COMPUTATIONAL CHALLENGES IN PORTFOLIO MANAGEMENT

The authors describe a relatively simple problem that all investors face—managing a portfolio of financial securities over time to optimize a particular objective function. They show how complex such a problem can become when real-world constraints are incorporated into its formulation.

The financial industry is one of the fastest-growing areas of scientific computing. Two decades ago, terms such as financial engineering, computational finance, and financial mathematics did not exist in common usage. Today, these areas are distinct and enormously popular academic disciplines with their own journals, conferences, and professional societies.

One explanation for this area's remarkable growth and the impressive array of mathematicians, computer scientists, physicists, and economists that are drawn to it is the formidable intellectual challenges intrinsic to financial markets. Many of the most basic problems in financial analysis are unsolved and surprisingly resilient to the onslaught of researchers from diverse disciplines.

In this article, we hope to give a sense of these challenges by describing a relatively simple problem that all investors face when managing a portfolio of financial securities over time. Such a problem becomes more complex once real-world considerations factor into its formulation. We present the basic dynamic portfolio optimization problem and then consider three aspects of it: taxes, investor preferences, and portfolio constraints. These three issues are by no means exhaustive—they merely illustrate examples of the kinds of challenges financial engineers face today. Examples of other computational issues in portfolio optimization appear elsewhere.[1,2]

1521-9615/01/$10.00 © 2001 IEEE

MARTIN B. HAUGH AND ANDREW W. LO
MIT Sloan School of Management and Operations Research Center

The portfolio optimization problem

Portfolio optimization problems are among the most studied in modern finance,[3] yet they continue to occupy the attention of financial academics and industry professionals because of their practical relevance and their computational intractabilities. The basic dynamic portfolio optimization problem consists of an individual investor's decisions for allocating wealth among various expenditures and investment opportunities over time, typically the investor's expected lifetime. The prices and price dynamics of goods and financial securities he or she purchases and any constraints such as tax liabilities, loan repayment provisions, income payments, and other cash inflows and outflows determine the investor's overall budget.

For expositional clarity, we start with a simple framework in which there are only two assets

available to the investor: a bond that yields a riskless rate of return and a stock that yields a random return of either 10 percent or –10 percent with probability p and $1 - p$, respectively. We denote the prices of the bond and stock at date t with B_t and S_t, respectively. Without loss of generality and for notational simplicity, we assume that S_0 = \$1 and the riskless rate of interest is 0 percent, so $B_t = 1$ for all $t \geq 0$. Finally, suppose that the investor's horizon spans only three dates, $t = 0$, 1, and 2, so that the possible paths for the evolving stock price S_t are given as in Figure 1. Of course, in practice, an investor has many assets to choose from over many dates, and the price of each asset can take on many values. For illustrative purposes, though, this simpler specification is ideal because it contains all the essential features of the dynamic portfolio optimization problem in a basic setting. Nevertheless, even in this simple framework, it will become apparent that practical considerations such as taxes, investor preferences, and portfolio constraints can create surprisingly difficult computational challenges.

Let C_t denote the value of the investor's consumption expenditures at date t, and let W_t denote the investor's wealth just prior to date-t consumption. We assume that the investor has a lifetime utility function $U(C_0, C_1, C_2)$ defined over each consumption path $\{C_0, C_1, C_2\}$ that summarizes how much he or she values the entire path of consumption expenditures. Then, ignoring market frictions and assuming that the investor's utility function is time-additive and time-homogenous—for example,

$$U(C_0, C_1, C_2) = u(C_0) + u(C_1) + u(C_2), \tag{1}$$

the investor's dynamic portfolio optimization problem at $t = 0$ is given by

$$V_0(W_0) = \underset{C_0, C_1, C_2}{\text{Max}} \, \mathrm{E}_0\left[u(C_0) + u(C_1) + u(C_2)\right] \tag{2}$$

subject to

$$W_t - C_t = x_t S_t + y_t B_t, \quad t = 0, 1, 2 \tag{3a}$$
$$W_{t+1} = x_t S_{t+1} + y_t B_{t+1}, \quad t = 0, 1 \tag{3b}$$
$$C_t \geq 0, \quad t = 0, 1, 2 \tag{3c}$$
$$x_t, y_t \in \mathbb{Z}^+, \quad t = 0, 1, 2 \tag{3d}$$
$$x_2 = y_2 = 0 \tag{3e}$$

where $\mathbb{Z}^+$ denotes the nonnegative integers, and x_t and y_t are the number of shares of stocks and bonds, respectively, that the investor holds in his or her portfolio immediately after date t. Of course, x_t and y_t may only depend on the information available at time t, a restriction that we impose throughout this article.

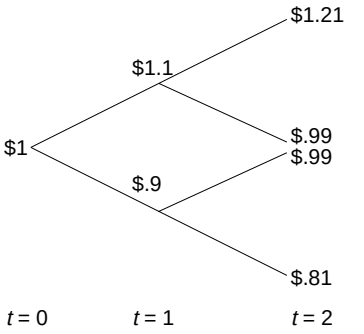


Figure 1. Stock price evolution.

The requirement that x_t and y_t are nonnegative means that borrowing and short sales are not allowed, a constraint that many investors face. That x_t and y_t are required to be integers simply reflects the fact that it is not possible to purchase a fractional number of stocks or bonds. The constraint in Equation 3b states that W_{t+1} is equal to W_t multiplied by the portfolio's gross return between dates t and $t + 1$.

We can easily solve this problem numerically by using the standard technique of stochastic dynamic programming.[4] In particular, because $V_2(W_2) = u(W_2)$, we can compute $V_1(W_1)$ using the Bellman equation so that

$$V_1(W_1) = \underset{C_1}{\text{Max}}\left\{u(C_1) + \mathrm{E}_1\left[V_2(W_2)\right]\right\} \tag{4}$$

subject to the constraints in Equation 3. An aspect of Equation 4 that makes it particularly easy to solve is the fact that the value function $V_1(\bullet)$ depends on only one state variable, W_1. This enables us to solve Equation 4 numerically without too many computations. Suppose, for example, that W_0 = \$1,000. Then, because $x_t, y_t \in \mathbb{Z}^+$, there are only 1,100 possible values that W_1 can take, so the right side of Equation 4 must be evaluated for only these 1,100 values. In contrast, when there are market frictions or when the investor has a more complex utility function, the computational requirements increase dramatically, reflecting Bellman's "curse of dimensionality."[4]

Taxes

Most seasoned investors are painfully aware of the substantial impact that taxes can have on the performance of their investment portfolio, so taxes play a major role in most dynamic portfolio optimization problems.[5–7]

To see how taxes can increase the computational complexity of such problems, let trading profits in the stock be subject to a capital gains tax in the portfolio optimization problem described earlier. Because this model has only two future periods, we do not distinguish between short-

term and long-term capital gains. For expositional simplicity, we also assume that we cannot use capital losses from one period to offset gains from a later one. Although these simplifying assumptions make the problem easier to solve, it will be apparent that the computations are considerably more involved than in the no-tax case.

To solve the dynamic portfolio optimization problem with taxes, we use dynamic programming as we did earlier. However, the value function is now no longer a function of wealth only, but it also depends on past stock prices and the number of shares purchased at each of those prices. In other words, the value function is now *path-dependent*. If we use $N_{s,t}$ to denote the number of shares of stock purchased at date $s \le t$ and that are still in the investor's portfolio immediately after trading at date t, then we can express the portfolio optimization problem as

$$V_0(W_0) = \underset{C_0, C_1, C_2}{\text{Max}} \; \mathrm{E}_0\left[u(C_0) + u(C_1) + u(C_2)\right] \quad (5)$$

subject to

$$W_t - C_t - \text{Max}\left\{0, \sum_{s=0}^{t-1} \tau(S_t - S_s)(N_{s,t-1} - N_{s,t})\right\} = x_t S_t + y_t B_t, \; t = 0, 1, 2 \quad (6a)$$

$$W_{t+1} = x_t S_{t+1} + y_t B_{t+1}, \quad t = 0, 1 \quad (6b)$$

$$x_t = \sum_{s=0}^{t} N_{s,t}, \; t = 0, 1, 2 \quad (6c)$$

$$C_t \ge 0, \quad t = 0, 1, 2 \quad (6d)$$

$$N_{0,0} \ge N_{0,1} \ge N_{0,2} \ge 0 \quad (6e)$$

$$N_{1,1} \ge N_{1,2} \ge 0 \quad (6f)$$

$$x_t, y_t \in \mathbb{Z}^+, \quad t = 0, 1, 2 \quad (6g)$$

$$x_2 = y_2 = 0 \quad (6h)$$

where τ is the capital gains tax rate. When $t = 2$, the value function depends on $(W_2, S_1, S_2, N_{0,1}, N_{1,1})$, and we obtain the relation

$$V_2(W_2, S_1, S_2, N_{0,1}, N_{1,1}) = u\left(W_2 - \text{Max}\left\{0, \sum_{s=0}^{1} \tau(S_2 - S_s) N_{s,1}\right\}\right) \quad (7)$$

so that date-2 consumption is simply W_2 less any capital gains taxes that must be paid. At $t = 1$, the value function depends on $(W_1, S_1, N_{0,0})$, and we can write the Bellman equation as

$$V_1(W_1, S_1, N_{0,0}) = \underset{C_1}{\text{Max}}\left\{u(C_1) + \mathrm{E}_1\left[V_2(W_2, S_1, S_2, N_{0,1}, N_{1,1})\right]\right\}. \quad (8)$$

If we compare Equation 8 with Equation 4, we see that the presence of taxes has made the dynamic portfolio optimization problem considerably more difficult. Specifically, in solving Equation 4 numerically, $V_1(\bullet)$ is computed for only 1,100 possible values of W_1. In contrast, solving Equation 8 numerically requires the evaluation of $V_1(\bullet)$ for all possible combinations of $\{W_1, S_1, N_{0,0}\}$, of which there are 1,001,000!

(We can verify this by noting the one-to-one correspondence between $\{W_1, S_1, N_{0,0}\}$ and $\{C_0, S_1, N_{0,0}\}$, and counting the possible combinations of $\{C_0, S_1, N_{0,0}\}$. Assuming, as before, that W_0 = \$1,000, we see that there are 1,001 possible choices for C_0. If $C_0 = i$, then there are $1{,}000 - i$ possible values for $N_{0,0}$. For each combination of $(C_0, N_{0,0})$, there are two possible values of S_1. This means that in total, there are $2\sum_{i=0}^{1{,}000}(1{,}000 - i) = 1{,}001{,}000$ combinations of $\{W_1, S_1, N_{0,0}\}$.)

Even in a simple two-period two-asset model, the portfolio optimization problem with taxes becomes considerably more complex. Indeed, in the T-period N-asset case, it is easy to see that by date T, there are $\mathbf{O}(NT)$ state variables, and if each state variable can take m distinct values, then there will be $\mathbf{O}(m^{NT})$ possible states at date T. For an investor with a 20-year horizon, an annual trading interval, and a choice of 25 assets, and assuming that the state variables take on only four distinct values at the end of the horizon, the number of possible states at the end will be of the order 10^{301}. (The number of distinct values that a state variable can take on depends, of course, on the precise nature of the state variable. For example, for a binomial state variable that is not "recombining," the number of distinct values it can take at date t is 2^T; if it is recombining, this is reduced to $T + 1$. We use four distinct values only for illustrative purposes; in most practical applications, the number is considerably larger.)

Preferences

Another important aspect of portfolio optimization problems is the objective function that represents the investor's preferences. Traditionally, these preferences are represented by time-additive time-homogeneous utility functions, which yield important computational advantages because they imply that the value function at date t does not depend on the investor's consumption choices prior to date t.

Unfortunately, the assumptions of time-additivity and time-homogeneity seem to be inconsistent with the empirical evidence on the consumption and portfolio choices of investors. For example, in-

dividuals tend to grow accustomed to their level of consumption over a period of time, implying that preferences depend not only on today's consumption level but also on levels of past consumption.[8]

Commonly known as *habit formation*, such preferences imply that the value function $V_t(\bullet)$ is a function of $(C_0, C_1, ..., C_{t-1})$ in addition to any other relevant state variables. As in the case with taxes, these problems quickly become intractable as the number of time periods increases. Of course, closed-form solutions are available for a few highly parameterized models of habit formation,[9] but in general these models must be solved numerically.[10] There are many other empirical regularities of investors' preferences that can induce path dependence in the value function, and for each of these cases, the computational demands quickly become intractable.

Portfolio constraints

When constraints are imposed on a portfolio optimization problem, their impact on the problem's computational complexity is not obvious. On the one hand, some unconstrained problems that admit closed-form solutions fail to do so once constraints are added. In practice, however, closed-form solutions are rarely available for realistic portfolio optimization problems, with or without portfolio constraints. We must solve such problems numerically, in which case, imposing constraints can sometimes reduce the number of computations because they limit the feasible region over which we must evaluate the value function. An example of this is the impact of the constraints in Equation 3 on the basic portfolio optimization problem. In that case, we imposed the constraint that x_t and y_t are nonnegative integers, eliminating the possibility of borrowing or shortselling. This implies that only a finite number of values for W_1 are possible, and as a result, the number of computations needed to evaluate $V_1(W_1)$ is greatly reduced.

On the other hand, in some cases constraints can greatly increase the number of computations, despite the fact that they limit the feasible set. This typically occurs when the constraints increase the problem's dimensionality. For example, in the basic portfolio optimization problem, consider imposing the additional constraint that the cumulative number of shares transacted—both purchased and sold—up to date t is bounded by some function, $f(t)$. (If 500 shares were purchased at $t = 0$ and 200 shares were sold at $t = 1$, the cumulative number of shares transacted as of date $t = 1$ is 700.)

In practice, these types of constraints are often imposed on investment funds to reduce transaction costs and the risk of churning. When such a constraint is imposed, the value function is no longer a function of only W_t but also of the cumulative number of shares transacted up to date t. By creating path dependence in the value function, constraints can substantially increase the computational complexity of even the simplest portfolio optimization problems.

Possible solution techniques

The most natural technique for solving dynamic portfolio optimization problems is stochastic dynamic programming. However, this approach is often compromised by several factors such as the curse of dimensionality when too many state variables are involved. In general, practical considerations such as taxes, transactions costs, indivisibilities and integer constraints, non-time-additive utility functions, and other institutional features of financial markets tend to create path dependencies in portfolio optimization problems—this increases the number of state variables in the value function. Such problems are difficult to solve in all but the simplest cases, with computational demands that become prohibitive as the number of time periods and assets increases.

Closed-form solutions are rarely available for realistic portfolio optimization problems.

Here, we briefly outline an alternative that might produce good approximate solutions to otherwise intractable portfolio optimization problems. This approach—called approximate dynamic programming, neuro-dynamic programming, or reinforcement learning—has had much success recently in solving challenging dynamic optimization problems in several contexts, including financial economics.[11–16] Although there are many different algorithms that we could categorize as "approximate dynamic programming," we will look at just one such algorithm: approximate value iteration.

Suppose the optimal value function at date t of a T-period dynamic optimization problem is given by $V_t(X_t)$ where $X_t \in \mathbb{R}^n$ is an n-dimensional vector of state variables. We assume that because of the computational intractabilities, it is impossible to determine $V_t(\bullet)$ exactly. Therefore,

we define a parameterized class of functions

$$\left\{\tilde{V}_t(X_t;\beta_t):\beta_t \in \mathbb{R}^p\right\} \quad (9)$$

which is called the approximation architecture. We then select our estimate $\tilde{V}_t$ of V_t from this class of functions by selecting some β_t from $\mathbb{R}^p$, the space of p-dimensional real-valued vectors.

Now suppose that we have applied the backward recursion of the Bellman equation to obtain estimators $\tilde{V}_{t+1},...,\tilde{V}_T$. Then we can use an approximate Bellman equation to compute an estimator of the value function at time t so that

$$\hat{V}_t(X_t) = \text{Max } \mathrm{E}_t\left[g_t(X_t) + \tilde{V}_{t+1}\left(X_{t+1};\hat{\beta}_{t+1}\right)\right] \quad (10)$$

where $g_t(X_t)$ is the reward at time t, possibly representing consumption as in our earlier examples, and where the maximization is with respect to the decision variables (suppressed for notational simplicity). Computing $\hat{V}_t(\bullet)$ often entails extensive Monte Carlo simulations because it is generally not possible to compute expectations over a high-dimensional space. In these cases, we estimate $\hat{V}_t(\bullet)$ at a fixed number of "training points" $(P_1, ..., P_m)$, where each $P_i \in \mathbb{R}^n$ represents a possible realization of the state vector X_t at date t. Once we have estimated $\hat{V}_t(P_i)$ for $i = 1, ..., m$, we obtain $\tilde{V}_t(\bullet)$ by solving the following least-squares problem:

$$\hat{\beta}_t = \arg\operatorname*{Min}_{\beta_t} \sum_{i=1}^{m} \left(\hat{V}_t(P_i) - \tilde{V}_t(P_i;\beta_t)\right)^2 . \quad (11)$$

With $\tilde{V}_t$ now determined, we then proceed to compute $\tilde{V}_{t-1}$ in a similar fashion and continue in this manner of approximate value iteration until we find $\tilde{V}_0$. The computational requirements of this algorithm might be considerably less demanding than if we were to solve the problem exactly. This is because we now only estimate the value function at a small representative subset of the state space rather than computing it exactly at all points in the state space.

Once we implement an approximate dynamic programming algorithm, we obtain an estimator $\{\tilde{v}_t\}$ of the value function. The natural question that follows is whether this estimator is "good." Although some theoretical results partially answer this question,[11] it is often difficult in practice to determine the accuracy of an approximate solution to a particular problem. One possibility is to try to derive lower and upper bounds on the true value function, V_t. Deriving a lower bound is typically straightforward—the sequence $\{V_t: t = 1,...,T\}$ defines a feasible trading strategy; therefore, the value of this strategy, which we can estimate through simulation, is a lower bound for V_0. However, deriving an upper bound for V_0 is generally not so straightforward. One possibility is to try to apply stochastic duality theory, which has already been studied extensively in the context of portfolio optimization.[17] In addition, other researchers have successfully employed duality theory in conjunction with approximate dynamic programming to construct lower and upper bounds on the prices of American options.[18] A similar approach might also work for portfolio optimization problems, which we are investigating in ongoing research.

There are many other approximate dynamic programming solutions, including algorithms based on approximate policy iteration and on Q-learning.[11] These approaches generally share the common feature of resorting to function approximation and simulation techniques to deal with computational intractabilities. As computing power continues its remarkable growth, we believe that these techniques will become increasingly important in addressing many of the challenges of financial computing over the next few decades.

Acknowledgment

This research was partially supported by the MIT Laboratory for Financial Engineering.

References

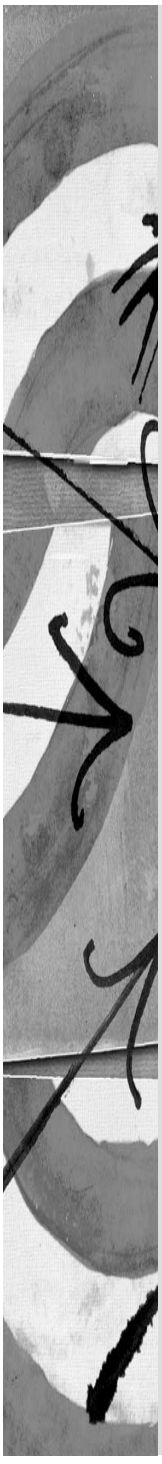
1. M. Haliassos and A. Michaelides, "Calibration and Computation of Household Portfolio Models," *Household Portfolios*, L. Guiso, M. Haliassos, and T. Japelli, eds., MIT Press, Cambridge, Mass., 2000.
2. M. Haugh and A. Lo, "Asset Allocation and Derivatives," *Quantitative Finance*, vol. 1, no. 1, Jan. 2001, pp. 42–75.
3. R. Merton, *Continuous-Time Finance*, Basil Blackwell, Oxford, UK, 1990.
4. D. Bertsekas, *Dynamic Programming and Optimal Control*, Vol. I, Athena Scientific, Belmont, Mass., 1995.
5. D. Bertsimas, A.W. Lo, and G. Mourtzinou, *Tax-Aware Multiperiod Portfolio Optimization*, MIT Sloan School of Management, Cambridge, Mass., 1998.
6. P.H. Dybvig and H.K. Koo, *Investment with Taxes*, Olin School, Washington Univ., St. Louis, 1996.
7. G.M. Constantinides, "Capital Market Equilibrium with Personal Tax," *Econometrica*, vol. 51, no. 3, May 1983, pp. 611–636.

8. D. Kahneman, P. Slovic, and A. Tversky, *Judgment Under Uncertainty: Heuristics and Biases*, Cambridge Univ. Press, Cambridge, UK, 1982.
9. G.M. Constantinides, "Habit Formation: A Resolution of the Equity Premium Puzzle," *J. Political Economy*, vol. 98, no. 3, June 1990, pp. 519–543.
10. J. Heaton, "An Empirical Investigation of Asset Pricing with Temporally Dependent Preference Specifications," *Econometrica*, vol. 63, no. 3, May 1995, pp. 681–717.
11. D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, Mass., 1996.
12. F. Longstaff and E. Schwartz, "Valuing American Options by Simulation: A Simple Least Squares Approach," to be published in *Rev. of Financial Studies*.
13. B. Van Roy, "Temporal Difference Learning and Applications in Finance," *Computational Finance*, Y.S. Abu-Mostafa et al., eds., MIT Press, Cambridge, Mass., 1999.
14. J. Moody et al., "Performance Functions and Reinforcement Learning for Trading Systems and Portfolios," *J. Forecasting*, vol. 17, 1998, pp. 441–470.
15. J.N. Tsitsiklis and B. Van Roy, *Regression Methods for Pricing Complex American-Style Options*, Laboratory for Information and Decision Systems, MIT, Cambridge, Mass., 2000.
16. R. Neuneier, "Optimal Asset Allocation Using Adaptive Dynamic Programming," *Advances in Neural Information Processing Systems*, D. Touretzky, M. Mozer, and M. Hasselmo, eds., MIT Press, Cambridge, Mass., 1996.
17. I. Karatzas and S. Shreve, *Methods of Mathematical Finance*, Springer-Verlag, New York, 1998.
18. M. Haugh and L. Kogan, *Pricing American Options: A Duality Approach*, Wharton School, Univ. of Pennsylvania, Philadelphia, 2001.

Martin B. Haugh is a PhD candidate in operations research at MIT. Starting January 2002, he will be an assistant professor in industrial engineering and operations research at Columbia University. His technical interests include financial engineering and operations research. He received his BSc and MSc in mathematics and statistics at University College, Cork and his MSc in applied statistics at the University of Oxford. Contact him at the MIT Operations Research Center, 1 Amherst St., E40-149, Cambridge, MA 02139; mhaugh@mit.edu.

Andrew W. Lo is the Harris and Harris Group Professor of Finance and director of the MIT Laboratory for Financial Engineering. His technical interests include financial engineering, computational finance, risk management, stochastic processes, nonparametric statistics, optimization, data mining, and computational learning. He received his BA in economics from Yale and his MA and PhD in economics from Harvard. He belongs to the IEEE, the American Economic Association, the American Finance Association, the American Statistical Association, the Econometric Society, the Institute of Mathematical Statistics, and the Society of Industrial and Applied Mathematics. Contact him at the MIT Sloan School of Management, 50 Memorial Dr., E52-432, Cambridge, MA 02142; alo@mit.edu.

EDITORIAL CALENDAR 2001



JANUARY/FEBRUARY

Usability Engineering in Software Development

When usability is cost-justified, it can be integrated into the development process; it can even become one of the main drivers of software development. How can you avoid conflict between your usability and development staff and build even stronger teams?

MARCH/APRIL

Global Software Development

What factors are enabling some multinational and virtual corporations to operate successfully across geographic and cultural distances, while others struggle and fail? Software development is increasingly becoming a multisite, multicultural, globally distributed undertaking.

MAY/JUNE

Organizational Change

Today's organizations must cope with reorganization, process improvement initiatives, mergers and acquisitions, and ever-changing technology. Through experience reports, case studies, and position papers, we will look at what organizations are doing and can do to cope.

JULY/AUGUST

Fault Tolerance

We used to think of fault-tolerant systems as ones built from parallel, redundant components. Today, it's much more complicated. Software is fault-tolerant when it can compute an acceptable result even if it receives incorrect data during execution or suffers from incorrect logic. How do you plan for this? How do you build it in?

SEPTEMBER/OCTOBER

Software Organizational Benchmarking

How do you decide what to benchmark and how much detail is necessary? How do you identify the right information sources?

NOVEMBER/DECEMBER

Lightweight Processes

Do lightweight processes such as eXtreme Programming and adaptive software development lower the ceiling or raise the floor? Here's a look at new flexible and agile methods as well as some fresh techniques that are applicable to any software methodology.