Guide to Using

Evolver

The Genetic Algorithm Solver for Microsoft Excel

Version 5.7
September, 2010

Palisade Corporation
798 Cascadilla St.
Ithaca, NY USA 14850
(607) 277-8000
(607) 277-8001 (fax)
http://www.palisade.com (website)
sales@palisade.com (e-mail)

Table of Contents

Chapter 1: Introduction

Introduction

Evolver represents the fastest, most advanced commercial genetic algorithm-based optimizer ever available. Evolver, through the application of powerful genetic algorithm-based optimization techniques, can find optimal solutions to problems which are "unsolvable" for standard linear and non-linear optimizers. Evolver is offered in two versions - professional and industrial - to allow you to select the optimizer with the capacity you need

The **Evolver User's Guide**, which you are reading now, offers an introduction to Evolver and the principles behind it, then goes on to show several example applications of Evolver's unique genetic algorithm technology. This complete manual may also be used as a fully-indexed reference guide, with a description and illustration of each Evolver feature.

Before You Begin

Before you install and begin working with Evolver, make sure that your Evolver package contains all the required items, and check that your computer meets the minimum requirements for proper use.

What the Package Includes

Evolver may be purchased on its own and also ships with the DecisionTools Suite Professional and Industrial versions. The Evolver CD-ROM contains the Evolver Excel add-in, several Evolver examples, and a fully-indexed Evolver on-line help system. The DecisionTools Suite Professional and Industrial versions contain all of the above plus additional applications.

About This Version

This version of Evolver can be installed as a 32-bit program for Microsoft Excel 2000 or higher.

Working with your Operating Environment

This User's Guide assumes that you have a general knowledge of the Windows operating system and Excel. In particular:

- *You are familiar with your computer and using the mouse.*
- *You are familiar with terms such as icons, click, double-click, menu, window, command and object.*
- *You understand basic concepts such as directory structures and file naming.*

If You Need Help

Technical support is provided free of charge for all registered users of Evolver with a current maintenance plan, or is available on a per incident charge. To ensure that you are a registered user of Evolver, **please register online at http://www.palisade.com/support/register.asp**.

If you contact us by telephone, please have your serial number and User's Guide ready. We can offer better technical support if you are in front of your computer and ready to work.

Before Calling

Before contacting technical support, please review the following checklist:

- *Have you referred to the on-line help?*
- *Have you checked this User's Guide and reviewed the on-line multimedia tutorial?*
- *Have you read the README.WRI file? It contains current information on* Evolver *that may not be included in the manual.*
- *Can you duplicate the problem consistently? Can you duplicate the problem on a different computer or with a different model?*
- *Have you looked at our site on the World Wide Web? It can be found at* ***http://www.palisade.com****. Our Web site also contains the latest FAQ (a searchable database of tech support questions and answers) and* Evolver *patches in our Technical Support section. We recommend visiting our Web site regularly for all the latest information on* Evolver *and other Palisade software.*

Contacting Palisade

Palisade Corporation welcomes your questions, comments or suggestions regarding Evolver. Contact our technical support staff using any of the following methods:

- *Email us at support@palisade.com.*
- *Telephone us at (607) 277-8000 any weekday from 9:00 AM to 5:00 PM, EST. Follow the prompt to reach technical support.*
- *Fax us at (607) 277-8001.*
- *Mail us a letter at:*

  Technical Support
  Palisade Corporation
  798 Cascadilla St.
  Ithaca, NY 14850 USA

If you want to contact Palisade Europe:

- *Email us at support@palisade-europe.com.*
- *Telephone us at +44 1895 425050 (UK).*
- *Fax us at +44 1895 425051 (UK).*
- *Mail us a letter at:*

  Palisade Europe
  31 The Green
  West Drayton
  Middlesex
  UB7 7PN
  United Kingdom

If you want to contact Palisade Asia-Pacific:

- *Email us at support@palisade.com.au.*
- *Telephone us at + 61 2 9252 5922 (AU).*
- *Fax us at + 61 2 9252 2820 (AU).*
- *Mail us a letter to:*

  Palisade Asia-Pacific Pty Limited
  Suite 404, Level 4
  20 Loftus Street
  Sydney NSW 2000
  Australia

Regardless of how you contact us, please include the product name, version and serial number. The exact version can be found by selecting the Help About command on the Evolver menu in Excel.

Student Versions

Telephone support is not available with the student version of Evolver. If you need help, we recommend the following alternatives:

- *Consult with your professor or teaching assistant.*
- *Log on to http://www.palisade.com for answers to frequently asked questions.*
- *Contact our technical support department via e-mail or fax.*

Evolver System Requirements

System requirements for Evolver include:

- *Pentium PC or faster with a hard disk.*
- *Microsoft Windows 2000 SP4 or higher.*
- *Microsoft Excel Version 2000 or higher.*

Installation Instructions

Evolver is an add-in program to Microsoft Excel. By adding additional commands to the Excel menu bars, Evolver enhances the functionality of the spreadsheet program.

General Installation Instructions

The Setup program copies the Evolver system files into a directory you specify on your hard disk. To run the Setup program in Windows 2000 or higher:

1) *Insert the Evolver or DecisionTools Suite Professional or Industrial version CD-ROM in your CD-ROM drive*
2) *Click the Start button, click Settings and then click Control Panel*
3) *Double-click the Add/Remove Programs icon*
4) *On the Install/Uninstall tab, click the Install button*
5) *Follow the Setup instructions on the screen*

If you encounter problems while installing Evolver, verify that there is adequate space on the drive to which you're trying to install. After you've freed up adequate space, try rerunning the installation.

Removing* Evolver *from Your Computer

If you wish to remove Evolver (or the DecisionTools Suite) from your computer, use the Control Panel's Add/Remove Programs utility and select the entry for @RISK or the DecisionTools Suite.

The DecisionTools Suite

Evolver can be used with the DecisionTools Suite, a set of products for risk and decision analysis available from Palisade Corporation. The default installation procedure of Evolver puts Evolver in a subdirectory of a main "Program Files\Palisade" directory. This is quite similar to how Excel is often installed into a subdirectory of a "Microsoft Office" directory.

One subdirectory of the Program Files\Palisade directory will be the Evolver directory (by default called Evolver5). This directory contains the Evolver add-in program file (EVOLVER.XLA) plus example models and other files necessary for Evolver to run. Another subdirectory of Program Files\Palisade is the SYSTEM directory which contains files needed by every program in the DecisionTools Suite, including common help files and program libraries.

Setting Up the Evolver Icons or Shortcuts

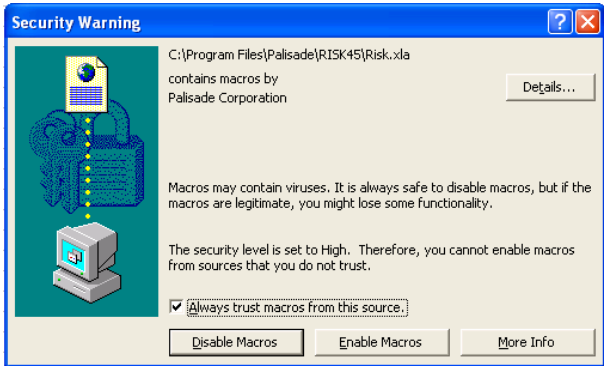
In Windows, setup automatically creates a Evolver command in the Programs menu of the Taskbar. However, if problems are encountered during Setup, or if you wish to do this manually another time, follow these directions:

1) *Click the Start button, and then point to Settings.*
2) *Click Taskbar, and then click the Start Menu Programs tab.*
3) *Click Add, and then click Browse.*
4) *Locate the file EVOLVER.EXE and double click it.*
5) *Click Next, and then double-click the menu on which you want the program to appear.*
6) *Type the name* "Evolver"*, and then click Finish.*

Macro Security Warning Message on Startup

Microsoft Office provides several security settings (under **Tools>Macro>Security**) to keep unwanted or malicious macros from being run in Office applications. A warning message appears each time you attempt to load a file with macros, unless you use the lowest security setting. To keep this message from appearing every time you run a Palisade add-in, Palisade digitally signs their add-in files. Thus, once you have specified **Palisade Corporation** as a trusted source, you can open any Palisade add-in without warning messages. To do this:

- Click **Always trust macros from this source** when a Security Warning dialog (such as the one below) is displayed when starting Evolver.



Other Evolver Information

Additional information on Evolver can be found in the following sources:

Evolver *Readme*

This file contains a quick summary of Evolver, as well as any late-breaking news or information on the latest version of your software. View the Readme file by selecting the Windows Start Menu/ Programs/ Palisade DecisionTools/ Readmes and clicking on Evolver 5.0 – Readme. It is a good idea to read this file before using Evolver.

Evolver *Tutorial*

The Evolver on-line tutorial provides first-time users with a quick introduction of Evolver and genetic algorithms. The presentation takes only a few minutes to view. See the Learning Evolver section below for information on how to access the tutorial.

Learning Evolver

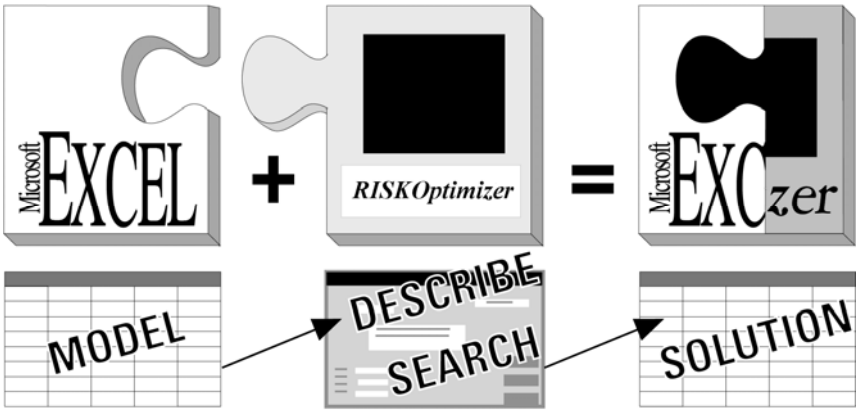
The quickest way to become familiar with Evolver is by using the on-line Evolver Tutorial, where experts guide you through sample models in movie format. This tutorial is a multi-media presentation on the main features of Evolver.

The tutorial can be run by selecting the **Evolver Help menu Getting Started Tutorial command**.

Chapter 2: Background

What Is Evolver?

The Evolver software package provides users with an easy way to find optimal solutions to virtually any type of problem. Simply put, Evolver finds the best inputs that produce a desired output. You can use Evolver to find the right mix, order, or grouping of variables that produces the highest profits, the lowest risk, or the most goods from the least amount of materials. Evolver is most often used as an add-in to the Microsoft Excel spreadsheet program; users set up a model of their problem in Excel, then call up Evolver to solve it.



You must first model your problem in Excel, then describe it to the Evolver add-in.

Excel provides all of the formulas, functions, graphs, and macro capabilities that most users need to create realistic models of their problems. Evolver provides the interface to describe the uncertainty in your model and what you are looking for, and provides the engines that will find it. Together, they can find optimal solutions to virtually any problem that can be modeled.

How does Evolver work?

Evolver uses a proprietary set of ***genetic algorithms*** to search for optimum solutions to a problem.

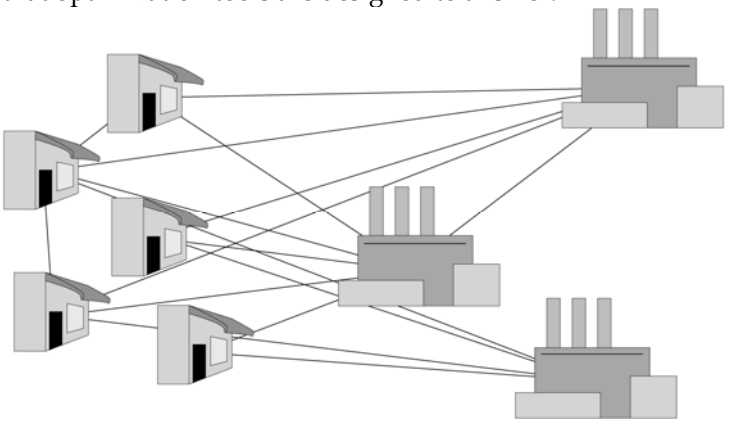
Genetic Algorithms

Genetic algorithms are used in Evolver to find the best solution for your model. Genetic algorithms mimic Darwinian principles of natural selection by creating an environment where hundreds of possible solutions to a problem can compete with one another, and only the "fittest" survive. Just as in biological evolution, each solution can pass along its good "genes" through "offspring" solutions so that the entire population of solutions will continue to evolve better solutions.

As you may already realize, the terminology used when working with genetic algorithms is often similar to that of its inspiration. We talk about how "crossover" functions help focus the search for solutions, "mutation" rates help diversify the "gene pool", and we evaluate the entire "population" of solutions or "organisms". To learn more about how Evolver's genetic algorithm works, see Chapter 7 - Genetic Algorithms.

What Is Optimization?

Optimization is the process of trying to find the best solution to a problem that may have many possible solutions. Most problems involve many variables that interact based on given formulas and constraints. For example, a company may have three manufacturing plants, each manufacturing different quantities of different goods. Given the cost for each plant to produce each good, the costs for each plant to ship to each store, and the limitations of each plant, what is the optimal way to adequately meet the demand of local retail stores while minimizing the transportation costs? This is the sort of question that optimization tools are designed to answer.



Optimization often deals with searching for the combination that yields the most from given resources.

In the example above, each proposed solution would consist of a complete list of what goods made by what manufacturing plant get shipped in what truck to what retail store. Other examples of optimization problems include finding out how to produce the highest profit, the lowest cost, the most lives saved, the least noise in a circuit, the shortest route between a set of cities, or the most effective mix of advertising media purchases. An important subset of optimization problems involves scheduling, where the goals may include maximizing efficiency during a work shift or minimizing schedule conflicts of groups meeting at different times. To learn more about optimization, see Chapter 6 - Optimization.

Why Build Excel Models?

To increase the efficiency of any system, we must first understand how it behaves. This is why we construct a working model of the system. Models are necessary abstractions when studying complex systems, yet in order for the results to be applicable to the "real-world," the model must not oversimplify the cause-and-effect relationships between variables. Better software and increasingly powerful computers allow economists to build more realistic models of the economy, scientists to improve predictions of chemical reactions, and business people to increase the sensitivity of their corporate models.

In the last few years computer hardware and software programs such as Microsoft Excel, have advanced so dramatically that virtually anyone with a personal computer can create realistic models of complex systems. Excel's built-in functions, macro capabilities and clean, intuitive interface allow beginners to model and analyze sophisticated problems. To learn more about building a model, see Chapter 9 - Evolver Extras.

Why Use Evolver?

Evolver's unique technology allows anyone with a PC and Excel for Windows to enjoy the benefits of optimization. Before Evolver, those who wished to increase efficiency or search for optimum solutions had three choices: guess, use low-powered problem-solving software, or hire experts in the optimization consulting field to design and build customized software. Here are a few of the most important advantages to using Evolver:

No More Guessing

When you are dealing with large numbers of interacting variables, and you are trying to find the best mix, the right order, the optimum grouping of those variables, you may be tempted to just take an "educated guess". A surprising number of people assume that any kind of modeling and analysis beyond guessing will require complicated programming, or confusing statistical or mathematical algorithms. A good optimized solution might save millions of dollars, thousands of gallons of scarce fuel, months of wasted time, etc. Now that powerful desktop computers are increasingly affordable, and software like Excel and Evolver are readily available, there is little reason to guess at solutions, or waste valuable time trying out many scenarios by hand.

More Accurate, More Meaningful

Evolver allows you to use the entire range of Excel formulas and even macros to build more realistic models of any system. When you use Evolver, you do not have to "compromise" the accuracy of your model because the algorithm you are using can not handle real world complexities. Traditional "baby" solvers (statistical and linear programming tools) force the user to make assumptions about the way the variables in their problem interact, thereby forcing users to build over-simplified, unrealistic models of their problem. By the time the user has simplified a system enough that these solvers can be used, the resulting solution is often too abstract to be practical. Any problems involving large amounts of variables, non-linear functions, lookup tables, if-then statements, database queries, or stochastic (random) elements cannot be solved by these methods, no matter how simply you try to design your model.

More Flexible

There are many solving algorithms which do a good job at solving small, simple linear and non-linear types of problems, including hill-climbing, baby-solvers, and other mathematical methods. Even when offered as spreadsheet add-ins, these general-purpose optimization tools can only perform numerical optimization. For larger or more complex problems, you may be able to write specific, customized algorithms to get good results, but this may require a lot of research and development. Even then, the resulting program would require modification each time your model changed.

Not only can Evolver handle numerical problems, it is the only commercial program in the world that can solve most combinatorial problems. These are problems where the variables must be shuffled around (permuted) or combined with each other. For example, choosing the batting order for a baseball team is a combinatorial problem; it is a question of swapping players' positions in the lineup. Complex scheduling problems are also combinatorial. The same Evolver can solve all these types of problems, and many more that nothing else can solve. Evolver's unique *genetic algorithm* technology allows it to optimize virtually any type of model; any size and any complexity.

More Powerful

Evolver finds better solutions. Most software derives optimum solutions mathematically and systematically. Too often these methods are limited to taking an existing solution and searching for the closest answer that is better. This "local" solution may be far from the optimal solution. Evolver intelligently samples the entire realm of possibilities, resulting in a much better "global" solution.

Easier to Use

In spite of its obvious power and flexibility advantages, Evolver remains easy to use because an understanding of the complex genetic algorithm techniques it uses is completely unnecessary. Evolver doesn't care about the "nuts and bolts" of your problem; it just needs a spreadsheet model that can evaluate how good different scenarios are. Just select the spreadsheet cells that contain the variables and tell Evolver what you are looking for. Evolver intelligently hides the difficult technology, automating the "what-if" process of analyzing a problem.

Although there have been many commercial programs developed for mathematical programming and model-building, spreadsheets are by far the most popular, with literally millions being sold each month. With their intuitive row and column format, spreadsheets are easier to set up and maintain than other dedicated packages. They are also more compatible with other programs such as word processors and databases, and offer more built-in formulas, formatting options, graphing, and macro capabilities than any of the stand-alone packages. Because Evolver is an add-in to Microsoft Excel, users have access to the entire range of functions and development tools to easily build more realistic models of their system.

Cost Effective

Many companies have hired trained consultants to provide customized optimization systems. Such systems will often perform quite well, but may require many months and a large investment to develop and implement. These systems are also difficult to learn, and therefore require costly training and constant maintenance. If your system must be altered, you may need to develop a whole new algorithm to find optimal solutions. For a considerably smaller investment, Evolver supplies the most powerful genetic algorithms available and allows for quick and accurate solutions to a wide variety of problems. Because it works in an intuitive and familiar environment, there is virtually no costly training and maintenance.

You may even wish to add Evolver's optimization power to your own custom programs. In just a few days, you could use Visual Basic to develop your own scheduling, distribution, manufacturing or financial management system. See the Evolver Developer Kit for details on developing an Evolver-based application.

Chapter 3: Evolver: Step-by-Step

Introduction

In this chapter, we will take you through an entire Evolver optimization one step at a time. If you do not have Evolver installed on your hard drive, please refer to the installation section of Chapter 1: Introduction and install Evolver before you begin this tutorial.

We will start by opening a pre-made spreadsheet model, and then we will define the problem to Evolver using probability distributions and the Evolver dialogs. Finally we will oversee Evolver's progress as it is searching for solutions, and explore some of the many options in the Evolver Watcher. For additional information about any specific topic, see the index at the back of this manual, or refer to Chapter 5: Evolver Reference.

***NOTE:** The screens shown below are from Excel 2007. If you are using other versions of Excel, your windows may appear slightly different from the pictures.*

The problem-solving process begins with a model that accurately represents your problem. Your model must be able to evaluate a given set of input values (adjustable cells) and produce a numerical rating of how well those inputs solve the problem (the evaluation or "fitness" function). As Evolver searches for solutions, this fitness function provides feedback, telling Evolver how good or bad each guess is, thereby allowing Evolver to breed increasingly better guesses. When you create a model of your problem you must pay close attention to the fitness function, because Evolver will be doing everything it can to maximize (or minimize) this cell.

The Evolver Tour

Starting Evolver

To start Evolver, either: ***1) click the* Evolver *icon in your Windows desktop, or 2) select* Palisade DecisionTools *then* Evolver 5.0 *in the Windows Start menu Programs entries. Each of these methods starts both Microsoft Excel and* Evolver.**

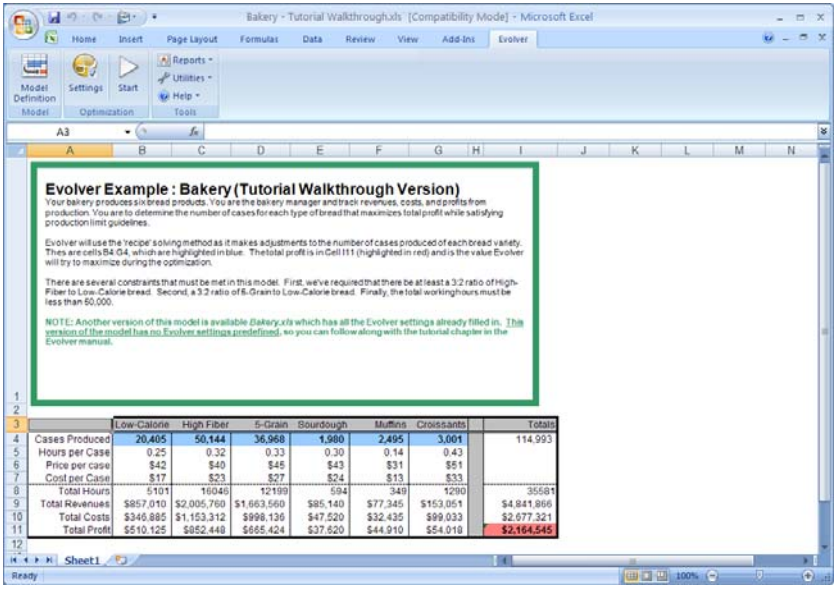
The Evolver *Toolbar*

When Evolver is loaded, a new Evolver ribbon or toolbar is visible in Excel. This toolbar contains buttons which can be used to specify Evolver settings and start, pause, and stop optimizations.

Opening an Example Model

To review the features of Evolver, you'll examine an example model that was installed when you installed Evolver. To do this:

1) ***Open the Bakery – Tutorial Walkthrough.XLS worksheet using the Help menu Example Spreadsheets command.***



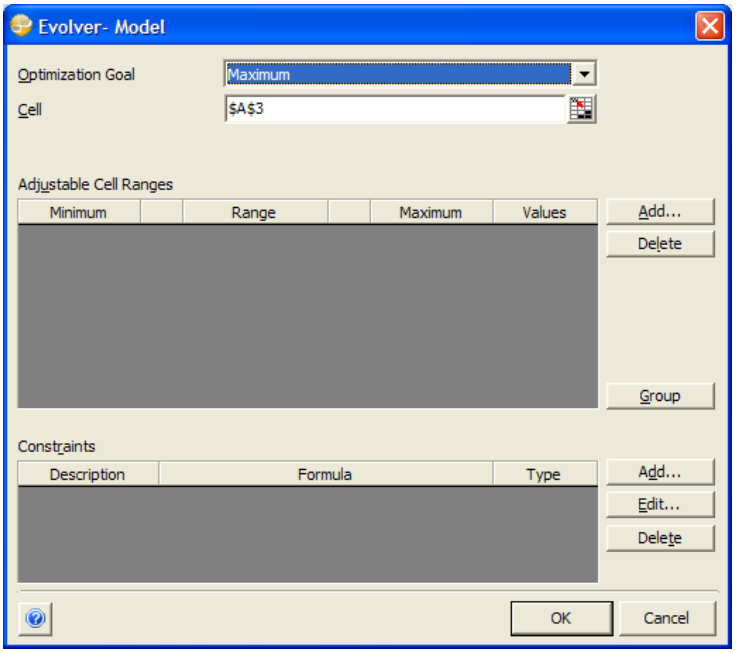
This example sheet contains a simple profit maximization problem for a bakery business. Your bakery produces 6 bread products. You are the bakery manager and track revenues, costs, and profits from production. You are to determine the number of cases for each type of bread that maximizes total profit while satisfying production limit guidelines. The guidelines you face include 1) *meeting the production quota for low calorie bread*, 2) *maintaining an acceptable ratio of high fiber to low calorie*, 3) *maintaining an acceptable ratio of 5 grain to low calorie*, and 4) *keeping production time within limits for person hours used*.

The Evolver Model Dialog

To set the Evolver options for this worksheet:

1) Click the* Evolver Model *icon on the* Evolver *toolbar (the one on the far left).

This displays the following Evolver Model dialog box:



The Evolver Model Dialog is designed so users can describe their problem in a simple, straightforward way. In our tutorial example, we are trying to find the number of cases to produce for the different bread products in order to maximize overall total profit.

Selecting the Target Cell

The "total profit" in the example model is what's known as the target cell. This is the cell whose value you are trying to minimize or maximize, or the cell whose value you are trying to make as close as possible to a pre-set value. To specify the target cell:

1) ***Set the "Optimization Goal" option to "Maximum."***

2) ***Enter the target cell, $I$11, in the "Cell" field.***

Cell references can be entered in Evolver dialog fields two ways: 1) You may click in the field with your cursor, and type the reference directly into the field, or 2) with your cursor in the selected field, you may click on Reference Entry icon to select the worksheet cell(s) directly with the mouse.

Adding Adjustable Cell Ranges

Now you must specify the location of the cells that contain values which Evolver can adjust to search for solutions. These variables are added and edited one block at a time through the *Adjustable Cells Ranges* section of the Model Dialog. The number of cells you can enter in Adjustable Cells Ranges depends on the version of Evolver you are using.

1) ***Click the "Add" button in the "Adjustable Cell Ranges" section.***

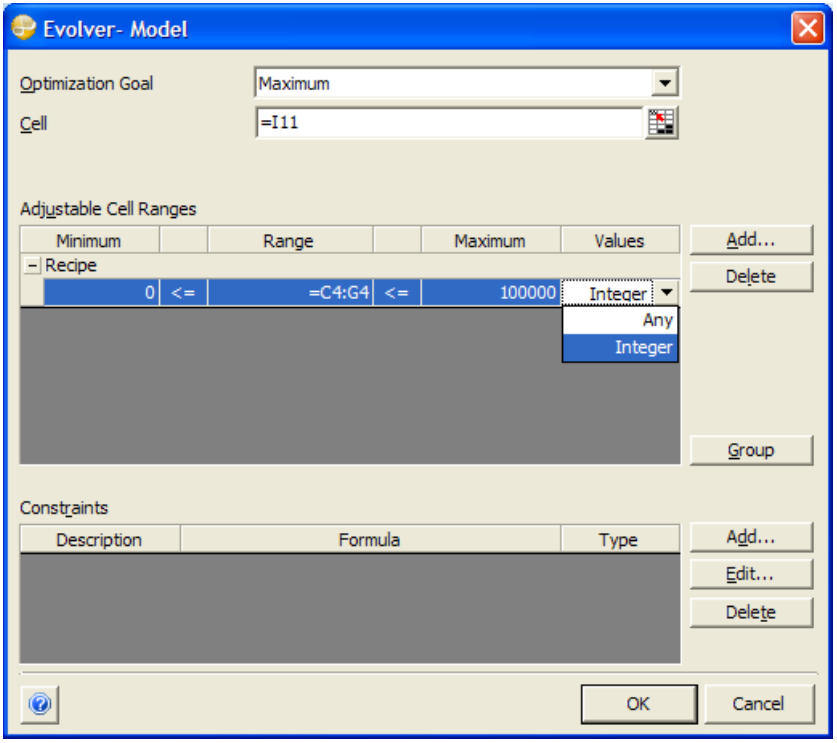
2) ***Select $C$4:$G$4 as the cells in Excel you want to add as an adjustable cell range.***

Entering the Min-Max Range for Adjustable Cells

Most of the time you'll want to restrict the possible values for an adjustable cell range to a specific minimum-maximum range. In Evolver this is known as a "range" constraint. You can quickly enter this min-max range when you select the set of cells to be adjusted. For the Bakery example, the minimum possible value for cases produced for each of the bread products in this range is 0 and the maximum is 100,000. To enter this range constraint:

1) ***Enter 0 in the Minimum cell and 100,000 in the Maximum cell.***

2) ***In the Values cell, select Integer from the drop-down list***

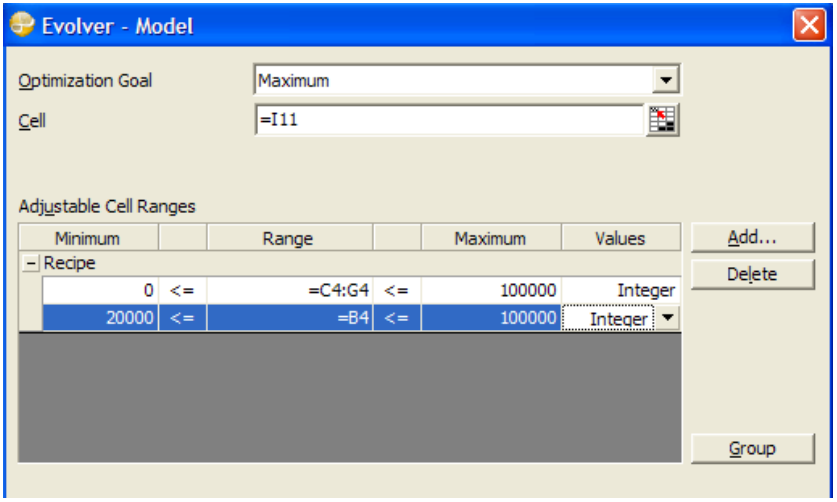


Now, enter a second cell range to be adjusted:

1) ***Click Add to enter a second adjustable cell.***

2) ***Select cell B4.***

3) ***Enter 20,000 as the Minimum and 100,000 as the Maximum.***



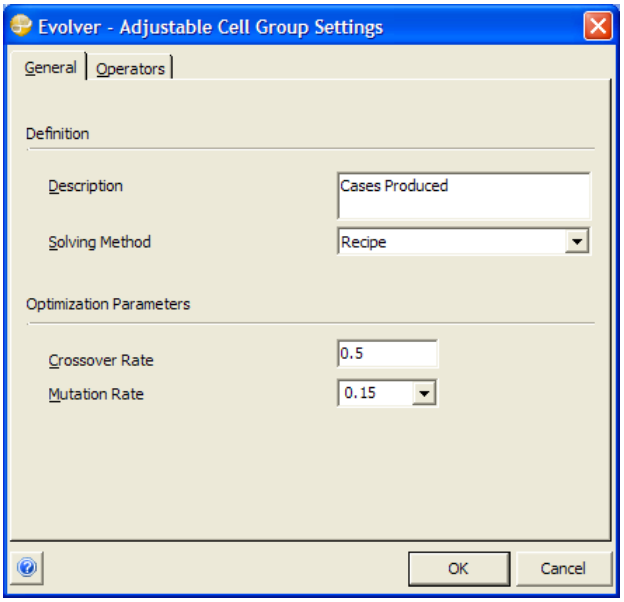
This specifies the last adjustable cell, B4, representing the production level for low calorie bread.

If there were additional variables in this problem, we would continue to add sets of adjustable cells. In Evolver, you may create an unlimited number of groups of adjustable cells. To add more cells, click the "Add" button once again.

Later, you may want to check the adjustable cells or change some of their settings. To do this, simply edit the min-max range in the table. You may also select a set of cells and delete it by clicking the "Delete" button.

Selecting a Solving Method

When defining adjustable cells, you can specify a ***solving method*** to be used. Different types of adjustable cells are handled by different solving methods. Solving methods are set for a Group of adjustable cells and are changed by clicking the ***"Group"*** button and displaying the **Adjustable Cell Group Settings** dialog box. Often you'll use the default "recipe" solving method where each cell's value can be changed independently of the others. Since this is selected as the default method, you don't have to change it.



The "recipe" and "order" solving methods are the most popular and they can be used together to solve complex combinatorial problems. Specifically, the "recipe" solving method treats each variable as an ingredient in a recipe, trying to find the "best mix" by changing each variable's value independently. In contrast, the "order" solving

method swaps values between variables, shuffling the original values to find the "best order."

For this model, leave the Solving Method as Recipe and simply:

- **Enter the label "Cases Produced" in the Description field.**

Constraints

Evolver allows you to enter constraints which are conditions that must be met for a solution to be valid. In this example model there are three additional constraints that must be met for a possible set of production levels for each of the bread products to be valid. These are in addition to the range constraints we already entered for the adjustable cells. They are:

1) **Maintaining an acceptable ratio of high fiber to low calorie bread (high fiber cases produced >= 1.5 * low calorie cases produced)**
2) **Maintaining an acceptable ratio of 5 grain to low calorie bread (5 grain cases produced >= 1.5 * low calorie cases produced)**
3) **Keeping production time within limits for person hours used (total person hours used < 50,000)**

Each time Evolver generates a possible solution to your model it checks that the constraints you have entered are valid.

Constraints are displayed in the bottom *Constraints* section of the Evolver Model dialog box. Two types of constraints can be specified in Evolver:

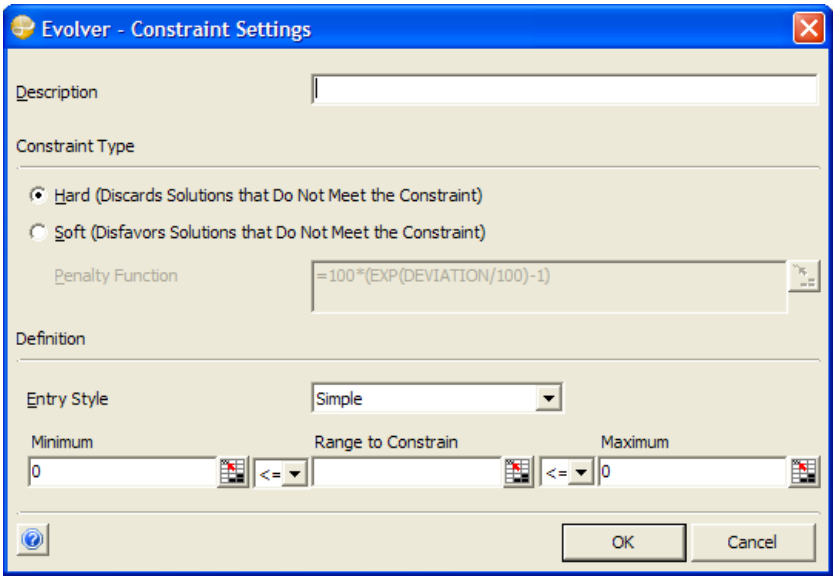
- **Hard**. These are conditions that must be met for a solution to be valid (i.e., a hard iteration constraint could be C10<=A4; in this case, if a solution generates a value for C10 that is greater than the value of cell A4, the solution will be thrown out)
- **Soft.** These are conditions which we would like to be met as much as possible, but which we may be willing to compromise for a big improvement in fitness or target cell result. (i.e., a soft constraint could be C10<100. In this case, C10 could go over 100, but when that happens the calculated value for the target cell would be decreased according to the penalty function you have entered).

Adding a Constraint

To add a constraint:

1) ***Click the Add button in the Constraints section of the main* Evolver *dialog.***

This displays the Constraint Settings dialog box, where you enter the constraints for your model.



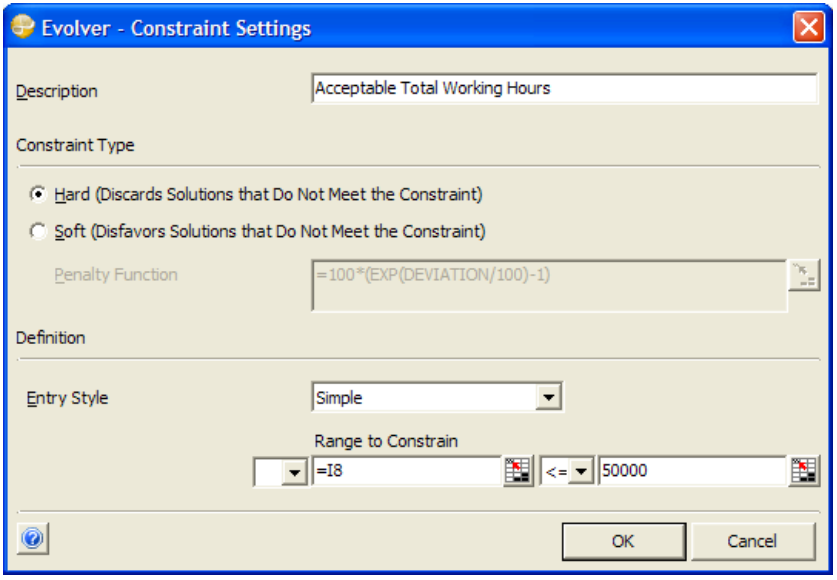
Simple Range of Values and Formula Constraints

Two formats – *Simple* and *Formula* – can be used for entering constraints. The Simple Range of Values format allows constraints to be entered using simple <,<=, >, >= or = relations. A typical Simple Range of Values constraint would be *0<Value of A1<10*, where A1 is entered in the *Cell Range* box, 0 is entered in the *Min* box and 10 is entered in the *Max* box. The operator desired is selected from the drop down list boxes. With a Simple Range of Values format constraint, you can enter just a Min value, just a Max or both.

A formula constraint, on the other hand, allows you to enter any valid Excel formula as a constraint, such as A19<(1.2*E7)+E8. For each possible solution Evolver will check whether the entered formula evaluates to TRUE or FALSE to see if the constraint has been met. If you want to use a boolean formula in a worksheet cell as a constraint, simply reference that cell in the *Formula* field of the Constraint Settings dialog box.

To enter the constraints for the Bakery model you'll specify three new hard constraints. These are hard constraints as the entered conditions must be met or the possible solution will be discarded by Evolver. First, enter the Simple Range of Values format hard constraints:

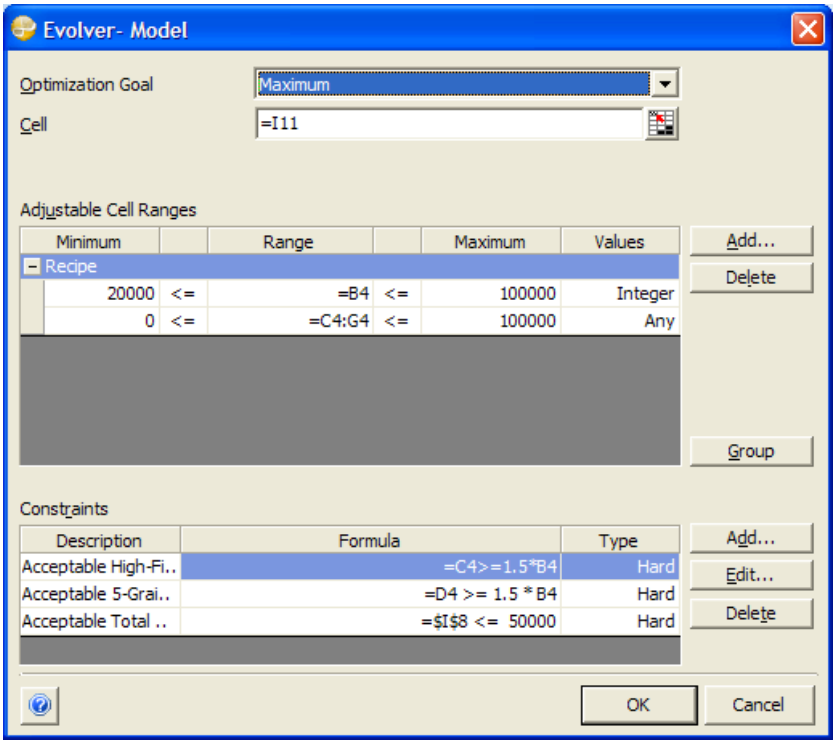
1) ***Enter " Acceptable Total Working Hours" in the description box.***
2) ***In the Range to Constrain box, enter I8.***
3) ***Select the <= operator to the right of the Range to Constrain.***
4) ***Enter 50,000 in the Maximum box.***
5) ***Clear the default value of 0 in the Minimum box.***
6) ***To the left of Range to Constrain, clear the operator by selecting a blank from the drop down list***
7) ***Click OK to enter this constraint.***



Now, enter the formula format hard constraints:

1) *Click Add to display the Constraint Settings dialog box again.*
2) *Enter "Acceptable ratio of high fiber to low calorie" in the description box.*
3) *In the Entry Style box, select Formula.*
4) *In the Constraint Formula box, enter C4>= 1.5*B4.*
5) *Click OK.*
6) *Click Add to display the Constraint Settings dialog box again.*
7) *Enter "Acceptable ratio of 5-grain to low calorie" in the description box.*
8) *In the Entry Style box, select Formula.*
9) *In the Constraint Formula box, enter D4>= 1.5*B4.*
10) *Click OK*

Your Model dialog with the completed constraints section should look like this.



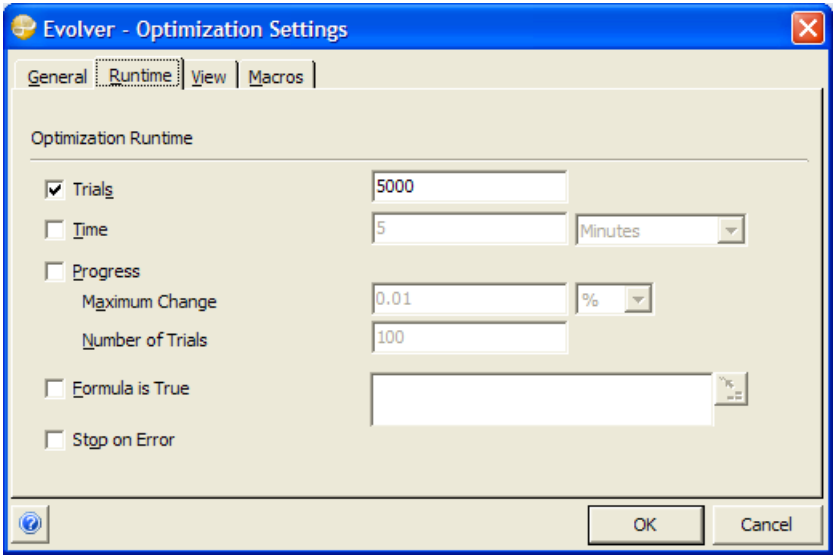
Other Evolver Options

Options such as *Update the Display, Random Number Seed and Stopping Conditions* are available to control how Evolver operates during an optimization. Let's specify some stopping conditions and display update settings.

Stopping Conditions

Evolver will run as long as you wish. The stopping conditions allow Evolver to automatically stop when either: a) *a certain number of scenarios or "trials" have been examined*, b) *a certain amount of time has elapsed*, c) *no improvement has been found in the last n scenarios* or d) *the entered Excel formula evaluates to TRUE*. To view and edit the stopping conditions:

1) ***Click the Optimization Settings icon on the* Evolver *toolbar.***
2) ***Select the Runtime tab.***



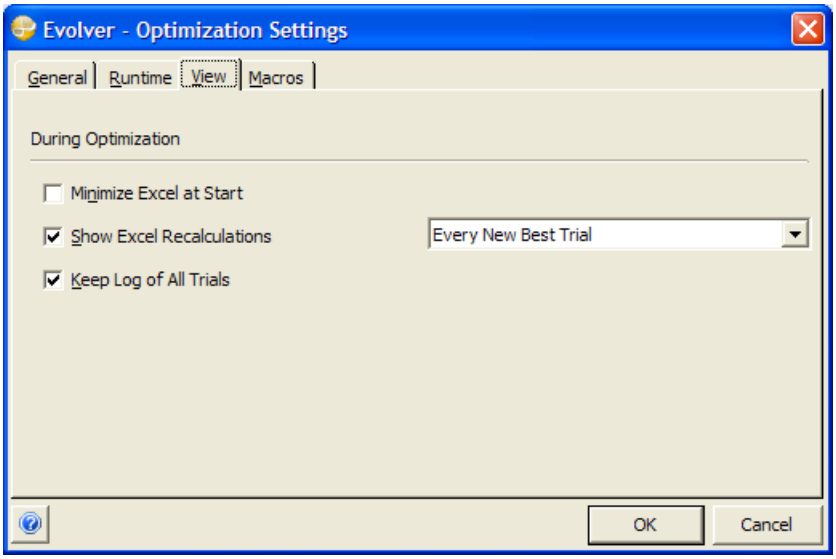
In the Optimization Settings dialog you can select any combination of these optimization stopping conditions, or none at all. If you select more than one stopping condition, Evolver will stop when any one of the selected conditions are met. If you do not select any stopping conditions, Evolver will run forever, until you stop it manually by pressing the "stop" button in the Evolver toolbar.

Trials	Minutes	Change in last	Formula is True
This option sets the number of "trials" that you would like Evolver to run. In each trial, Evolver evaluates one complete set of variables or one possible solution to the problem.	Evolver will stop after the specified amount of time has elapsed. This number can be a fraction (4.25).	This stopping condition is the most popular because it keeps track of the improvement and allows Evolver to run until the rate of improvement has decreased. For example, Evolver could stop if 100 trials have passed and we still haven't had any change in the best scenario found so far.	Evolver will stop if the entered Excel formula evaluates to TRUE in a model recalculation.

- **Turn off all stopping conditions to let Evolver run freely**.

View Options

While Evolver runs, a set of options are available on the View Tab to determine what you will see on-screen.



The During Optimization options include:

Every Trial	Every New Best Trial	Never
This option redraws the screen after each calculation, allowing you to see Evolver adjusting the variables and calculating the output. We suggest this option be turned on while you are learning Evolver, and also each time you use Evolver on a new model, to verify that your model is calculating correctly.	This option redraws the screen each time Evolver generates a new best answer, allowing you to see the current optimal solution at any time during the optimization.	This option never redraws the screen during the optimization. This results in the fastest possible optimizations but provides little feedback on calculated results during the run.

- **Turn on the "Every Trial"**

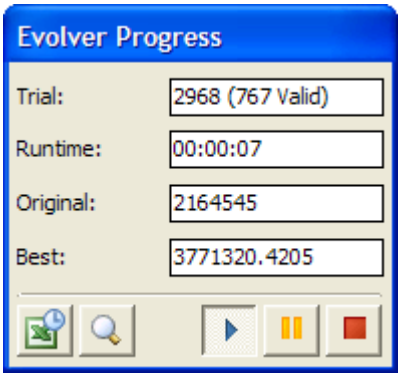
Running the Optimization

Now, all that remains is to optimize this model to maximize total profit while satisfying production limit guidelines. To do this:

1) ***Click OK to exit the Optimization Settings dialog.***

2) ***Click the Start Optimization icon***

As Evolver begins working on your problem, you will see the current best values for your adjustable cells – *Cases Produced* - in your spreadsheet. The best value for *Total Profit* is shown in the highlighted cell.



During the run, the Progress window displays: 1) the best solution found so far, 2) the original value for the target cell when the Evolver optimization began, 3) the number of trials of your model that have been executed and number of those trials which were valid; i.e., all constraints were met and 4) the time that has elapsed in the optimization.

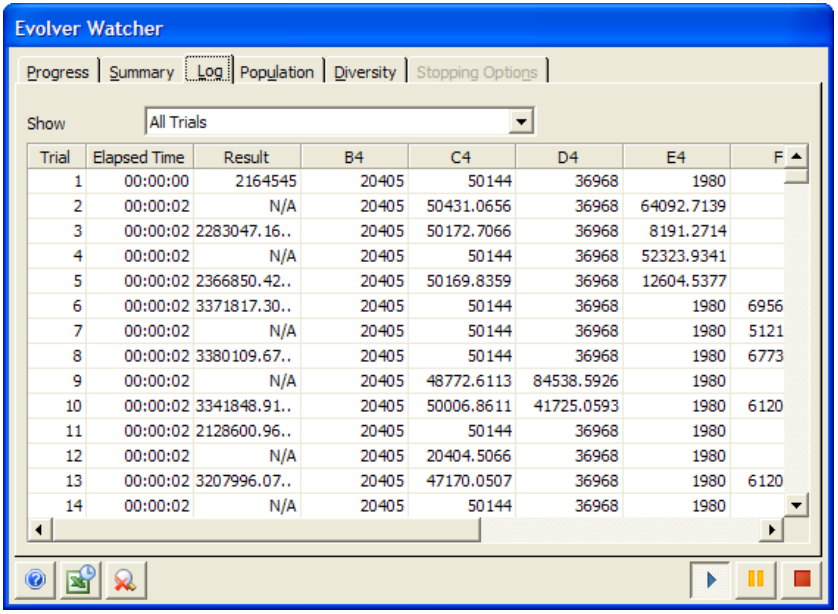
Any time during the run you can click the **Excel Updating Options** icon to see a live updating of the screen each trial.

The Evolver Watcher

Evolver can also display a running log of each trial solution. This is displayed in the Evolver Watcher while Evolver is running. The Evolver Watcher allows you to explore and modify many aspects of your problem as it runs. To view a running log of the trials:

1) ***Click the Watcher (magnifying glass) icon in the Progress window to display the Evolver Watcher***

2) ***Click the Log tab.***



Trial	Elapsed Time	Result	B4	C4	D4	E4	F
1	00:00:00	2164545	20405	50144	36968	1980	
2	00:00:02	N/A	20405	50431.0656	36968	64092.7139	
3	00:00:02	2283047.16..	20405	50172.7066	36968	8191.2714	
4	00:00:02	N/A	20405	50144	36968	52323.9341	
5	00:00:02	2366850.42..	20405	50169.8359	36968	12604.5377	
6	00:00:02	3371817.30..	20405	50144	36968	1980	6956
7	00:00:02	N/A	20405	50144	36968	1980	5121
8	00:00:02	3380109.67..	20405	50144	36968	1980	6773
9	00:00:02	N/A	20405	48772.6113	84538.5926	1980	
10	00:00:02	3341848.91..	20405	50006.8611	41725.0593	1980	6120
11	00:00:02	2128600.96..	20405	50144	36968	1980	
12	00:00:02	N/A	20405	20404.5066	36968	1980	
13	00:00:02	3207996.07..	20405	47170.0507	36968	1980	6120
14	00:00:02	N/A	20405	50144	36968	1980	

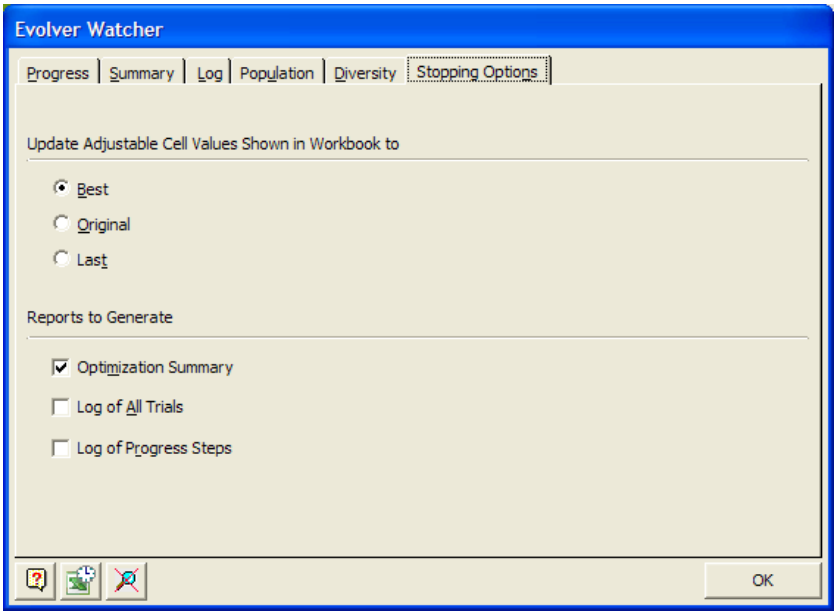
In this report the results of each trial solution is shown. The column for *Result* shows by trial the value of the target cell that you are trying to maximize or minimize - in this case, the Total Profit in $I$11. The columns for *C4* through G4 identify the values used for your adjustable cells.

Stopping the Optimization

After five minutes, Evolver will stop the optimization. You can also stop the optimization by:

1) *Clicking the Stop icon in the* **Evolver Watcher** *or* **Progress** *windows.*

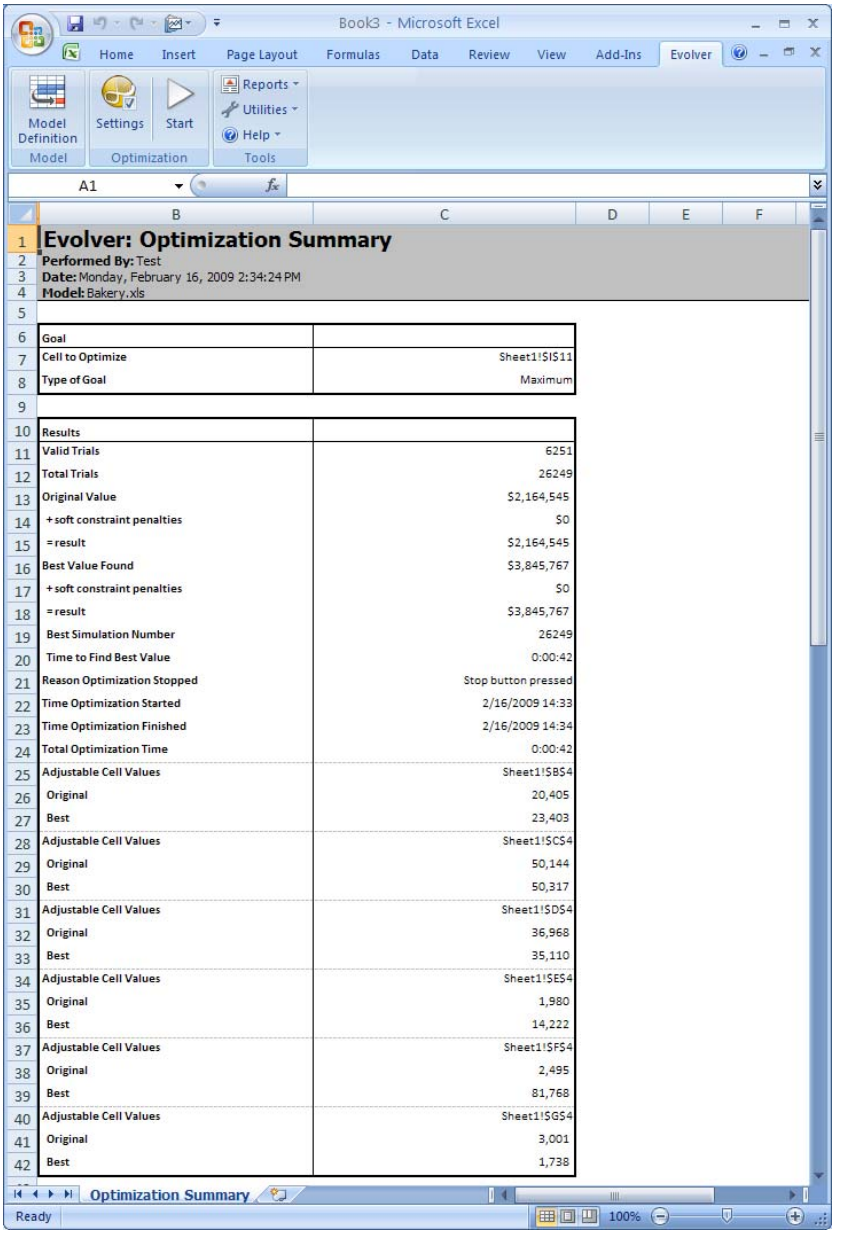
When the Evolver process stops, Evolver displays the Stopping Options tab which offers the following choices:



These same options will automatically appear when any of the stopping conditions that were set in the Evolver Optimization Settings dialog are met.

Summary Report

Evolver can create an optimization summary report that contains information such as date and time of the run, the optimization settings used, the value calculated for the target cell and the value for each of the adjustable cells.



Evolver: Optimization Summary

Performed By: Test
Date: Monday, February 16, 2009 2:34:24 PM
Model: Bakery.xls

Goal	
Cell to Optimize	Sheet1!$I$11
Type of Goal	Maximum

Results	
Valid Trials	6251
Total Trials	26249
Original Value	$2,164,545
+ soft constraint penalties	$0
= result	$2,164,545
Best Value Found	$3,845,767
+ soft constraint penalties	$0
= result	$3,845,767
Best Simulation Number	26249
Time to Find Best Value	0:00:42
Reason Optimization Stopped	Stop button pressed
Time Optimization Started	2/16/2009 14:33
Time Optimization Finished	2/16/2009 14:34
Total Optimization Time	0:00:42
Adjustable Cell Values	Sheet1!$B$4
Original	20,405
Best	23,403
Adjustable Cell Values	Sheet1!$C$4
Original	50,144
Best	50,317
Adjustable Cell Values	Sheet1!$D$4
Original	36,968
Best	35,110
Adjustable Cell Values	Sheet1!$E$4
Original	1,980
Best	14,222
Adjustable Cell Values	Sheet1!$F$4
Original	2,495
Best	81,768
Adjustable Cell Values	Sheet1!$G$4
Original	3,001
Best	1,738

This report is useful for comparing the results of successive optimizations.

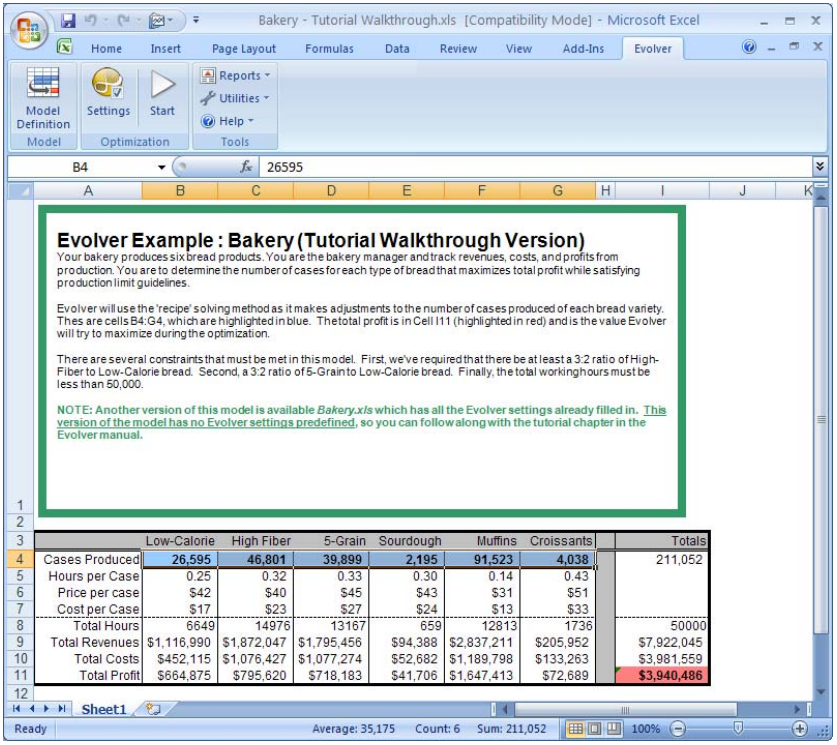
Placing the Results in Your Model

To place the new, optimized mix of production levels for the bakery to each of the six types of bread in your worksheet:

1) ***Click on the "Stop" button.***

2) ***Make sure the "Update Adjustable Cell Values Shown in Workbook to" option is set to "Best"***

You will be returned to the BAKERY – TUTORIAL WALKTHROUGH.XLS spreadsheet, with all of the new variable values that created the best solution.



IMPORTANT NOTE: Although in our example you can see that* Evolver *found a solution which yielded a total profit of 3,940,486, your result may be higher or lower than this. These differences are due to an important distinction between Evolver and all other problem-solving algorithms: it is the random nature of Evolver's genetic algorithm engine that enables it to solve a wider variety of problems, and find better solutions.

When you save any sheet after Evolver has run on it (even if you "restore" the original values of your sheet after running Evolver), all of the Evolver settings in the Evolver dialogs will be saved along with that sheet. The next time that sheet is opened, all of the most recent Evolver settings load up automatically. All of the other example worksheets have the Evolver settings pre-filled out and ready to be optimized.

NOTE: If you want to take a look at the Bakery model with all optimization settings pre-filled out, open the example model Bakery.XLS

Chapter 4: Example Applications

Introduction

This chapter explains how Evolver can be used in a variety of applications. These example applications may not include all of the features you would want in your own models, and are most effective as idea generators and templates. All examples illustrate how Evolver finds solutions by relying on the relationships that already exist in your worksheet, so it is important that your worksheet model accurately portray the problem you are trying to solve.

All Excel worksheet examples can be found within your EVOLVE32 directory, in a sub-directory called "EXAMPLES". They are listed alphabetically in this chapter. Examples use the following color-coding conventions:

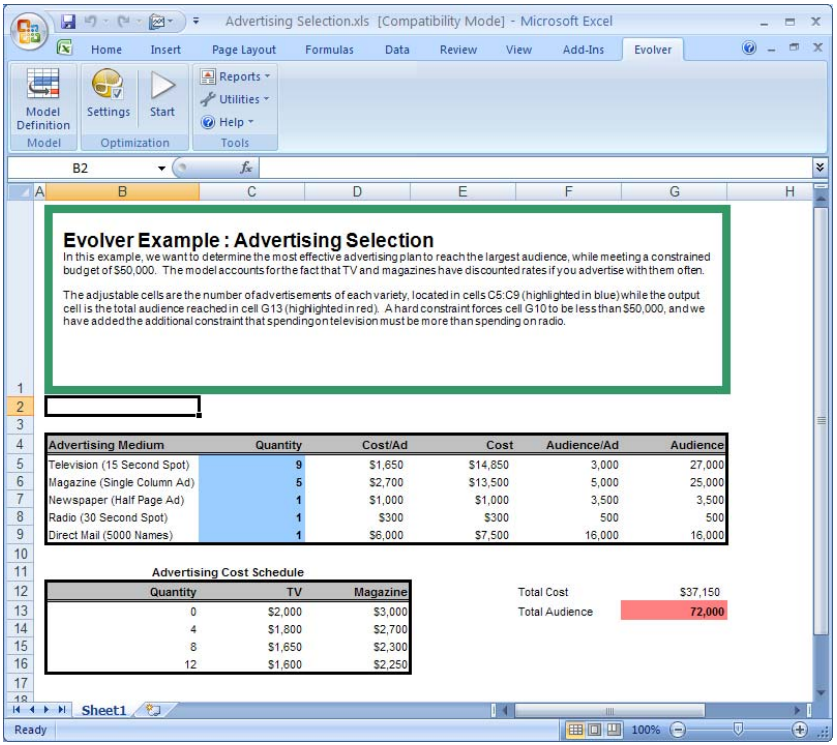
- *blue outlined cells. . . . . adjustable cells that Evolver will be adjusting.*
- *red outlined cells .. . . . . the target or goal cell.*

Each example comes with all Evolver settings pre-selected, including the target cell, adjustable cells, solving methods and constraints. You are encouraged to examine these dialog settings before optimizing. By studying the formulas and experimenting with different Evolver settings, you can get a better understanding of how Evolver is used. The models also let you replace the sample data with your own "user" data. If you decide to modify or adapt these example sheets, you may wish to save them with a new name to preserve the original examples for reference.

Advertising Selection

An ad agency must figure out the most efficient way to spend its advertising dollars to maximize the coverage for its target audience. It must not spend over its budget, and the amount spent on TV must be more than the amount spent on radio.

Example file:	Advertising Selection.xls
Goal:	Allocate advertising purchases, within your budget, among media which have various price breaks. Maximize people reached.
Solving method:	budget
Similar problems:	budget-type problems with additional constraints.



How The Model Works

The first thing we need to do is choose a solving method that tells Evolver what to do with the variables. See Chapter 5: Complete Reference for descriptions of the different solving methods.

This is basically a budget-type problem with the additional constraint that TV spending must be more than radio spending.

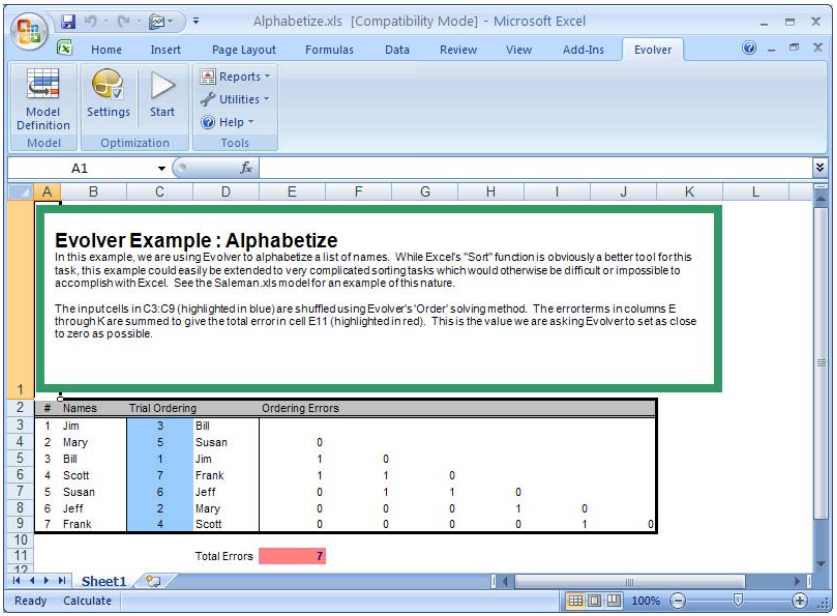
How To Solve It

The variables to be adjusted by Evolver are in cells C5:C9. We will ask Evolver to juggle them using the "budget" method, to allow each variable to be an independent value. The total audience is calculated with the SUM function in cell G13; this is the cell we will ask Evolver to maximize. The hard constraints specify that TV spending must be more than radio spending.

Alphabetize

This is a list of seven names which we would like Evolver to alphabetize. Although this example is simple, Evolver could handle complex sorts where data was interdependent, or names were weighted more heavily based upon other information in the model.

Example file:	Alphabetize.xls
Goal:	Alphabetize the list of names.
Solving method:	order
Similar problems:	Any sorting problem that is beyond the capability of Excel.

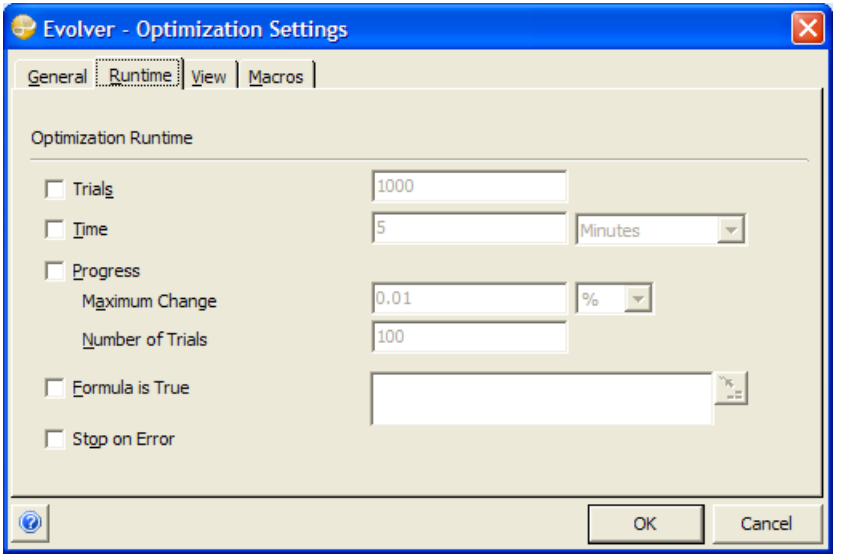


How The Model Works

The "Alphabetize.xls" file is a very simple model which illustrates Evolver's sorting possibilities. Column B contains the first names of seven people, and column A contains the corresponding "ID"" number for each person. Column D uses the VLOOKUP function in Excel to translate whatever number is chosen in Column C into its corresponding name. Cells E4:E9 use a simple penalty function which assigns a value of 1 each time an earlier name gets listed after a later name. The sum of all these errors is in cell E11, our target cell.

How To Solve It

In this model, the variables to be adjusted are located in column C (C3:C9). We will ask Evolver to juggle cells C3:C9 using the "order" solving method. The "order" solving method tells Evolver to rearrange the order of the selected values, trying different permutations of those variables rather than trying out new values. We will ask Evolver to find the value closest to 0 for the total error in cell E11, because when this target cell hits 0, that means that all the names are in the correct order.



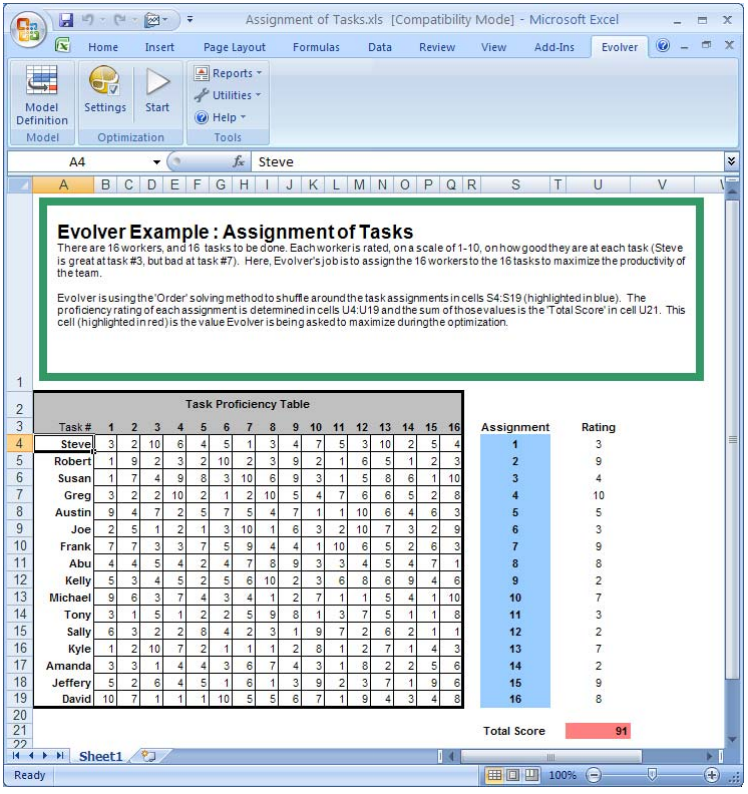
By not selecting any stopping criteria in the Evolver Options dialog, you are telling Evolver to keep working forever until it is manually stopped by clicking the "stop" button on the Evolver toolbar. But in this model we have selected the "value closest to" option, so Evolver will automatically stop if it finds a solution that meets your "value closest to" value of 0.

We are using a smaller population size because although there are no fast rules about choosing an optimal population size, generally, we can select a smaller population size when working with problems that have a smaller number of total possible solutions, so we focus more quickly on breeding the top performing solutions. In this problem, there are only 5040 possible orders of the 7 names.

Assignment of Tasks

This example models a common problem involving resource allocation. In this problem, a manager has 16 workers to perform 16 tasks. Each worker's ability to perform each task has been rated on a scale of 1 to 10 (1= cannot do the task, 10= perfect at the task). The challenge here is to match each worker to a task so that the overall productivity of the workers is maximized.

Example file:	Assignment of Tasks.xls
Goal:	Assign 16 workers to 16 tasks so the overall efficiency is maximized.
Solving method:	order
Similar problems:	assignment problems, scheduling meetings at times when the most workers would be happiest to meet, finding the best machines for a series of jobs.



The model provides a 16 by 16 grid in cells B4:Q19 where each worker has been rated for each task. The "chosen task" column (column S) to the right of the grid arbitrarily assigns each worker to one task. The next column over (column U) checks what task was assigned, and enters each worker's rating for that task. Finally, the total score of the entire solution (in cell U21) is the sum of adding up all the individual ratings.

How The Model Works

There is only one person for each task, so no numbers can be duplicates, and each number must be used once. Each worker's rating at that task is recorded in column U using the INDEX() function. These scores are summed in cell U21 to figure out the total score for that set of assignments.

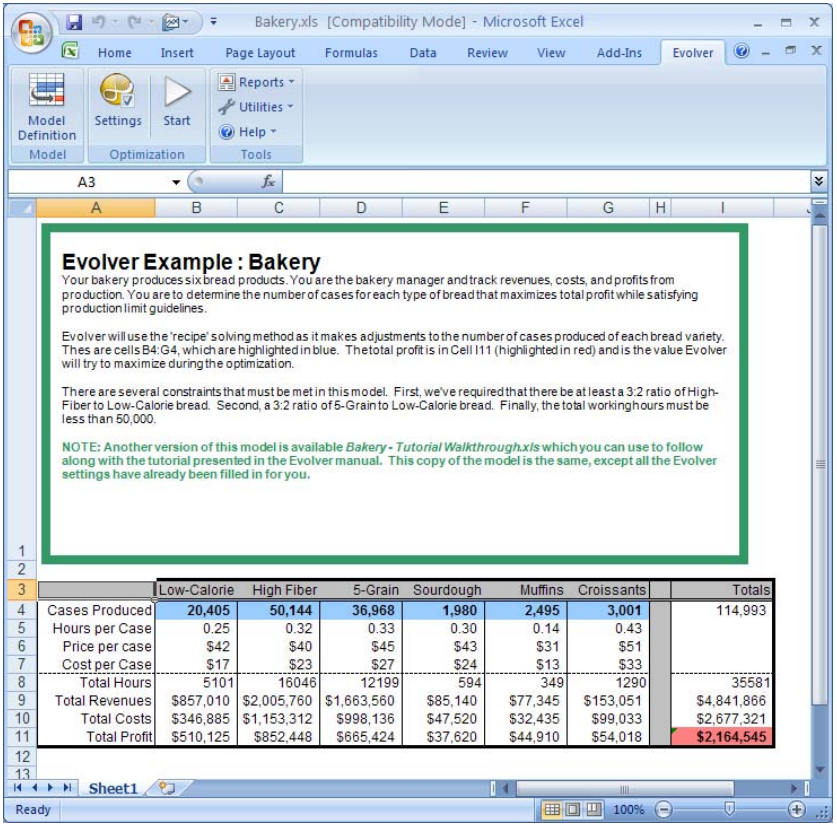
How To Solve It

Evolver is asked to juggle the "chosen task" variables, located in column S (S4:S19). We will ask Evolver to juggle these cells using the "order" solving method. This method will shuffle the existing values in those cells around, so be sure that there is only one of each value represented before you begin the optimization. We will ask Evolver to find the maximum value for cell U21, the target cell, because the higher this cell gets, the better the overall assignment.

Bakery

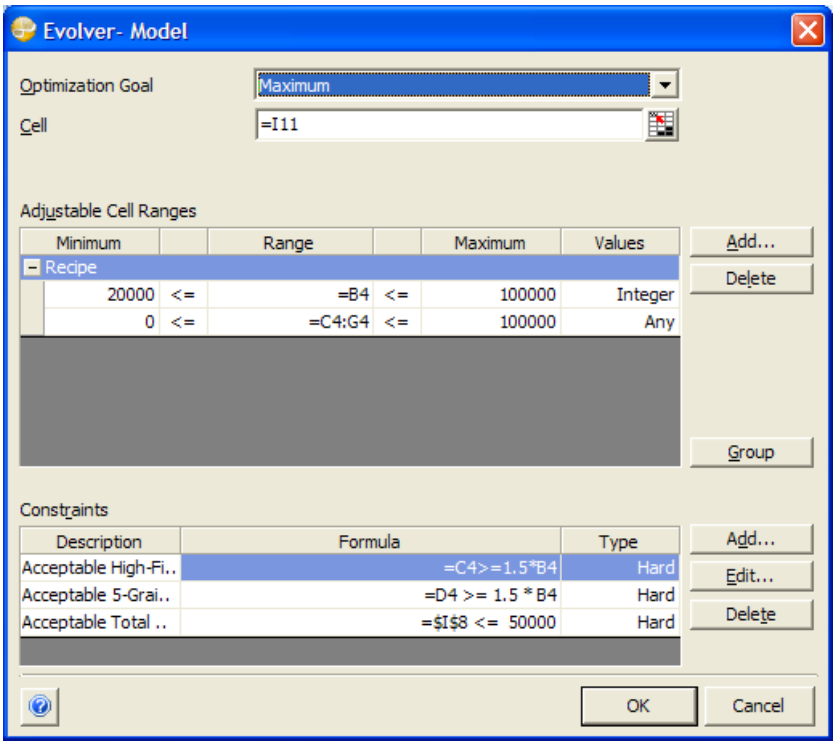
This example illustrates a common problem in production decision problems, where finding the right amount of each product to produce becomes very difficult... even with only a few items. A bakery owner must determine the number of cases to produce for each kind of bread, in order to maximize the total profit of the bakery. Be sure to also observe the limitations outlined, such as the total number of employee hours, and the correct ratios of products to be produced. (Note: this model is covered in detail in *Chapter 3: Evolver Step-by-Step*)

Example file:	Bakery.xls
Goal:	Find the optimal amount of each kind of bread to bake to satisfy all quotas and maximize profits.
Solving method:	recipe
Similar problems:	developing portfolios, manufacturing planning



How The Model Works

This problem lists the amount of each bread product to be produced across the top of the chart in row 4. When we adjust these quantity variables (B4:G4), the model computes the hours and costs it would take, as well as the profit that would be generated from baking that amount. The profit (in cells B11:G11) are added together in cell I11, which becomes the target cell to maximize.



The model also has three constraints. Each constraint listed is a hard constraint. One is a Simple Range of Values format constraint and two are constraints entered as Excel formulas.

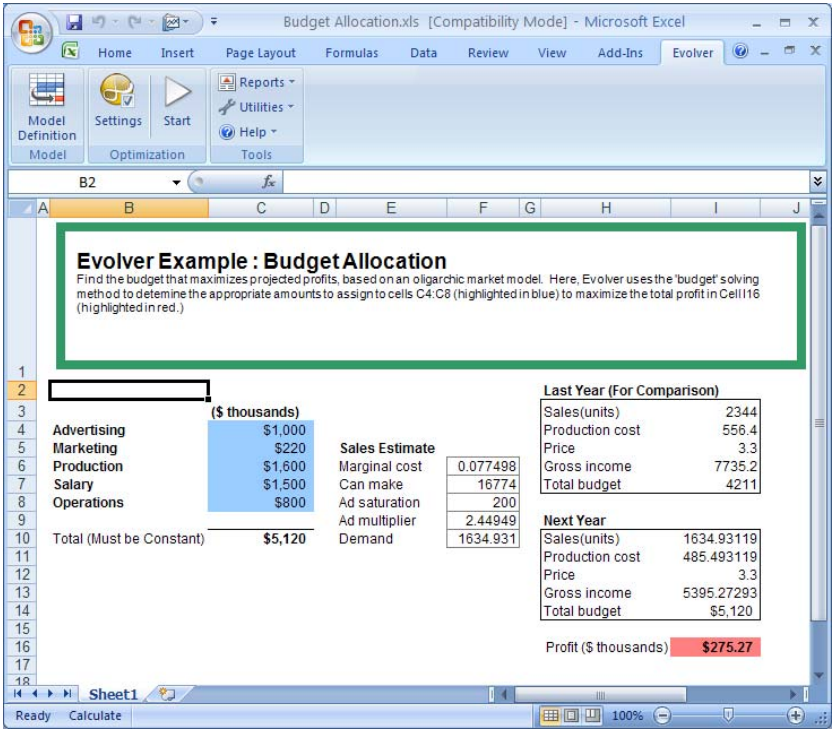
How To Solve It

Evolver is asked to find the values for cells B4:G4 (the amounts to make) that will maximize the value in cell I11 (the total profit). Since each value it finds can be independent of the others, we will use the "recipe" solving method. We will also ask Evolver to observe the constraints for cells C4, D4 and I8.

Budget Allocation

A senior executive wants to find the most effective way to distribute funds among the various departments of the company to maximize profit. Below is a model of a business and its projected profit for the next year. The model estimates next year's profit by examining the annual budget and making assumptions about, for example, how advertising affects sales. This is a simple model, but it illustrates how you can set up any model and use Evolver to feed inputs into it to find the best output.

Example file:	Budget Allocation.xls
Goal:	Allocate the annual budget among five departments to maximize next year's profits.
Solving method:	budget
Similar problems:	Allocate any scarce resource (such as labor, money, gas, time) to entities that can use them in different ways or with different efficiencies.



How The Model Works

The file "Budget Allocation.xls" models the effects of a company's budget on its future sales and profit. Cells C4:C8 (the variables) contain the amounts to be spent on each of the five departments. These values total the amount in cell C10, the total annual budget for the company. This budget is set by the company and is unchangeable.

Cells F6:F10 compute an estimate of the demand for the company's product next year, based on the advertising and marketing budgets. The amount of actual sales is the minimum of the calculated demand and the supply. The supply is dependent upon the money allocated to the production and operations departments.

How To Solve It

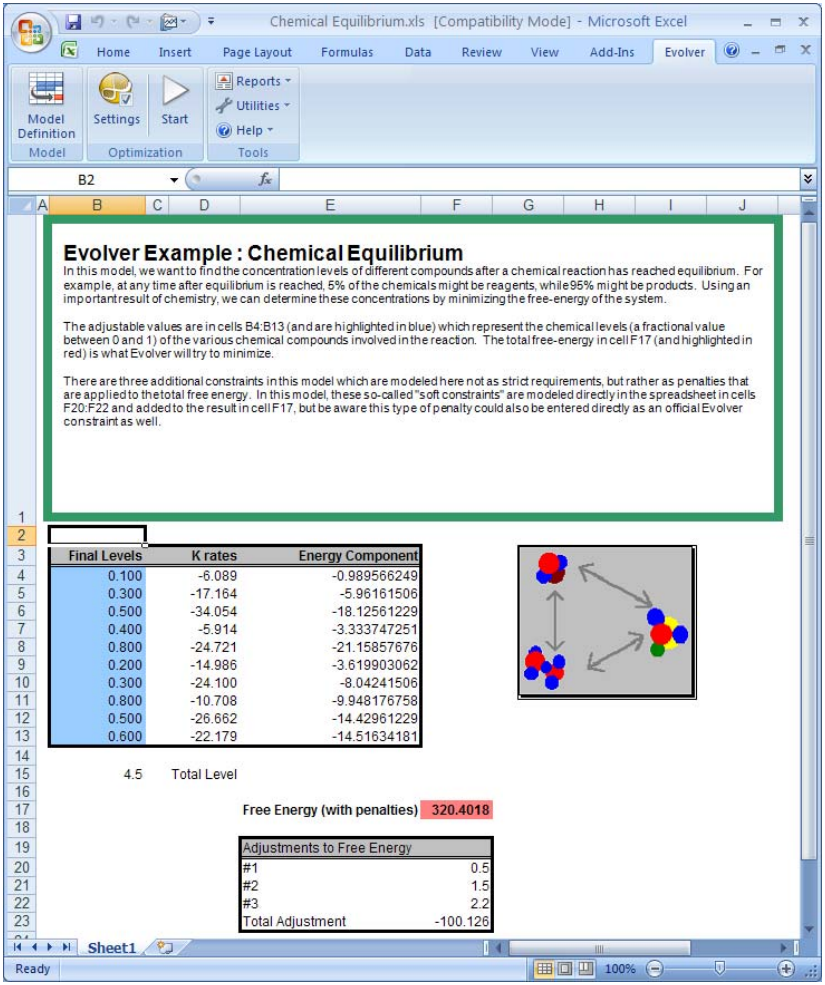
Maximize the profit in cell I16 by using the "budget" solving method to adjust the values in cells C4:C8. Set the independent ranges for each of the adjustable cells for the budget for each department, to keep Evolver from trying negative numbers, or numbers which would not make suitable solutions (e.g., all advertising and no production) for the departmental budget.

The "budget" solving method works like the "recipe" solving method, in that it is trying to find the right "mix" of the chosen variables. When you use the budget method, however, you add the constraint that all variables must sum up to the same number as they did before Evolver started optimizing.

Chemical Equilibrium

Any process which can be modeled to produce a result given some initial conditions can be optimized by Evolver. This example shows how Evolver can find levels of different chemicals (products and reactants) that minimizes the free energy after a reaction has reached equilibrium. In complicated chemical processes the ingredients (reagents) and the products continually re-form into one another until the concentration of the compounds becomes constant; when "equilibrium" is reached. At any time after equilibrium is reached, a steady percentage of the equilibrium chemicals might be reagents (e.g. 5%), and a steady percentage would be products (95%).

Example file:	Chemical Equilibrium.xls
Goal:	Compute the free energy of the reaction environment and find the levels for the chemicals, subject to the soft constraints (some chemical levels are proportional to others).
Solving method:	recipe
Similar problems:	determining conditions of the most stable market equilibrium.



How The Model Works

The variables of this problem in cells B4:B13 are the chemical levels to be mixed. Cell B15 calculates the total amount, which must be kept within a given range, according to the penalties.

Constraints in F20:F22 are soft constraints, meaning that we will not force Evolver to only accept valid solutions, but instead we will calculate penalties if certain chemicals are out of the desired proportion to other chemicals. These soft constraints use penalty functions built directly in the worksheet model. The penalties are added to the total free energy cell in F17, so when Evolver is minimizing the target, it will be looking for solutions that do not produce the penalties.

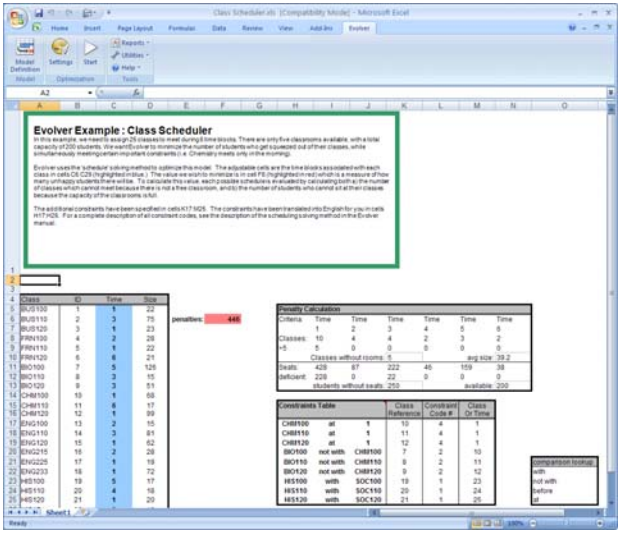
How To Solve It

Use the recipe solving method for cells B4:B13. Minimize cell F17.

Class Scheduler

A university must assign 25 different classes to 6 pre-defined time blocks. Each class lasts exactly one time block. Normally, this would allow us to treat the problem with the "grouping" solving method. However, there are a number of constraints that must be met while the classes are being scheduled. For example, biology and chemistry should not occur at the same time so that pre-medical students can take both classes in the same semester. To meet such constraints, we use the "schedule" solving method instead. The "schedule" solving method is like the "grouping" method, only with the constraint that certain tasks must (or must not) occur before (or after or during) other tasks.

Example file:	Class Scheduler.xls
Goal:	Assign 25 classes to 6 time periods to minimize the number of students who get squeezed out of their classes. Meet a number of constraints regarding which classes can meet when.
Solving method:	schedule
Similar problems:	Any scheduling problem where all tasks are the same length and can be assigned to any of a number of discrete time blocks. Also, any grouping problem where constraints exist as to which groups certain items can be assigned.



How The Model Works

The "Class Scheduler.xls" file contains a model of a typical scheduling problem where many constraints must be met. Cells C5:C29 assign the 25 classes to the 6 time blocks. There are only five classrooms available, so assigning more than five classes to one time block means that at least one of the classes cannot meet.

Cells K17:M25 contain the constraints; to the left of the constraints are English descriptions of the constraints. You can use either the number code or the english description as the constraint. The list of constraint codes for scheduling problems can be found in more detail in the "Solving Methods" section of Chapter 5: Complete Reference.

Each possible schedule is evaluated by calculating both a) the number of classes which cannot meet, and b) the number of students who cannot sit at their classes because the capacity of the classrooms is full. This last constraint keeps Evolver from scheduling all the large classes at the same time. If only one or two large classes meet during a time block, the larger classrooms can be used for them.

Cells I8:N8 uses the DCOUNT Excel function to count up how many classes are assigned to each time block. Right below cells I9:N9 then compute how many classes did not get assigned a room for that time block. All the classes that are without rooms are totaled in cell K10.

If the number of seats required by a given class exceeds the number of seats available, cells I12:N12 calculate by how much, and the total number of students without seats is calculated in cell K13. In cell F6, this total number of students without seats is added to the average class size, and multiplied by the number of classes without rooms. This way, we have one cell which combines all penalties such that a lower number in this cell always indicates a better schedule.

How To Solve It

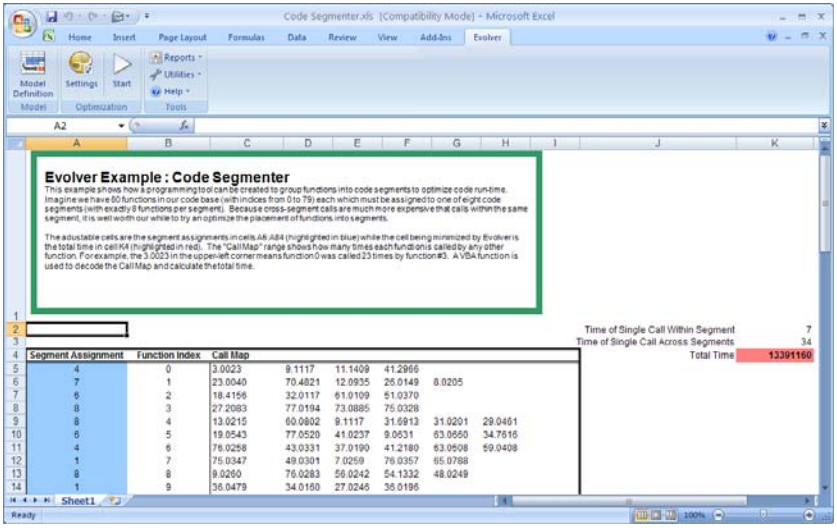
Minimize the value of the penalties in F6 by changing cells C5:C29. Use the "schedule" solving method. When this solving method is chosen, you will see a number of related options appear in the lower "options" section of the dialog box. Set the number of time blocks to 6, and set the constraints cells to K17:M25.

Code Segmenter

A Windows programmer wants to break a program up into several code segments, so that Windows can use memory more efficiently by only keeping in memory the code segments currently being used.

This is an example of collecting similar items into groups. The items can interact efficiently with others in the same group, but it is difficult for items in different groups to interact. When there are natural barriers to letting every item interact directly with every other (say all computer users wanted to be directly connected to one printer), it is necessary to break the items up into groups. An efficient grouping can have a significant effect on the overall productivity of the system.

Example file:	Code Segmenter.xls
Goal:	Group program routines into eight different code segments so that the program executes as quickly as possible.
Solving method:	grouping
Similar problems:	Collect workstations into LAN clusters, or circuits into areas on microchips, so the cost of the communication between groups is minimized.



How The Model Works

Windows programmers often break programs up in this way to increase program efficiency. When a routine in another segment needs to run, Windows will throw out the calling segment and read in the called segment from the disk. If a 2 Mb program is broken up into

80 segments of 20 Kb each, the program can run if only 20 Kb of memory is available. In order to run with acceptable performance, however, the code segments must be carefully organized. Calling a function in another segment takes more time than calling one in the same segment as the caller. Minimizing the number of cross-segment calls is referred to as the code segmentation problem.

Since it is possible to optimize some parts of an application at the expense of the whole application, we will use Evolver to perform a global optimization.

The "Code Segmenter.xls" example file assumes that an application has been compiled with a certain segmentation. The application was run just like a user would run it, while a performance tracing routine kept track of the number of times each function called every other function. These results thus represent the nature of calls in the typical usage of the application. From them we can make predictions about the speed of the application with different segmentation strategies.

This worksheet uses the custom "SegCost" function. SegCost computes the time it would take the user to run the program the same way as when the typical usage statistics were acquired. It does this by counting the number of inter- and intra-segment calls, and multiplying each by the cost of each kind of call. Here we assume an inter-segment call (or near call) takes seven clock cycles, and an intra-segment call (or far call) takes 34 cycles, which is the case for any 386 computer.

The SegCost function is written as an Excel VBA macro, as shown here:

```
Function segCost(segs, calls, inP, outP) As Double

Dim inCost#, outCost#, total#, temp#, tempPtr#
Dim i%, j%, wide%, funcNumber%, ThisSeg%, OtherSeg%
Dim NumCalls%, NumInCall%, NumOutCall%,
SegOrder$, CallOrder$

SegOrder = Application.Names("segs").RefersTo
CallOrder = Application.Names("calls").RefersTo
NumInCall = 0
NumOutCall = 0
inCost = Range("k2")
outCost = Range("k3")
total = 0
wide = Range(CallOrder).Columns.Count
For i = 1 To Range(SegOrder).Rows.Count
  ThisSeg = Range(SegOrder).Rows(i)
  For j = 1 To wide
    temp = Range(CallOrder).Rows(i).Columns(j)
```

```
      If temp <> 0 Then
        funcNumber = Int(temp)
        OtherSeg = Range(SegOrder).Rows(funcNumber + 1)
        NumCalls = 10000 * (temp - funcNumber)
        If ThisSeg = OtherSeg Then
          temp = NumCalls * inCost
          NumInCall = NumInCall + 1
        Else
          temp = NumCalls * outCost
          NumOutCall = NumOutCall + 1
        End If
        total = total + temp
      End If
    Next
  Next
  segCost = total
End Function
```

The sample application has 80 functions. The number of times each function calls each other is stored in the "calls" range (C5:I104). We could create a 80 by 80 matrix to represent the calling pattern, but this *n* by *n* approach would become unusable after about 250 functions, because Excel has a limit of 256 columns (and because the approach would need an exponential amount of memory).

Instead, we use a condensed notation to represent the calling pattern. We first assume that no function calls more than a certain number of other functions. In the example file, we assume seven is the upper limit; that is why the calls range is seven columns wide, but this limit is arbitrary. We also assume that no function is called by any other function more than 9999 times.

Let us look at function 1, starting at cell C5. Function 1 calls four functions: 3, 9, 81, and 41. C5:I5, the first row in calls, contains one real number for each function called (e.g. 3.0023). The integer portion (e.g. 3) represents the function that is called, and the fraction multiplied by 10,000 (e.g. .0023 x 10,000 = 23) represents the number of times function 1 called function 3 in the typical usage of the application. Thus, 9.1117 means that the function called function #9 1,117 times, and so on. This concise format lets us save memory and make the best use of the limited number of columns available in Excel.

Cell A5:A104 (the "segs" range) contains the number of the segment each function is assigned to. Cell K4 calls "SegCost" to compute the overall performance of the current segmentation strategy.

How To Solve It

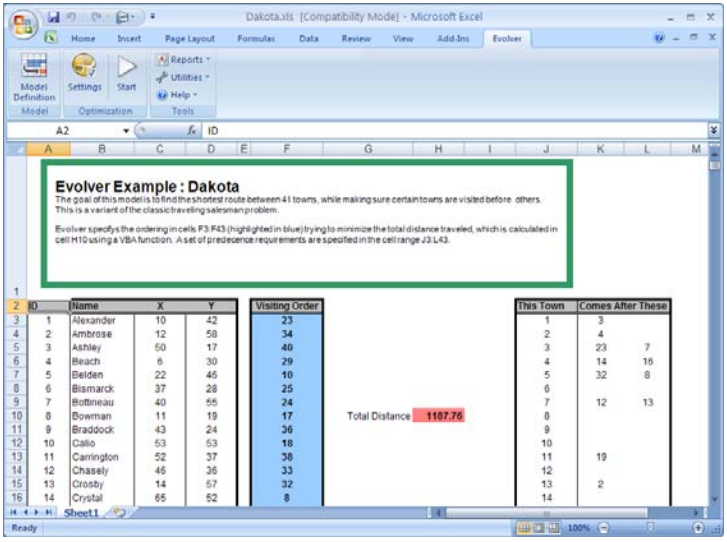
Minimize the value in cell K4 by adjusting the cells in A5:A104. Use the "grouping" method. The "grouping" solving method tells Evolver to arrange variables into *x* groups, where *x* is the number of different values in the adjustable cells at the start of an optimization.

Dakota: Routing With Constraints

A real-estate firm needs to assess each of its properties throughout North Dakota in a certain order, so that certain properties are visited before others. Similar to the classic traveling salesman problem, the goal of this problem is to find the shortest route among a set of cities that ensures that each city is visited once. However, here we add the constraint that certain cities must be visited before certain other cities (such as town #2 coming after town #4). This means that instead of the "order" solving method we will use the "project" solving method.

A project is an ordering for a set of tasks where certain tasks must precede other tasks. You could use the "project" solving method, in conjunction with your own custom functions, to find the best timing for a project (based on a combination of any number of criteria, such as time to finish, resource utilization, etc.).

Example file:	Dakota.xls
Goal:	Plan a route among 41 towns in North Dakota which finds the shortest route between all cities while making sure some cities are visited before others.
Solving method:	project
Similar problems:	Re-schedule a project to balance resource utilization. Schedule the flow of jobs in a machine shop to reduce total time while ensuring that some jobs are done before others.



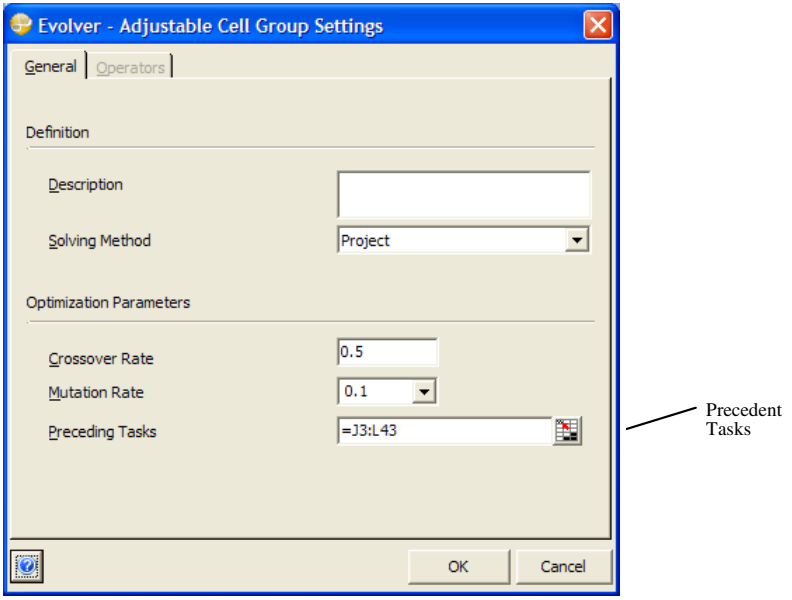
How The Model Works

Cells F3:F43 contain the order in which the cities will be visited. Cell H10 calculates the total length of the route, based on the order and the x,y locations of the cities (in C3:D43). Cell H10 uses the custom function "BigRouteLength" to speed up the computation of the total route length.

Cells J3:L43 contain the precedence tasks. This is a table showing which cities (tasks) must be preceded by other cities. Eight cities (1,2,3,4,5,7, 11 and 13) must have certain cities that are visited before them.

How To Solve It

Minimize the route length in H10 by changing the cells F3:F43. Use the "project" solving method and set the precedence tasks to J3:L43. These precedents are set in the Preceding Tasks field of the Adjustable Cell Group Settings Dialog:

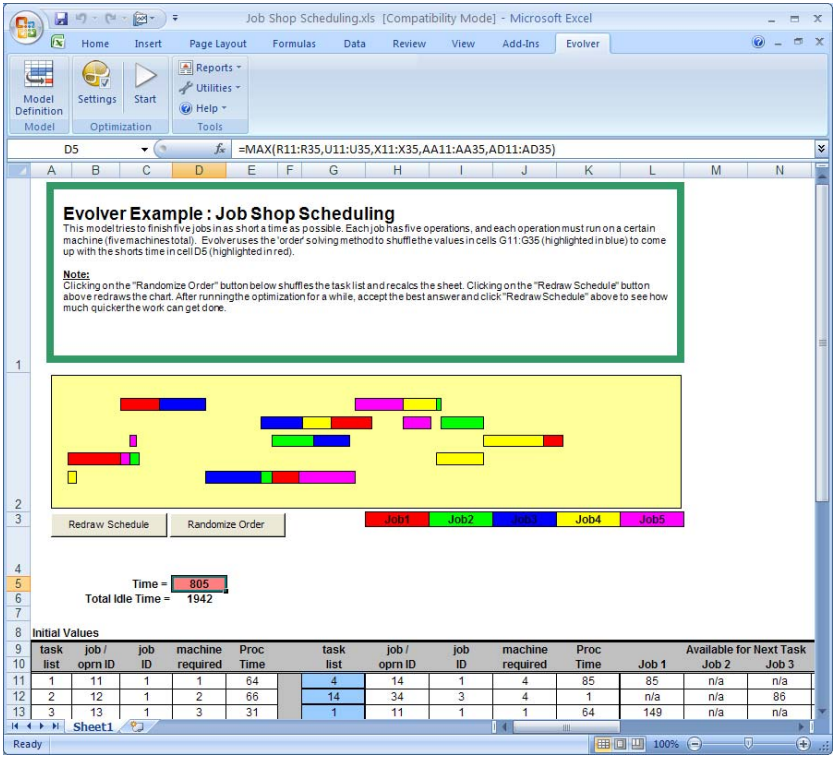


Job Shop Scheduling

A metalworking shop needs to find the best way to schedule a set of jobs that can be broken down into steps that can be run on different machines. Each job is composed of five tasks, and the tasks must be completed in order. Each task must be done on a specific machine, and takes a specific amount of time to complete. There are five jobs and five machines.

Clicking the Draw Schedule button at the top of the sheet will redraw the bar chart to show when each of the job tasks is scheduled to run.

Example file:	Job Shop Scheduling.xls
Goal:	Assign job pieces (tasks) to machines so total time for all jobs to finish is minimized.
Solving method:	order
Similar problems:	Scheduling or project-management problems



How The Model Works

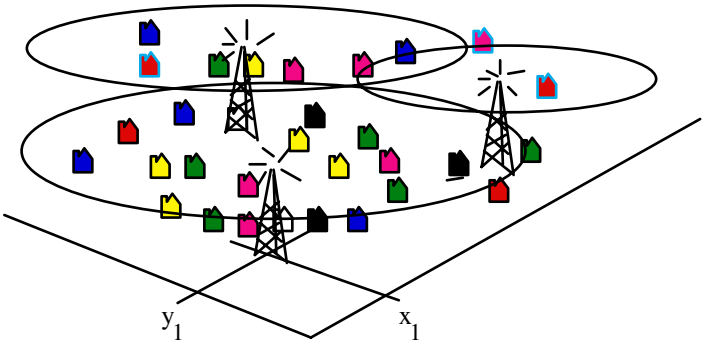
Cell D5 computes the makespan, or how much time elapses between the start of the first scheduled task and the end of the last scheduled task. This total time is what we wish to minimize. Cells G11:G35 hold the variables (the tasks) to be shuffled to find the best assignment order. The equations on the sheet figure out how soon each task can run on the machine that it needs.

How To Solve It

Select a set of adjustable cells G11:G35 and select the order solving method. Minimize cell D5.

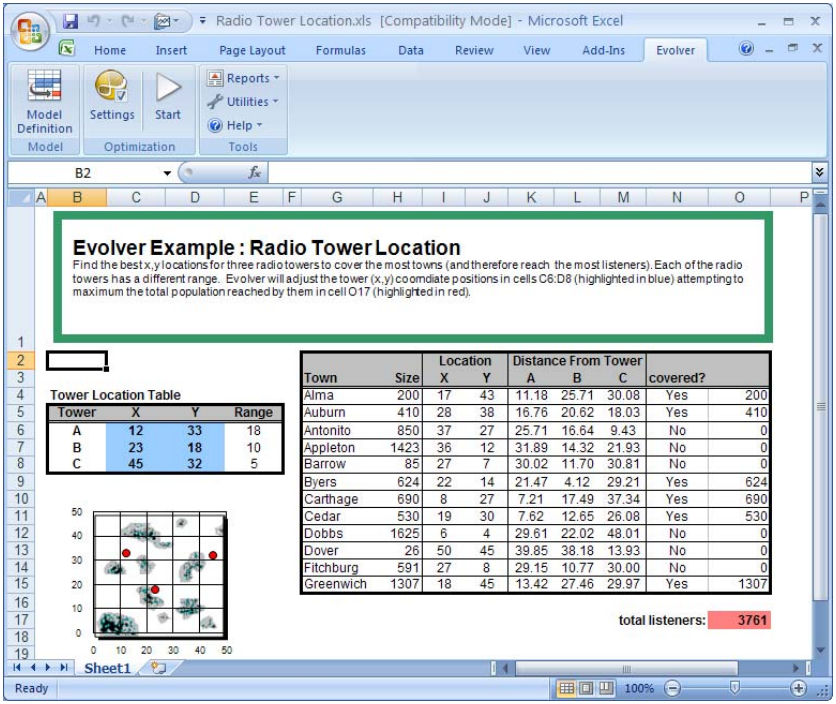
Radio Tower Location

A radio network wants to build three radio towers in a region that has twelve major communities. Each community has a different population size, and each radio tower has a different strength broadcast range. The goal is to place the towers so that the maximum number of potential listeners fall inside the broadcast radii of the towers.



A more complicated example of a location problem might be to locate several factories so that they are a) in the vicinity of both vendors and customers, b) in affordable, open land, and c) near a large, technically trained work force. Any number of additional influences on the best locations, such as tax incentives, can also be added to such a model. Evolver can then find the best locations in x,y or even x,y,z coordinate space.

Example file:	Radio Tower Location.xls
The Goal:	Find the best x,y coordinates for three radio towers so that the maximum potential listening population falls inside their broadcasting range.
Solving method:	recipe
Similar problems:	Find sites for warehouses that minimize the shipping necessary between warehouses and stores. Locate fire stations so that populations are best covered with a limited number of stations, including factors such as housing density.



How The Model Works

The file "Radio Tower Location.xls" models a two-dimensional landscape where the placement of five radio towers determines how many listeners are reached. Cells C6:D8 contain the x,y coordinates for the three towers. The illustration in the model consists of two elements: one is a bitmap picture of the population densities (in green) pasted from the Windows Paintbrush program; the other is an Excel scatter graph that re-calculates automatically to show the locations of the towers.

Ten communities are represented as single-point locations. The Excel model computes the distance between the communities and the towers in K4:M15 to determine if each community is covered (yes) or not covered (no). The total population of all the covered communities (the number we want to maximize) is calculated in cell O17.

How To Solve It

Maximize the population reached in cell O17 by adjusting the tower location cells C6:D8. Use the "recipe" solving method and set the ranges for the variables from 0 to 50 (the limits of our location area).

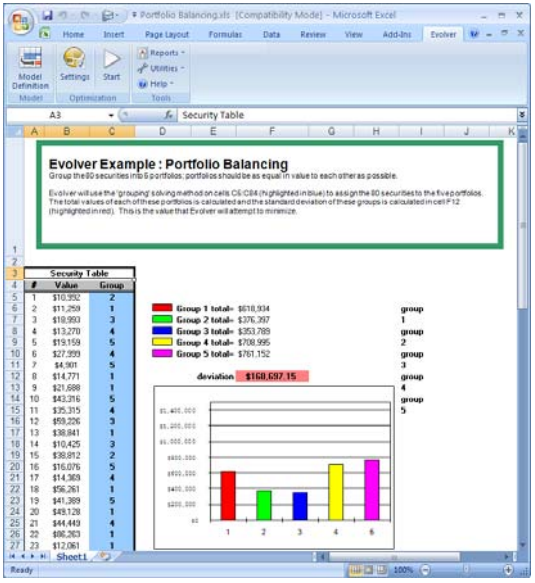
The "recipe" solving method tells Evolver to adjust the variables chosen in any way it sees fit. As is the case with a recipe for baking, we are trying to find the right mix of "ingredients" (x,y coordinates) to produce the optimum solution.

Portfolio Balancing

A broker has a list of 80 securities, each worth a different amount of money. The broker wants to group these securities into five packages (portfolios) that are as close to each other in total value as possible.

This is an example of a general class of problems called bin packing problems. Packing the holds of a cargo ship, so that each hold weighs as much as the others is another example. If there are millions of small items to be packaged into a few groups, such as grains of wheat into ship holds, a roughly equal distribution can be guessed at without a big difference in weight. However, several dozen packages of different weights and/or sizes can be packed in very different ways, and efficient packing can improve the balance that would be found manually.

Example file:	Portfolio Balancing.xls
Goal:	Break a list of securities up into five different portfolios whose total values are as close as possible to each other.
Solving method:	grouping
Similar problems:	Create teams that have roughly equivalent collective skills. Pack containers into holds of a ship so that the weight is evenly distributed.



How The Model Works

The "Portfolio Balancing.xls" file models a typical grouping assignment. Column A contains identification numbers to specific securities, and column B contains the dollar value of each security. Column C assigns each security to one of the five portfolios. When setting a grouping or bin packing type of problem and using the grouping solving method, you must be sure that before you start Evolver each group (1-5) is represented in the current scenario at least once.

Cells F6:F10 calculate the total value of each of the five portfolios. This is done with database criteria offscreen (in column I) and "DSUM()" formulas in cells F6:F10. Thus, cell F6, for example, calculates DSUM of all the values in column B that have been assigned to group *5* (in column C).

Cell F12 computes the standard deviation among the total portfolio values using the "STDEV()" function. This provides a measure of how close in total value to each other the portfolios are. The graph shows the total value of each portfolio, with a reference line drawn at the goal number where each portfolio would be if they were all even.

How To Solve It

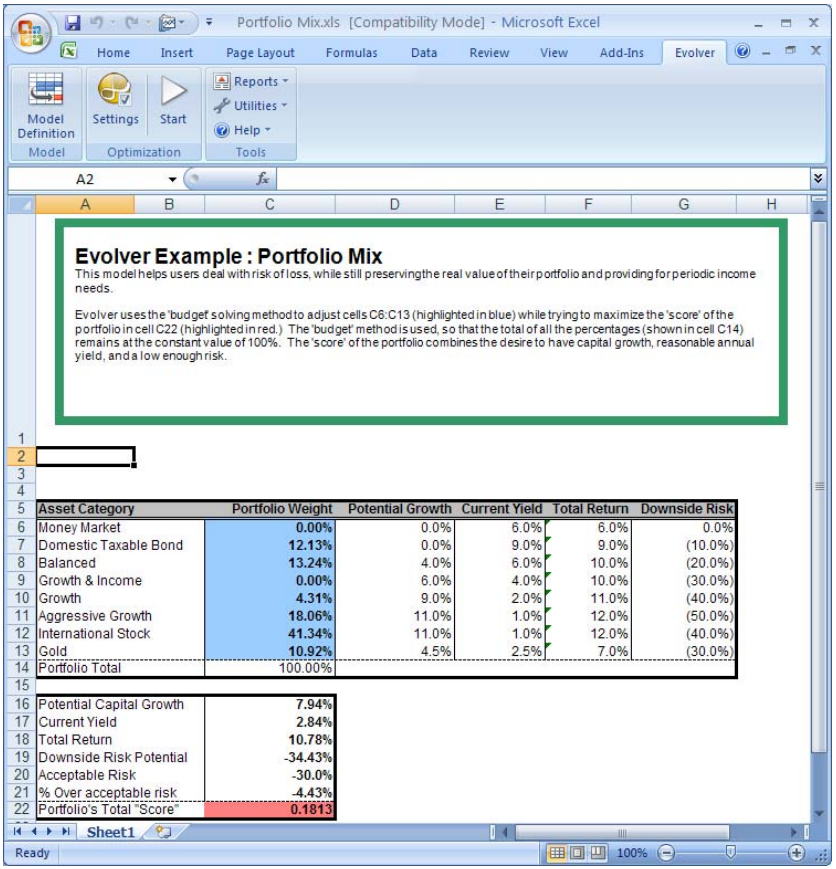
Minimize the value in cell F12 by adjusting the cells in C5:C104. Use the "grouping" method and make sure the values 1, 2, 3, 4, and 5 each appear at least once in column C.

The "grouping" solving method tells Evolver to arrange variables into *x* groups, where *x* is the number of different values in the adjustable cells at the start of an optimization.

Portfolio Mix

A young couple has assets in many different types of investments, each with its own yield, potential growth, and risk. By combining several formulas which multiply various weights, they have customized a sort of "score" which shows how well any particular mix of investments satisfies their needs.

Example file:	Portfolio Mix.xls
The Goal:	Find the optimal mix of investments to maximize your profit, given your current risk/return needs.
Solving method:	budget



How The Model Works

This is a classic financial model which attempts to balance the risk of loss against the return on investment. Each asset listed in column A is assigned some weight in column C. The model multiplies the return

percentages by the weight each asset carries in the portfolio to yield a total return in cell C18. We also calculate a total risk number in cell C19, which should not be higher than the acceptable risk listed in cell D19.

How To Solve It

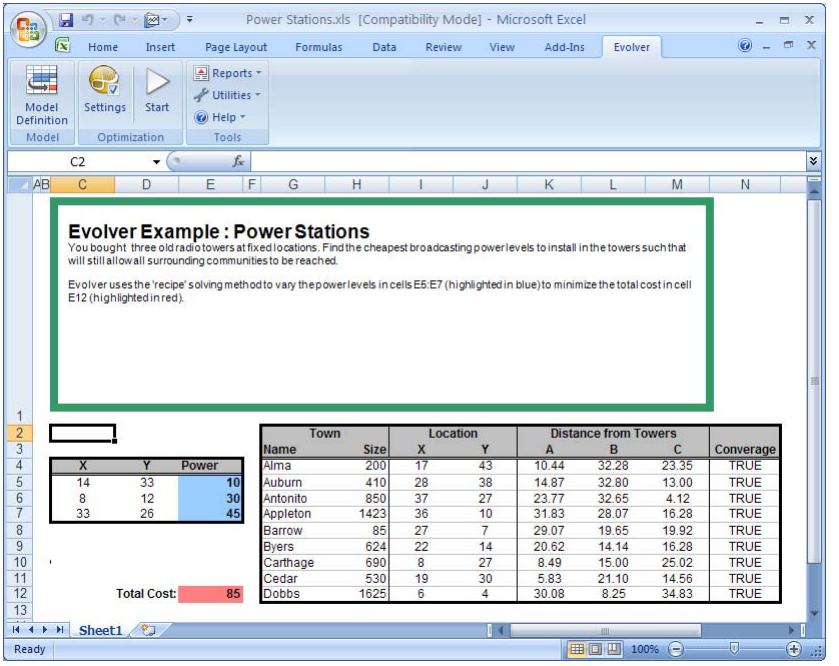
The total "score" in cell C22 reflects the total return minus a penalty for any risk above the acceptable percentage. We maximize this score.

Power Stations

A radio network buys three abandoned, non-working radio towers in a region that has ten major communities. The network wants to purchase brand new broadcast transmitters and install them in the towers to get them broadcasting again.

Because there is a limited budget, the goal is to spend the least amount of money on transmitters that will still cover all 9 surrounding communities. We assume a linear pricing model where the cost of a transmitter is directly related to its power, so we'll be looking for the lowest amount of power to purchase, but it would be just as easy to create a lookup chart of actual transmitter types and prices.

Example file:	Power Stations.xls
The Goal:	Find the smallest (cheapest) transmitter for each of the old towers that will still cover the entire ten surrounding communities.
Solving method:	recipe
Similar problems:	set-covering problems, where a bunch of elements need to be described by a small number of well-defined sets.



How The Model Works

This is very similar to the radio tower location example (Radio Tower Location.xls), except that here the locations are frozen, and it is the tower's power ranges in cells E5:E7 that are the variables to be adjusted. We add up the power cost of the three towers in cell E12, the target cell to be minimized.

Cells K4:M12 calculate how far away each community is from a tower, and column N returns a TRUE if a community is near enough to one of the transmitters to be covered. All of these constraints are checked in a single hard constraint named *All Areas Covered?*. This constraint has the formula AND($N$4:$N$12) which returns TRUE only if all values in column N are TRUE.

How To Solve It

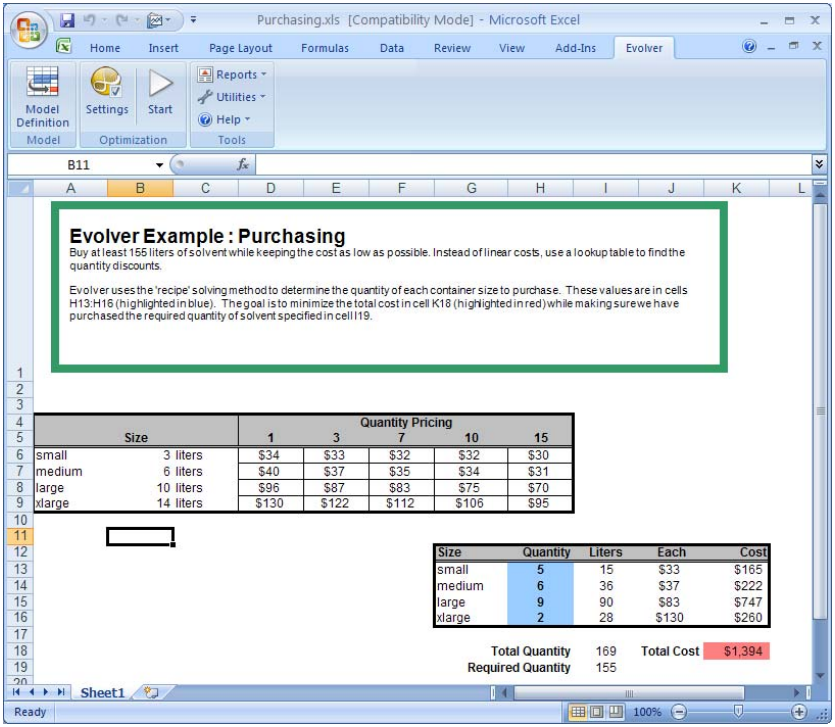
Minimize the power required in cell E12 by adjusting the radii of the towers in cells E5:E7. Use the "recipe" solving method and set the ranges for the variables from 0 to 100. The single hard constraint, entered using the Excel formula format, is described above.

Purchasing

Any time you have many possible ways to order items the quantity discounts make it difficult to determine the most cost effective way to buy the items. This model contains a simple price table, listing quantity discount prices for a special solvent. You must buy at least 155 liters of this solvent, which comes in small, medium, large and extra-large barrels.

Try to purchase the right number of each barrel size to minimize your cost.

Example file:	Purchasing.xls
The Goal:	Spend the least amount of money buying 155 liters of solvent.
Solving method:	recipe
Similar problems:	The opposite: create a pricing table that most consistently and fairly rewards higher quantity orders.



How The Model Works

This solvent comes in 3, 6, 10 and 14-liter barrels. The table of prices for each size is listed in cells D6:H9. Cells H13:H16 contain the amounts to buy of each size. Column K calculates the cost for each purchase, and cell K18 is the total cost. This model allows you to change the required amount to be purchased (cell I19) from 155 to whatever you wish. Cell I18 contains the total liters that were purchased, and so this cell must be at least the required number in cell I19 (155). The single hard constraint is that the amount purchased exceeds the amount required.

Since we need 155 liters, we might just think of buying 11 extra-large barrels (154 liters), plus one small barrel (3 liters) for a total of 157 liters. According to the price table, that would cost $1,200 total. But running the optimization will give you an even more cost-effective combination.

How To Solve It

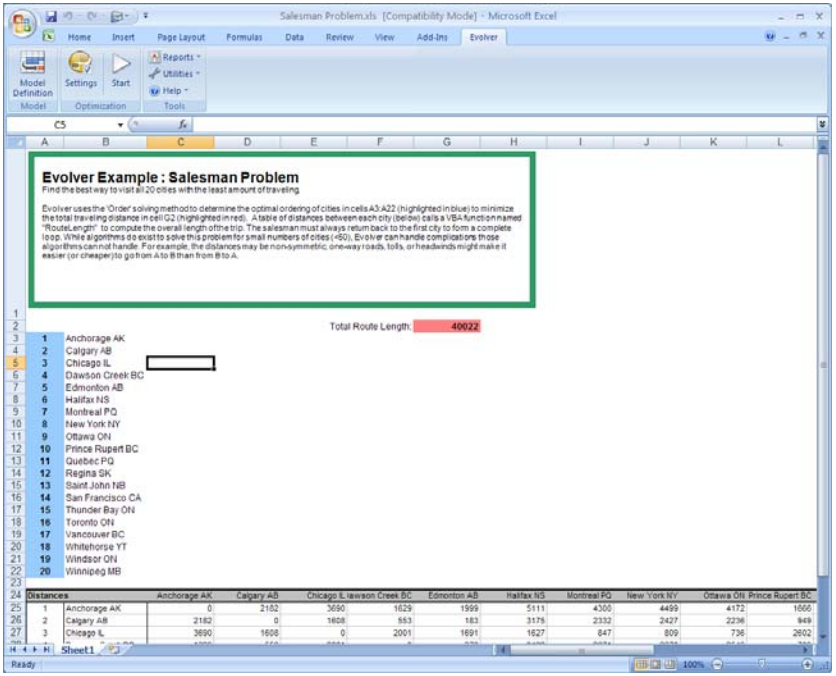
Minimize the cost in cell K18 by adjusting the quantities to buy in cell H13:H16. Use the recipe solving method to adjust values, and set the ranges of these variables to be between 1 and 20. You can not buy just a part of one barrel, so we will ask Evolver to try only integers by checking the "integers" option in the Adjustable Cells Dialog. Since we cannot purchase less than 155 liters, enter a single hard constraint specifying that I18>155.

Salesman Problem

A salesman is required to visit every city in the assigned territory once. What is the shortest route possible that visits every city? This is a classic optimization problem and one that is extremely difficult for conventional techniques to solve if there are a large (>50) number of cities involved.

A similar problem might be finding the best order to perform tasks in a factory. For example, it might be much easier to apply black paint after applying white paint than the other way around. In Evolver, these types of problems can be best solved by the ***order*** solving method.

Example file:	Salesman Problem.xls
Goal:	Find the shortest route among n cities that visits each city once.
Solving method:	order
Similar problems:	Plan the drilling of circuit board holes in the fastest way.



How The Model Works

The file "Salesman Problem.xls" calculates the route length of a trip to various cities by looking up the distances in a table. Column A contains identifying numbers for specific cities. Column B contains the names that those numbers represent (with a lookup function). The order in which the cities (and their numbers) appear from top to bottom represents the order in which the cities are visited. For example, if you entered a "9" into cell A3, then Ottawa would be the first city visited. If A4 contained "6" (Halifax), then Halifax would be the second city visited.

The distances between cities are represented in the table beginning at C25. The distances in the table are symmetric (distance from A to B is the same as from B to A). However, more realistic models may include non-symmetric distances to represent greater difficulty of traveling in one direction (because of tolls, available transportation, headwinds, slope, etc.).

A function now must be used to calculate the length of the route between these cities. The total route length will be stored in cell G2, the cell we wish to optimize. To do this, we use the "RouteLength" function. This is a custom VBA function in Salesman Problem.xls.

How To Solve It

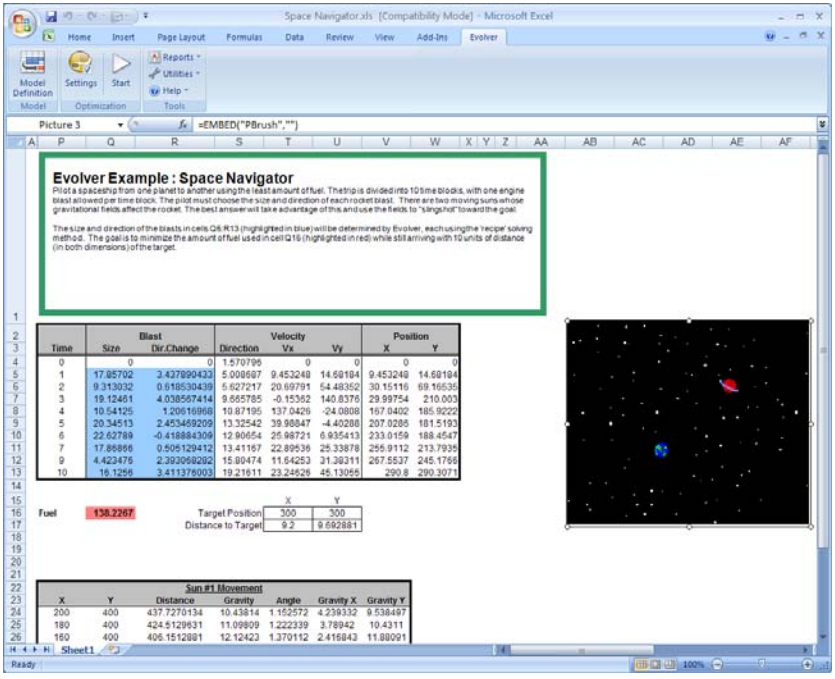
Minimize the value in cell G2 by adjusting the cells in A3:A22. Use the "order" method and make sure the values 1 through 20 exist in the adjustable cells (A3:A22) before you start optimizing.

The "order" solving method tells Evolver to rearrange the chosen variables, trying different permutations of existing variables.

Space Navigator

As the launching crew of the space shuttle "Evolver III", you must figure out the amount and direction of each rocket thrust to reach your destination using the least amount of fuel. The better solutions will probably exploit the gravitational "whip" effect of nearby suns to conserve fuel.

Example file:	Space Navigator.xls
Goal:	Get a spaceship to its destination using as little fuel as possible. Take advantage of the gravity of stars moving through your neighborhood.
Solving method:	recipe
Similar problems:	process control problems



How The Model Works

Cells Q5:R15 hold the blast size and direction values for each of ten time steps. Cell Q16, which we want to minimize, is simply the sum of all the fuel burned in the ten time steps (Q4:Q13).

The hard constraints are: a) that the ship's final position be within 10 horizontal units of its destination, and b) that it be within 10 vertical units.

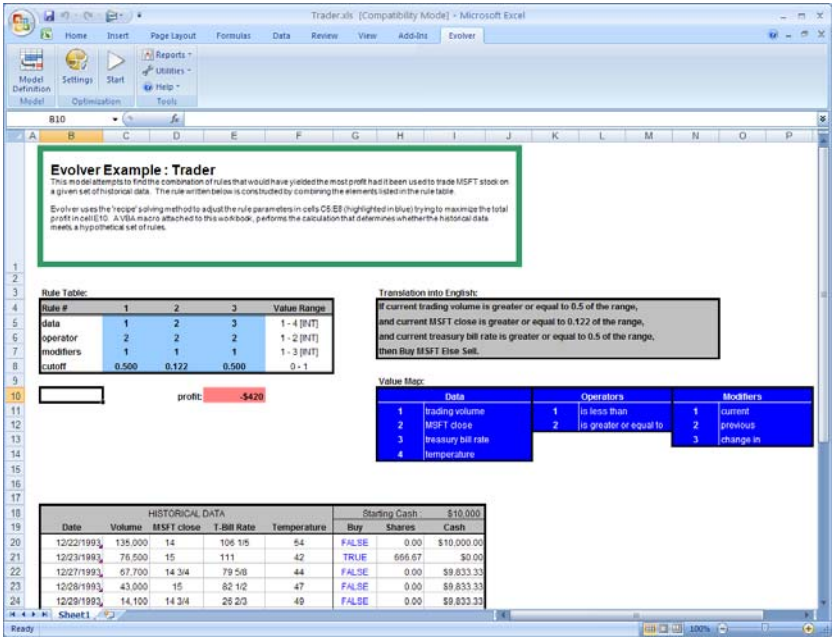
How To Solve It

Minimize cell Q16. Create an adjustable cells group that uses the recipe solving method using cells Q5:R13. The Blast cells (Q5:Q13) should range between 0 and 300 and the Direction cells (R5:R13) should range between -3 and 3, since it uses Radians to represent the direction of the blasts. One Radian is about 57 degrees.

Trader

You are trading on the S&P 500, and you have determined that technical analysis provides more accurate forecasting of stocks than traditional fundamental analysis, and can save you time once you build a system. It seems there are an infinite number of possible rules by which you could trade, but only a few of them would have made you a tidy profit if you had been following them. An intelligent computer search could help you determine what rules would have made the most money over a certain historical period.

Example file:	Trader.xls
Goal:	Find a set of three rules which would have yielded the highest return over a certain time period.
Solving method:	recipe
Similar problems:	find optimal moving averages that would have yielded the best result; any rule-finding or criteria-finding problems



How The Model Works

This model uses several adjustable cell groups to solve the overall problem. There are three rules that are evaluated for each trading day. If the conditions of all three rules are true, then the computer will buy on that day, otherwise it will sell. (A more realistic trading system would not just buy or sell, but also sometimes hold onto what it has.)

Each rule is described by a set of four numbers in cells C5:E8 which indicate several things: 1) which data source the rule refers to, 2) whether the data value should be above or below a cutoff value, 3) the cutoff value that determines if the rule is true, and 4.) a modifier value that determines if the value itself should be examined, or if the last day's value or the change since the last day should be examined.

The cutoff values range from 0 to 1, and represent the percentage of the data source's range. For example, if volume ranges from 5,000 to 10,000, then a cutoff value of 0.0 would match a volume of 5,000, a cutoff value of 1.0 would match a volume of 10,000, and a cutoff value of 0.5 would match a volume of 7,500. This system allows the rules to refer to any data source, regardless of the values it takes on.

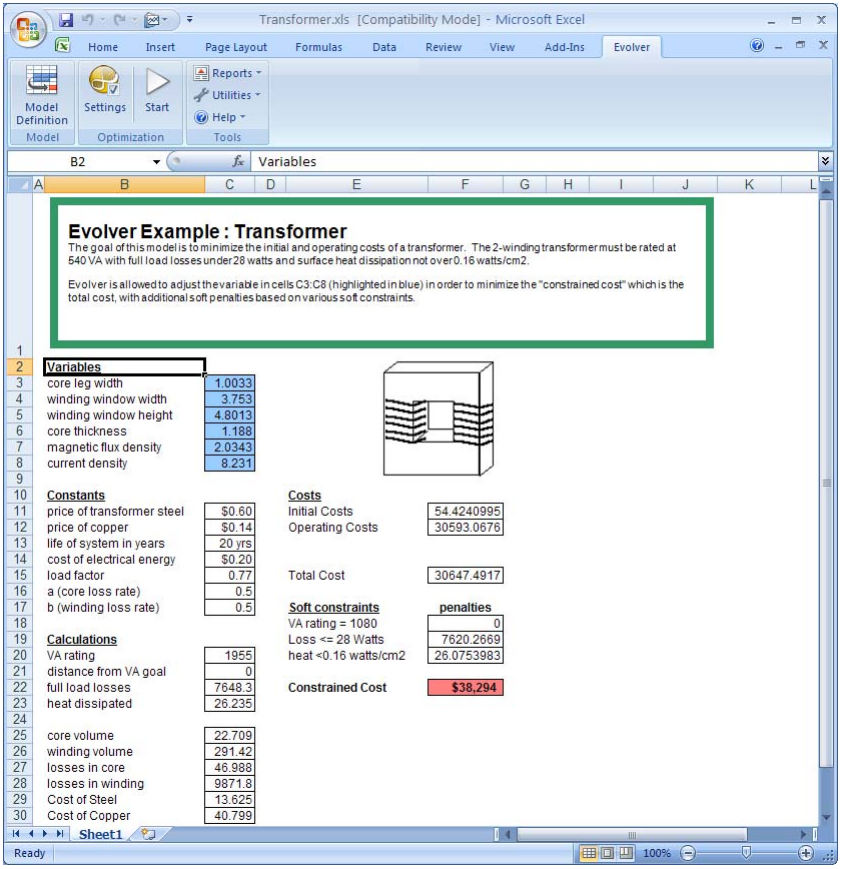
How To Solve It

Create adjustable cell groups, all using the "recipe" solving method. Each row in C5:E5, C6:E6, C7:E7, and C8:E8 should be created separately, so that each group can easily be assigned its own options such as integer and ranges. The settings for each set of variables are listed in F5:F8. Maximize on cell E10, which calls a macro to simulate trading with those rules. The total profit made after simulating trading on each day in the historical database is returned in cell E10.

Transformer

The 2-winding transformer must be rated at 1080 VA with full load losses under 28 watts and surface heat dissipation not over 0.16 watts/cm2. Minimize costs while observing the performance criteria.

Example file:	Transformer.xls
Goal:	Minimize the initial and operating cost of a transformer.
Solving method:	recipe
Similar problems:	circuit design, bridge design



How The Model Works

The rating, load loss, and heat dissipation constraints are coded as soft constraints. We create a soft constraint by penalizing those solutions which do not meet our requirements, and are invalid. Unlike a hard constraint which must be met, Evolver is allowed to try

out some invalid solutions, but because these invalid solutions are penalized by a function in your model which checks for violations, they will produce poor results in your target cell. Thus, over time, these invalid solutions will be discarded from the evolving population of possible solutions.

A soft-constraint model may work better than a hard-constraint, if the problem is less heavily constrained. It also allows Evolver to accept a really great solution even if it may fall a little short of the constraints, which could be more valuable than a not-so-great solution that meets all the constraints.

How To Solve It

Compute material cost (initial cost) and operating costs (cost of electricity * electricity wasted) in cells F11 and F12. Combine these with penalty functions set in F18:F20 to form a final constrained cost in cell F22. Minimize this target cell using the recipe solving method.

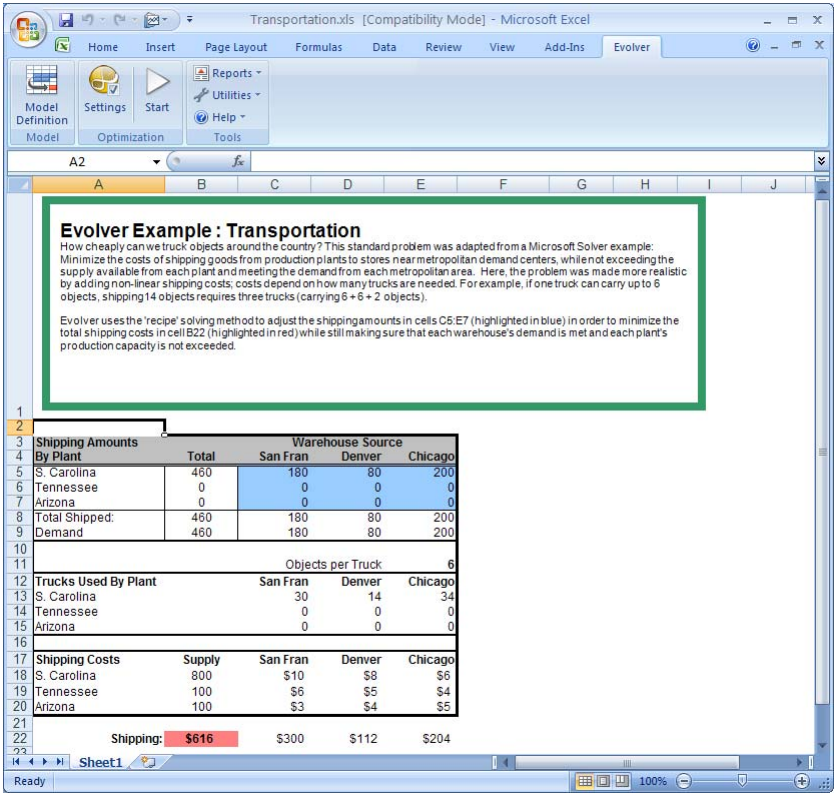
Transportation

How cheaply can we truck objects around the country? This standard problem was expanded from an older Microsoft Solver example.

"Minimize the costs of shipping goods from production plants to warehouses near metropolitan demand centers, while not exceeding the supply available from each plant and meeting the demand from each metropolitan area."

To make the problem more realistic, the shipping costs were changed so they are no longer linear, but depend on how many trucks are needed. A truck can carry up to 6 objects, so shipping 14 objects requires 3 trucks (carrying 6 + 6 + 2 objects).

Example file:	Transportation.xls
Goal:	Truck objects from three plants to five warehouses in the cheapest way possible.
Solving method:	recipe
Similar problems:	design communications networks



How The Model Works

Cells C5:G7 contain the number of objects shipped from each plant to each warehouse. C13:G13 compute the number of trucks that would be needed to ship those objects. The hard constraints are: 1) that the total shipped from each plant is less than or equal to the supply on hand at the plant, and 2) that the total shipped from all plants to each warehouse is greater than or equal the amount that warehouse requires. This ensures that every warehouse will get what it needs, and no plant is overtaxed.

How To Solve It

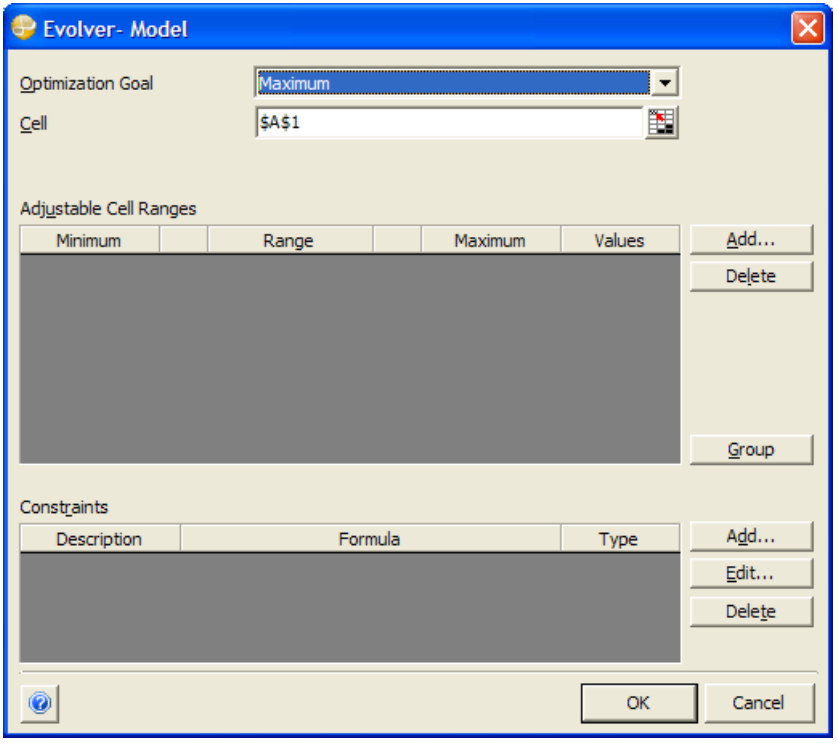
Use the recipe solving method on cells C5:G7, using integers between 0 and 500. A set of hard constraints are entered for each plant specifying that plant shipments<=plant supply. A second set of hard constraints are entered for each warehouse specifying that total shipments to warehouse>=warehouse demands. Minimize the shipping cost in cell B22.

Chapter 5: Evolver Reference Guide

Model Definition Command

Defines the goal, adjustable cells and constraints for a model

Selecting the Evolver Model Definition command (or clicking the Model icon on the Evolver toolbar) displays the Model Dialog.



The Evolver Model Dialog.

The Evolver Model Dialog is used to specify or describe an optimization problem to Evolver. This dialog starts empty with each new Excel workbook, but saves its information with each workbook. That means that when the sheet is opened again, it will be filled out the same way. Each component of the dialog is described in this section.

Options in the Model dialog include:

- **Optimization Goal.** The *Optimization Goal* option determines what kind of answer Evolver is to search for. If *Minimum* is selected, Evolver will look for variable values that produce the smallest possible value for the target cell (all the way down to -1e300). If *Maximum* is selected, Evolver will search for the variable values that result in the largest possible value for the target cell (up to +1e300).

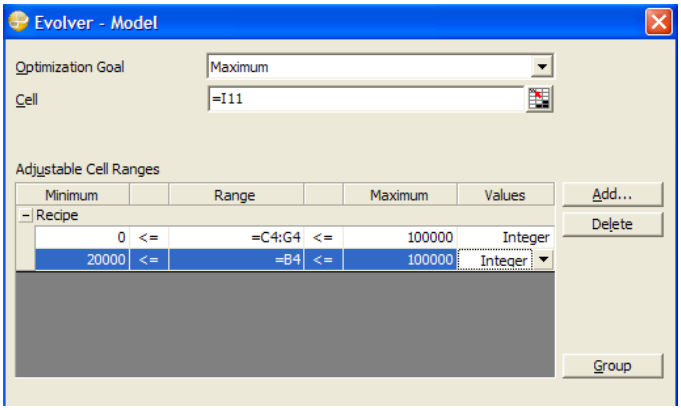
  If *Target Value* is selected, Evolver will search for variable values that produce a value for the target cell as close as possible to the value you specify. When Evolver finds a solution which produces this result, it will automatically stop. For example, if you specify that Evolver should find the result that is closest to 14, Evolver might find scenarios that result in a value such as 13.7 or 14.5. Note that 13.7 is closer to 14 than 14.5; Evolver does not care whether the value is greater or less than the value you specify, it only looks at how close the value is.

- **Cell.** The cell or ***target cell*** contains the output of your model. A value for this target cell will be generated for each "trial solution" that Evolver generates (i.e., each combination of possible adjustable cell values). The target cell should contain a formula which depends (either directly or through a series of calculations) on the adjustable cells. This formula can be made with standard Excel formulas such as SUM() or user-defined VBA macro functions. By using VBA macro functions you can have Evolver evaluate models that are very complex.

  As Evolver searches for a solution it uses value of the target cell as a rating or "fitness function" to evaluate how good each possible scenario is, and to determine which variable values should continue cross-breeding, and which should die. In biological evolution, death is the "fitness function" that determines what genes continue to flourish throughout the population. When you build your model, your target cell must reflect the fitness or "goodness" of any given scenario, so as Evolver calculates the possibilities, it can accurately measure its progress.

Adjustable Cell Ranges

The *Adjustable Cell Ranges* table displays each range which contains the cells or values that Evolver can adjust, along with the description entered for those cells. Each set of adjustable cells is listed in a horizontal row. One or more adjustable cell ranges can be included in an **Adjustable Cell Group**. All cell ranges in an Adjustable Cell Group share a common solving method, crossover rate, mutation rate and operators.



Because the adjustable cells contain the variables of the problem, you must define at least one group of adjustable cells to use Evolver. Most problems will be described with only one group of adjustable cells, but more complex problems may require different blocks of variables to be solved with different solving methods simultaneously. This unique architecture allows for highly complex problems to be easily built up from many groups of adjustable cells.

The following options are available for entering Adjustable Cell Ranges:

- **Add**. You can add new adjustable cells by clicking on the "Add" button next to the Adjustable Cells list box. Select the cell or cell range to be added, and a new row will appear in the **Adjustable Cell Ranges** table. In the table, you can enter a **Minimum** and **Maximum** value for the cells in the range, along with the type of Values to test – **Integer** values across the range, or **Any** values.
- **Minimum and Maximum.** After you have specified the location of the adjustable cells, the Minimum and Maximum entries set the range of acceptable values for each adjustable cell. By default, each adjustable cell takes on a real-number (double-precision floating point) value between -infinity and +infinity.

Range settings are constraints that are strictly enforced. Evolver will not allow any variable to take on a value outside the set ranges. You are encouraged to set more specific ranges for your variables whenever possible to improve Evolver's performance. For example, you may know that the number cannot be a negative, or that Evolver should only try values between 50 and 70 for a given variable.

- **Range**. The reference for the cell(s) to be adjusted is entered in the *Range* field. This reference can be entered by selecting the region in the spreadsheet with the mouse, entering a range name or typing in a valid Excel reference such as Sheet1!A1:B8. The **Range** field is available for all solving methods. For recipe and budget methods, however, *Minimum, Maximum* and *Values* options can be added to allow the entry of a range for the adjustable cells.

> ***NOTE: By assigning tight ranges to your variables, you can limit the scope of the search, and speed up* Evolver*'s convergence on a solution. But be careful not to limit the ranges of your variables too tightly; this may prevent* Evolver *from finding optimal solutions.***

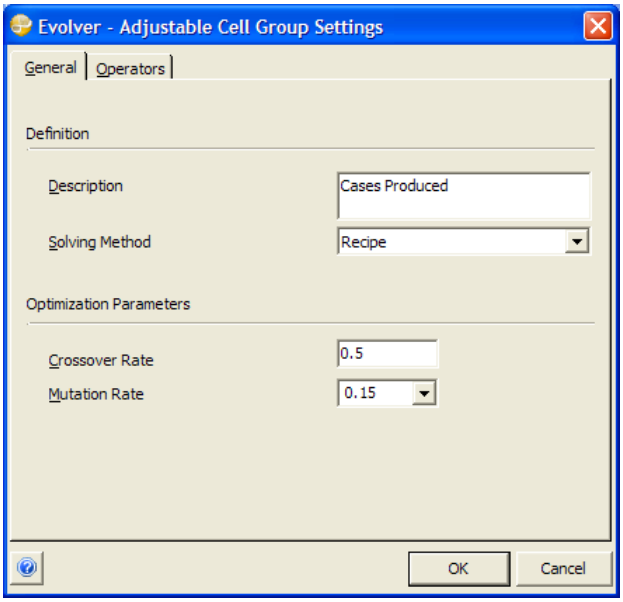
- **Values.** The Values entry allows you to specify that Evolver should treat all of the variables in the specified range as integers (e.g., 22), rather than as real numbers (e.g., 22.395). This option is only available when using the "recipe" and "budget" solving methods. The default is to treat the variables as real numbers.

Be sure to turn on the Integers setting if your model uses variables to lookup items from tables (HLOOKUP(), VLOOKUP(), INDEX(), OFFSET(), etc.). Note that the Integers setting affects all of the variables in the selected range. If you want to treat some of your variables as reals and some as integers, you can create two groups of adjustable cells instead of one, and treat one block as integers and the other block as reals. Simply "Add" a recipe group of adjustable cells, and leave the Values entry as Any. Next, "Add" another cell range, this time selecting the Integers setting and selecting only the integer adjustable cells.

Adjustable Cell Groups

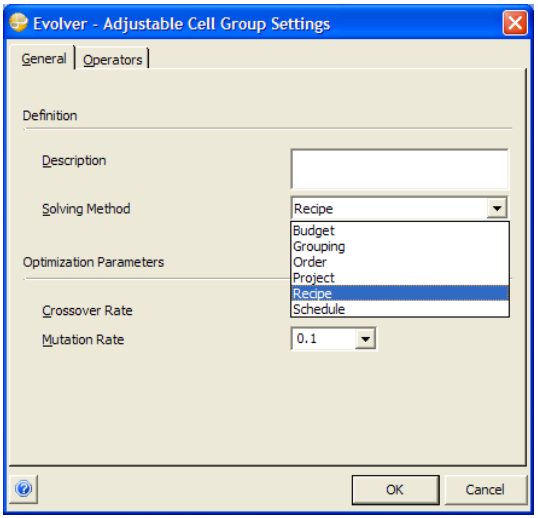
Each group of adjustable cells can contain multiple cell ranges. This allows you to build a "hierarchy" of groups of cell ranges that are related. Within each group, each cell range can have its own Min-Max range constraint.

All cell ranges in an Adjustable Cell Group share a common **solving method, crossover rate, mutation rate and operators**. These are specified in the **Adjustable Cell Group Settings dialog**. This dialog is accessed by clicking the **Group** button next to the **Adjustable Cell Ranges** table. You may create a new Group to which you can add adjustable cell ranges or edit the settings for an existing group.



Options on the **General tab** in the Adjustable Cell Group Settings dialog include**:**

- **Description.** Describes the group of adjustable cell ranges in dialogs and reports.
- **Solving Method.** Selects the Solving Method to be used for each of the adjustable cell ranges in the group.



When you select a range of cells to be adjusted by Evolver, you also are specifying a "solving method" you wish to apply when adjusting those adjustable cells. Each solving method is, in essence, a completely different genetic algorithm, with its own optimized selection, crossover and mutation routines. Each solving method juggles the values of your variables a different way.

The "recipe" solving method, for example, treats each variable selected as an ingredient in a recipe; each variable's value can be changed independently of the others'. In contrast, the "order" solving method swaps values between the adjustable cells, reordering the values that were originally there.

There are six solving methods that come with Evolver. Three of the solving methods (recipe, order, and grouping) use entirely different algorithms. The other three are ***descendants*** of the first three, adding additional constraints.

The following section describes the function of each solving method. To get a better understanding of how each solving method is used, you are also encouraged to explore the example files included with the software (see Chapter 4: Example Applications).

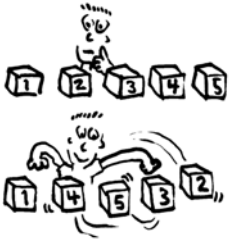
Recipe Solving Method

The "recipe" solving method is the most simple and most popular type of solving method. Use recipe whenever the set of variables that are to be adjusted can be varied independently of one another. Think of each variable as the amount of an ingredient in a cake; when you use the "recipe" solving method, you are telling Evolver to generate numbers for those variables in an effort to find the best mix. The only constraint you place on recipe variables is to set the range (the highest and lowest value) that those values must fall between. Set these values in the *Min* and *Max* fields in the Adjustable Cells dialog (e.g. 1 to 100), and also indicate whether or not Evolver should be trying integers (1, 2, 7) or real numbers (1.4230024, 63.72442).

Below are examples of a set of variable values as they might be in a sheet before Evolver is called, and what two new scenarios might look like after using the recipe solving method.

Original Set of Variable Values	One Set of Possible Recipe Values	Another Set of Possible Recipe Values
23.472	15.344	37.452
145	101	190
9	32.44	7.073
65,664	14,021	93,572

Order Solving Method



The "order" solving method is the second most popular type, after "recipe". An order is a permutation of a list of items, where you are trying to find the best way to arrange a set of given values. Unlike "recipe" and "budget" solving methods, which ask Evolver to generate values for the chosen variables, this solving method asks Evolver to use the existing values in your model.

An order could represent the order in which to perform a set of tasks. For example, you might wish to find the order in which to accomplish five tasks, numbered 1,2,3,4, and 5. The "order" solving method would scramble those values, so one scenario might be 3,5,2,4,1. Because Evolver is just trying variable values from your initial sheet, there is no Min - Max range entered for adjustable cells when the Order solving method is used.

Below are examples of a set of variable values as they might be in a sheet before Evolver is called, and what two new scenarios might look like after using the order solving method.

Original Set of Variable Values	One Set of Possible Order Values	Another Set of Possible Order Values
23.472	145	65,664
145	23.472	9
9	65,664	145
65,664	9	23.472

Grouping Solving Method

The "grouping" solving method should be used whenever your problem involves multiple variables to be grouped together in sets. The number of different groups that Evolver creates will be equal to the number of unique values present in the adjustable cells at the start of an optimization. Therefore, when you build a model of your system, be sure that each group is represented at least once.

For example, suppose a range of 50 cells contains only the values 2, 3.5, and 17. When you select the 50 cells and adjust the values using the "grouping" solving method, Evolver will assign each of the fifty cells to one of the three groups, 2, 3.5 or 17. All of the groups are represented by at least one of the adjustable cells; just like tossing each of the 50 variables in one of several "bins", and making sure there is at least one variable in each bin. Another example would be assigning 1s, and 0s, and -1s to a trading system to indicate buy, sell and hold positions. Like the "order" solving method, Evolver is arranging existing values, so there is no min-max range or integers option to define.

***NOTE:** When using the "grouping" solving method, do not leave any cells blank, unless you would like 0.0 to be considered one of the groups.*

You may realize that the "grouping" solving method could be approximated by using the "recipe" solving method with the integers option "on" and the ranges set from 1 to 3 (or whatever number of groups there are). The difference lies in the way a recipe and a grouping perform their search. Their ***selection***, ***mutation*** and ***crossover*** routines are different; a grouping is much more concerned with the values of all the variables, because it can swap a set of variables from one group with a set of variables from another group.

Below are examples of a set of variable values as they might be in a sheet before Evolver is called, and what two new scenarios might look like after using the grouping solving method.

Original Set of Variable Values	One Set of Possible Grouping Values	Another Set of Possible Grouping Values
6	6	8
7	6	7
8	8	6
8	7	7

When using the Grouping solving method, there are 2 additional settings in the Adjustable Cell Group Settings dialog:

- **Group Names (Optional)**. This setting allows a user to specify a range containing numeric group IDs. Normally Evolver reads group IDs from the adjustable range. For example, if the adjustable range is A1:D1, and it contains numbers 1, 1, 3, 2, then Evolver with use 1, 2, and 3 as group IDs. However, there may be more groups than there are adjustable cells; for example, we may want to assign items represented by cells A1:D1 to groups numbered 1 to 5. In this case, the Group Names setting will allow the user to specify a range containing five cells with numbers 1 to 5 be used as group IDs during optimization.
- **All Groups Must Be Used**. If this option is checked, every solution will have members from every group. For example, if the adjustable cells are A1:D1, and group IDs are 1, 2, and 3, then Evolver will not try a solution with 1 assigned to all four cells (with 2 and 3 missing). On the other hand this solution may be tried if the check box is not selected.

Budget Solving Method

A "budget" is similar to a "recipe" except that all of the variables' values must total to a certain number. That number is the total of the variables' values at the time an optimization is started.

For example, you might want to find the best way to distribute an annual budget among a number of departments. The "budget" solving method will take the total of the current values for the departments, and use that sum as the total budget to be optimally distributed. Below are examples of what two new scenarios might look like after using the budget solving method.

Original Set of Budget Values	One Set of Possible Budget Values	Another Set of Possible Budget Values
200	93.1	223.5
3.5	30	0
10	100	-67
10	.4	67

Many values are being tried, but the sum of all values remains 223.5.

Project Solving Method

The "project" solving method is similar to the "order" solving method except that certain items (tasks) must precede others. The "project" solving method can be used in project management to rearrange the order in which tasks are carried out, but the order will always meet the precedence constraints.

A problem modeled using the *Project* solving method will be much easier to work with and understand if the adjustable cells containing the task order are in a single column, rather than in a row. This is because the solving method expects the preceding tasks cells to be arranged vertically rather than horizontally, and it will be easier to examine your worksheet if the adjustable cells are also vertical.

After you have specified the location of the adjustable cells, you should specify the location of the preceding tasks cells in the *Preceding Tasks* section of the dialog. This is a table of cells that describes which tasks must be preceded by which other tasks. The solving method uses this table to rearrange the order of variables in a scenario until the precedence constraints are met. There should be one row in the preceding tasks range for each task in the adjustable cells. Starting in the first column of the preceding tasks range, the identifying number of each task on which that row's task depends should be listed in separate columns.

This Item	Must Comes After These...		
1	6	9	
2	1	6	3
3	1		
4	9	12	
5			
6	9	1	2
7	3	4	
8			
9	12	3	1

Example of how to set up precedents for Project solving method.

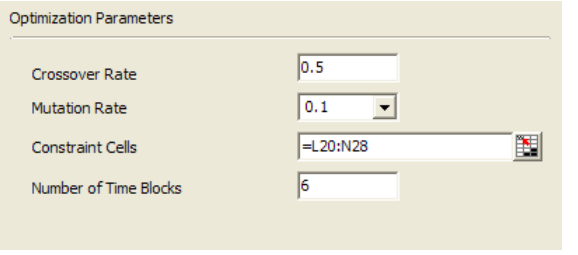
The precedence tasks range should be specified as being ***n*** rows by ***m*** columns, where ***n*** is the number of tasks in the project (adjustable cells), and ***m*** is the largest number of preceding tasks that any one task has.

Below are examples of a set of variable values as they might be in a sheet before Evolver is called, and what two new scenarios might look like after using the Project solving method, with the constraint that 2 must always come after 1, and 4 must always come after 2.

Original Set of Variable Values	One Set of Possible Project Values	Another Set of Possible Project Values
1	1	1
2	3	2
3	2	4
4	4	3

Schedule Solving Method

A schedule is similar to a grouping; it is an assignment of tasks to times. Each task is assumed to take the same amount of time, much as classes at a school are all of the same length. Unlike a grouping, however, the Adjustable Cell Group Settings Dialog for the "schedule" solving method lets you directly specify the number of time blocks (or groups) to be used. Notice when you select the "schedule" method, several related options appear in the lower portion of the dialog box.



In the *Optimization Parameters* section, you will notice that you can also have a constraint cell range attached to it. This range can be of any length, but must be exactly three columns wide. Eight kinds of constraints are recognized:

1) *(with) The tasks in the 1st & 3rd columns must occur in the same time block.*
2) *(not with) The tasks in the 1st & 3rd columns must not occur in the same time block.*
3) *(before) The task in the 1st column must occur before the task in the 3rd column.*
4) *(at) The task in the 1st column must occur in the time block in the 3rd column.*

5) *(not after) The task in 1st column must occur at the same time or before the task in the 3rd column.*
6) *(not before) The task in 1st column must occur at the same time or after the task in the 3rd column.*
7) *(not at) The task in the 1st column must not occur in the time block in the 3rd column.*
8) *(after) The task in the 1st column must occur after the task in the 3rd column.*

Either a numeric code (1 through 8) or the English description (*after, not at, etc.*) can be entered for a constraint. (Note: All language versions of the Evolver will recognize the English description entered for a constraint as well as the its translated form). All of the constraints specified in your problem will be met. To create constraints, find an empty space on your worksheet and create a table where the left and right columns represent tasks, and the middle column represents the type of constraints. A number from 1 to 8 represents the kind of constraint listed above. The cells in the constraint range must have the constraint data in them before you start optimizing.

This Task	Constraint	This Task
5	4	2
12	2	8
2	3	1
7	1	5
6	2	4
9	3	1

Below are examples of a set of variable values as they might be in a sheet before Evolver is called, and what two new scenarios might look like after using the Schedule solving method.

Original Set of Variable Values	One Set of Possible Schedule Values	Another Set of Possible Schedule Values
1	1	1
2	1	3
3	3	1
1	1	2
2	2	2
3	3	2

NOTE: When you select the schedule solving method, integers starting from 1 are always used (1,2,3...), regardless of the original values in the adjustable cells.

Crossover and Mutation Rate

One of the most difficult problems with searching for optimal solutions, when your problem has seemingly endless possibilities, is in determining where to focus your energy. In other words, how much computational time should be devoted to looking in new areas of the "solution space", and how much time should be devoted to fine-tuning the solutions in our population that have already proven to be pretty good?

A big part of the genetic algorithm success has been attributed to its ability to preserve this balance inherently. The structure of the GA allows good solutions to "breed", but also keeps "less fit" organisms around to maintain diversity in the hopes that maybe a latent "gene" will prove important to the final solution.

Crossover and *Mutation* are two parameters that affect the scope of the search, and Evolver allows users to change these parameters before, and also during the evolutionary process. This way, a knowledgeable user can help out the GA by deciding where it should focus its energy. For most purposes, the default crossover and mutation settings (.5 and .1 respectively) do not need adjustment. In the event that you wish to fine-tune the algorithm to your problem, conduct comparative studies, or just to experiment, here is a brief introduction to these two parameters:

- **Crossover.** The crossover rate can be set to between 0.01 and 1.0, and reflects the likelihood that future scenarios or "organisms" will contain a mix of information from the previous generation of parent organisms. This rate can be changed by experienced users to fine-tune Evolver's performance on complex problems.

  In other words, a rate of 0.5 means that an offspring organism will contain roughly 50% of its variable values from one parent and the remaining values from the other parent. A rate of 0.9 means that roughly 90% of an offspring organism's values will come from the first parent and 10% will come from the second parent. A Crossover rate of 1 means that no crossover will occur, so only clones of the parents will be evaluated.

  The default rate used by Evolver is 0.5. Once Evolver has started solving a problem, you can change the crossover rate by using the Evolver Watcher (see the Evolver Watcher section in this chapter).

- **Mutation Rate.** The mutation rate can be set to between 0.0 and 1.0, and reflects the likelihood that future scenarios will contain some random values. A higher mutation rate simply means that more mutations or random "gene" values will be introduced into the population. Because mutation occurs after crossover, setting the mutation rate to 1 (100% random values) will effectively prevent the crossover from having any effect, and Evolver will generate totally random scenarios.

  If all the data of the optimal solution was somewhere in the population, then the crossover operator alone would be enough to eventually piece together the solution. Mutation has proven to be a powerful force in the biological world for many of the same reasons that it is needed in a genetic algorithm: it is vital to maintaining a diverse population of individual organisms, thereby preventing the population from becoming too rigid, and unable to adapt to a dynamic environment. As in a genetic algorithm, it is often the genetic mutations in animals which eventually lead to the development of critical new functions.

  For most purposes, the default mutation setting does not need adjustment, but can, however, be changed by experienced users to fine-tune Evolver's performance on complex problems. The user may wish to boost the mutation rate if Evolver's population is fairly homogenous, and no new solutions have been found in the last several hundred trials. Typical setting changes are from .06 to .2. Once Evolver has started solving a problem, you can change the mutation rate dynamically by using the Evolver Watcher (see the Evolver Watcher section later in this chapter).

  By selecting *Auto* from the drop down list in the Mutation rate field, auto-mutation rate adjustment is selected. Auto-mutation rate adjustment allows Evolver to increase the mutation rate automatically when an organism "ages" significantly; that is, it has remained in place over an extended number of trials. For many models, especially where the optimal mutation rate is not known, selecting Auto can give better results faster.

Number of Time Blocks and Constraint Cells

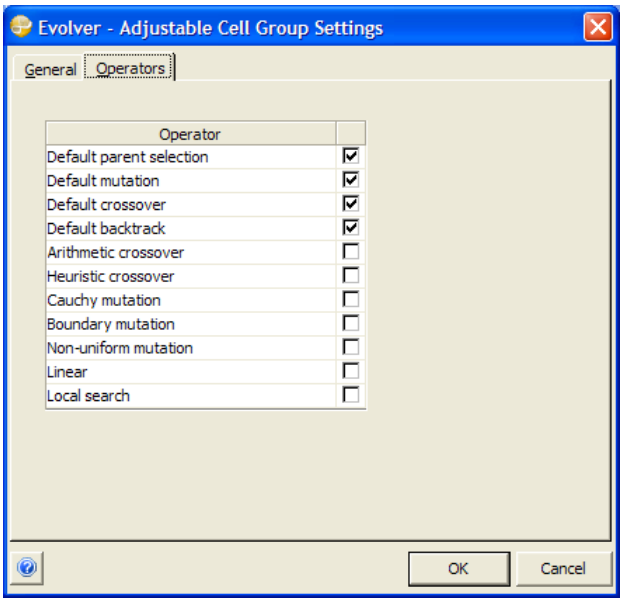
For more information on these options, see the *Schedule Solving* method in the *Solving Methods* section of this chapter.

Preceding Tasks

For more information on these options, see the *Project Solving* method in the *Solving Methods* section of this chapter.

Operators

Evolver includes selectable genetic operators when used with the Recipe solving method. Clicking the **Operators tab** in the Adjustable Cell Group Settings Dialog allows you to select a specific genetic operator (such as heuristic crossover or boundary mutation) to be used when generating possible values for a set of adjustable cells. In addition, you can have Evolver automatically test all available operators and identify the best performing one for your problem.



Genetic algorithms use genetic operators to create new members of the population from current members. Two of the types of genetic operators Evolver employs are *mutation* and *crossover*. The mutation operator determines if random changes in "genes" (variables) will occur and how they occur. The crossover operator determines how pairs of members in a population swap genes to produce "offspring" that may be better answers than either of their "parents".

Evolver includes the following specialized genetic operators:

- **Arithmetic Crossover** – Creates new offspring by arithmetically combining the two parents (as opposed to swapping genes).
- **Heuristic Crossover** – Uses values produced by the parents to determine how the offspring is produced. Searches in the most promising direction and provides fine local tuning.
- **Cauchy Mutation** – Designed to produce small changes in variables most of the time, but can occasionally generate large changes.

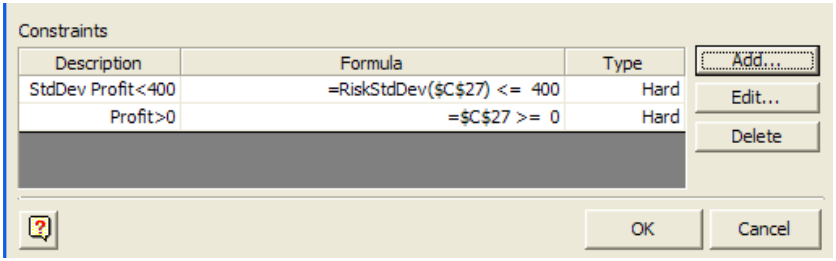
- **Boundary Mutation –** Designed to quickly optimize variables that affect the result in a monotonic fashion and can be set to the extremes of their range without violating constraints.
- **Non-uniform Mutation –** Produces smaller and smaller mutations as more trials are calculated. This allows Evolver to "fine tune" answers.
- **Linear –** Designed to solve problems where the optimal solution lies on the boundary of the search space defined by the constraints. This mutation and crossover operator pair is well suited for solving linear optimization problems.
- **Local search** - Designed to search the solution space in the neighborhood of a previous solution, expanding in directions that provide improvement, and contracting in directions that produce a worse result.

Depending on the type of optimization problem, different combinations of mutation and crossover operators may produce better results than others. In the Operators tab of the Adjustable Cell Group Settings dialog, when using the Recipe solving method, any number of operators may be selected. When multiple selections are made, Evolver will test valid combinations of the selected operators to identify the best performing ones for your model. After a run, the *Optimization summary worksheet* ranks each of the selected operators by their performance during the run. For subsequent runs of the same model, selecting just the top performing operators may lead to faster, better performing optimizations.

***NOTE:** When creating multiple groups of adjustable cells, check to be sure that no spreadsheet cell is included in several different groups of adjustable cells. Each group of adjustable cells should contain unique adjustable cells because the values in the first group of adjustable cells would be ignored and overwritten by the values in the second group of adjustable cells. If you think a problem needs to be represented by more than one solving method, consider how to break up the variables into two or more groups.*

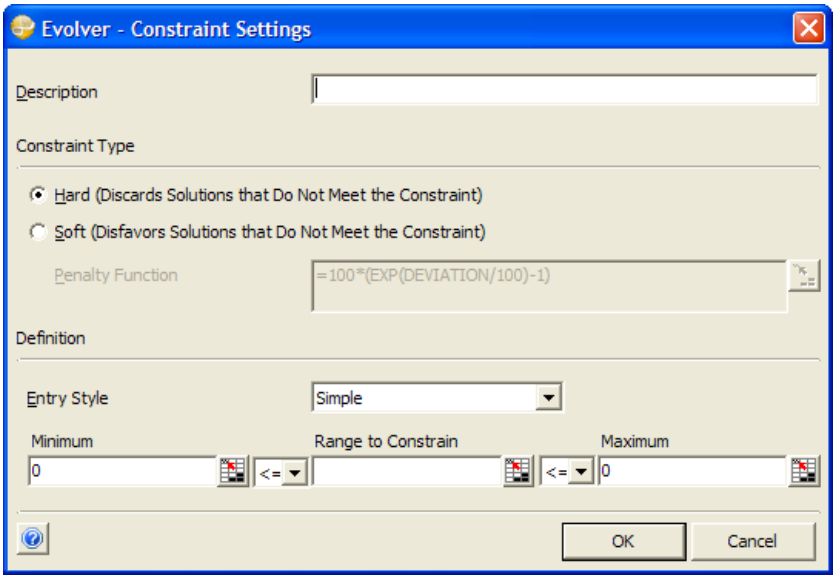
Constraints

Evolver allows you to enter constraints, or conditions that must be met for a solution to be valid. Constraints you have entered are shown in the **Constraints table** in the Model Definition dialog box.



Add - Adding Constraints

Clicking the *Add* button next to the Constraints table displays the **Constraint Settings** dialog box where constraints are entered. Using this dialog box the type of constraint desired, along with its description, type, definition and evaluation time can be entered.



Constraint Type

Two types of constraints can be specified in Evolver:

- **Hard**, or conditions that must be met for a solution to be valid (i.e., a hard constraint could be C10<=A4; in this case, if a solution generates a value for C10 that is greater than the value of cell A4, the solution will be thrown out).
- **Soft**, or conditions which we would like to be met as much as possible, but which we may be willing to compromise for a big improvement in fitness or target cell result (i.e., a soft constraint could be C10<100; however, C10 could go over 100, but when that happened the calculated value for the target cell would be decreased based on the penalty function you have entered).

Simple and Formula Constraints

Two formats – ***Simple*** and ***Formula*** -- can be used for entering constraints. The type of information you can enter for a constraint depends on the format you select.

- **Simple Format** - The S*imple* format allows constraints to be entered using simple <, <=, >, >= or = relations where a cell is compared with an entered number. A typical Simple constraint would be:

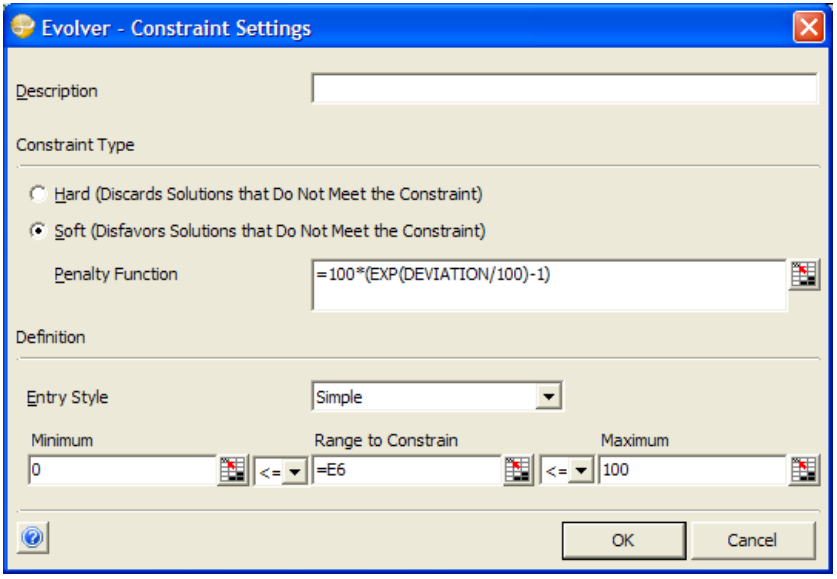
  0<Value of A1<10

  where A1 is entered in the *Cell Range* box, 0 is entered in the *Min* box and 10 is entered in the *Max* box. The operator desired is selected from the drop down list boxes. With a simple range of values format constraint, you can enter just a Min value, just a Max or both. The entered Min and Max values must be numeric in the simple range of values constraint format.

- **Formula Format** - The Formula format allows you to enter any valid Excel formula as a constraint, such as A19<(1.2*E7)+E8. Evolver will check whether the entered formula evaluates to TRUE or FALSE to see if the constraint has been met

Soft Constraints

Soft Constraints are conditions which we would like to be met as much as possible, but which we may be willing to compromise for a big improvement in fitness or target cell result. When a soft constraint is not met it causes a change in the target cell result away from its optimal value. The amount of change caused by an unmet soft constraint is calculated using a penalty function that is entered when you specify the soft constraint.

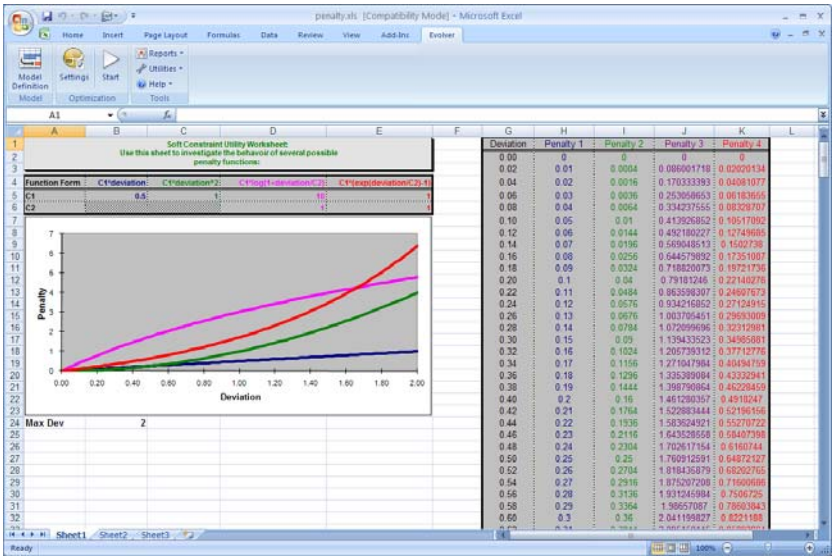


More information about penalty functions is as follows:

- **Entering a Penalty Function.** Evolver has a default penalty function which is displayed when you first enter a soft constraint. Any valid Excel formula, however, may be entered to calculate the amount of penalty to apply when the soft constraint is not met. An entered penalty function should include the keyword ***deviation*** which represents the absolute amount by which the constraint has gone beyond its limit. With each recalculation Evolver checks if the soft constraint has been met; if not, it places the amount of deviation in the entered penalty formula and then calculates the amount of penalty to apply to the target cell.

  The penalty amount is either added or subtracted from the calculated target cell value in order to make it less "optimal". For example, if *Maximum* is selected in the *Optimization Goal* field in the Evolver Model Dialog, the penalty is subtracted from the calculated target cell value.

- **Viewing the Effects of an Entered Penalty Function.** Evolver includes an Excel worksheet PENALTY.XLS which can be used to evaluate the effects of different penalty functions on specific soft constraints and target cell results.



PENALTY.XLS allows you to select a soft constraint from your model whose effects you wish to analyze. You can then change the penalty function to see how the function will map a specific value for the unmet soft constraint into a specific penalized target value. For example, if your soft constraint is A10<100, you could use PENALTY.XLS to see what the target value would be if a value of 105 was calculated for cell A10.

- **Viewing the Penalties Applied.** When a penalty is applied to the target cell due to an unmet soft constraint, the amount of penalty applied can be viewed in the Evolver Watcher. In addition, penalty values are shown in Optimization Log worksheets, created optionally after optimization.

NOTE: If you place a solution in your worksheet using the* Update Adjustable Cell Values *options in the* Stop *dialog, the calculated target cell result shown in the spreadsheet will not include any penalties applied due to unmet soft constraints. Check the Optimization Log worksheet to see the penalized target cell result and the amount of penalty imposed due to each unmet soft constraint.

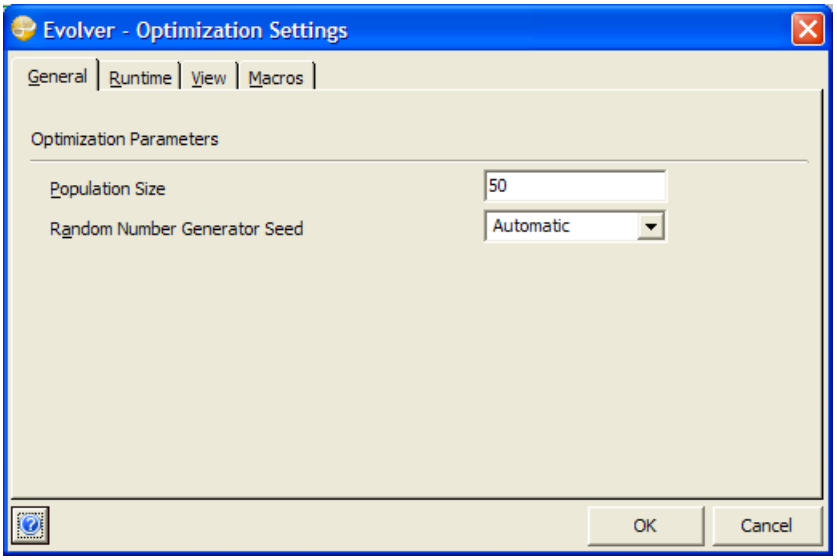
- **Implementing Soft Constraints in Worksheet Formulas.** Penalty functions can be implemented directly in the formulas in your worksheet. If soft constraints are implemented directly in the worksheet they should not be entered in the main Evolver dialog. For more information on implementing penalty functions in your worksheet, see the section *Soft Constraints* in Chapter 9: Evolver Extras.

Optimization Settings Command

Optimization Settings Command – General Tab

Defines the general settings for an optimization

The Optimization Settings dialog General tab displays settings for population size, display update, and random number generator seed.



Optimization Parameter Options on the General tab include:

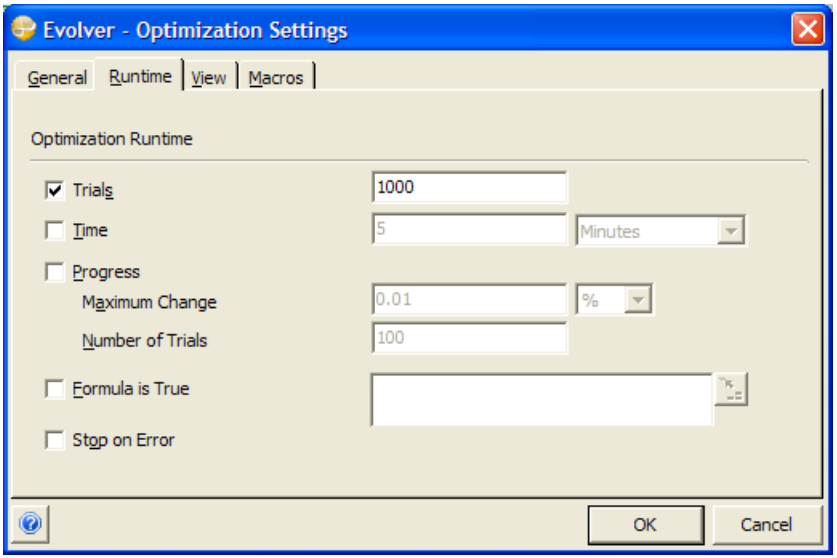
- **Population Size.** The population size tells Evolver how many organisms (or complete sets of variables) should be stored in memory at any given time. Although there is still much debate and research regarding the optimal population size to use on different problems, generally we recommend using 30-100 organisms in your population, depending on the size of your problem (bigger population for larger problems). The common view is that a larger population takes longer to settle on a solution, but is more likely to find a global answer because of its more diverse gene pool.

- **Random Number Generator Seed.** The Random Number Generator Seed option allows you to set the starting seed value for the random number generator used in Evolver. When the same seed value is used, an optimization will generate the exact same answers for the same model as long as the model has not been modified. The seed value must be an integer in the range 1 to 2147483647.

Optimization Settings Command – Runtime Tab

Defines the runtime settings for an optimization

The Optimization Settings dialog Runtime tab displays Evolver settings that determine the runtime of the optimization. These stopping conditions specify how and when Evolver will stop during an optimization. Once you select the Start Optimization command, Evolver will continuously run, searching for better solutions and running trials until the selected stopping criteria are met. You can turn on any number of these conditions, or none at all if you want Evolver to search indefinitely (until you stop it). When multiple conditions are checked, Evolver stops as soon as one of the chosen conditions is met. You may also override these selections and stop Evolver at any time manually using the stop button in the Evolver Watcher or Progress windows.



Optimization Runtime Options

Optimization Runtime options on the Runtime tab include:

- **Trials** - This option, when set, stops Evolver when the given number of trial solutions are generated by Evolver.

  The Trials setting is particularly useful when comparing Evolver's efficiency when trying different modeling methods. By changing the way you model a problem, or by choosing a different solving method, you may increase Evolver's efficiency. Having a model run a specified number of trials will indicate how efficiently Evolver is converging on a solution, regardless of any differences in the number of variables chosen, the speed of the computer hardware being used, or the screen re-drawing time. The Evolver optimization summary worksheet is also useful in comparing results between runs. For more information on Optimization Summary worksheets, see the Evolver Watcher – Stopping Options section in this chapter.

- **Time** - This option, when set, stops Evolver from optimizing scenarios after the given number of hours, minutes or seconds has elapsed. This entry can be any positive real number (600, 5.2, etc.).

- **Progress** - This option, when set, stops Evolver from optimizing scenarios when the improvement in the target cell is less than the specified amount (change criterion). You can specify, as an integer, the number of trials over which to check the improvement. A percentage value - such as 1% - can be entered as the maximum change value in the *Maximum Change* field.

  Suppose that we are trying to maximize the mean of the target cell, and after 500 trials, the best answer found so far is 354.8. If the "*Progress*" option is the only stopping condition selected, Evolver will pause at trial #600 and will only continue if it is able to find an answer of at least 354.9 during those last 100 trials. In other words, Evolver's answers have not improved at least 0.1 over the last 100 trials, so it assumes there is little more improvement to be found, and stops the search. For more complex problems, you may want to boost the number of trials that Evolver runs through (500) before deciding whether there is still sufficient improvement to go on.

  This is the most popular stopping condition, because it gives the user an effective way to stop Evolver after the improvement rate is slowing down, and Evolver is not seeming to find any better solutions. If you are viewing the graphs of the best results on the Progress tab of the Evolver

Watcher, you will see the graphs plateau or flatten out for a while before this condition is met and Evolver stops. "*Progress*" is really just an automatic way to do what you could do yourself with the graph -- let it run until the improvement levels off.

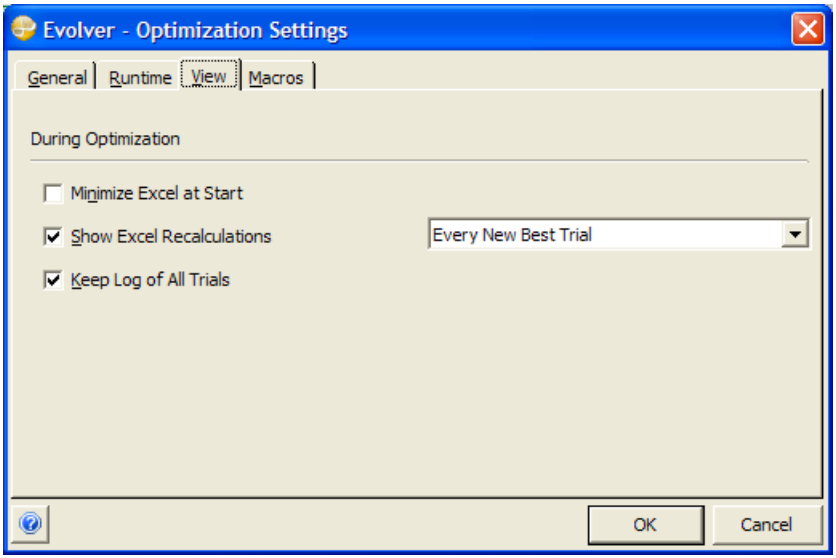
- **Formula is True.** This stopping condition causes the optimization to stop whenever the entered (or referenced) Excel formula evaluates to TRUE during the optimization.
- **Stop on Error.** This stopping condition causes the optimization to stop whenever an Error value is calculated for the target cell.

NOTE: ***You can also select no stopping conditions, and*** **Evolver** ***will run forever until you press the stop button on the*** **Evolver Watcher** ***window.***

Optimization Settings Command – View Tab

Defines the view settings for an optimization

The Optimization Settings dialog View tab displays Evolver settings that determine what will be shown during an optimization.



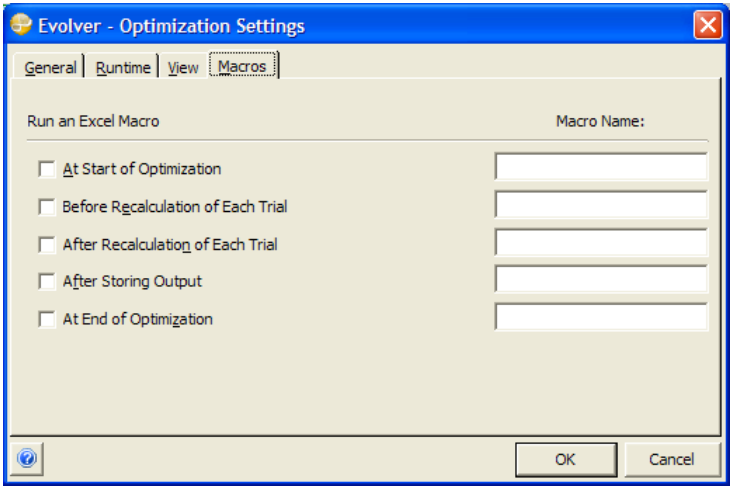
Options on the View tab include:

- **Minimize Excel at Start**. This option selects to minimize Excel when an optimization starts.
- **Show Excel Recalculations**. This specifies to update Excel either with **Every New Best Trial**, or at the end of **Every Trial**.
- **Keep Log of Trials**. This option specifies that Evolver keeps a running log of each new trial performed. This log can be viewed in the Evolver Watcher Window.

Optimization Settings Command – Macros Tab

Defines macros to be run during an optimization

VBA macros can be run at different times during an optimization and during each trial solution. This allows the development of custom calculations that will be invoked during an optimization.



Macros may be executed at the following times during an optimization:

- **At the Start of the Optimization** - macro runs after the Run icon is clicked; prior to the first trial solution being generated.
- **Before Recalculation of Each Trial** - macro runs before recalculation of each trial that is executed.
- **After Recalculation of Each Trial**- macro runs after recalculation of each trial that is executed
- **After Storing Output** - macro runs after each trial that is executed and after the value for the target cell's is stored.
- **At the End of the Optimization** - macro runs when the optimization is completed.

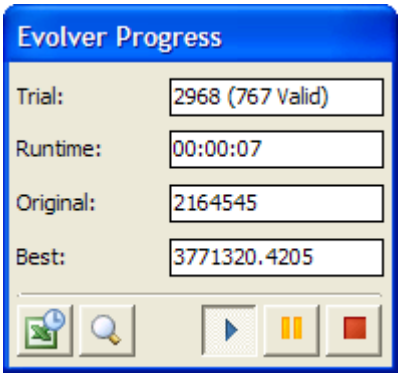
This feature allows calculations which only can be performed through the use of a macro to be made during an optimization. Examples of such macro-performed calculations are iterative "looping" calculations and calculations which require new data from external sources.

The **Macro Name** defines the macro to be run.

Start Optimization Command

Starts an optimization

Selecting the Start Optimization command or clicking the Start Optimization icon starts an optimization of the active model and workbook. As soon as Evolver is running, you will see the following Evolver **Progress window**.



The Progress window displays:

- **Trial** or the total number of trials that have been executed and **#Valid** indicates the number of those trials for which all constraints were met.
- **Runtime** or the elapsed time in the run
- **Original** or the original value for the target cell.
- **Best** or the current best value for the target cell that is being minimized or maximized.

Options on the Evolver Toolbar of the Progress window include:

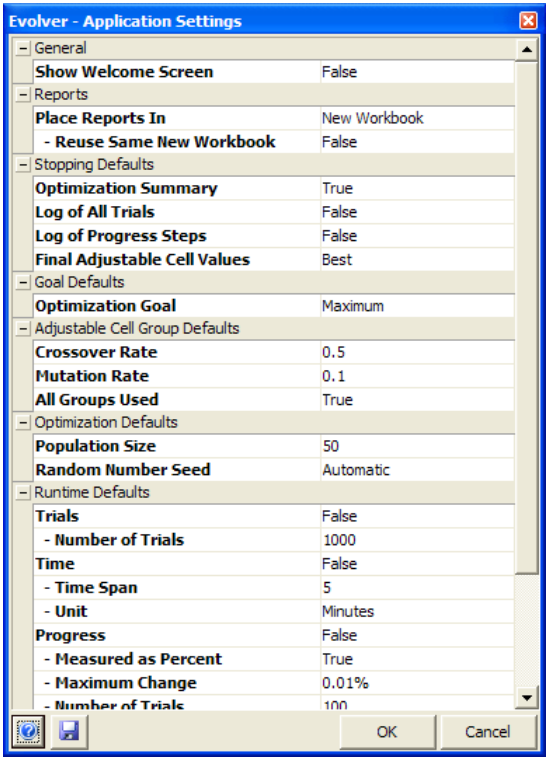
- **Display Excel Updating Options**. Selects to update the Excel display **Every Trial**, on **Every New Best Trial** or **Never**. Note that in some situations the screen will be updated independently of these settings, for example when optimization has been paused.
- **Display Evolver Watcher**. Displays the full Evolver Watcher window.
- **Run.** Clicking the Run icon causes Evolver to begin searching for a solution based on the current description in the Evolver Model Dialog. If you pause Evolver you will still be able to click the Run icon to continue the search for better solutions.
- **Pause.** If you would like to pause the Evolver process, just click the Pause icon, and you temporarily "freeze" the Evolver process. While paused, you may wish to open and explore the Evolver Watcher and change parameters, look at the whole population, view a status report, or copy a graph.
- **Stop.** Stops the optimization.

Utilities Commands

Application Settings Command

Displays the Application Settings dialog where program defaults can be set

A wide variety of Evolver settings can be set at default values that will be used each time Evolver runs. These include Stopping Defaults, Default Crossover and Mutation Rates and others.

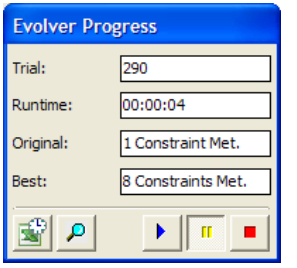


Constraint Solver Command

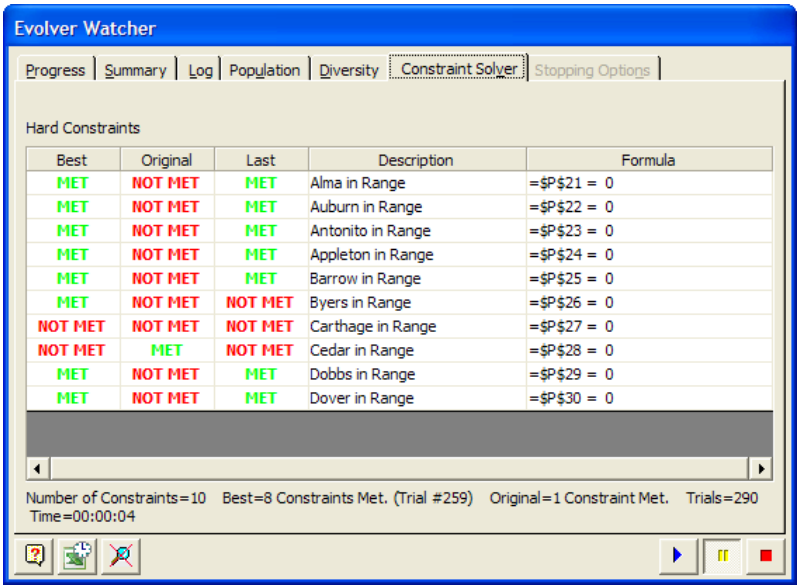
Runs the Constraint Solver

The Constraint Solver enhances Evolver's ability to handle model constraints. When Evolver runs an optimization, it is assumed that the original adjustable cell values meet all the hard constraints, i.e. that the original solution is valid. If that is not the case, the algorithm may run very many trials before finding a first valid solution. However, if a model contains multiple constraints, then it may not be obvious what adjustable cell values will meet all of them.

If a Evolver model contains multiple hard constraints, and optimizations fail with all solutions invalid, you will be notified and the Constraint Solver can be run. The Constraint Solver runs an optimization in a special mode, in which the objective is to find a solution meeting all the hard constraints. The optimization progress is shown to the user in the same way as in regular optimizations. The **Progress Window** shows the number of constraints that are met in the original and best solutions.



A button in the Progress Window allows the user to switch to the Evolver Watcher. In the Constraint Solver mode the details of optimization progress are available like in regular mode optimizations, in **Progress**, **Summary**, **Log**, **Population** and **Diversity** tabs. In the Constraint Solver mode the Watcher contains an additional tab, entitled **Constraint Solver**. This tab shows the status of each hard constraint (**Met** or **Not Met**) for the Best, Original, and Last solution.



A Constraint Solver optimization stops automatically when a solution meeting all the hard constraints is found; it can also be stopped by clicking a button in the progress window or in the Evolver Watcher. After a Constraint Solver run, in the Evolver Watcher Stopping Options tab you can choose to keep the Best, Original, or Last solution, like in regular-mode optimizations.

Note there is no need to set up the Constraint Solver before a run. It uses the settings specified in the model, only changing the optimization objective: the new objective is to find a solution meeting all the hard constraints.

Evolver Watcher

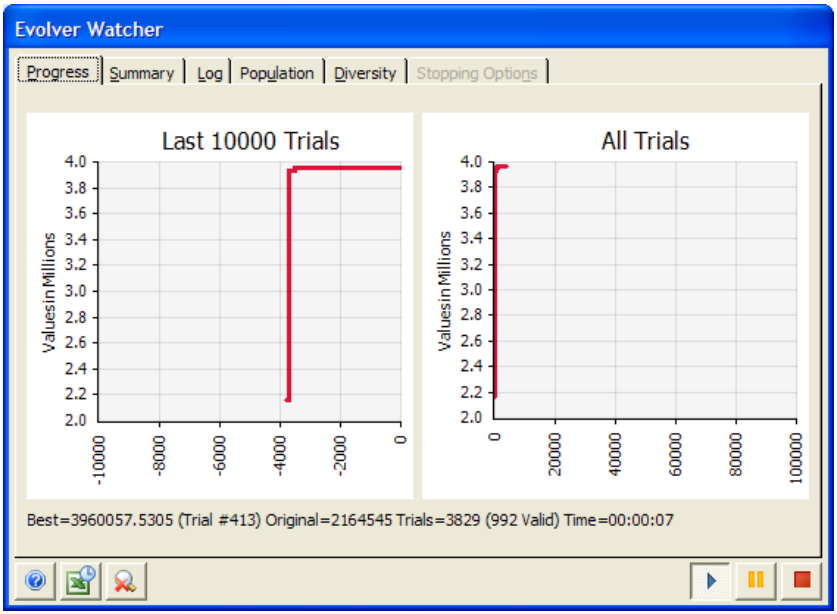
The magnifying glass icon on the Evolver Progress window toolbar displays the Evolver Watcher. Evolver Watcher is responsible for regulating and reporting on all Evolver activity.

From Evolver Watcher, you can change parameters and analyze the progress of the optimization. You can also see real-time information about the problem and information on Evolver's progress in the status bar across the bottom of Evolver Watcher.

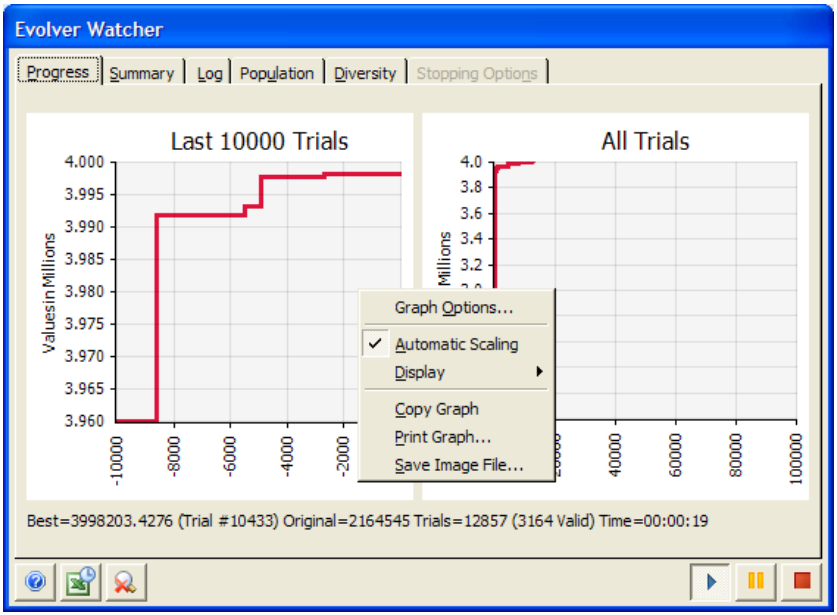
Evolver Watcher – Progress Tab

Displays progress graphs for target cell value

The **Progress Tab** in the Evolver Watcher graphically shows how results are changing, by trial, for the selected target cell.

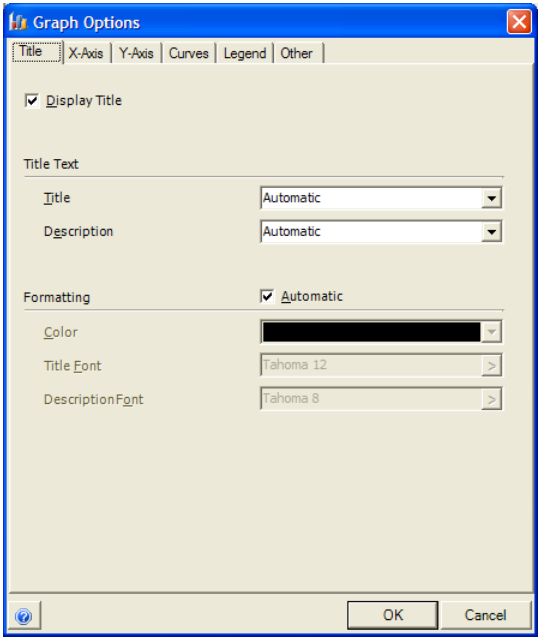


Progress graphs show the trial count on the X-axis and target cell value on the Y-axis. Progress graphs can be rescaled by clicking on the axis limits and dragging the axis to the new scale value. Alternatively, right-clicking on the Progress graph can display the **Graph Options** dialog where further customization of the graphs is allowed.



Graph Options Dialog

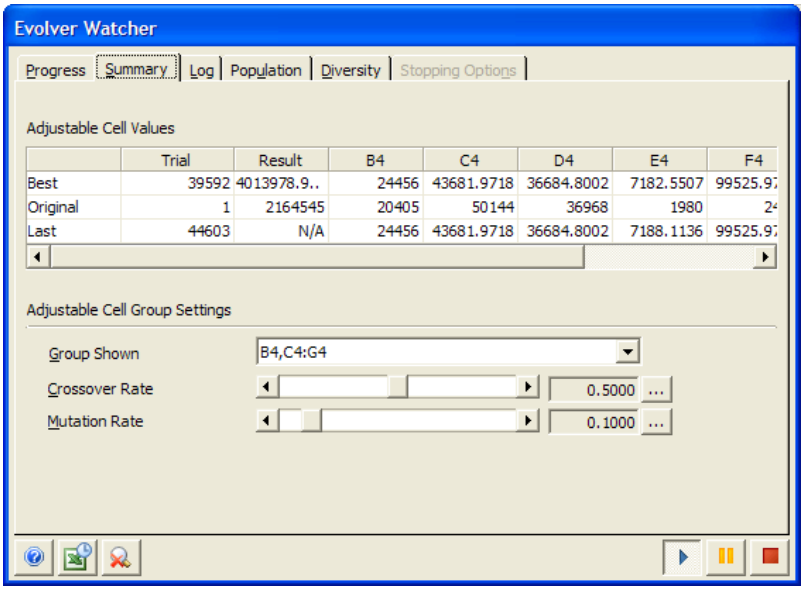
The Graph Options dialog displays settings that control the titles, legends, scaling and fonts used on the displayed graph.



Evolver Watcher – Summary Tab

Displays details for adjustable cell values

The **Summary Tab** in the Evolver Watcher displays a summary table of adjustable cell values tested during the optimization, along with tools for adjusting the crossover and mutation rate for each Adjustable Cell Group in the model.



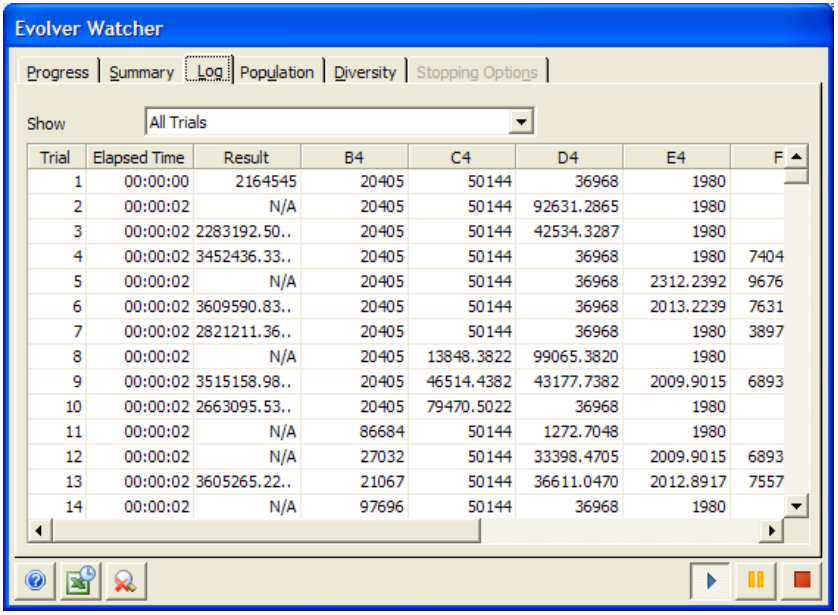
The **Adjustable Cell Group Settings** allows you to change the Crossover and Mutation rates of the genetic algorithm as the problem is in progress. Any changes made here will override the original setting of these parameters and will take place immediately, affecting the population (or group of adjustable cells) that was selected in the **Group Shown** field.

We almost always recommend using the default crossover of 0.5. For mutation, in many models you may turn it up as high as about 0.4 if you want to find the best solution and are willing to wait longer for it. Setting the mutation value to 1 (the maximum) will result in completely random guessing, as Evolver performs mutation after it performs crossover. This means that after the two selected parents are crossed over to create an offspring solution, 100% of that solution's "genes" will mutate to random numbers, effectively rendering the crossover meaningless (see "crossover rate, what it does" and "mutation rate, what it does" in the index for more information).

Evolver Watcher – Log Tab

Displays a log of each trial run during the optimization

The **Log Tab** in the Evolver Watcher displays a summary table of each trial run during the optimization. The log includes the results for the target cell, each adjustable cell and entered constraints. A log is only available if the option **Keep a Log of All Trials** is selected in the Otimization Settings dialog **View** tab.



Trial	Elapsed Time	Result	B4	C4	D4	E4	F
1	00:00:00	2164545	20405	50144	36968	1980	
2	00:00:02	N/A	20405	50144	92631.2865	1980	
3	00:00:02	2283192.50..	20405	50144	42534.3287	1980	
4	00:00:02	3452436.33..	20405	50144	36968	1980	7404
5	00:00:02	N/A	20405	50144	36968	2312.2392	9676
6	00:00:02	3609590.83..	20405	50144	36968	2013.2239	7631
7	00:00:02	2821211.36..	20405	50144	36968	1980	3897
8	00:00:02	N/A	20405	13848.3822	99065.3820	1980	
9	00:00:02	3515158.98..	20405	46514.4382	43177.7382	2009.9015	6893
10	00:00:02	2663095.53..	20405	79470.5022	36968	1980	
11	00:00:02	N/A	86684	50144	1272.7048	1980	
12	00:00:02	N/A	27032	50144	33398.4705	2009.9015	6893
13	00:00:02	3605265.22..	21067	50144	36611.0470	2012.8917	7557
14	00:00:02	N/A	97696	50144	36968	1980	

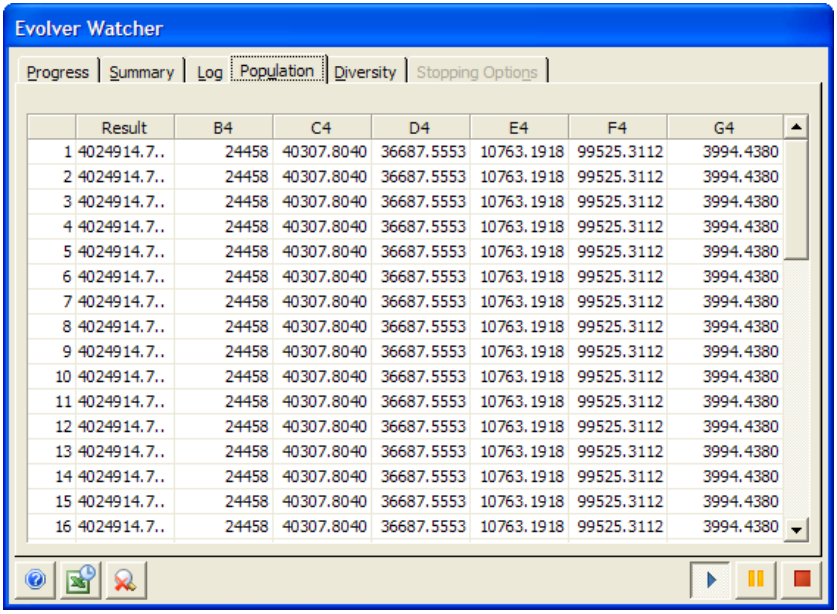
The Show options select to show a log of **All Trials** or only those Trials where there was a **Progress Step** (i.e. where the optimization result improved). The log includes:

1) **Elapsed Time**, or the start time of the optimization
2) **Iters**, or the number of iterations run
3) **Result**, or the value of the target cell that you are trying to maximize or minimize, including penalties for soft constraints
4) **Input columns**, or the values used for your adjustable cells
5) **Constraint columns** showing whether your constraints were met

Evolver Watcher – Population Tab

Lists all the variables of each organism (each possible solution) in the current population

The population table is a grid which lists all the variables of each organism (each possible solution) in the current population. These organisms ("Org *n*") are ranked in order from worst to best. Since this table lists all organisms in the population, the "population size" setting in the Evolver Settings dialog determines how many organisms will be listed here (default 50). In addition, the first column of the chart shows the resulting value of the target cell for each organism.

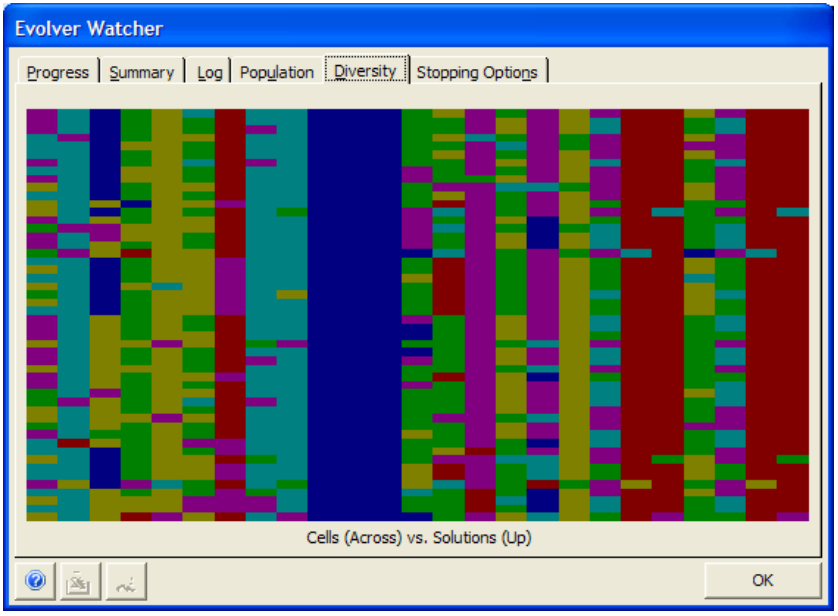


	Result	B4	C4	D4	E4	F4	G4
1	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
2	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
3	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
4	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
5	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
6	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
7	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
8	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
9	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
10	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
11	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
12	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
13	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
14	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
15	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380
16	4024914.7..	24458	40307.8040	36687.5553	10763.1918	99525.3112	3994.4380

Evolver Watcher – Diversity Tab

Displays a color plot of all variables in the current population

The plot on the Diversity tab assigns colors to adjustable cell values, based on how much the value of a given cell differs across the population of organisms (solutions) that are stored in memory at a given point. (Using the genetic optimization terminology, this is an indication of the diversity that exists in the gene pool.) Each vertical bar in the plot corresponds to one adjustable cell. Horizontal stripes within each bar represent the values of that adjustable cell in different organisms (solutions). The colors of the stripes are assigned by dividing the range between the minimum and maximum value for a given adjustable cell into 16 equal-length intervals; each of the intervals is represented by a different color. For example, in the picture the fact that the vertical bar representing the second adjustable cell is single-color means that the cell has the same value in each solution in memory.

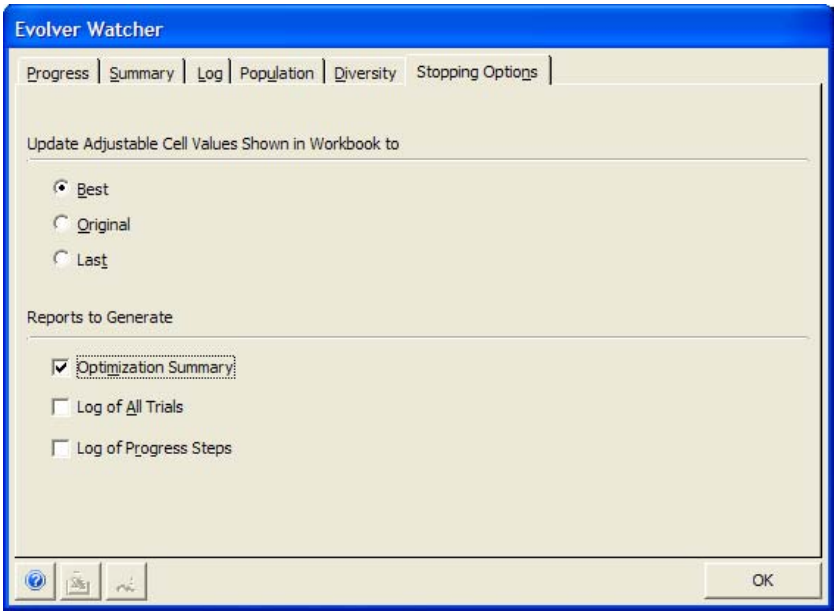


Evolver Watcher – Stopping Options Tab

Displays stopping options for the optimization

When you click the **Stop** button, the Evolver Watcher dialog **Stopping Options tab** is displayed. This includes the options available for updating your worksheet with the best calculated values for adjustable cells, restoring original values, and generating an optimization summary report.

Clicking OK destroys Evolver's population of solutions and places the selected values in your spreadsheet. If you wish to save any information about the Evolver session, including the population values, the time and number of trials run, be sure to select to create an optimization summary report.



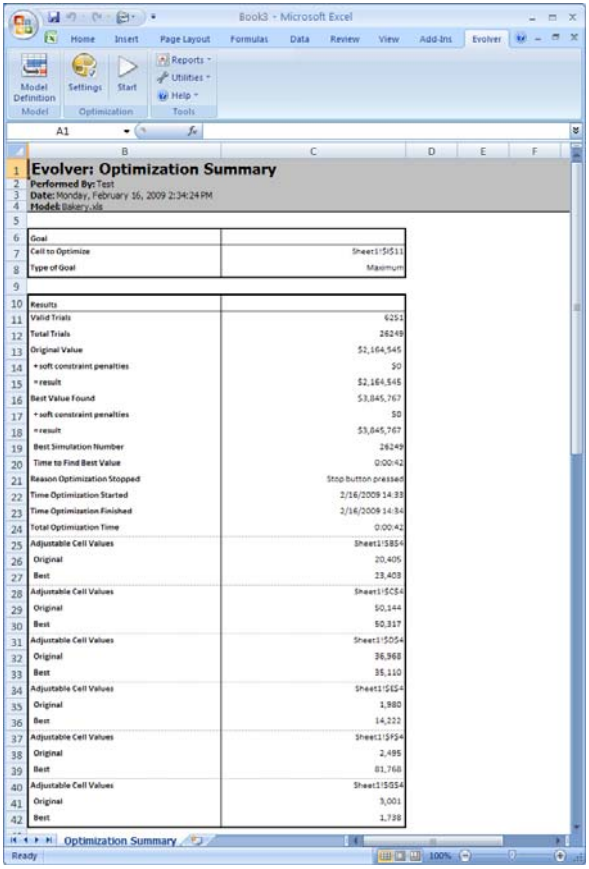
This dialog will also appear if one of the user specified stopping conditions has been met (number of requested trials have been evaluated, minutes requested have elapsed, etc.). The stop alert offers three choices for updating the adjustable cell values in your spreadsheet: **Best, Original and Last.**

- **Best .** This accepts Evolver's results and ends Evolver's search for better solutions. When you choose this option, the values of the best scenario Evolver has found in its search are placed into the adjustable cells of your spreadsheet.

- **Original**. This restores the adjustable cells to their original values before Evolver was run, and ends Evolver's search for better solutions.
- **Last**. This causes Evolver to place the last calculated values in the optimization in the worksheet. The Last Calculated Values option is particularly useful when debugging models.

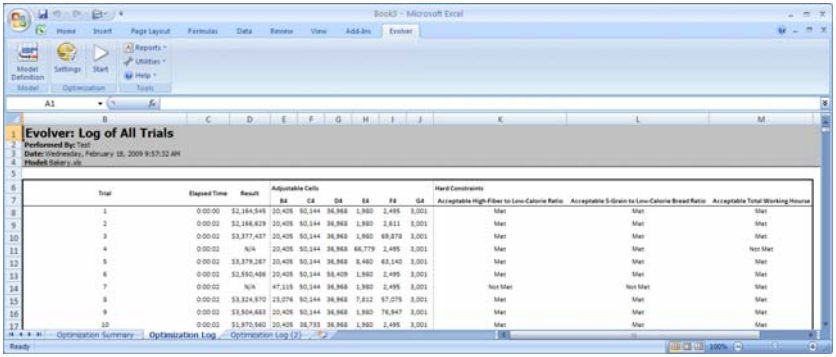
The Reports to Generate options can generate optimization summary worksheets that can be used for reporting on the results of a run and comparing the results between runs. Report options include:

- **Optimization Summary.** This summary report contains information such as date and time of the run, the optimization settings used, the value calculated for the target cell and the value for each of the adjustable cells.

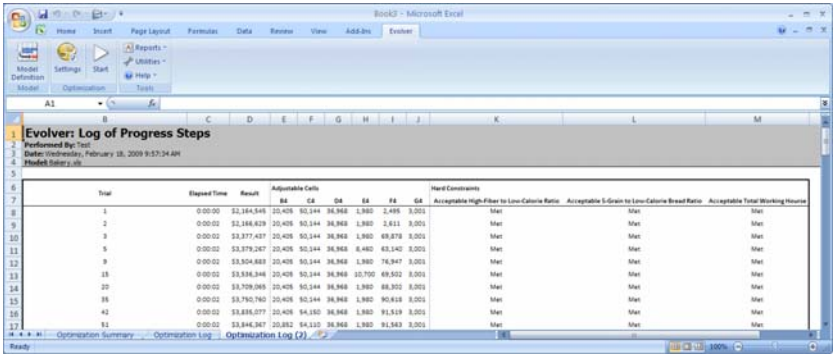


This report is useful for comparing the results of successive optimizations.

- **Log of All Trials.** This report logs the results of all trials performed.



- **Log of Progress Steps.** This report logs the results of all trials that improved the result for the target cell.



Chapter 6: Optimization

Optimization Methods

We have already seen a few examples of optimization problems in the tutorials. Some optimization problems are much harder than others to solve. For tough problems, such as finding the shortest route between 1000 cities, it is not feasible to examine every possible solution. Such an approach would require years of calculations on the fastest computers.

To solve such problems, it is necessary to search through a subset of all possible solutions. By examining these solutions, we can get an idea of how to find better solutions. This is accomplished with an *algorithm*. An algorithm is simply a step-by-step description of how to approach a problem. All computer programs, for example, are built by combining numerous algorithms.

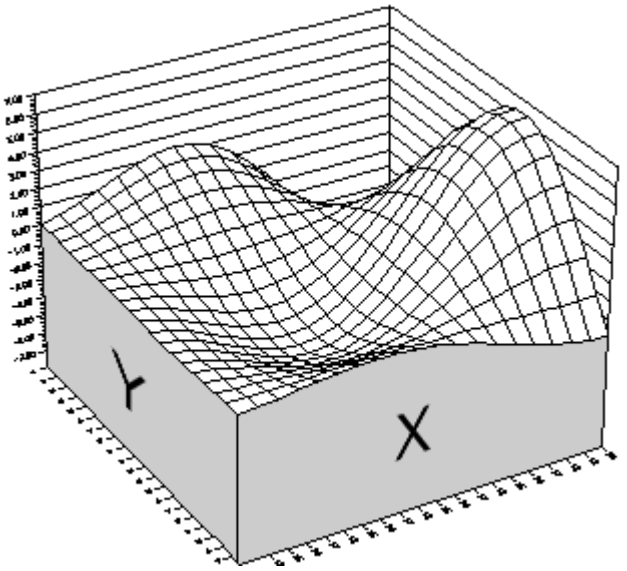
Let us start by exploring how most problem-solving algorithms represent a problem. Most problems can be divided into three basic components: inputs, a function of some kind, and a resulting output.

	Looking for:	Given this:	To get the best:
Problem Components	Inputs	Function	Output
In Evolver/Excel	Variables	Model	Goal

Let us assume that our optimization problem involves two variables, X and Y. When placed in an equation, these two variables produce a result =Z. Our problem is to find the value for X and Y that produces the largest Z value. We can think of Z as a "rating", which indicates how good any particular X,Y pairing is.

	Looking for:	Given this:	To get the best:
In this example	X and Y	Equation	Z

A plot of every single set of Xs,Ys, and the resulting Zs would produce a three-dimensional surface graph such as the one shown below.



A "landscape" of possible scenarios or solutions.

Each intersection of an X and Y value produces a Z height. The peaks and valleys of this "landscape" represent good and bad solutions respectively. Searching for the maximum or highest point on this function by examining each solution would take far too much time, even with a powerful computer and the fastest program.* Remember that we are giving Excel just the function itself, not a graph of the function, and that we could just as easily be dealing with a 200-dimensional problem as with this two-dimensional problem. Thus, we need a method that will let us do fewer calculations and still find the maximum productivity.

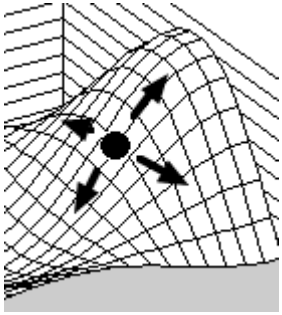
* In our diagram, the function is shown as a smooth landscape. In the rare cases where we deal with simple, smooth (differentiable) functions, it is possible to use calculus to find minima and maxima. However, most realistic problems are not described by such smooth functions.

About Hill Climbing Algorithms

Let us look at a simple algorithm called hill-climbing. Hill-climbing is an algorithm that works like this:

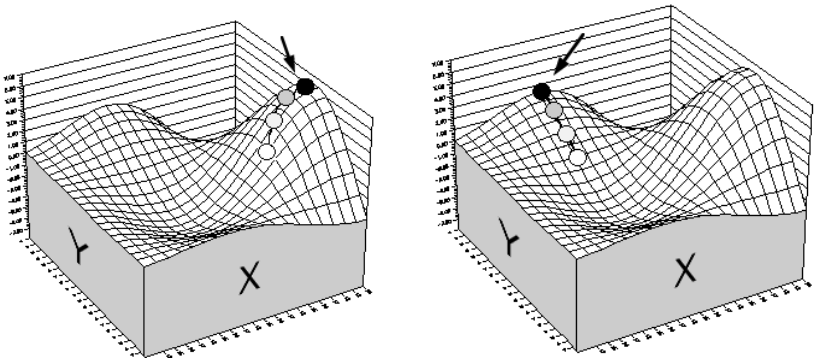
1) *Start at a random point on the landscape (take a random guess).*
2) *Walk a small distance in some arbitrary direction.*
3) *If you have walked to a new point that is higher, stay and repeat step 2. If your new point is lower, go back to your original point and try again.*

Hill-climbing tries only one solution or scenario at a time. We will use a black dot (•) to represent one possible solution (a set of X, Y and Z values). If we place the dot at the random starting point, we hope that our hill-climbing method will bring the dot to the highest point on the graph.



From the diagram above we can clearly see that we want the dot to go up the high hill to the right. However, we only know that because we have already seen the entire landscape. As the algorithm runs, it sees the landscape immediately around it, but not the entire landscape; it sees the trees but not the forest.

In most real-world problems, the landscape is not so smooth, and would require years to calculate, so we only calculate the current scenario and the immediately surrounding scenarios. Imagine that the dot is a blindfolded man standing amidst smooth, rolling hills. If the man employed the hill-climbing algorithm, this man would put one foot in each direction, and only move when he felt higher ground. This man would successfully step his way upwards, and eventually would come to rest on the hilltop where the ground all around him was lower than the ground he was on. This seems simple enough. However, we get into a very serious problem if the man starts out in another place... he climbs up the wrong hill! (see the diagram below).



Even with a smooth function, hill climbing can fail
if you start from a slightly different position (right).

Hill-climbing only finds the nearest hilltop, or ***local maximum***. Thus, if your problem has a very rough and hilly solution landscape, as most realistic models do, hill-climbing is not likely to find the highest hill, or even one of the highest.

Hill-climbing has another problem; how do we actually find the terrain around our current location? If the landscape is described by a smooth function, it may be possible to use differentiation (a calculus technique) to find out which direction has the steepest slope. If the landscape is discontinuous or not differentiable (as is more likely in real-world problems), we need to calculate the "fitness" of surrounding scenarios.

For example, lets say a bank hires one security guard from 9:00am to 5:00pm to guard the bank, but the bank must give the officer two (2) half-hour breaks. We must try to find the optimum break times, given general rules about performance/fatigue ratios, and considering the different levels of customer activity throughout the day. We may start by trying out different combinations of duty breaks and evaluate them. If we currently use a schedule where the breaks are timed at 11:00am and 3:00pm, we might calculate the productivity of the surrounding scenarios:

Direction	Break 1 (x)	Break 2 (y)	–Score" (z)
Current Solution	11:00am	3:00pm	= 46.5
West Scenario	10:45am	3:00pm	= 44.67
East Scenario	11:15am	3:00pm	= 40.08
North Scenario	11:00am	3:15pm	= 49.227
South Scenario	11:00am	2:45pm	= 43.97

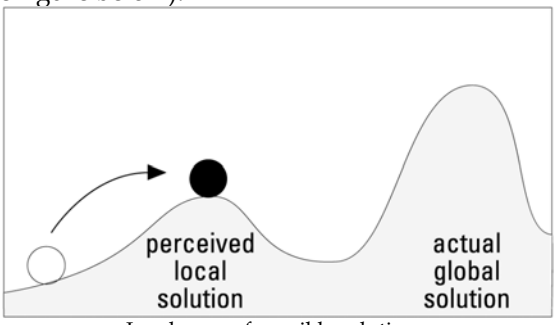
If we had three adjustable cells (breaks) instead of two, we would need to look at eight different directions. In fact, if we had just fifty variables, (quite realistic for a medium-sized problem), we would need to calculate productivity for 2^{50}, or over one quadrillion scenarios, just for one guard!!

There are modifications that can be made to hill-climbing to improve its ability to find global maxima (the highest hills on the entire landscape). Hill-climbing is most useful for dealing with unimodal (one-peak) problems, and that is why some analysis programs use the technique. Nevertheless, it is very limited for complex and/or large problems.

Excel Solver

Excel includes an optimization utility called ***Solver***. It serves a somewhat similar purpose as Evolver: to find optimal solutions. Solver can solve two kinds of problems: linear problems and simple non-linear problems. It solves linear problems using a linear programming routine. This classic mathematical technique is often called the Simplex method, and it will always find perfect answers to small, purely linear problems.

Like most other ***baby solvers***, the Microsoft Solver also solves non-linear problems, using a ***hill climbing*** routine (specifically, the GRG2 routine). A hill climbing routine starts with the current variable values and slowly adjusts them until the output of the model does not improve anymore. This means that problems with more than one possible solution may be impossible for Solver to solve well, because Solver ends up at a ***local*** solution and cannot jump over to the ***global*** solution (see figure below).



Landscape of possible solutions.

In addition, Solver requires that the function represented by your model be continuous. This means the output should change smoothly as the inputs are adjusted. If your model uses lookup tables, acquires noisy, real-time data from another program, contains random elements, or involves if-then rules, your model will be jumpy and discontinuous. Solver would not be able to solve such a problem.

Solver also puts a limit on the number of variables and the number of constraints in your problem (200) above which you must turn to a more powerful technique.

Evolver vs. Solver

The Excel Solver and Evolver each has its strengths and weaknesses. Generally speaking, Solver is faster for solving small and simple problems, while Evolver is the only way to solve many other kinds of problems. In addition, you may find Evolver will find much better answers than Solver for larger, more complex problems, the kind often seen in the "real world".

Evolver can find answers for many more kinds of problems than Solver. Almost any numerical situation that you can model in Excel can be optimized with Evolver.

Specifically, Evolver finds optimal solutions to linear, non-linear, table-based, or stochastic (random) numerical problems. It can solve problems with any combination of these qualities. Evolver can also generate permutations of existing values, re-order the values, or group the values together in different ways to find the optimum solution. In fact, wherever you have a spreadsheet model with variables that you can adjust to influence the model's output, Evolver can automate the search process for you by intelligently crunching through thousands of scenarios and keeping track of the best ones.

Evolver's genetic algorithm process is more suitable than Solver to interruptions; you may stop the Evolver process at any time and see the best solution Evolver has found so far. You can then make changes to the problem itself, and continue the process right from where you left off. For example, in a job scheduling problem, you may wish to find the best tasks to assign your machines. When one machine is available, you may stop the genetic algorithm process to find the optimal task to assign to that machine. Then the task may be omitted from the problem, and the optimization can continue with the remaining jobs.

The genetic algorithm that gives Evolver the ability to handle all of those kinds of problems will usually be slower than the Solver and other traditional mathematical or statistical methods. Some classes of problems have well-known and finely-tuned optimization routines available. In such cases, you will find answers faster by using the custom methods, rather than the general-purpose method used in Evolver.

When to Use Evolver

Generally speaking, Evolver should be used when:

1) Traditional algorithms fail to produce good, global solutions.

2) The problem is too large and/or contains more variables than your algorithm can handle.

3) Your problem is too complex to be solved by any other method.

4) Your model involves random numbers, lookup tables, if-then statements or any other discontinuous functions which prohibit the use of traditional solvers.

5) You have no idea what the solution could be, and therefore have no starting guess from which a traditional solver must begin its search.

6) You want the option of making some "hard" constraints in your problem (X must equal Y) more "soft", and therefore more accurate (X should equal Y, because otherwise I lose some Z), exploring a much wider range of possibly better solutions, even if a few constraints are violated to get them.

7) You would rather get a pretty good solution to your problem quickly than to wait a long time for the absolute best solution.

8) You need many alternative solutions that are near to the best solution.

9) You wish to imbed a simple, robust search algorithm into your own custom application (see the Evolver Developer Kit for details).

> ***NOTE: When time permits, Evolver can always be used in addition to other methods to double-check their accuracy. Although it may take more time than other methods, the better solution that Evolver may find is most likely worth the investment.***

Remember, because Evolver does not need to know the "nuts and bolts" of your problem, Evolver can solve problems where the user has no knowledge of linear programming techniques, optimization theory, mathematics or statistics. Using Evolver requires only that the user set the variables (the cells which contain values that can be adjusted), the goal (the cell that contains the output), and a description of what values Evolver may use when searching for optimal solutions.

Types of Problems

Several different types of problems are typically optimized. If you understand these types of problems, you'll be better equipped to apply Evolver to your own models.

Linear Problems

In linear problems, all the outputs are simple linear functions of the inputs, as in y=mx+b. When problems only use simple arithmetic operations such as addition, subtraction, and Excel functions such as TREND() and FORCAST() it indicates there are purely linear relationships between the variables.

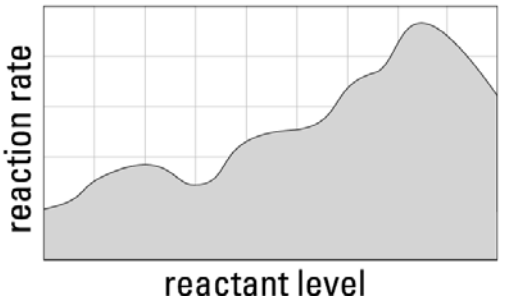
Linear problems have been fairly easy to solve since the advent of computers and the invention by George Dantzig of the Simplex Method. A simple linear problem can be solved most quickly and accurately with a linear programming utility. The Solver utility included with Excel becomes a linear programming tool when you set the "Assume Linear Model" checkbox.[*] Solver then uses a linear programming routine to quickly find the perfect solution. If your problem can be expressed in purely linear terms, you should use linear programming. Unfortunately, most real-world problems cannot be described linearly.

Non-linear Problems

If the cost to manufacture and ship out 5,000 widgets was $5,000, would it cost $1 to manufacture and ship 1 widget? Probably not. The assembly line in the widget factory would still consume energy, the paperwork would still need to be filled out and processed through the various departments, the materials would still be bought in bulk, the trucks would require the same amount of gas to deliver the widgets, and the truck driver would still get paid a full day's salary no matter how full the load was. Most real-world problems do not involve variables with simple linear relationships. These problems involve multiplication, division, exponents, and built-in Excel functions such as SQRT() and GROWTH(). Whenever the variables share a disproportional relationship to one another, the problem becomes non-linear.

[*] For more specifics on Microsoft's Solver utility, see the Excel User's Guide.

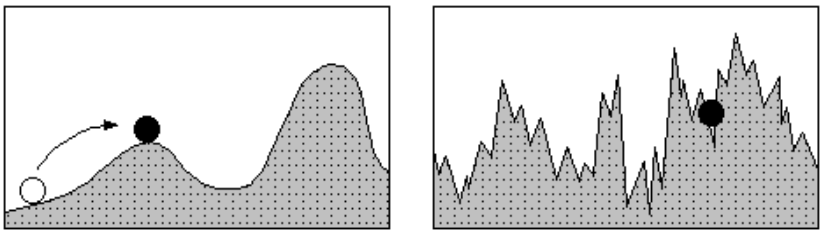
A perfect example of a non-linear problem is the management of a manufacturing process at a chemical plant. Imagine that we want to mix some chemical reactants together and get a chemical product as the result. The rate of this reaction may vary non-linearly with the amount of reactants available; at some point the catalyst becomes saturated and excess reactant just gets in the way. The following diagram shows this relationship:



If we simply need to find the minimum level of reactants that will give us the highest rate of reaction, we can just start anywhere on the graph and climb along the curve until we reach the top. This method of finding an answer is called hill climbing.

Hill climbing will always find the best answer if a) the function being explored is smooth, and b) the initial variable values place you on the side of the highest mountain. If either condition is not met, hill climbing can end up in a local solution, rather than the global solution.

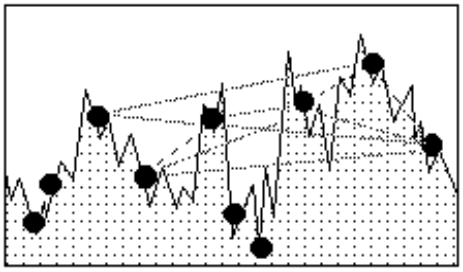
Highly non-linear problems, the kind often seen in practice, have many possible solutions across a complicated landscape. If a problem has many variables, and/or if the formulas involved are very noisy or curvy, the best answer will probably not be found with hill climbing, even after trying hundreds of times with different starting points. Most likely, a sub-optimal, and extremely local solution will be found (see figure below).



Hill climbing finds the local, but not global maximum.

Noisy data: Hill climbing not effective, even with multiple tries.

Evolver does not use hill climbing. Rather, it uses a stochastic, directed search technique, dubbed a genetic algorithm. This lets Evolver jump around in the solution space of a problem, examining many combinations of input values without getting stuck in local optima. In addition, Evolver lets good scenarios "communicate" with each other to gain valuable information as to what the overall solution landscape looks like, and then uses that information to better guess which scenarios are likely to be successful. If you have a complex or highly non-linear problem, you should, and often must, use Evolver.



Evolver generates many possible scenarios, then refines the search based on the feedback it receives.

Table-based problems

Many problems require the use of lookup tables and databases. For example, in choosing the quantities of different materials to buy, you might need to look up the prices charged for different quantities.

Tables and databases make problems discontinuous (non-smooth). That makes it difficult for hill-climbing routines like Solver to find optimal solutions. Evolver, however, does not require continuity in the functions it evaluates, and it can find good solutions for table-based problems, even problems that use many large, interrelated tables.

If your problem involves looking up values from a database, or a table of data in Excel, where the index of the table item is a variable or a function of a variable, you need to use Evolver. If you only look up a single, constant item in a table (the same record is retrieved from the table regardless of the input variables' values), then you are really only dealing with a constant, and you can probably use Solver effectively.

Combinatorial problems

There is a large class of problems that are very different from the numerical problems examined so far. Problems where the outputs involve changing the order of existing input variables, or grouping

subsets of the inputs are called combinatorial problems. These problems are usually very hard to solve, because they often require exponential time; that is, the amount of time needed to solve a problem with 4 variables might be 4 x 3 x 2 x 1, and doubling the number of variables to 8 raises the solving time to 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1, or a factor of 1,680. The number of variables doubles, but the number of possible solutions that must be checked increases 1,680 times. For example, choosing the starting lineup for a baseball team is a combinatorial problem. For 9 players, you can choose one out of the 9 as the first batter. You can then choose one out of the remaining 8 as the second batter, one of the remaining 7 will be the third, and so on. There are thus 9x8x7x6x5x4x3x2x1 (9 factorial) ways to choose a lineup of 9 players. This is about ***362,880*** different orderings. Now if you double the number of players, there are 18 factorial possible lineups, or ***6,402,373,705,000,000*** possible lineups!

Evolver's genetic algorithm intelligently searches through the possible permutations. This is much more practical than searching through *all* possibilities, and it is much more efficient than examining purely random permutations; sub-orders from good scenarios can be retained and used to create even better scenarios.

Chapter 7: Genetic Algorithms

Introduction

Evolver uses genetic algorithms to search for optimal answers for models. The genetic algorithms used are adapted from Evolver, an optimization add-in to Excel from Palisade Corporation. This chapter provides background information on genetic algorithms to give insights on how they are used for optimizing models.

History

The first genetic algorithms were developed in the early 1970s by John Holland at the University of Michigan. Holland was impressed by the ease in which biological systems could perform tasks which eluded even the most powerful super-computers; animals can flawlessly recognize objects, understand and translate sounds, and generally navigate through a dynamic environment almost instantaneously.

For decades, scientists have promised to replicate these capabilities in machines, but we are beginning to recognize just how difficult this task is. Most scientists agree that any complex biological system that exhibits these qualities has evolved to get that way.

Evolution Theory

Evolution, so the theory goes, has produced systems with amazing capabilities through relatively simple, self-replicating building blocks that follow a few simple rules:

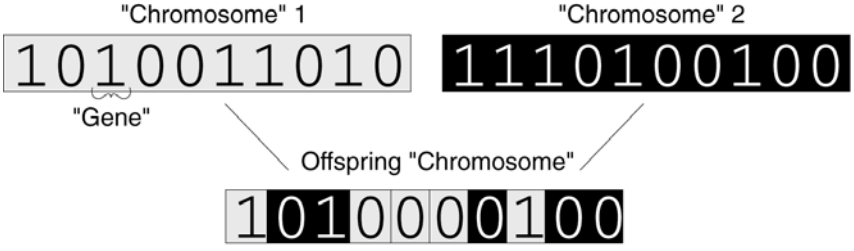
1) Evolution takes place at the level of the chromosome. The organism doesn't evolve, but only serves as the vessel in which the genes are carried and passed along. It is the chromosomes which are dynamically changing with each re-arrangement of genes.

2) Nature tends to make more copies of chromosomes which produce a more "fit" organism. If an organism survives long enough, and is healthy, its genes are more likely to be passed along to a new generation of organisms through reproduction. This principle is often referred to as "survival of the fittest". Remember that "fittest" is a relative term; an organism only needs to be fit in comparison to others in the current population to be "successful".

3) Diversity must be maintained in the population. Seemingly random mutations occur frequently in nature that ensure variation in the organisms. These genetic mutations often result in a useful, or even vital feature for a species' survival. With a wider spectrum of possible combinations, a population is also less susceptible to a common weakness that could destroy them all (virus, etc.) or other problems associated with inbreeding.

Once we break down evolution into these fundamental building blocks, it becomes easier to apply these techniques to the computational world, and truly begin to move towards more fluid, more naturally behaving machines.

Holland began applying these properties of evolution to simple strings of numbers that represented chromosomes. He first encoded his problem into binary strings (rows of "1s" and "0s") to represent the chromosomes, and then had the computer generate many of these "bit" strings to form a whole population of them. A fitness function was programmed that could evaluate and rank each bit string, and those strings which were deemed most "fit" would exchange data with others through a "crossover" routine to create "offspring" bit strings. Holland even subjected his digital chromosomes to a "mutation" operator, which injected randomness into the resulting "offspring" chromosomes to retain diversity in the population. This fitness function replaced the role of death in the biological world; determining which strings were good enough to continue breeding and which would no longer be kept in memory.



The program kept a given number of these "chromosomes" in memory, and this entire "population" of strings continued to evolve until they maximized the fitness function. The result was then decoded back to its original values to reveal the solution. John Holland remains an active pioneer in this field, and is now joined by hundreds of scientists and scholars who have devoted the majority of their time toward this promising alternative to traditional linear programming, mathematical, and statistical techniques.

Holland's original genetic algorithm was quite simple, yet remarkably robust, finding optimal solutions to a wide variety of problems. Many custom programs today solve very large and complex real-world problems using only slightly modified versions of this original genetic algorithm.

Modern Adaptations of Genetic Algorithms

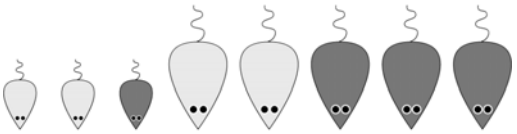
As interest swelled in academic circles, as serious computational power began moving its way into mainstream desktop machines, standards like Microsoft Windows and Excel made design and maintenance of complex models easier. The use of real numbers rather than bit string representations eliminated the difficult task of encoding and decoding chromosomes.

The popularity of the genetic algorithm is now growing exponentially, with seminars, books, magazine articles, and knowledgeable consultants popping up everywhere. The International Conference of Genetic Algorithms is already focusing on practical applications, a sign of maturity that eludes other "artificial intelligence" technologies. Many Fortune 500 companies employ genetic algorithms regularly to solve real-world problems, from brokerage firms to power plants, phone companies, restaurant chains, automobile manufacturers and television networks. In fact, there is a good chance that you have already indirectly used a genetic algorithm before!

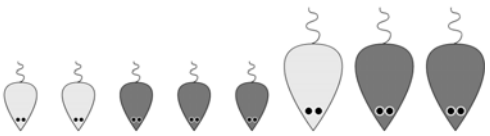
A Biological Example

Let us look at a simple example of evolution in the biological world (on a small scale). By "evolution" here we mean any change in the distribution or frequency of genes in a population. Of course, the interesting thing about evolution is that it tends to lead to populations that are constantly adapting to their environments.

Imagine that we are looking at a population of mice. These mice exhibit two sizes, small and large, and they exhibit two colors, light or dark. Our population consists of the following eight mice:



One day, cats move into the neighborhood and start eating mice. It turns out that darker mice and smaller mice are harder for the cats to find. Thus, different mice have different odds of avoiding the cats long enough to reproduce. This affects the nature of the next generation of mice. Assuming the old mice die soon after reproducing, the next generation of mice looks like this:



Notice that large mice, light mice, and especially large, light mice, are having trouble surviving long enough to reproduce. This continues in the next generation.

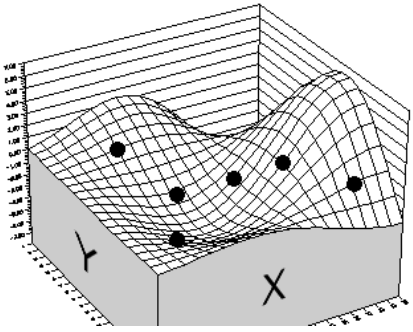
Now the population consists mostly of small, dark mice, because these mice are better suited for survival in this environment than other kinds of mice. Similarly, as the cats begin to go hungry with less mice to eat, perhaps those cats who prefer a lunch of grass will be better adapted, and pass along their grass-loving gene to a new

generation of cats. This is the central concept of "survival of the fittest". More precisely, it could be phrased "survival until reproduction". In evolutionary terms, being the healthiest bachelor in the population is worthless, since you must reproduce in order for your genes to influence future generations.

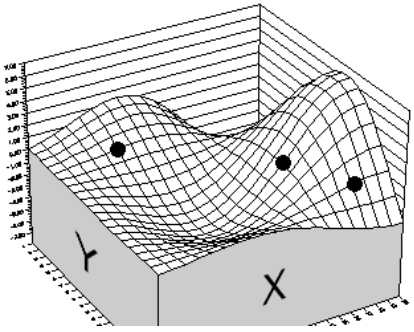
A Digital Example

Imagine a problem with two variables, X and Y, that produce a result Z. If we calculated and plotted the resulting Z for every possible X and Y values, we would see a solution "landscape" emerge (discussed also in Chapter 6: Optimization). Since we are trying to find the maximum "Z", the peaks of the function are "good" solutions, and the valleys are "bad" ones.

When we use a genetic algorithm to maximize our function, we start by creating several possible solutions or scenarios at random (the black dots), rather than just one starting point. We then calculate the function's output for each scenario and plot each scenario as one dot. Next we rank all of the scenarios by altitude, from best to worst. We keep the scenarios from the top half, and throw out the others.



First, create a whole "population" of possible solutions. Some will be better (higher) than others.

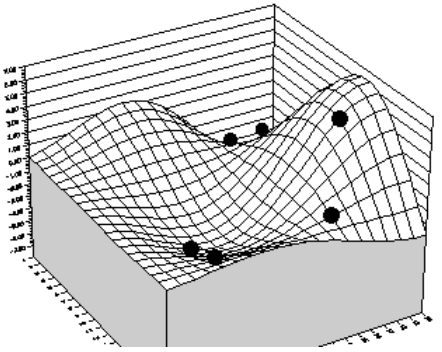


Next we rank them all and keep the solutions which yield better results.

Each of the three remaining scenarios duplicates itself, bringing the number of scenarios back up to six. Now comes the interesting part: Each of the six scenarios is made up of two adjustable values (plotted as an X and a Y coordinate). The scenarios pair off with each other at random. Now each scenario exchanges the first of its two adjustable values with the corresponding value from its partner. For example:

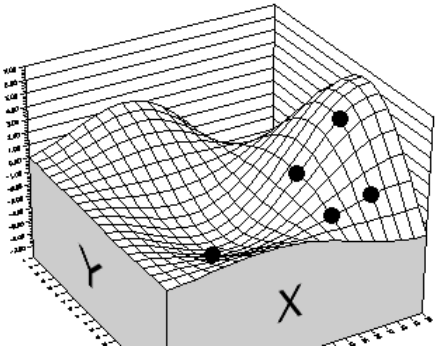
	Before	After
Scenario 1	3.4, 5.0	2.6, 5.0
Scenario 2	2.6, 3.2	3.4, 3.2

This operation is called crossing over, or ***crossover***. When our six scenarios randomly mate and perform crossover, we may get a new set of scenarios such as this:



In the above example, we assume that the original three scenarios, a, b, and c, paired up with the duplicates, A, B, C, to form the pairs aB, bC, bA. These pairs then switched values for the first adjustable cell, which is equivalent in our diagram to exchanging the x and y coordinates between pairs of dots. The population of scenarios has just lived through a generation, with its cycle of "death" and "birth".

Notice that some of the new scenarios result in lower output (lower altitude) than any we saw in the original generation. However, one scenario has moved high up on the tallest hill, indicating progress. If we let the population evolve for another generation, we may see a scene like the following:



You can see how the average performance of the population of scenarios increases over the last generation. In this example, there is not much room left for improvement. This is because there are only two genes per organism, only six organisms, and no way for new genes to be created. This means there is a limited ***gene pool.*** The gene pool is the sum of all the genes of all organisms in the population.

Genetic algorithms can be made much more powerful by replicating more of the inherent strength of evolution in the biological world; increasing the number of genes per organism, increasing the number of organisms in a population, and allowing for occasional, random mutations. In addition, we can choose the scenarios that will live and reproduce more like they occur naturally: with a random element that has a slight bias towards those that perform better, instead of simply choosing the best performers to breed (even the biggest and strongest lion may get hit with lightning)!

All of these techniques stimulate genetic refinement, and help to maintain diversity in the gene pool, keeping all kinds of genes available in case they turn out to be useful in different combinations. Evolver automatically implements all of these techniques.

Chapter 8: Evolver Extras

Adding Constraints

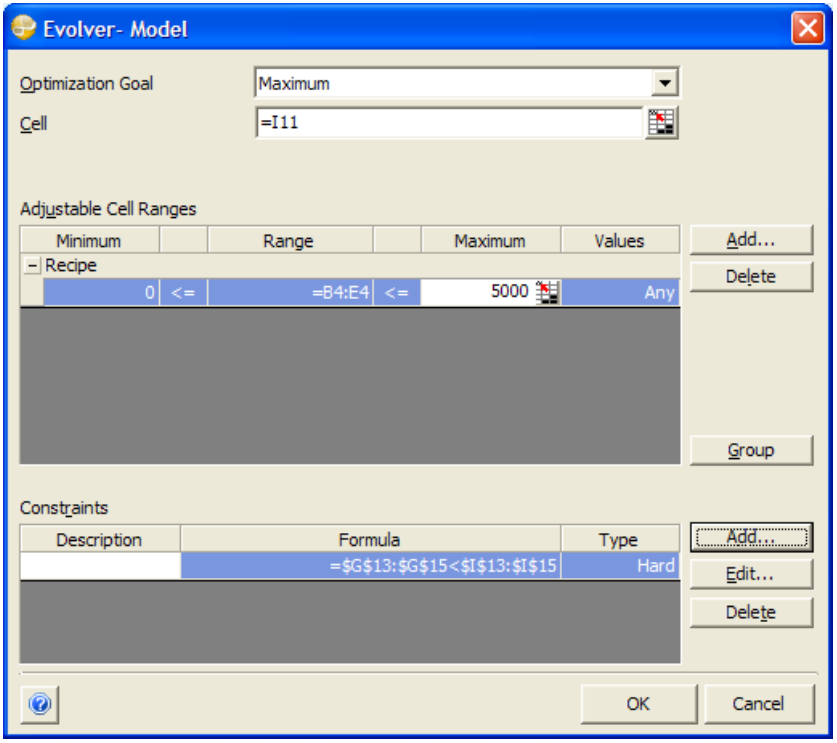
Realistic problems often have a number of constraints that must be met while we search for optimal answers. For example, in the tutorial which seeks the transformer design with the lowest cost, one of the constraints is that the transformer must remain cool, radiating no more than 0.16 watts/cm^2.

A scenario which meets all the constraints in a model is said to be a viable or "valid" solution. Sometimes it is difficult to find viable solutions for a model, much less to find the optimal viable solution. This may be because the problem is very complex, and only has a few viable solutions, or because the problem is over-specified (there are too many constraints, or some constraints conflict with others), and there are no viable solutions.

There are three basic kinds of constraints: ***range*** constraints, or min-max ranges placed on adjustable cells, ***hard*** constraints, which must always be met, and ***soft*** constraints which we would like to be met as much as possible, but which we may be willing to compromise for a big improvement in fitness.

Range Constraints

The simplest hard constraints are the ones that are placed on the variables themselves. By setting a certain range on each variable, we can limit the overall number of possible solutions Evolver will search through, resulting in a more efficient search. Enter Min and Max values in the Model window's Adjustable Cell Ranges section to tell Evolver the range of values that are acceptable for each variable.



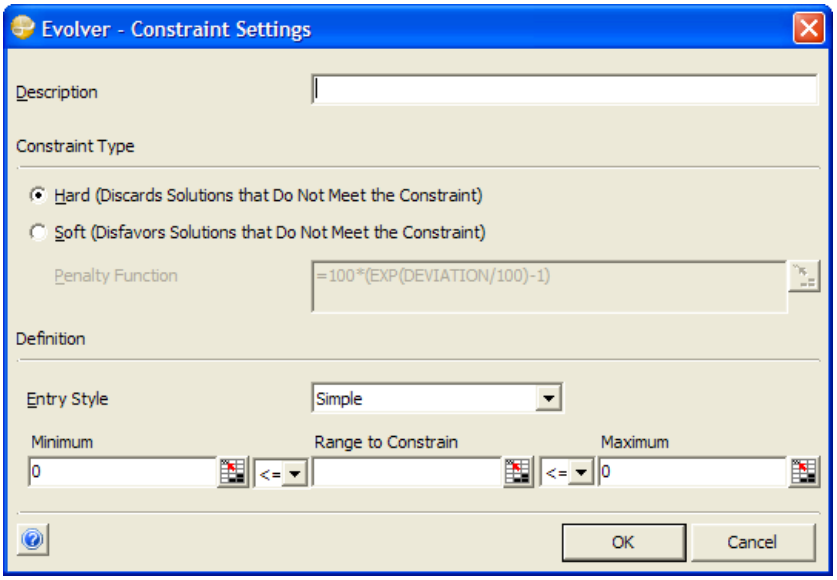
Evolver will only try values between 0 and 5,000 for the specified cells.

A second type of hard constraint placed on the variables is built in to each of Evolver's solving methods (recipe, order, grouping, etc.). For example, when we adjust variables using the budget solving method, that means Evolver is hard constrained to try only sets of values that add up the same amount. Like the Ranges setting, this hard constraint also reduces the number of possible scenarios that must be searched.

The integer option in the Model dialog box is also a hard constraint, telling Evolver to try only integer values (1, 2, 3 etc.) instead of real numbers (1.34, 2.034, etc.) when adjusting the variable values.

Hard Constraints - customized

Any constraint that falls outside the Evolver variable constraints can be entered using the Constraint Settings dialog.



***NOTE:** Like evolution in nature, a genetic algorithm's problem-solving power lies primarily in its ability to freely explore many combinations of likely solutions, and naturally lean towards the best ones. If we forbid* **Evolver** *to even look at solutions that do not meet our demands, the genetic algorithm optimization process can be crippled.*

It is always easier for Evolver to find solutions that meet the hard constraints if the initial scenario in the worksheet does itself meet the constraints. That lets Evolver know a starting point in the space of valid solutions. If you do not know of a scenario which meets the constraints, run Evolver with any initial scenario and it will do its best to find scenarios which meet the constraints.

Soft Constraints

Forcing a program to find only solutions that meet all constraints can result in no viable solutions being found. Often, it is more useful to have an approximately viable solution, where maybe a few solutions fall short of meeting the constraints.

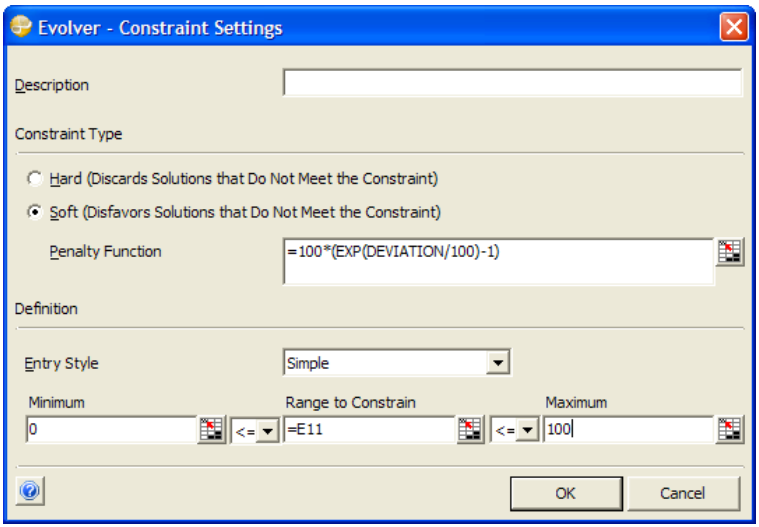
An alternative to the use of "hard constraints" that must be met is to reconfigure the problem with "soft constraints"; constraints that Evolver will ***tend to meet***. These soft constraints are often more realistic, and allow Evolver to try many more options. In the case of a highly constrained problem (where there are not very many possible solutions that would meet all your requirements), Evolver's genetic algorithm will be more likely to find the best solution if it is allowed to get feedback on some solutions that are *close* to satisfying the constraints.

When constraints are design goals, such as "produce twice as many forks as knives", it is often not so important to meet them exactly: especially if getting a perfectly balanced production schedule required a day-long optimization process. In this case, a good solution to the problem, that *almost* meets the constraint (production is 40% forks, 23% knives, 37% spoons), is usually better than waiting all day to find out that maybe there is no solution, because *all* the constraints could not possibly be met.

Penalty Functions

Soft constraints can easily be implemented in Excel through the use of ***penalty functions***. Instead of telling Evolver that it absolutely cannot use certain values when looking for solutions, we allow those "invalid" values to be explored, but we will penalize such solutions accordingly. For example, your problem may involve finding the most efficient way to distribute goods with the constraint that you use only three trucks. A more accurate model would include a penalty function that allowed you to use more trucks, but added the tremendous cost to the bottom line. Penalty functions can be specified in the Constraint Settings dialog or entered directly in your model by adding formulas to represent the penalty functions.

Entering a Penalty Function

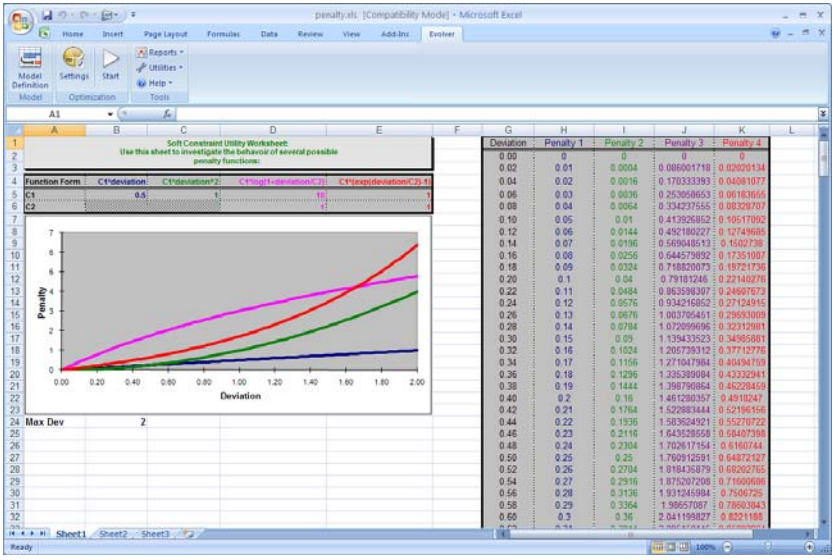


Evolver has a default penalty function which is displayed when you first enter a soft constraint. Any valid Excel formula, however, may be entered to calculate the amount of penalty to apply when the soft constraint is not met. An entered penalty function should include the keyword *deviation* which represents the absolute amount by which the constraint has gone beyond its limit. At the end of a trial solution Evolver checks if the soft constraint has been met; if not, it places the amount of deviation in the entered penalty formula and then calculates the amount of penalty to apply to the target cell value that is being minimized or maximized.

The penalty amount is either added or subtracted from the value for the target cell in order to make it less "optimal." For example, if *Maximum* is selected in the *Find the* field in the Evolver Model Dialog, the penalty is subtracted from the value for the target cell.

Viewing the Effects of an Entered Penalty Function

Evolver includes an Excel worksheet PENALTY.XLS which can be used to evaluate the effects of different penalty functions on specific soft constraints and target cell results.



PENALTY.XLS allows you to select a soft constraint from your model whose effects you wish to analyze. You can then change the penalty function to see how the function will map a specific value for the unmet soft constraint into a value for the target cell. For example, if your soft constraint is A10<100, you could use PENALTY.XLS to see what the target value would be if a value of 105 was calculated for cell A10.

Viewing the Penalties Applied

When a penalty is applied to the target cell due to an unmet soft constraint, the amount of penalty applied can be viewed in the Evolver Watcher. In addition, penalty values are shown in Optimization Log worksheets, created optionally after optimization.

Entering Soft Constraints In Your Worksheet

Penalty functions may also be entered directly in your worksheet. A Boolean penalty function will assign a set penalty on any scenario which does not meet the specified constraint. For example, if you wanted the value in cell B1(supply) to be at least as great as the value in cell A1(demand), you could create this penalty function in another cell: *=IF(A1>B1, -1000, 0)*. If the result of this cell were added to the value for the target cell, than every time Evolver tried a solution which violated that constraint (i.e. the supply did not meet the demand), the value for the target cell being maximized would show a value 1,000 lower than the real result. Any solution which violated this constraint would produce a low value for the value for the target cell, and eventually Evolver would "breed out" these organisms.

You can also use a scaling penalty function, which more accurately penalizes the solution relative to how badly it violates the constraint. This is often more practical in the real world, because a solution where supply did not quite meet demand would be better than a solution where supply didn't even come close to the demand. A simple scaling penalty function computes the absolute difference between the constraint's goal value and it's actual value. For example, in the same problem where A1(demand) should not exceed B1(supply), we could assign the following penalty function: *=IF(A1>B1, (A1-B1)^2, 0)*. This kind of penalty function measures how close a constraint is to being met, and exaggerates that difference by squaring it. Now our penalty changes based on how badly a solution violates the constraint.

More Examples of Penalty Functions

For example, suppose you have created a manufacturing model where one of the constraints is that the amount of wood used should be equal to the amount of plastic used. This constraint is met when "AmountWood" = "AmountPlastic". We want to find solutions that include the same amount of both materials, so we create a penalty function to discourage solutions that stray from our goal. The formula "=ABS(AmountWood-AmountPlastic)" computes the absolute (non-negative) difference between the amount of wood and the amount of plastic being used. By using the ABS() function, we arrive at the same penalty value if AmountWood is 20 greater than AmountPlastic, or if AmountPlastic is 20 less than AmountWood. Now when we optimize the model, our goal is to minimize this absolute difference.

Suppose instead we impose the following constraint: The amount of wood must be twice the amount of plastic. The penalty function would then be:

```
=ABS(AmountWood-AmountPlastic*2)
```

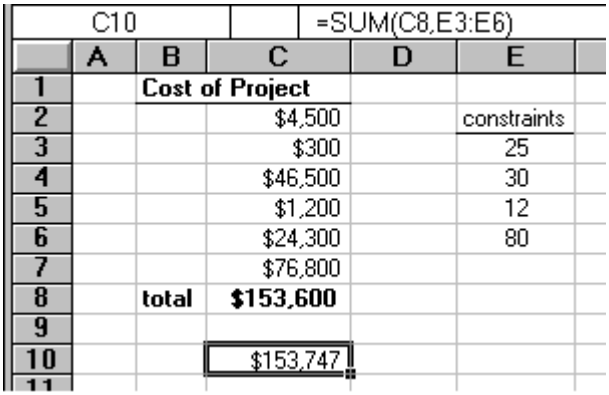
A different possible constraint is that the amount of wood should be ***no less than*** twice the amount of plastic. While the previous example produced a penalty if there was too much wood, in this case we only care if there is not enough wood; if AmountWood is ten times AmountPlastic, we want no penalty to be applied. The appropriate penalty function would then be:

```
=IF(AmountWood<AmountPlastic*2,
ABS(AmountPlastic*2-AmountWood),0)
```

If AmountWood is at least twice as great as AmountPlastic, the penalty function returns 0. Otherwise, it gives a measure of how much less than twice AmountPlastic the AmountWood value is.

Using Penalty Functions

After you have created penalty functions to describe the soft constraints in your model, you can combine them with your normal target cell formula to obtain a constrained target cell formula. In the example illustrated below, if cell C8 computes the total cost of a project, and cells E3:E6 contain five penalty functions, then you can create a formula in cell C10 such as =SUM(C8, E3:E6).



C10		=SUM(C8,E3:E6)		
	A	B	C	D	E
1		**Cost of Project**			
2			$4,500		constraints
3			$300		25
4			$46,500		30
5			$1,200		12
6			$24,300		80
7			$76,800		
8		**total**	**$153,600**		
9					
10			$153,747		

Create a cell that adds the constraints to your total, and minimize the values for this cell.

This adds the penalties in column E to the cost in C8 to obtain a constrained or penalized cost function in C10. Note that if this were a maximization problem, you would subtract, rather than add, the penalties to the original target cell. Now when you use Evolver, you simply select this constrained cell, C10, as the target cell to be whose value will be optimized.

When Evolver tries to optimize a constrained value for the target cell, the penalty functions will tend to force the search towards scenarios that meet the constraints. Eventually Evolver will end up with solutions that are good answers and that meet or nearly meet all constraints (the penalty functions will have values near 0).

Multiple Goal Problems

You may only specify one cell in the target cell field of Evolver, but you can still solve for multiple goals by creating a function that combines the two goals into one goal. For example, as a polymer scientist, you may be trying to create a substance that is flexible, but also strong. Your model computes the resulting strength, flexibility and weight that would result from a given mix of chemical combinations. The amounts of each chemical to use are the adjustable variables of the problem.

Since you want to maximize the Strength of the substance (in cell S3) but also maximize its Flexibility (in cell F3), you would create a new cell with the formula: =(S3+F3). This would be your new target cell, for the higher this number went, the better the overall solution.

If the flexibility was more important than the strength, we could change the formula in the target cell to read =(S3+(F3*2)). This way, scenarios which increased the flexibility by a certain amount would look better (produce a higher fitness "score") than scenarios which increased the strength by the same amount.

If you wanted to maximize the Strength of a substance (in cell S5) but also minimize its Weight (in cell W5), you would create a new cell with the following formula: =(S5^2)-(W5^2). This formula would produce a higher number when the structure was both strong-and-light, a lower number when the structure was weak-and-heavy, and equally average numbers for weak-but-light and strong-but-heavy scenarios. You would therefore use this new cell as your target, and maximize its mean to satisfy both goals.

Improving Speed

When you use Evolver to solve a problem, you are using both the Evolver library of compiled routines to control the process and Excel's spreadsheet evaluation function to examine different scenarios. A large percentage of the time used by Evolver is actually used by Excel as it recalculates your spreadsheet. There are a number of things that can be done to speed up Evolver optimization and Excel's recalculation process.

- The speed of Evolver is directly related to the speed of your computer processor. A Pentium/2.0ghz will be roughly twice as fast as the Pentium/1.0ghz. This means that Evolver will be able to evaluate twice as many trials in the same amount of time.
- Try to avoid re-drawing in your window. Drawing graphics and numbers on the screen takes time, sometimes more than half the time spent optimizing! If you have charts or graphs on the sheet, they will slow down the re-calculate time significantly. You can tell Excel not to spend time drawing while Evolver is solving a problem by turning off the *Update Display* option in the Evolver Model Dialog or by minimizing the Excel sheet. You can see how much faster your problem is working by watching the status bar.
- Once Evolver has more or less converged on a solution, and there has been no improvement on the best solution in a while (e.g. last 1000 trials), you may want to increase the mutation rate to allow Evolver to broaden its search for solutions, rather than continuing to refine solutions in the current population using primarily crossover. You can increase mutation rate through the Evolver Watcher using the Population Settings command.
- Set more tightly the ranges that the adjustable cells must fall between; this will create a smaller area in which Evolver must search for solutions, and should therefore speed up the process. Make sure that your ranges allow enough freedom for Evolver to explore all realistic solutions.

How Evolver's Optimization is Implemented

In this section we describe more specifically how Evolver's optimization algorithms are implemented.

***NOTE:** You do not need to know this material in order to use* **Evolver.**

The majority of Evolver's genetic algorithm technology such as the ***recipe*** and ***order*** solving methods are based on academic work in the genetic algorithm field over the last ten years. However, most of the descendant solving methods included with Evolver, and the multiple groups of adjustable cells, backtracking, strategy, and probability features are unique to Evolver.

Evolver uses a steady-state approach. This means that only one organism is replaced at a time, rather than an entire "generation" being replaced. This steady state technique has been shown to work as well or better than the generational replacement method. To find out the equivalent number of "generations" Evolver has run, take the number of individual trials it has explored and divide that by the size of the population.

Selection

When a new organism is to be created, two parents are chosen from the current population. Organisms that have high fitness scores are more likely to be chosen as parents.

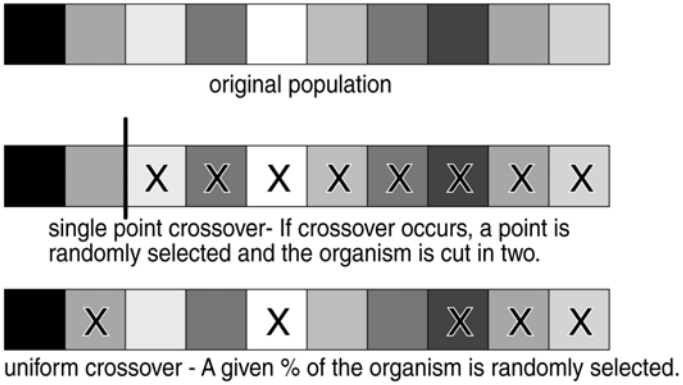
In Evolver, parents are chosen with a rank-based mechanism. Instead of some genetic algorithm systems, where a parent's chance to be selected for reproduction is directly proportional to its fitness, a ranking approach offers a smoother selection probability curve. This prevents good organisms from completely dominating the evolution from an early point.

Crossover

Since each solving method adjusts the variables in different ways, Evolver employs a different crossover routine optimized for that type of problem.

The basic recipe solving method performs crossover using a uniform crossover routine. This means that instead of chopping the list of variables in a given scenario at some point and dealing with each of the two blocks (called "single-point" or "double-point" crossover), two groups are formed by randomly selecting items to be in one group or another. Traditional *x*-point crossovers may bias the search with the irrelevant position of the variables, whereas the uniform

crossover method is considered better at preserving schema, and can generate any schema from the two parents.



The order solving method performs crossover using a similar algorithm to the order crossover operator described in L. Davis' *Handbook of Genetic Algorithms*.[*] This selects items randomly from one parent, finds their place in the other parent, and copies the remaining items into the second parent in the same order as they appear in the first parent. This preserves some of the sub-orderings in the original parents while creating some new sub-orderings.

Mutation

Like crossover, mutation methods are customized for each of the different solving methods. The basic recipe solving method performs mutation by looking at each variable individually. A random number between 0 and 1 is generated for each of the variables in the organism, and if a variable gets a number that is less than or equal to the mutation rate (for example, 0.06), then that variable is mutated. The amount and nature of the mutation is automatically determined by a proprietary algorithm. Mutating a variable involves replacing it with a randomly generated value (within its valid min-max range).

To preserve all the original values, the order solving method performs mutation by swapping the positions of some variables in the organism. The number of swaps performed is increased or decreased proportionately to the increase and decrease of the mutation rate setting (from 0 to 1).

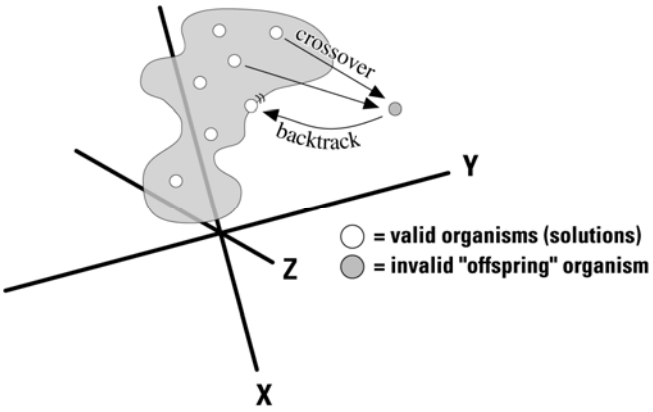
Replacement

Since Evolver uses a rank-ordered rather than generational replacement method, the worst-performing organisms are always replaced with the new organism that is created by selection, crossover, and mutation, regardless of its fitness "score".

[*] Davis, Lawrence (1991). Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold.

Constraints

Hard constraints are implemented with Palisade's proprietary "backtracking" technology. If a new offspring violates some externally imposed constraints, Evolver backtracks towards one of the parents of the child, changing the child until it falls within the valid solution space.



Appendix A: Automating Evolver

VBA

Evolver comes with a complete macro language for building custom applications which use Evolver's capabilities. Evolver's custom functions can be used in Visual Basic for Applications (VBA) for setting up and running optimizations and displaying the results from optimizations. For more information on this programming interface, see the Evolver Developer Kit help document, available via the Evolver Help menu.

Appendix B: Troubleshooting / Q&A

Troubleshooting / Q&A

This section answers some commonly asked questions regarding Evolver and keeps you up to date on common questions, problems and suggestions. After reading through this section, you may call Palisade customer support at the numbers listed in the beginning chapter of this manual.

Q: Why am I having trouble getting a valid answer from Evolver?

A: Make sure that the Evolver dialog is set up correctly. Most of the problems are associated with the setting of the variables. Each group of adjustable cells should be exclusive, in that no single cell or range of cells is being treated with more than one solving method.

Q: Can Evolver deal with concepts or categories instead of just numbers?

A: Evolver can indirectly deal with any kind of data, since numbers are just symbols. Use a lookup table in Excel to translate between integers and strings of text. Evolver (like all computer programs) ultimately can only deal with numbers, but your interface may use those numbers to represent and display any strings.

Q: Even though I'm filling in the dialogs the same way, and letting Evolver run the same amount of time, why does Evolver sometimes find different solutions?

A: As is the case with natural selection in the biological world, the Evolver genetic algorithm will not always follow the same path when searching for solutions (unless you use a fixed random number generator seed). Ironically it is this "unpredictability" that allows Evolver to solve more types of problems, and often find better solutions than traditional techniques. Evolver's genetic algorithm engine is not just executing a series of pre-programmed commands, or plugging values through a mathematical formula, but it is efficiently experimenting with many random hypothetical scenarios simultaneously, and then refining the search through many "survival-of-the-fittest" operators which also contain random elements.

Q: Why is the best solution found not changing?

A: You may have specified the wrong target cell in the Evolver Model Dialog. Evolver is looking at this blank cell and the value does not change because there is no formula. To fix this, display the Evolver Model Dialog and select a proper target cell; i.e. one that accurately reflects how good/bad each possible solution is. A proper target cell has a formula which depends, directly or indirectly, on the variables Evolver is adjusting (adjustable cells).

Q: Some of the cells in my spreadsheet model contain "####" symbols.

A: If the cell is too small to display all of its contents, it will display several #### signs. Increase the size of the cell.

Q: Evolver is working OK, but is there any simple way to get better results?

A: Consider loosening the constraints in the problem, including variable ranges. Change some of your hard constraints to soft constraints via penalty functions (see Adding Constraints in Chapter 8: Evolver Extras). Too many restrictions on what Evolver can try may be preventing Evolver from exploring an area of possibilities that may yield better results. Remember, the longer you let Evolver explore the possibilities, the more likely it is to find the optimal solution. For more ideas on how to fine-tune Evolver, see Chapter 8: Evolver Extras.

The more scenarios Evolver can run through, the better. Speed up the Evolver process by turning off the "Every Recalculation" option for display update.

Appendix C: Additional Resources

Additional Learning Resources

The following list represents a select sampling of genetic algorithm and artificial-life-related materials. A star (*) indicates a Palisade favorite.

Books

• Bolles, R.C., & Beecher, M.D. (Eds.). (1988). Evolution and Learning. Lawrence Erlbaum.

• Beer, R.D. (1990). Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology. Academic Press.

• Davis, Lawrence (1987). Genetic Algorithms and Simulated Annealing. Palo Alto, CA: Morgan Kaufman.

* Davis, Lawrence (1991). Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold.

• Darwin, Charles (1985). On The Origin of Species. London: Penguin Classics. (originally 1859)

* Dawkins, Richard. (1976). The Selfish Gene. Oxford University Press.

• Eldredge, N. (1989). Macroevolutionary Dynamics: Species, Niches, and Adaptive Peaks. McGraw-Hill.

• Fogel, L., Owens, J., and Walsh, J. (1966). Artificial Intelligence through Simulated Evolution. New York: John Wiley and Sons.

• Goldberg, David (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley Publishing.

• Holland, J.H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press.

• Koza, John (1992). Genetic Programming. Cambridge, MA: MIT Press.

* Langton, C.L. (1989). Artificial Life. MIT Press. [ALife I]

• Levy, Steven (1992). Artificial Life. New York: Pantheon.

• Meyer, J.-A., & S.W. Wilson (Eds.). (1991). Proceedings of the First International Conference on Simulation of Adaptive Behavior: From Animals to Animats. MIT Press/Bradford Books.

* Proceedings of the Sixth International Conference (ICGA) on Genetic Algorithms (1995). San Mateo, CA: Morgan Kaufman Publishing. (Also available; the first five ICGA proceedings).

• Proceedings of the Workshop on Artificial Life (1990). Christopher G. Langton, Senior Editor. Reading, MA: Addison-Wesley Publishing.

• Rawlins, Gregory (1991). Foundations of Genetic Algorithms. San Mateo, CA: Morgan Kaufman Publishing.

• Richards, R.J. (1987). Darwin and the Emergence of Evolutionary Theories of Mind and Behavior. U. Chicago Press.

• Williams, G.C. (1966). Adaptation and Natural Selection. Princeton U. Press.

Articles

* Antonoff, Michael (October, 1991). Software by Natural Selection. Popular Science, p. 70-74.

• Arifovic, Jasmina (January, 1994). Genetic Algorithm Learning and the Cobweb Model. In Journal of Economic Dynamics & Control v18 p.3

* Begley, S (May 8, 1995). "Software au Naturel" In Newsweek p. 70

• Celko, Joe (April, 1993). Genetic Algorithms and Database Indexing. In Dr. Dobb's Journal p.30

• Ditlea, Steve (November, 1994). Imitation of Life. In Upside Magazine p.48

• Gordon, Michael (June, 1991). User-based Document Clustering by Redescribing Subject Descriptions with a Genetic Algorithm. In Journal of the American Society for Information Science v42 p.311

• Hedberg, Sara (September, 1994). Emerging Genetic Algorithms. In AI Expert, p. 25-29.

• Hinton, G.E., & Nowlan, S.J. (1987). How Learning Can Guide Evolution. In Complex Systems 1: p.495-502.

* Kennedy, Scott (June, 1995). Genetic Algorithms: Digital Darwinism. In Hitchhicker's Guide to Artificial Intelligence Miller Freeman Publishers

• Kennedy, Scott (December, 1993). Five Ways to a Better GA. In AI Expert, p. 35-38

• Lane, A (June, 1995). The GA Edge in Analyzing Data. In AI Expert p.11

• Lee, Y.C. (Ed.). (1988). Evolution, learning, and cognition. In World Scientific.

• Levitin, G and Rubinovitz, J (August, 1993). Genetic Algorithm for Linear and Cyclic Assignment Problem. In Computers & Operations Research v20 p.575

• Marler, P., & H.S. Terrace. (Eds.). (1984). The Biology of Learning. Springer-Verlag.

• Mendelsohn, L. (December, 1994) Evolver Review In Technical Analysis of Stocks and Commodities. p.33

• Maynard Smith, J. (1987). When Learning Guides Evolution. In Nature 329: p.761-762.

- Murray, Dan (June, 1994). Tuning Neural Networks with Genetic Algorithms. In AI Expert p.27
- Wayner, Peter (January, 1991). Genetic Algorithms: Programming Takes a Valuable Tip from Nature. In Byte Magazine v16 p.361

Magazines & Newsletters

- Advanced Technology for Developers (monthly newsletter). Jane Klimasauskas, Ed., High-Tech Communications, 103 Buckskin Court, Sewickley, PA 15143 (412) 741-7699
- AI Expert (monthly magazine). Larry O'Brien, Ed., 600 Harrison St., San Francisco, CA 94107 (415) 905-2234. *Although AI Expert ceased publishing in the spring of 1995, its back issues contain many useful articles. Miller-Freeman, San Francisco.
- Applied Intelligent Systems (bimonthly newsletter). New Science Associates, Inc. 167 Old Post Rd., Southport, CT 06490 (203) 259-1661
- Intelligence (monthly newsletter). Edward Rosenfeld, Ed., PO Box 20008, New York, NY 10025-1510 (212) 222-1123
- PC AI Magazine (monthly magazine). Joseph Schmuller, Ed., 3310 West Bell Rd., Suite 119, Phoenix, AZ 85023 (602) 971-1869
- Release 1.0 (monthly newsletter). Esther Dyson, Ed., 375 Park Avenue, New York, NY 10152 (212) 758-3434
- Sixth Generation Systems (monthly newsletter). Derek Stubbs, Ed., PO Box 155, Vicksburg, MI, 49097 (616) 649-3592

Introduction to Simulation

If you are new to Simulation or if you would just like some more background information on the technique, the following books and articles might be helpful:

* Baird, Bruce F. Managerial Decisions Under Uncertainty: John Wiley & Sons, Inc. 1989.

* Clemen, Robert T. Making Hard Decisions: Duxbury Press, 1990.

- Hertz, D.B. "Risk Analysis in Capital Investment": HBR Classic, Harvard Business Review, September/October 1979, pp. 169-182.

- Hertz, D.B. and Thomas, H. Risk Analysis and Its Applications: John Wiley and Sons, New York, NY, 1983.

- Megill, R.E. (Editor). Evaluating and Managing Risk: PennWell Books, Tulsa, OK, 1984.

- Megill, R.E. An Introduction to Risk Analysis, 2nd Ed.: PennWell Books, Tulsa, OK, 1985.

- Morgan, M. Granger and Henrion, Max, with a chapter by Mitchell Small, Uncertainty: Cambridge University Press, 1990.

- Newendorp, P.D. Decision Analysis for Petroleum Exploration: Petroleum Publishing Company, Tulsa, Okla., 1975.

- Raiffa, H. Decision Analysis: Addison-Wesley, Reading, Mass., 1968.

Technical References to Simulation and Monte Carlo Techniques

If you would like a more in depth examination of simulation, sampling techniques and statistical theory, the following books may be useful:

- Iman, R. L., Conover, W.J. "A Distribution-Free Approach To Inducing Rank Correlation Among Input Variables": Commun. Statist.-Simula. Computa.(1982) 11(3), 311-334

* Law, A.M. and Kelton, W.D. Simulation Modeling and Analysis: McGraw-Hill, New York, NY, 1991,1982.

Rubinstein, R.Y. Simulation and the Monte Carlo Method: John Wiley and Sons, New York, NY, 1981.

Technical References to Latin Hypercube Sampling Techniques

If you are interested in the relatively new technique of Latin Hypercube sampling, the following sources might be helpful:

- Iman, R.L., Davenport, J.M., and Zeigler, D.K. "Latin Hypercube Sampling (A Program Users Guide)": Technical Report SAND79-1473, Sandia Laboratories, Albuquerque (1980).
- Iman, R.L. and Conover, W.J. "Risk Methodology for Geologic Displosal of Radioactive Waste: A Distribution - Free Approach to Inducing Correlations Among Input Variables for Simulation Studies": Technical Report NUREG CR 0390, Sandia Laboratories, Albuquerque (1980).
- McKay, M.D, Conover, W.J., and Beckman, R.J. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code": Technometrics (1979) 211, 239-245.
- Startzman, R. A. and Wattenbarger, R.A. "An Improved Computation Procedure for Risk Analysis Problems With Unusual Probability Functions": SPE Hydrocarbon Economics and Evaluation Symposium Proceedings, Dallas (1985).

Examples and Case Studies Using Simulation

If you would like to examine case studies showing the use of Simulation in real life situations, see the following:

Hertz, D.B. and Thomas, H. Practical Risk Analysis - An Approach Through Case Histories: John Wiley and Sons, New York, NY, 1984.

* Murtha, James A. Decisions Involving Uncertainty, An @RISK Tutorial for the Petroleum Industry: James A. Murtha, Houston, Texas, 1993

• Newendorp, P.D. Decision Analysis for Petroleum Exploration: Petroleum Publishing Company, Tulsa, Okla., 1975.

• Pouliquen, L.Y. Risk Analysis in Project Appraisal: World Bank Staff Occasional Papers Number Eleven. John Hopkins Press, Baltimore, MD, 1970.

* Trippi, Robert R. and Truban, Efraim, Neural Networks: In Finance and Investing: Probus Publishing Co., 1993

Glossary

For additional information on any term, refer to the Evolver index in the following chapter.

Algorithm A mathematically based step-by-step method of solving a certain kind of problem. All computer programs are built by combining many algorithms.

Adjustable Cell A spreadsheet cell whose value can be adjusted by Evolver to try to optimize the value of the target cell. An adjustable cell is a variable value and should always contain a simple number, rather than an equation.

Baby Solver *slang* Simple software programs that find the inputs which produce a desired output using a combination of linear programming techniques, or basic hill-climbing algorithms. Baby solvers often take guesses, then refine their answer to arrive at a "local" solution rather than a "global" solution.

Cell The cell is the basic unit of a spreadsheet in which data is stored. There are up to 256 columns and 16,000 rows, for a total of more than 4 million cells, in each Excel worksheet.

Constraints Constraints are conditions which should be met (soft constraints) or must be met (hard constraints) for a scenario to be considered valid.

Continuous Distribution A probability distribution where any value between the minimum and maximum is possible (has finite probability).
See discrete distribution

Crossover In a genetically based context, crossing over is an exchange of equivalent genetic material between homologous chromatids during meiosis. In Evolver, the term crossover is used to express the computational equivalent to crossing over, where an exchange between variables yields new combinations of scenarios.

Cumulative Distribution A cumulative distribution, or a cumulative distribution function, is the set of points, each of which equals the integral of a probability distribution starting at the minimum value and ending at the associated value of the random variable.
See cumulative frequency distribution, probability distribution

Cumulative Frequency Distribution

A cumulative frequency distribution is the term for the output and the input cumulative distributions of Evolver. A cumulative distribution is constructed by cumulating the frequency (progressively adding bar heights) across the range of a frequency distribution. A cumulative distribution can be an "upwardly sloping" curve, where the distribution describes the probability of a value less than or equal to any variable value. Alternatively, the cumulative curve may be a "downwardly sloping" curve, where the distribution describes the probability of a value greater than or equal to any variable value.
See cumulative distribution

Dependent Variable

A dependent variable is one that depends in some way on the values of other variables in the model under consideration. In one form, the value of an uncertain dependent variable can be calculated from an equation as a function of other uncertain model variables. Alternatively, the dependent variable may be drawn from a distribution based on the random number which is correlated with a random number used to draw a sample of an independent variable.
See independent variable

Deterministic

The term deterministic indicates that there is no uncertainty associated with a given value or variable.

Dialog

The window on a computer screen that requests the user to provide information. Also called dialog box. Evolver contains two major dialogs; the Evolver Model Dialog, and the Adjustable Cells Dialog.

Discrete Distribution

A probability distribution where only a finite number of discrete values are possible between the minimum and maximum.
See continuous distribution

Field

The basic unit of data entry. Depending on its field type, a field can contain text, pictures, or numbers. Most fields in the Evolver dialogs ask the user to input the location of spreadsheet cells, or options regarding how Evolver should behave.

Fitness Function

This is a formula which can calculate how good or bad any proposed solution is to a given problem. The term is often used in the genetic algorithm field as an analogy to “fitness” in biological selection. Designing an accurate fitness function is critical when using a genetic algorithm to solve a problem.

Functions

In Excel, a function is a pre-defined formula that takes a value, performs an operation, and returns a value. Excel contains hundreds of built-in formulas (like "SUM") that save time, space, and are faster. For example, instead of typing A1+ A2+ A3+ A4+ A5+ A6, you can type SUM(A1:A6) and get the same result.

Frequency Distribution

Frequency distribution is the proper term for the output probability distributions and the input histogram distributions (HISTOGRM) of Evolver. A frequency distribution is constructed from data by arranging values into classes and representing the frequency of occurrence in any class by the height of the bar. The frequency of occurrence corresponds to probability.

Genetic Algorithm

A procedure for improving results of some operation by repeatedly trying several possible solutions and reproducing and mixing the components of the better solutions. The process is inspired by, and crudely similar to, the process of evolution in the biological world, where the fittest survive to reproduce.

Generation

In the field of genetic algorithms, each completely new population of "offspring" solutions is a new "generation". Some genetic algorithm routines mate all members of a population at once, creating a whole new "generation" of offspring organisms that replaces the previous population. Evolver evaluates and replaces one organism at a time (rank-ordered) and thus does not use the term "generation" in its documentation. This steady state technique works as well as generational replacement.

Genotype

In biology, this is the genetic constitution of an individual. The term usually refers to the sum total of the individual's genes. In the study of GAs, genotype is used to describe the artificial "chromosome" that is evaluated as a possible solution to the problem.

Global Maximum

The largest possible value for a given function. Complex functions or models may have many local maxima but only one global maximum.

Group of Adjustable cells

Each set of variables, along with the way they will be treated, is one group of adjustable cells. Evolver will list all groups of adjustable cells in the variables section of the Evolver Model dialog. This architecture allows complex problems to be built and described as several groups of adjustable cells.

Hard Constraints

A constraint that must always be met. For example, the ranges for variables in a recipe problem are hard constraints; a variable set to range between 10 and 20 can never have a value less than 10 or greater than 20. See also *soft constraints*.

Higher Moments Higher moments are statistics of a probability distribution. The term generally refers to the skewness and kurtosis, the third and fourth moments respectively. The first and second moments are the mean and the standard deviation respectively. *See skewness, kurtosis, mean, standard deviation*

Hill-Climbing Algorithm An optimization procedure that starts from a given scenario and repeatedly moves the scenario in small steps in the direction that will most improve it. Hill-climbing algorithms are fast and simple, but have two drawbacks. First, much work may be needed to find the direction of most improvement. Second, the algorithms usually climb the nearest hill, or local maximum. This prevents the algorithm from finding the global maximum in a difficult problem.

Independent Variable An independent variable is one that does not depend in any way on the values of any other variable in the model under consideration. The value of an uncertain independent variable is determined by drawing a sample from the appropriate probability distribution. This sample is drawn without regard to any other random sample drawn for any other variable in the model.
See dependent variable

Iteration An iteration is one recalculation of the user's model during a simulation. A simulation consists of many recalculations or iterations. During each iteration, all uncertain variables are sampled once according to their probability distributions, and the model is recalculated using these sampled values.
Also known as a simulation trial

Kurtosis Kurtosis is a measure of the shape of a distribution. Kurtosis indicates how flat or peaked the distribution is. The higher the kurtosis value, the more peaked the distribution.
See skewness

Latin Hypercube Latin Hypercube sampling is a relatively new stratified sampling technique used in simulation modeling. Stratified sampling techniques, as opposed to Monte Carlo type techniques, tend to force convergence of a sampled distribution in fewer samples.
See Monte Carlo

Local Maximum The largest possible value for a given function within a given range of values. A local maximum exists at a set of values for variables in a function if slightly changing any or all of the variables' values produces a smaller result from the function. (Compare with global maximum).

Mean

The mean of a set of values is the sum of all the values in the set divided by the total number of values in the set. *Synonym: expected value*

Model

For the purposes of this manual, a model is a numeric representation, in Excel, of a real-world situation.

Monte Carlo

Monte Carlo refers to the traditional method of sampling random variables in simulation modeling. Samples are chosen completely randomly across the range of the distribution, thus necessitating large numbers of samples for convergence for highly skewed or long-tailed distributions.
See Latin Hypercube

Most Likely Value

The most likely value or mode is the value that occurs most often in a set of values. In a histogram and a result distribution, it is the center value in the class or bar with the highest probability.

Mutation

In the biological world, gene mutation is the source of variation needed for effective natural selection. Likewise, a genetic algorithm uses mutation techniques to maintain diversity in a population of possible scenarios.

Optimization

The process of finding values for variables so that the output of a function can be maximized (made as large as possible) or minimized (made as small as possible). Optimization by equation solving is easy for smoothly changing functions with few variables, but extremely difficult for many real-world problems. Tough problems generally need a search mechanism. Evolver uses an optimizing search mechanism based upon a genetic algorithm.

Organism

A block of memory in a population that stores a set of variable values (scenario).

Penalty Function

A spreadsheet equation that Evolver can use to penalize scenarios that fail to meet some criteria. Penalty functions are used to help minimize side effects from scenarios or to achieve multiple goals. Unlike a hard constraint, a penalty function does allow invalid solutions to be explored; it just makes those solutions look bad so the population will evolve away from those solutions. Boolean penalties are either on or off, penalizing all invalid solutions by the same amount. Scaling penalties are more fluid, assigning a penalty in proportion to how badly a constraint is violated.

Percentile

A percentile is an increment of the values in a data set. Percentiles divide the data into 100 equal parts, each containing one percent of the total values. The 60th percentile, for example, is the value in the data set for which 60% of the values are below it and 40% are above.

Phenotypes

In biology, this is an observable trait of an individual which arises from interactions between genes, and between genes and the environment. In the study of GAs, phenotype is used to describe the individual variables or "genes" that make up one complete solution or "chromosome". (see Genotype)

Population

The entire set of scenarios that Evolver keeps in memory from which new scenarios are generated. Evolver keeps one population of possible solutions for each group of adjustable cells in a system.

Probability

Probability is a measure of how likely a value or event is to occur. It can be measured from simulation data as frequency by calculating the number of occurrences of the value or event divided by the total number of occurrences. This calculation returns a value between 0 and 1 which then can be converted to percentage by multiplying by 100.
See frequency distribution, probability distribution

Probability Distribution

A probability distribution or probability density function is the proper statistical term for a frequency distribution constructed from an infinitely large set of values where the class size is infinitesimally small.
See frequency distribution

Random Number Generator

A random number generator is an algorithm for choosing random numbers, typically in the range of 0 to 1. These random numbers are equivalent to samples drawn from a uniform distribution with a minimum of 0 and a maximum of 1. Such random numbers are the basis for other routines that convert them into samples drawn from specific distribution types.
See random sample, seed

Random Sample

A random sample is a value that has been chosen from a probability distribution describing a random variable. Such a sample is drawn randomly according to a sampling "algorithm". The frequency distribution constructed from a large number of random samples drawn by such an algorithm will closely approximate the probability distribution for which the algorithm was designed.

Ranges

In Evolver:

The user sets the range, or the highest and lowest value that Evolver is allowed to try when adjusting a certain variable. Although this is not necessary to solve a problem, setting these ranges limits the possibilities and hence narrows Evolver's search.

In Excel:

A block of contiguous cells in a worksheet that is defined by the upper left cell and the lower right cell (e.g. A5:C9 describes a range of 15 cells).

Scenario A set of values for the variables in a spreadsheet model. Each scenario most often represents one possible solution.

Simulation Simulation is a technique whereby a model, such as a Excel worksheet, is calculated many times with different input values with the intent of getting a complete representation of all possible scenarios that might occur in an uncertain situation.

Skewness Skewness is a measure of the shape of a distribution. Skewness indicates the degree of asymmetry in a distribution. Skewed distributions have more values to one side of the peak or most likely value — one tail is much longer than the other. A skewness of 0 indicates a symmetric distribution, while a negative skewness means the distribution is skewed to the left. Positive skewness indicates a skew to the right. *See kurtosis*

Solution Any given system contains many input variables producing an output. In Evolver, a "solution" will more often refer to one of the possible combinations of variables rather than *the* best combination.

Soft Constraints When constraints do not necessarily have to be met, they can be made soft instead of hard. This is done by specifying a penalty function in Evolver or adding a penalty function to the target cell's fitness function.

It is often better for constraints to be soft if possible. This is because: 1. Evolver can usually solve softly-constrained problems faster, and 2. a soft-constraint model often will find a great solution that almost meets the soft constraints, which can be more valuable than a not-so-great solution that does meet hard constraints.

Solving Method Evolver includes six of these methods, each using a customized algorithm to solve a specific type of problem. For each set of variables selected in a problem, the user must assign the solving method to be used on those variables. The six solving methods are: grouping, order, recipe, budget, project, and schedule.

Standard Deviation The standard deviation is a measure of how widely dispersed the values are in a distribution. Equals the square root of the variance. *See variance*

Stochastic Stochastic is a synonym for uncertain, risky. *See risk, deterministic*

Status Bar The status bar appears at the bottom of the Excel window, and displays Evolver's current activity.

Survival of the Fittest	The idea that organisms better suited to an environment are more likely to live long enough to reproduce and spread their genes through the population's next generation.
Target Cell	The spreadsheet cell whose value we want to minimize or maximize. This cell is set in the Evolver Model dialog (select Evolver Model Definition command or the Model icon).
Trials	The process of Evolver generating a value for each variable in the problem, then recalculating the scenario for evaluation.

Index

P

R

S

T

V

W