURUSD60S (the "S" at the end stands for

source, in other words what we'll feed into the GGO Smart Editor to create our target GA-enabled code, which we'll call GGO\_EURUSD60T).

Now copy and paste the Keltner Channel long and short entry code into your new document and edit out all of the comment text just to clean it up a bit. It should look like this:

```
[IntrabarOrderGeneration = false]
inputs:
    Length( 20 ),
    NumATRs( 1.5 ) ;

variables:
    Avg( 0 ),
    UpperBand( 0 ),
    Setup( false ),
    CrossingHigh( 0 ) ;

Avg = AverageFC( Close, Length ) ;
UpperBand = Avg + NumATRs * AvgTrueRange( Length ) ;

if CurrentBar > 1 and Close crosses over UpperBand then
    begin
        SetUp = true ;
        CrossingHigh = High ;
    end
else if Setup and ( Close < Avg or High >= CrossingHigh + 1 point )
then
    Setup = false ;

if Setup then
    Buy ( "KltChLE" ) next bar at CrossingHigh + 1 point stop ;

[IntrabarOrderGeneration = false]
inputs:
    Length( 20 ),
    NumATRs( 1.5 ) ;

variables:
    Avg( 0 ),
    LowerBand( 0 ),
    Setup( false ),
    CrossingLow( 0 ) ;

Avg = AverageFC( Close, Length ) ;
LowerBand = Avg - NumATRs * AvgTrueRange( Length ) ;

if CurrentBar > 1 and Close crosses under LowerBand then
    begin
        SetUp = true ;
```

```

        CrossingLow = Low ;
    end
else if Setup and ( Close > Avg or Low <= CrossingLow - 1 point ) then
    Setup = false ;

if Setup then
    Sell Short ( "KltChSE" ) next bar at CrossingLow - 1 point stop ;

```

Next, add an “L” prefix to each user-defined variable for the long part and an “S” prefix for each user-defined variable in the short part. Your code should now look like this:

```

[IntrabarOrderGeneration = false]
inputs:
    LLength( 20 ),
    LNumATRs( 1.5 ) ;

variables:
    LAvg( 0 ),
    UpperBand( 0 ),
    LSetup( false ),
    CrossingHigh( 0 ) ;

LAvg = AverageFC( Close, LLength ) ;
UpperBand = LAvg + LNumATRs * AvgTrueRange( LLength ) ;

if CurrentBar > 1 and Close crosses over UpperBand then

    begin
        LSetUp = true ;
        CrossingHigh = High ;
    end
else if LSetup and ( Close < LAvg or High >= CrossingHigh + 1 point )
then
    LSetup = false ;

if LSetup then
    Buy ( "KltChLE" ) next bar at CrossingHigh + 1 point stop ;

[IntrabarOrderGeneration = false]
inputs:
    SLength( 20 ),
    SNumATRs( 1.5 ) ;

variables:
    SAvg( 0 ),
    LowerBand( 0 ),
    SSetup( false ),
    CrossingLow( 0 ) ;

SAvg = AverageFC( Close, SLength ) ;
LowerBand = SAvg - SNumATRs * AvgTrueRange( SLength ) ;

if CurrentBar > 1 and Close crosses under LowerBand then

```

```

begin
  SSetup = true ;
  CrossingLow = Low ;
end
else if SSetup and ( Close > SAvg or Low <= CrossingLow - 1 point )
then
  SSetup = false ;

if SSetup then
  Sell Short ( "KltChSE" ) next bar at CrossingLow - 1 point stop ;

```

Verify your strategy, apply it to your chart and check its operation. It should behave identically to Keltner Channel LE and SE stacked in the TS Strategy Builder separately except that all the inputs are made to one strategy. Notice the relative unimportance of the order of the lines of code in reference to the event-driven properties of Easy Language we discussed previously.

Now, is this any way to structure code, event-driven or not? No. From an internal documentation standpoint, especially as we add other strategy components to it, this code is poorly structured. So, let's eliminate redundancies, consolidate the inputs and variable declarations and document and place the Keltner LE and SE logic at the end. Remember, we're doing this for us, not for the computer.

Now our code looks like this:

```

[IntrabarOrderGeneration = false]

inputs:
  LLength( 20 ),
  LNumATRs( 1.5 ),
  SLength( 20 ),
  SNumATRs( 1.5 ) ;

variables:
  LAvg( 0 ),
  UpperBand( 0 ),
  LSetup( false ),
  CrossingHigh( 0 ),
  SAvg( 0 ),
  LowerBand( 0 ),
  SSetup( false ),
  CrossingLow( 0 ) ;

{Long and Short Entry Setup}
LAvg = AverageFC( Close, LLength ) ;
UpperBand = LAvg + LNumATRs * AvgTrueRange( LLength ) ;

SAvg = AverageFC( Close, SLength ) ;
LowerBand = SAvg - SNumATRs * AvgTrueRange( SLength ) ;

{Long Entry Logic}

```

```

if CurrentBar > 1 and Close crosses over UpperBand then
    begin
        LSetup = true ;
        CrossingHigh = High ;
    end
else if LSetup and ( Close < LAvg or High >= CrossingHigh + 1 point )
then
    LSetup = false ;

if LSetup then
    Buy ( "KltChLE" ) next bar at CrossingHigh + 1 point stop ;

{Short Entry Logic}

if CurrentBar > 1 and Close crosses under LowerBand then

    begin
        SSetup = true ;
        CrossingLow = Low ;
    end
else if SSetup and ( Close > SAvg or Low <= CrossingLow - 1 point )
then
    SSetup = false ;

if SSetup then
    Sell Short ( "KltChSE" ) next bar at CrossingLow - 1 point stop ;

{***End of Entries***}

```

This code looks quite a bit different from our previous version, but its logic is absolutely unchanged. It is however, considerably easier to maintain and enhance this Easy Language Document than the previous one. We have established a structure that will make it far easier for us to add new strategy components.

As long as we add inputs to the input section, variables to the variables declaration section, setup logic (if any) to the setup logic section and entry or exit logic at the end, it will be relatively easy to build up our single-document strategy so that its logic is identical to what we put into the strategy builder. Just remember to add “L” or “S” prefixes if long or short variables need to be separately named.

Usually, I like to eat the steer one bite at a time and add, edit, verify and test as I go along. So I add and integrate each strategy component one at a time. That way I can catch any syntax or logic errors I make right away and fix them.

And that’s our next step. We can now add ATR Trailing long and short exits and our Percent Trailing long and short exits the same way. When we’re done, our new, customized strategy will look like this:

```
[IntrabarOrderGeneration = false]
```

```

inputs:
    LLength( 20 ),
    KNumATRs( 1.5 ),
    SLength( 20 ),
    KNumATRs( 1.5 ),
    LATRLength( 10 ),
    LNumATRs( 3 ),
    SATRLength( 10 ),
    SNumATRs( 3 ),
    LFloorAmt( 1 ),
    LTrailingPct( 20 ),
    SFloorAmt( 1 ),
    STrailingPct( 20 ),
    PositionBasis( false ) ;

variables:
    LAvg( 0 ),
    UpperBand( 0 ),
    LSetup( false ),
    CrossingHigh( 0 ),
    SAvg( 0 ),
    LowerBand( 0 ),
    SSetup( false ),
    CrossingLow( 0 ),
    LATRCalc( 0 ),
    PosHigh( 0 ),
    SATRCalc( 0 ),
    PosLow( 0 ),
    MP( 0 );

{Long and Short Entry Setup}
LAVg = AverageFC( Close, LLength ) ;
UpperBand = LAvg + KNumATRs * AvgTrueRange( LLength ) ;

SAvg = AverageFC( Close, SLength ) ;
LowerBand = SAvg - KNumATRs * AvgTrueRange( SLength ) ;

{ATR Trailing Long and Short Exits Setup}

LATRCalc = AvgTrueRange( LATRLength ) * LNumATRs ;
SATRCalc = AvgTrueRange( SATRLength ) * SNumATRs ;

MP = MarketPosition ;

{Percent Trailing Long and Short Exits Setup}

if PositionBasis then
    SetStopPosition
else
    SetStopShare ;

{Long Entry Logic}

if CurrentBar > 1 and Close crosses over UpperBand then
    begin
        LSetup = true ;

```

```

        CrossingHigh = High ;
        end
else if LSetup and ( Close < LAvg or High >= CrossingHigh + 1 point )
then
    LSetup = false ;

if LSetup then
    Buy ( "KltChLE" ) next bar at CrossingHigh + 1 point stop ;

{Short Entry Logic}

if CurrentBar > 1 and Close crosses under LowerBand then

    begin
        SSetup = true ;
        CrossingLow = Low ;
    end
else if SSetup and ( Close > SAvg or Low <= CrossingLow - 1 point )
then
    SSetup = false ;

if SSetup then
    Sell Short ( "KltChSE" ) next bar at CrossingLow - 1 point stop ;

{***End of Entries***}

{ATR Long Exit Logic}

if MP = 1 then
    begin
        if MP[1] <> 1 or High > PosHigh then
            PosHigh = High ;
        Sell ( "AtrLX" ) next bar at PosHigh - LATRCalc stop ;
    end
else
    Sell ( "AtrLX-eb" ) next bar at High - LATRCalc stop ;

{ATR Short Exit Logic}

if MP = -1 then
    begin
        if MP[1] <> -1 or Low < PosLow then
            PosLow = Low ;
        Buy To Cover ( "AtrSX" ) next bar at PosLow + SATRCalc stop ;
    end
else
    Buy To Cover ( "AtrSX-eb" ) next bar at Low + SATRCalc stop ;

{Percent Trailing Long Exit Logic}

if MarketPosition = 1 then
    SetPercentTrailing( LFloorAmt, LTrailingPct ) ;

{Percent Trailing Short Exit Logic}

if MarketPosition = -1 then
    SetPercentTrailing( SFloorAmt, STrailingPct ) ;

```

We've kept it structured and eliminated redundancies. Insert the new EURUSD60 strategy into your EURUSD 60 minute chart. Double check to make sure that the Properties of this strategy, especially the Commission and Slippage numbers are correct and within real-world limits.

Realistic settings for the Commission and Slippage are just as much a part of the "personality" of your market as its price action. (See Pardo, Chapter 4, subchapter Practical Considerations, Transaction Costs). Furthermore, as an experienced system trader, I have observed that adding real-world "commish" and slippage numbers to a strategy can actually make it less "nervous" and more robust.

Everything in the Strategy Performance Report should still come out the same as it did when the individual components were stacked up in the Strategy Builder. Now with the strategy open in the Easy Language editor, do an Edit, Select All and Copy your strategy into the paste buffer (the clipboard).

If your system makes money by placing very many trades with razor thin average profits, you've created a scalping system. Scalping systems can have wonderful looking equity curves made up of apparently solid lines of green dots. And by their very nature, scalping systems are extremely brittle, potentially non-robust and very sensitive to changing market or trading rule (or commission or slippage) conditions.

I will occasionally actually exaggerate slightly the commission and especially the slippage numbers when I'm in the discovery phase of a system build in order to purposely avoid ending up with a scalping system. Some traders actually like and are good at scalping either in real-time discretionary trading or with their systems. If you've discovered a scalping system and if your commission and slippage numbers are really honest and realistic and if you like that kind of system, then enjoy it and good luck!

Now invoke the GGO Smart Editor and make sure that you're in the Prepare Strategy view. Use the Mode Selector near the top next to the speed buttons. Click on the Retrieve TS Strategy from Clipboard applet button to pull in the strategy to the work area of GGO. You're now ready to prepare your strategy for genetic optimization.

## **GGO Strategy Preparation Procedure**

Now you will start to make choices that will affect how the GGO handles the optimization of your system. We will take a "preferred route" to our destination, if you will. While there are other ways to get there, the one we will choose has several advantages. It should be regarded as a default best-practices procedure. Note that I use that phrase "best-practices" with some trepidation, because what we're doing is as much art as it is science.

Still, the concept behind the proven path we will take is that we want to use the most GA-leveraged, computationally economical methods we can select and still get results that will turn out well. Remember that we're searching for robustness, not some fantasy of curve-fit perfection that will come apart when we turn it loose on unseen live data.

Let's start with Properties settings. In your chart in Format Strategies, leave the Inside Bar Back-Testing box left unchecked. This testing feature can be useful, especially on volatile markets like currencies. An intra-bar price fluctuation could very well cause your system to exceed the maximum open drawdown limits you wish to set for real-time trading.

This testing feature (Inside Bar Back-Testing) has only been implemented in TradeStation for a little over a year now. Let's quickly consider some of the comparative advantages of using it.

### **Inside Bar Back-Testing**

Inside bar testing was implemented about a year ago by TradeStation at user request so that during certain intrabar conditions that can occur during backtesting, there will be a reduced chance of overstating profits. This is particularly true if you are implementing certain types of relatively tight profit or protective stops in your strategy, and your strategy uses relatively long intrabar intervals on a relatively volatile market.

However, in real life, time is money. Thus, if one can safely cut down optimization time significantly by not using look inside bar testing, then it certainly makes sense to not use it. This not only was the only way you could run a TradeStation backtest/optimization in the recent past, but it is quite feasible and reasonably accurate if you understand the system to be optimized and if you also understand what the implications are of not using look inside bar testing.

The short version of the story is that it is only necessary to use inside-bar testing if the system that you are testing employs entry/exit signals which could trigger some of the TradeStation's OHLC (open, high, low, close) intra-bar calculation assumptions with a certain degree of frequency during the test run.

For example, we all know that Intra-bar order generation is strongly recommended by TradeStation when a strategy uses Easy Language functions like:  
`SetProfitTarget`, `SetStopLoss`, `SetPercentTrailing`,  
`SetDollarTrailing`, `SetBreakEven`. Well, if we didn't, we do now, don't we?  
(LOL!!!)

When evaluating a trading signal for a given bar, TradeStation assumes the following:  
If the Open of the bar is higher than the Close, then TS assumes the High of the bar was made before the Low. If the Open of the bar is lower than the Close, then TS assumes the Low of the bar was made before the High.

These assumptions were a calculation expedient before the implementation of inside-bar testing and obviously, they are not always true in real life, especially if the open and close are relatively close to each other, as would be the case in a bar with sideways market action relative to opening and closing prices but with a wide range between that bar's high and low.

Thus, using these assumptions can sometimes lead to inaccurate backtesting results. However, it is very important to realize that these assumptions are only triggered under the following circumstances: If say, the opening price is higher than the close, and intra-bar price movement is such that strategy components trigger two different signals in opposite directions for the same bar, then TS applies an assumption (that the high happened before the low whether it did or not!!!) to make an arbitrary determination as to which signal was triggered "first."

Whoa, you say! What if that arbitrary determination happens to be wrong? For instance, let's say you have a system on the S&P with both a 3 point profit target and 3 point stop loss. Now, what if within a single 60 minute bar, you had a range of more than 3 points from the open in both directions? In theory, both signals could have triggered within that bar. TradeStation isn't going to do that, so the assumption is now invoked.

Now if you carefully consider the "worst-case" scenario described above, it is also clear that there are very many systems with shorter bar intervals in which this never (or almost never) happens and where the assumptions will never be triggered. (That's how TradeStation sold thousands of copies over many years, even though they only introduced look inside bar testing about a year ago).

Over the years system developers have also learned that there are many ways to protect themselves (and their systems) against the potential backtesting errors that occur when The Dreaded Assumption happens to be wrong. For example, you can use a moving average filter such that the system can only go long if the price is above say, an 8-bar moving average, or go short if it's below that MA. Under these circumstances, unless there's a really anomalous bad tick up or down, then it's virtually impossible for the system to trigger both buy and sell entries intra-bar and the dreaded "the high happened before the low whether it did or not!!!!" (or vice-versa) assumption is never triggered.

Now if you're electing to use inside-bar back testing and you choose say, to look at 1 minute intra-bar intervals, just bear in mind that you're asking your optimization to examine many more data points than if you run your optimization on say, 60 minute bars! **IF YOU ARE REALLY EXPERT AT EASY LANGUAGE AND KNOW ALL THE IMPLICATIONS OF WHETHER OR NOT TO USE INSIDE-BAR TESTING**, then you can make your own decision!

Otherwise, we can suggest the following guidelines. If your system uses "assumption-susceptible" stops (e.g., percent trailing) and it optimizes out with very tight exits, (by "tight," for example, we're talking about \$100-200 on the S&P E-mini representing just 1

or 2 points,) and if the combination of your bar interval (let's say 20 minutes) AND current market volatility (which can vary greatly) are such that intra-bar price range is frequently exceeding 4 points, then it would be prudent to use look inside bar testing. Your test run will take longer, but it's likely to paint a more accurate picture of what's going to unfold for your system in real time. And real life trading unfolds in real time!!!

However, if you are using additional buy/sell filters and/or your exit optimization ranges come out large enough so that it is unlikely that two signals in opposite directions (e.g. a stop loss v. profit target) would ever trigger within the same bar thus invoking The Dreaded Assumption, then you can save yourself a lot of time by NOT selecting look inside bar testing.

The more you know about the characteristics of your system, (particularly if it's using a strategy that you hand-coded or hand-assembled), the better position you'll be in to make an informed decision about whether you can run an optimization faster by skipping the safeguards provided by intra-bar backtesting. If you are in serious doubt as to whether or not to use inside-bar backtesting, our recommendation and "default setting" if you will, is ON.

## **Proceeding With Your Genetic Optimization**

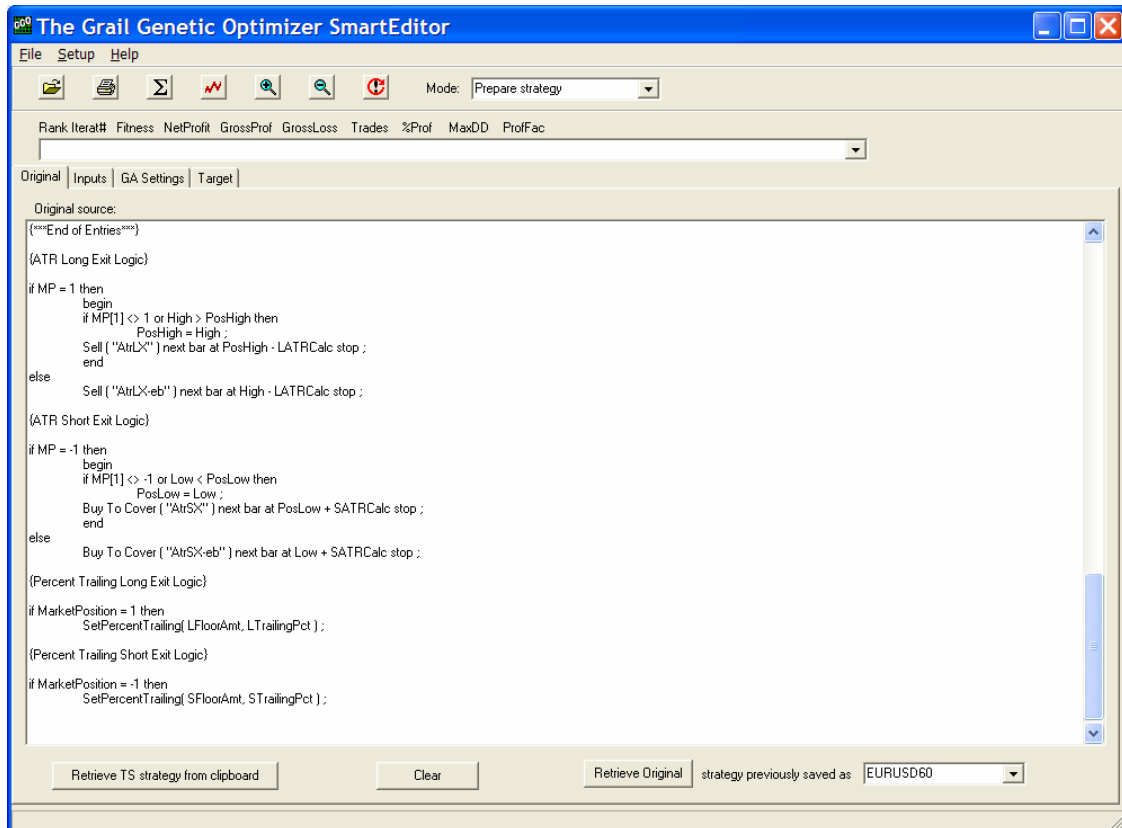
Now be careful! For the purposes of this modeling run, I left Inside Bar Back-Testing OFF. The underlying assumptions and test results obtained with Inside Bar Back-Testing ON may or may not be something you would want to trade. Furthermore, this example is intended primarily as an illustration. We at The Grail are NOT in the business of telling you what symbol to trade in what market in what time frame with what basic strategy components!!!!

Set the Maximum number of bars study will reference to 200 (this is a GGO standard setting). If you leave this setting at the default of 50, your optimization will crash.

In Position Limits, click the checkbox and allow up to 2 entry orders in the same direction, and select the "when the order is generated by a different order" radio button (this is another GGO standard setting). Since we have only one entry component in this strategy, this setting won't make any difference for this strategy. But if you had multiple entry types, it would make a difference, and for the CASB (Computer-Assisted Strategy Builder), it will make a difference, so it doesn't hurt to get accustomed to setting it for multiple entries.

Everything else in Properties (e.g., initial capital, interest rate) should be set as appropriate for any other optimization so that when you run TradeStation performance reports, they reflect realistic financial results. Click OK to exit the Properties window for your strategy. Make sure that all the automation related check boxes in the strategy builder are unchecked.

Open the GGO and look at the Original tab. The page with the code you brought in from the clipboard should look like this:



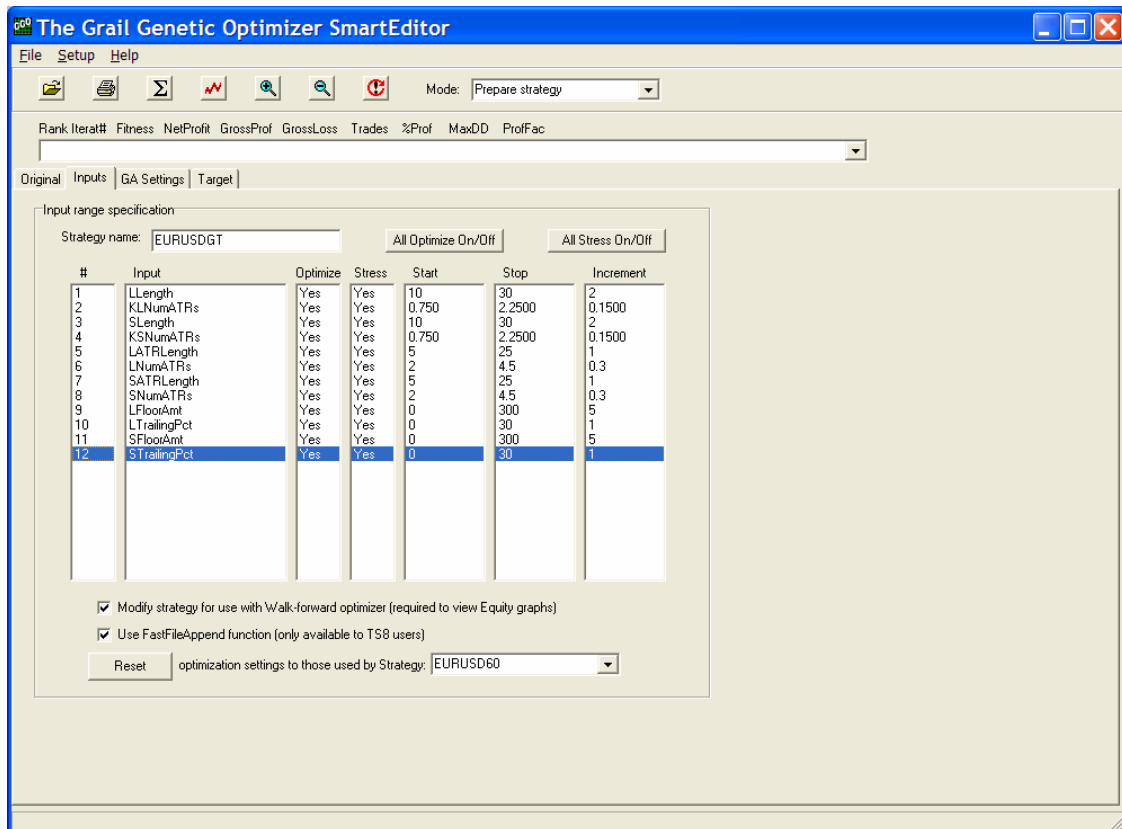
Now go to the Inputs tab. In the Strategy name field, you'll notice that we put our strategy name with the suffix "GT" for genetic target. You will also notice that we changed the Start, Stop and Increment values of some of our parameters in order to be more realistic for this market. Remember that for contracts, floor amounts typically must be able to optimize on a larger range than for individual equities (stocks).

The numbers I have put in this example are realistic guidelines. I always like to see final values within and not at the extreme limits of the optimization range. This tells me that the optimization found good values within the ranges I specified.

There are exceptions to this rule. For example, you'll notice that the length parameters for Keltner Channel Long and Short entries start at 10. This happens to be a TradeStation default, but I have also observed that attempting to calculate Keltner entries on fewer than 10 bars can lead to unstable or un-robust results. Hence, the values you see here. If the best result I can achieve is with a Keltner length of 10, I leave it alone and consider that to be the best realistic result I can achieve for that market.

Note that I have given the floor amounts on the Percent Trailing Stops more headroom since we're trading FOREX contracts and not the price of individual shares of stock.

Again, note these numbers and use them as guidelines. If you end up with disappointing results and one or more of your parameters are at the start or stop limits, consider expanding those limits and running the optimization again.



You will also note that we are checking the “Modify strategy for use with Walk-forward optimizer. This causes the generated Easy Language code to write results files that are used by the GWFO. The only time you would want to leave this box unchecked is if you bought the GGO as a stand-alone product, or if you simply have no intention of running a walk-forward optimization on your system. Not checking this box will make the optimization run faster. Note: This is NOT recommended.

If you’re using TradeStation Platform 8.x (this will apply to most users), then be sure to check the Use FastFileAppend box. This will speed up your optimization.

If you have created a similar strategy in the past and you wish to apply it to a different market, and if that strategy required the types of adjustments to the parameter ranges that we discussed above, you can choose that strategy from the list at the bottom right and click the Reset applet button. This will automatically set all the parameter ranges to those specified for the market you choose from the list. This can be very handy if you have several parameters and you made several adjustments to them.

Now having just said that, let me direct your attention to what's next!

## **Some Strategic Advice about Strategy Building**

The developer of The Grail suite was hired by a wealthy private investor to perform some proprietary tests on some of the more popular commercial trading strategies. The results were extremely interesting, and they can be summarized briefly as follows.

First, only about 10% of the commercial strategies that were tested did well enough to even warrant applying them to real-life markets and attempting to trade with them. Now, of those 10%, virtually all of them had the following characteristics:

- They were relatively simple
- They had relatively few inputs
- Those inputs had reasonable ranges (no extreme values permitted)
- They generated reasonable but not spectacular profits, profit factors, winning percentages, etc.
- They did reasonably well on unseen data
- Overall—get this—price and complexity correlated with performance...negatively

I mention this here because if you're up against the 50 input limit of the GGO and your optimization ranges are in the thousands with increments of 1 and you're glad you're able to capture the optimization parameter characteristics from the last strategy because it saves you from keying in about 9 pages of values for this one, you need to stop and ask yourself what you're doing. In the Grail user community, there's something called the Ten Percent Club.

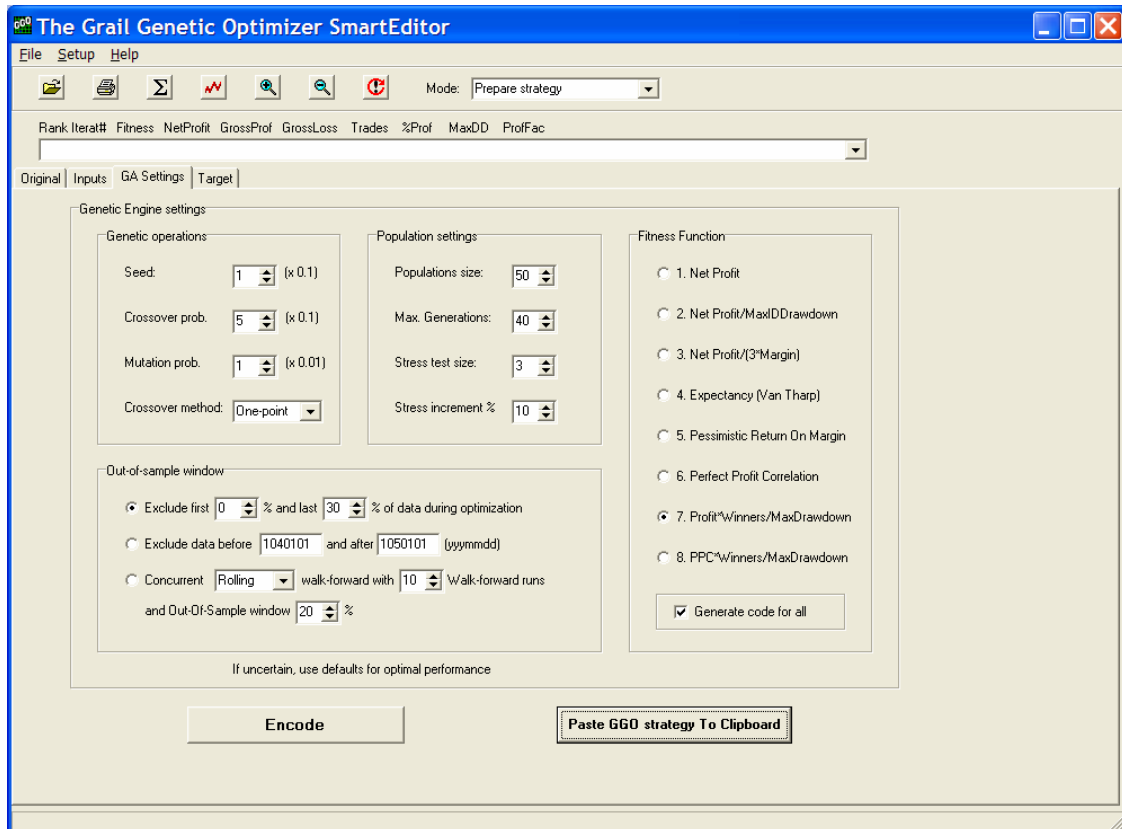
That's what you become part of when you achieve a system that passes all of The Grail performance criteria. It's called the Ten Percent Club because 90% of the systems everyone tests...don't make the cut. You can give yourself a tremendous head start at the Ten Percent Club by asking yourself if on the face of it, your potential strategy has much of a chance.

Now allow Your Author to share a little secret with you. Many of the strategies I test make the Ten Percent club, often on the first try. That's because I test dirt-simple strategies with few inputs that seem to fit the "personality" of the market I'm testing.

Very often, in fact almost always, I'm using the "freebie" strategy components that came with TradeStation. These rarely cause problems during release updates because TradeStation Development checks them as part of their pre-release testing. Otherwise, I'm using the freebie components I can download from TradeStation World.

Even if TradeStation Development has to modify them to use the new release, it's very easy to substitute the new code segments for the old ones in my strategies. Rarely, I'll code something myself for the same reason a rabbit stays ahead of a lawnmower. In other words, if it's something I need and nothing else out there does the job, I'll code it. But by using production code as an example and making every effort to make my own code "like that," I still manage to avoid most release update code compatibility problems.

Now, take a look at the following screen:



Above the Encode applet button, you'll notice some words that should get your attention: If uncertain, use defaults for optimal performance. Go ahead and do that.

You'll notice that we've selected Fitness Function 7. It's a good default to use. This fitness function serves two purposes. It acts as a goal for the genetic algorithm to seek. In other words, the GA determines whether or not a particular permutation of parameters is good or not based on the calculated value of this fitness function for this iteration.

Your optimization report will also be sorted by this fitness function. As you'll see soon, this fitness function isn't the Omnigod Criterion of how good your strategy is. It's a single-number condensation of the "goodness" of your system.

The only other thing we've changed from default values is selecting the Generate code for all box. That way you can quickly change fitness functions by making one edit in the generated Easy Language code. If you prefer to generate Easy Language target code separately for every (something I would recommend) then you can leave this box unchecked.

## **Walk-Forward Data Exclusion and Walk-Forward Method**

At the bottom left of the screen shot above, you'll notice a selection section called Out-of-sample window. I have left everything in this window at the default settings. At this stage of our testing, I want to allow my strategy to optimize on the first 70% of the data and then see how it walks forward on the remaining 30%. For the majority of markets and strategies you will test, we recommend these settings.

Now markets and their personalities can be complex. Sometimes a market will demonstrate an equity curve that has some very profitable and some not-very-profitable intervals within the time sample tested.

The majority of market/strategy combinations (systems) will yield the best test results with the default OOS (out-of-sample) and stress test settings shown. Experienced users may want to consider a Concurrent Rolling walk-forward for example, if the market shows considerable personality change across the tested time interval.

I will reserve a more extended discussion of the potential comparative advantages of these alternate optimization strategies for the GWFO tutorial since the selections you make on these options also affect how you will do subsequent walk-forward optimizations with the GWFO. Remember that a concurrent rolling walk-forward optimization done in the GGO is considerably more computer and file-intensive (and therefore, slower) than the default anchored walk-forward.

Permit me to quote directly from the GGO Help documentation: "Only select this option if you are an experienced user of GGO/GWFO and [have] a good understanding of how GGO interacts with GWFO." Most optimizations will yield the most robust walk-forward results if run with the default settings.

## **Let's Proceed!**

Now click the Encode applet button and then click the Paste GGO strategy to Clipboard applet button. The GA-enabled version of your strategy is now in the paste buffer (clipboard).

In TradeStation, create a new strategy, give it the same name as your original, except with the suffix "T" for target, and paste the generated code into the Easy Language editor. Verify the strategy (it should verify without any problems), and close it.

Now insert that strategy into your chart, delete any remaining strategies from your chart (doing things in this order will preserve the property settings from your previous strategies), and then verify that the Property settings for your strategy in TradeStation are where you want them.

Optimize the Iteration parameter with a start value of 1, and a stop value of 6000 with increments of 1. Remember, recommended iterations = MaxGen x PopSize x StressTestSize or  $50 \times 40 \times 3 = 6000$ ).

You can now start the optimization. The GGO will now iterate through 6,000 genetically sampled tests equivalent to many millions of times as many exhaustive, brute-force combinatorial iterations. (You're testing 12 simultaneous parameters!!!) It will take anywhere from about an hour to a few hours (depending on the speed of your computer) to genetically iterate through this optimization.

One nice feature about how the GGO shakes hands with the TradeStation optimization engine is that since the TS optimizer bases its completion time estimation on the physical rather than the genetic iterations (that's what it "sees" when it runs the optimization), the time estimate that it calculates is as accurate as for any normal (brute force) TS optimization.

## **A Brief Word About GA Iterations**

In terms of statistical precision or resolution, genetic algorithms are like digital photographs or pictures. If you were to specify fewer iterations than recommended in the run above, the "picture" of your combinatorial search would still be complete, but the picture elements (pixels) would be bigger. If you reduced the number of iterations sufficiently, you would discover that entire detail areas of your "picture" would be swallowed up in the increasingly large blobs of averaged color and brightness.

Television and news services do this "pixilation" intentionally to the faces of criminals or to license plates to make the resulting image unintelligible. Specifying an unrealistically small number of iterations in a GA combinatorial search can make it run very fast. The only problem is that insufficient detail is rendered to give you an accurate statistical picture of your results.

At the other extreme, adding iterations can add resolution to your statistical "picture." But beyond a certain point, you achieve rapidly diminishing returns for all your number-crunching. Furthermore, the purpose of our combinatorial searches is to find good-enough, robust parameters that have a good chance of walking forward.

"Perfect" optimizations are curve-fits that are almost by definition, brittle and unpredictable. In fact, most really competent system builders *automatically throw out*

results (profit factors, t-tests, etc.) that are “too good to be true” as statistical outliers, which they in fact, almost always are.

When you’re facing a testing run that will take all night, and if you’re armed with some knowledge of how genetic algorithms work, you might find it tempting to save a few hours of total testing time by reducing the number of iterations in your genetic combinatorial searches. But this is not a hint. It’s a warning not to short-circuit the genetic search process by specifying too few iterations to render a clear picture of your system.

Similarly, you’re not going to get “better” results by specifying overkill numbers of iterations. In fact, you’ll risk arriving at less-robust solutions because you’re exposing your system to possible curve-fitting. Baking that cake for five hours instead of 45 minutes isn’t going to make it “better.” The recommendations and methods for calculating GA search iterations in The Grail are based both on generally accepted theory and practice in genetic algorithms and on trading strategy testing application-specific research and experience. So use the recommended numbers.

By way of review, the recommended default is to set stress test size=3, resulting in 6000 iterations in total. Especially if you are going to use the GWFO to do a walk-forward analysis afterwards (recommended), the GA should perform no fewer than 6000 iterations under any circumstances. (The figure 6000 is arrived at by multiplying the default population size of 50 by the default max. generations of 40 and the default stress test size of 3. Thus  $50 \times 40 \times 3 = 6000$ ).

The default population size and number of generations were carefully chosen to provide optimal GA performance under most conditions. Therefore I recommend that you should only adjust stress test size and preferably use a stress test size of between 3-5 resulting in 6000-10000 iterations.

## **GAs, Heuristics and Convergence**

The genetic algorithms implemented in the GGO can be considered heuristic to the degree that they “learn” which parameter ranges result in a good computed goal (your chosen fitness function) and which do not. Towards the end of a typical GGO optimization, a slightly higher percentage of good fitness function results will be returned by the optimizer than at the beginning.

At the same time, we never turn off the pseudorandom functions that continue to explore new canonical permutations throughout the entire optimization run. Therefore, if you look for the GGO to converge on “the solution” the same way that an iterative solution to a CFD (computational fluid dynamics) calculation would converge, you might, no, you will get frustrated!

In a typical CFD (or similar engineering) calculation, quantities called “the residuals” of the equations which are the change in the equations over an iteration, are tracked. These

are usually scaled or normalized the way we normalize various coefficients to the hyperbolic tangent so that we can compare dissimilarly scaled parameters for our neural hybridizer.

In any case, in such an iterative solution algorithm, one usually looks for these residuals to reach a certain (low) level and then level-off as an indication of iterative convergence. For a time-marching, steady-state strategy, this involves examining whether the residual has been reduced by a certain number (usually 3-4) of orders of magnitude.

In plainer terms and for the non-mathematicians in the audience, as the residual values in a typical iterative calculation decrease to a certain threshold, the calculation is essentially telling you “OK, we’re home now!!!” This is NOT how the Grail GAs are implemented. The Grail GAs continue to explore new pseudo-random parameter values right up to the end of the run, resulting in the occasional ugly among the good, the bad and the ugly equity curves you’ll see even near the end of the specified number of iterations.

### ***Making Sure Your Train is On the Right Track:***

Now that we have this optimization run going, we can try out an interesting feature of the GGO Smart Editor, check and see how our file builds are progressing and learn a few things about how the GGO and the GWFO work. You will have plenty of time to read, follow along, and perform the checks you will be instructed to do next!

Every time the GGO goes through an iteration that meets certain criteria, it adds a record to one report designed to be read by the GGO Smart Editor, and it creates a results detail file designed to be used by both the GGO and the GWFO.

First, let’s check on the GGO report. Open an instance of Windows Explorer, and start drilling down through the directory hierarchy (e.g., My Computer -> MYHARDDISK(C:) -> Program Files -> TheGrailGGO->Reports) until you reach the Reports subdirectory under TheGrailGGO directory.

The GGO Optimization report file should have attributes that indicate that it was built at the time you started your optimization run if you right click on it to view its properties. The size of this file should be growing slowly as records that meet the MinBarsPerTrade, MaxBarsPerTrade, MinPercWinners, MaxPercWinners and MaxDDAllowed criteria set in the GGO are added.

If several hundred iterations go by and the size of this file remains static, you might want to save a few hours of computer time and Abort your optimization. If no records are ever added to this file during the building optimization run, you’ll get null results for the strategy build when you open the GGO to examine the results report. As you grow more experienced with the GGO, you can adjust those criteria to save a step and simply not

write records to the optimization report file that represent completely unsatisfactory strategy configurations.

Next, navigate over to TheGrailWFO/Data directory and check to make sure that the GGO logic has created a subdirectory under Data. The name of the strategy you're testing will be concatenated into the folder name. The GGO should be adding a file containing a report for each genetic iteration of the search to this directory. Again, if it isn't, save yourself some time and abort the optimization now until you find out what's wrong. These files contain the data that allows the GGO to reconstruct the equity curve (among other things) for each iteration of the run.

A newly enhanced feature of the GGO allows you to monitor the optimization process and it also serves as a nice cross-check that the GWFO file-building process is working. To sample the activity of the GGO in real time, invoke the GGO Smart Editor, select the Evaluate Strategy view, choose the Equity Graph tab and then select Real-time update in the Type window.

The default refresh rate of 5 seconds will keep this display current without unduly slowing down the optimization. You should see a succession of equity curves, good, bad and ugly, in the GGO Equity Graph window. Once again, if you observe this display as you eat lunch and every single equity curve looks hopeless or worse, you might want to consider an Abort until you figure out what is going wrong.

Typically, you should see several fairly upswept, reasonably linear equity curves over a 5 minute observation period. This tells you that from time to time, there's a genetically sampled permutation of parameters for your strategy that gives you good in-sample and out-of-sample (OOS) results.

Once you are confident that the GGO Smart Editor report is being built correctly and that the GWFO data directory is being properly appended to and that frequently, the genetically sampled parameters combinations are creating good looking equity curves, you can confidently let your optimization proceed.

## ***Interpreting the GGO Optimization Report***

When the optimization completes, you will have two sets of data to help you choose which strategy parameters you will want to test further. (Bear in mind that while you can in theory use the GGO as a stand-alone product, we highly recommend using your GGO optimization as the first step in walk-forward testing with the GWFO). The first is the file containing the GGO records of the genetic iterations that met your selection criteria (this is the first file we examined for proper updating).

The second data set is a directory containing the 4,000 or so files representing the optimization results that get written from the GGO during the optimization. The GGO uses these files to reconstruct the equity curve for every genetically sampled instance of

parameter permutations. Note that we're not yet ready to perform any walk-forward analysis because so far, we haven't completed the optimization phase.

As you saw earlier, one of the latest Grail enhancements to the GGO includes the ability to monitor equity curves in real time as we did above. One highly desirable spinoff of this enhancement is the ability to rapidly page up or down through strategy performance reports. Thus, by pressing the down cursor key repeatedly, you can rapidly review and examine successive equity curves.

The records in the GGO optimization report are sorted by the fitness type you chose when you were setting up the Easy Language strategy for conversion by the GGO Smart Editor. In this tutorial, we chose fitness type 7, which is as good a fitness type as any. Bear in mind that what this fitness type is attempting to do is to condense the "goodness" of the equity curve into a single, sort-able numerical coefficient. In this case, we're operationally defining goodness or fitness to be the product of the net profit and the winning percentage divided by the maximum drawdown.

The Fitness Function is exactly that—a computed value that's used to estimate the overall fitness of a strategy as applied to a market. On the one hand, it's an important calculated value because it tells the goal-seeking genetic optimizer engine whether it's getting closer to a desired goal, namely overall strategy fitness.

On the other hand, the Fitness Function is not Omnigod. In other words, if record 82 as sorted by your chosen fitness function, be it 7 or 6 or whatever, happens to have a drop-dead beautiful, consistent across varying market conditions, linear, smooth, upswept equity curve, then configuration 82 may very well be your first choice for further testing even if 81 records above it have nominally higher fitness function values.

So use the fitness function for exactly what it is: a useful if somewhat simplistic, single-numerical-value guideline for helping to determine how appropriately a particular strategy fits a market. Use it like an informed educator would use an SAT score. Or, use it the way the Genetic Optimizer uses it: as a derived, single-number goal guiding you towards "what's hot."

Normally, your best equity curves should be at or near the top of your fitness function sort. (In the example above, I would consider record 82 to be "near the top" of say, 4,000+ records in the optimization report).

Note that the number of records in the report will usually be between 70-80% of the total number of iterations performed by GGO. Thus if you have used the default settings, optimizing Iterations between 1...6000 will typically yield between 4000-5000 records in the GGO Optimization report. The reason why there are fewer records in the report than iterations performed can be explained by duplicates encountered during the GA optimization. Note that GGO is programmed to immediately identify duplicates and it will never attempt to evaluate the same parameter combination twice.

## ***GGO Optimization Report Interpretation: Selecting Your Parameter Sets***

When your GGO optimization is finished, you can invoke the GGO Smart Editor and open the optimization report that was created by your GGO run. To do this, select the Evaluate Strategy view in GGO, go to the File menu and in Open Optimization Report, select the file that was created for your optimization run to open.

The records in this file will rapidly populate the data matrix of the GGO. Records with the highest fitness function scores will sort to the top.

Generally, you will find the best equity curves at or near the top of this fitness function sort, but not always. Choosing a good equity curve is as much an art as it is a science. (Remember, we mentioned earlier that genetic optimization is a computer-assisted process). But there are some things you can look for.

A general direction of up is good. As you look across 4 years of performance of the system, are the drawdowns moderate, or are they sharp and catastrophic? How is the recent performance? A curve that starts out a bit flat but that makes steady upward progress for the next three years is probably good.

Pay special attention to the curve to the right of the yellow line that represents the start of previously unseen or OOS (out-of-sample) data. One of the single most impressive things you can observe among these thousands of equity curves is an equity curve whose characteristics remain essentially unchanged to the right of the yellow line.

In fact, a curve that bravely chops its way up right through the yellow line and does the exact same thing to the end of the graph is the most impressive equity curve there is. It's better than a curve that establishes a nearly solid line of new highs right up to the yellow line, then starts to chop. Continuity on both sides of the yellow line tells you that your system is doing as or nearly as well on unseen data as it does on in-sample data.

A curve that climbs like the Space Shuttle the first two years, tapers off and then goes flat to negative the last couple of quarters may disappoint you in the future no matter how good its overall fitness numbers look. Long before the appearance of The Grail, many a successful system trader (your author included) achieved very high audited ROAs by developing a good eye for steady, robust equity curves.

Now in case you're wondering if you could do all this without The Grail, remember that your author also wasted tremendous amounts of time manually slogging through procedures that The Grail automates. If you're not an experienced systems trader, try to imagine the work it took to do honest walk-forward sampling manually. And those optimizations didn't benefit from genetic algorithms.

Now think of the following concept in the context of sample size. If your equity curve looks good across a multi-year time frame, it means that the strategy is yielding good

returns across a variety of market conditions. Since certain outside factors that influence markets and even aspects of market personality inevitably change over a long enough time span, a steadily climbing equity curve across a few years means that the strategy is probably tapping into one or more underlying, robust “personality” traits of the market.

This also means that when the Grail Walk Forward Optimizer segments and masks off selected portions of the data, that the OOS (out of sample) blind walk-forward tests Grail performs will probably (but not always—that’s why you do walk-forward testing!) come out OK. Never underestimate the value of a robust looking, reasonably linear equity curve, particularly if it represents a statistically sound sample size.

A curve that chops and claws its way up consistently even in the face of frequent modest drawdowns has more promise than one that looks straight as a ruler part of the time and that goes flat or negative for extended periods, even if its net result, including a single-number fitness function, looks “better.”

In any case, your job at this point is to page through the top few hundred or so results sorted by fitness function, and to start looking for really promising equity curves. You might want to select half a dozen of the best looking ones for further testing. Write them down by record number for later reference. Most of the time, your best equity curve should be at or near the top of the fitness function sort, but this will not always be the case.

## **Which Path to Nirvana?**

In fact, The Grail has several alternatives for arriving at very similar end results. For this tutorial however, we’ll follow the recommended default procedure and perform stress and out-of-sample testing during this optimization run. I will comment here that the path we’re following leverages to the maximum the advantages of using genetic algorithms in the first place.

This procedure is also yielding the maximum amount of analytical information with a minimum amount of computer time. My philosophy in following this recommended path is that in the absence of a compelling reason to do otherwise, we might as well leverage all the GA technology we have at our disposal.

I will comment here that on a market like a currency pair that tends to have a relatively stable “personality” over time, the method we’re following here is especially appropriate to the extent that it’s very likely to discover a robust, stable system. On markets like E-mini futures that tend to be less stable in terms of strategy effectiveness from one week to the next, alternate testing approaches using rolling or concurrent genetic walk-forward optimizations may work better, and that’s why The Grail system offers those options.

## **Anchored v. Rolling Walk-Forward**

If a market appears to be changing personality over the time sample tested, then a rolling walk-forward may be more appropriate and yield results that might walk forward better in live trading than results obtained from an anchored walk-forward. The less stable the personality of your target market, the more it may make sense to test the rolling walk forward against the results obtained with the default anchored walk-forward.

Nevertheless, we regard the anchored walk-forward to be the benchmark test. Only performance reports can tell you whether the rolling or concurrent genetic alternate methods are better for a particular market.

## **Choosing OOS (Out-of-Sample) and Stress Parameters**

Under normal circumstances, use 30% unseen data for your GGO optimizations. A changing personality over time might suggest using a smaller exclusion percentage while an extremely stable market might give you the opportunity to extrapolate results over a larger percentage of your data.

Similarly, a market that creates smooth, steady equity curves can probably be safely tested with the stress testing parameter set to to the low range (2 or 3) while a market that exhibits a great deal of equity curve volatility should probably be tested with a higher stress test coefficient. While this risks rejecting the market as a system candidate, it also protects you from a potentially unstable market by rejecting it before you trade it.

## ***Deploying Your System***

At this point, you can take the parameters you discovered, plug them into the strategy you created and cross-check that the strategy performance and equity curves plotted by the RINA reports in TradeStation match the results you got with the Grail. At this point, you have also subjected your system to far more rigorous and comprehensive testing than the vast majority of systems ever get.

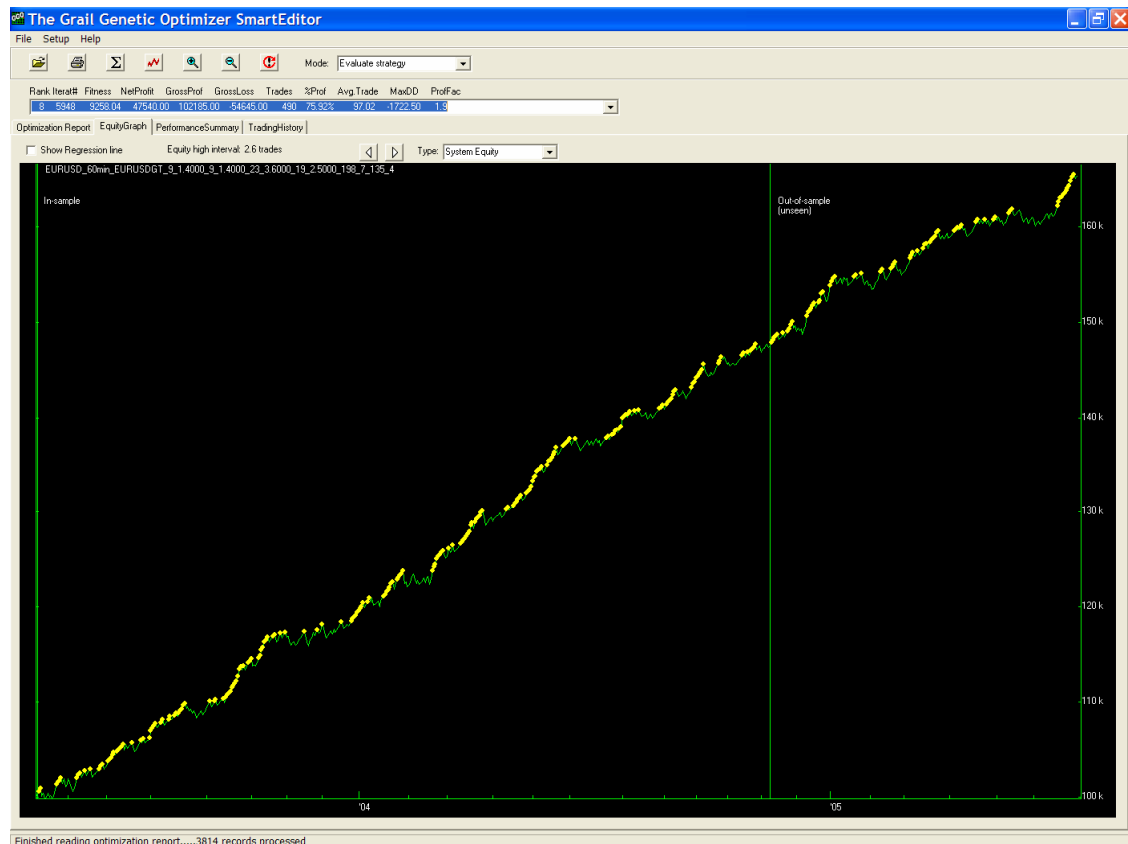
## **How to Set the Parameters in Your Easy Language Strategy**

The Evaluate Strategy mode of the GGO displays all the parameters of all the iterations of the strategy you tested. These are the actual parameters that created the test results for that iteration, that equity graph and that performance summary. Simply plug those parameters into your original Easy Language Strategy and apply them to your chart. The results you get on the TradeStation Strategy Performance Report should cross-check perfectly with the results posted in the GGO.

At your option and if you are particularly confident that your nice, linear equity curve has “success” written all over it, you can deploy your system in live trading, although we at

The Grail HIGHLY RECOMMEND performing additional walk-forward optimization before you deploy your strategy.

For the record, here's the equity curve of the system that I created by following to the letter the contents of this tutorial. Or more accurately, maybe I should say that this is the tutorial that resulted from documenting to the letter the procedures I followed to generate the following equity curve!



You'll notice that I used Record 8. I liked its equity curve both in and out of sample.

Bear in mind that the Grail Suite is designed primarily to be used as a series of integrated applications designed to progressively test and refine a trading system. Therefore we will include in this GGO tutorial brief versions of the GWFO and Cluster Analysis tutorials since it is intended that you take your original GGO results through two additional testing stages before you regard the resulting system as "portfolio-ready."

## ***Walk-Forward Optimization***

Now invoke the GWFO. Go to the File, Open menu and double click the directory containing the results from your fixed-logic stress tested and excluded-data GGO

optimization run. Select the first file in the list, then click the “Select all files for this system” button. All of the files created by the GGO optimization run should be selected.

Then click the “Open” button. You should see a message at the bottom of the GWFO window saying that “4,042 (or whatever the actual number is) files selected for Walk-forward.” The results of your selected GGO strategy optimization are now available to the GWFO.

Perform an Anchored WFO using the default values of 10 runs and the Out of Sample % set to 20. This walk-forward analysis will tell you whether your newly optimized strategy will probably robustly walk forward on unseen data. It’s a very satisfying feeling to see a strategy pass all the criteria in the GWFO. If one out of ten attempts to test a strategy passes all the GWFO robustness and reliability criteria, you’re doing well.

If your discovery/optimization process was successful in a negative sense to the extent that you discovered that your system is unstable or not robust, you could spend a tiny fraction of the trading capital you DIDN’T lose by attempting to trade a statistical deathtrap, and buy your favorite beverage so that you can toast your success at ruling out a losing system! Remember, in the same sense that professional traders consider cash as just as valid a position as long or short, to not attempt to trade a statistically indefensible system is a valid, rational action on the part of a responsible systems trader.

### ***Cluster Analysis: Determining the Optimal Walk-Forward Optimization***

When you read the Grail Thread in the TradeStation Forum

([https://www.TradeStation.com/Discussions/Topic.aspx?Topic\\_ID=31924](https://www.TradeStation.com/Discussions/Topic.aspx?Topic_ID=31924))

you will occasionally hear reference to a system passing the “Grail Triathlon.” This is insider’s shorthand for a system going through a Genetic Optimization, a Genetic Walk-Forward Optimization and finally a Cluster Analysis.

Cluster Analysis is a time-consuming but very thorough way to obtain further information about how often you should re-optimize your system as well as giving you a final level of refinement in parameter recommendations for trading your new system. It suggests optimal out-of-sample exclusion percentages and the number of runs you should use to walk-forward optimize your system in the future.

This is accomplished by performing 30 (did I mention this would be time-consuming?) complete GWFO optimizations each with a different combination of number of runs and OOS exclusion percentages. The end result is a series of 5 x 6 matrices with each cell representing of all the results you would obtain by doing a single GWFO run with a particular number of runs and OOS exclusion percentage. (Remember, 10 runs and a 20% OOS exclusion is the Grail default).

But it’s very likely that you’ll obtain better walk-forward optimization results and more optimal final strategy parameters by using more or fewer runs and/or a larger or smaller

exclusion percentage. And the only way to really find this out is to perform multiple GWFO walk-forward optimizations and compare the results. This is what a cluster analysis does.

Cluster analysis can reveal one other thing about your system. It is very unlikely, but possible that your system will PASS the 5 GWFO test criteria only if it is subjected to the 10-run 20% OOS exclusion of a default-values GWFO optimization or perhaps one or two GWFO optimizations with values very close to those. If say, 3 cluster analysis combinations PASS and the other 27 FAIL, you might want to re-consider whether you want to implement your new system!

But this is a very rare situation. More typically, a cluster analysis will simply reveal a better set of number of runs and OOS exclusion percentages to walk-forward analyze your system. And it will give you as close an estimate of an optimal re-optimization interval as can be determined statistically. If 2 or 3 cluster analysis combinations FAIL, you'll simply want to avoid those walk-forward run/exclusion% parameters as well as the parameters in adjacent cells even if they PASS.

### **Invoking a Cluster Analysis**

In the GWFO, select all the optimization report files and open them just as you would for a regular walk-forward analysis. Then go to the Optimize menu and select Cluster Analysis. The GWFO will now spin through 30 anchored walk-forward analyses.

For practical purposes, you should probably plan on doing this on one of your dedicated number-crunching PCs because this analysis could take the better part of a week to complete. You can at least monitor your progress by going to the Cluster Analysis tab in the GWFO and seeing how much of the cluster analysis matrix is complete.

### **Later, That Same Week**

When your cluster analysis is complete, you will have a report consisting of Re-optimized P&L (annualized), Walk-forward Overall result (Pass/Fail), Re-optimization interval (days).

Here's an example report:

Re-optimized P&L (annualized) (Matrix 1)

OOS% \ Runs	5	10	15	20	25	30
10	19291.11	18709.31	13809.98	17066.38	15482.08	14174.17
15	16341.30	14297.19	13152.52	12187.08	14114.28	13606.57
20	14208.05	15463.51	11390.87	11544.66	14597.98	13575.22
25	11135.77	12054.39	15473.71	13392.82	11878.76	15208.33
30	11482.46	11563.75	13434.59	10564.72	11821.49	12805.42

Walk-forward Overall result (Pass/Fail) (Matrix 2)

```
OOS% \ Runs 5 10 15 20 25 30
10 PASS PASS PASS PASS PASS PASS
15 PASS** PASS PASS PASS PASS PASS
20 PASS PASS PASS PASS PASS PASS
25 PASS PASS PASS PASS PASS PASS
30 PASS PASS PASS PASS PASS PASS
```

Walk-forward Robustness Index% (Matrix 3)

```
OOS% \ Runs 5 10 15 20 25 30
10 96.3% 99.2% 94.4% 82.1% 73.4% 91.9%
15 101.8% 90.3% 81.1% 94.1% 83.2% 74.2%
20 92.5% 91.2% 87.8% 88.4% 78.9% 77.2%
25 97.3% 95.1% 88.1% 86.0% 93.0% 83.2%
30 86.0% 80.1% 89.8% 82.3% 74.6% 74.1%
```

Re-optimization interval (days) (Matrix 4)

```
OOS% \ Runs 5 10 15 20 25 30
10 32 24 19 15 13 12
15 42 28 22 17 15 13
20 50 32 24 19 15 13
25 56 34 25 20 16 14
30 61 36 26 20 16 14
```

## Let's Start By Defining a Cluster

A cluster in this context is a 3 x 3 matrix of results. The value we're looking for is the center value of the cluster. The cluster is evaluated on the basis of its grand sum.

OK, let's back up a step. Remember what we're looking for: robustness. That means that to us, peak values are probably not optimal values. In fact, what we're looking for throughout the entire Grail evaluation process are values that are the least likely to come apart on us if and when something in our market changes over time.

With this general philosophy in mind, what we're looking for in our cluster analysis results are good values such that an excursion of one row or one column either way will still give us...good values! So with this in mind, let's look for the walk-forward parameters that give us the "best" net profit.

After a few great moments in matrix math, we can see that the following cluster produces the highest grand sum:

```
OOS% \ Runs 5 10 15 20 25 30
```

10 **19291.11 18709.31 13809.98** 17066.38 15482.08 14174.17  
15 **16341.30 14297.19 13152.52** 12187.08 14114.28 13606.57  
20 **14208.05 15463.51 11390.87** 11544.66 14597.98 13575.22  
25 11135.77 12054.39 15473.71 13392.82 11878.76 15208.33  
30 11482.46 11563.75 13434.59 10564.72 11821.49 12805.42

Now our job is easy. The value we're interested in is the *center* value of this cluster. It was obtained by using 10 runs and an out-of-sample exclusion percentage of 15. We can add or subtract one class interval (in other words, make a one row or column excursion in any direction) from *this* run / OOS% value pair and expect perfectly satisfactory results (everything in **bold**).

As a cross-check, examine Matrix 2. All of the GWFO test results for the values in this cluster PASS and one even gets a PASS\*\* (pass with distinction). It's nice that in this example, the robustness index percentages yield similar results to the net profits.

If you look at this cluster in Matrix 3, you'll observe that the center and adjacent robustness percentage values are all satisfactory to outstanding. At cell (10,15) we read a robustness index of 90.3%, a very respectable value. In other words, by GWFO optimizing our strategy with these run/OOS% values, we are estimating our actual blind walk-forward performance to be about 90% of observed performance.

Now look at Matrix 4. At these coordinates (x=10, y=15), we can also see that the suggested re-optimization interval is every 28 days. So the results from this cluster analysis suggest a re-op using 10 runs and 15% OOS exclusion every 28 days.

Now, there's one other thing you should consider before settling on these values, particularly if two or more clusters score closely on net profit or robustness, and that's the equity curve. All other factors being equal, pick the result that has the most linear, consistent, and for lack of a more objective descriptor, the best looking equity curve.

As with our initial GGO evaluation of the model Building test results, a really nice-looking equity curve can even override small advantages in a calculated fitness function.

Even at this stage of our strategy optimization, a good-looking equity curve can trump small numeric advantages in calculated, synthetic fitness functions or calculated performance values. In trading, the general rule is to make full use of all the significant information you have at your disposal to support your decision to execute a trade. Successful computer-assisted strategy building requires your best judgment based on every decision support resource you have, including the graphical linearity of your equity curve.

## **Conclusions and Comments**

The Grail is designed and built by honest, high integrity developers to very high mathematical and statistical standards. All of The Grail Executive Team actually use

The Grail tools and techniques to create real systems that they use to trade real markets with real money.

If this were an industry standard for trading software, we would hold a near-monopoly in our market. This tutorial, like all of the intellectual property of The Grail should be considered a work in progress. Because of the nature of our software, we are privileged to have a highly sophisticated user base.

I never cease to be amazed by the things that I learn every day about the markets, about systems and about software. We at The Grail are highly receptive to comments, suggestions, feedback and corrections. We invite your comments and suggestions, and we invite you to visit one of the most active and dynamic threads at TradeStation.com, namely:

[https://www.TradeStation.com/Discussions/Topic.aspx?Topic\\_ID=31924](https://www.TradeStation.com/Discussions/Topic.aspx?Topic_ID=31924)

Thank you and good luck and success in your trading!