September 2021

Investor Solutions

Designing robust strategic asset allocations under constraints and uncertainty

Keith R. Collier, CFA
Director, Asset Allocation Research
BNY Mellon Investor Solutions

In this whitepaper, we introduce BNY Mellon Investor Solutions' SAA design process in detail. Along the way, we highlight specific innovations that differentiate our SAA approach from traditional optimization techniques, and deliver intelligent, objective, bespoke client solutions.

Executive Summary

Strategic asset allocation (SAA)–or policy portfolio design–serves a central role as the touchstone of multi-asset investment: transforming long-term, forward-looking market forecasts into enduring portfolio allocations. However, forecasting is an inherently error-prone endeavor because financial market performance exhibits a high degree of noise. These forecast errors become even more protracted as time horizons extend. Regardless of horizon, the reality that forecast error exists means that an "optimal" solution is also a potentially precarious and elusive solution, as it requires near-complete certainty to successfully maximize utility across various investment allocation possibilities.

Thus, when designing a policy portfolio to weather the highs and lows of the coming market cycle, we propose investors consider a "robust" portfolio, rather than an "optimal" one. We define a *robust* portfolio as a portfolio that delivers higher utility (versus other portfolios) for a given probability of adverse market conditions. Said another way, the robust portfolio has less shortfall uncertainty than the non-robust portfolio (where *shortfall* refers to underperforming a given objective).

The technology to deliver a robust SAA design to investors should perform three key functions:

- Embrace the reality of forecast uncertainty and solve for resilient solutions despite that ambiguity
- Respect "common sense" practical or required portfolio allocation constraints
- Apply a multi-objective portfolio search methodology to balance multiple competing objectives

We have developed an innovative process to address these goals, which we explain in detail in this paper. The core of our process is based on an efficient search technique from the world of machine learning, known as *simulated annealing*. The algorithm is inspired by the field of metallurgy where the term *annealing* describes the strengthening process that occurs in metals as they are systematically

heated to a high temperature quickly, then cooled slowly from the furnace. In a similar way, the simulated annealing algorithm allows our portfolio allocation possibilities and gradually "harden" into the robust portfolio allocation result.

The value to the client is in the delivery of a portfolio solution that can consistently fulfill numerous competing objectives while maintaining an advantage to a vast number of alternative portfolio allocations despite the uncertainty of future market performance.

Introduction

Strategic Asset Allocation (SAA) is often referred to as the most important driver of investors' risk and return experience [Brinson, Hood, Beebower, "Determinants of. Portfolio Performance," 1986], yet SAA design approaches vary widely among practitioners–from intuitive/experience-based methods to advanced, non-linear optimizations. As stress periods–not least, the great financial crisis–have demonstrated, subjective methods may fail to consider counterintuitive solutions. Likewise, complex mathematical approaches are often highly sensitive to input parameters and forecast accuracy. Both clients and practitioners should rightfully be wary of both simple *and* complex approaches that each embed an overreliance on forecast accuracy–because all forecasts eventually fail. This is not an indictment of the asset allocation profession, but rather a practical reality that should be embraced and incorporated into best practices of portfolio design.

Consider this situation: You're a parent of a young child and need to serve dinner every evening. If you're lucky, your child is amenable to almost anything you put their way. But if the child is a picky eater, then you're not getting out of this so easily. Consider instead that you had *twins* – and this particular evening, they're especially cranky. One child wants pasta and the other wants potatoes. What could you possibly serve them to placate their shrieking cries? Spaghetti only solves half the problem and a dinner of French fries surely won't get you nominated for Parent of the Month. What to serve? Oh yes, gnocchi! …That delightful Italian potato dumpling creation. They had it for the first time two weeks ago and they both cleaned their plates. So gnocchi it is, and everyone's happy.

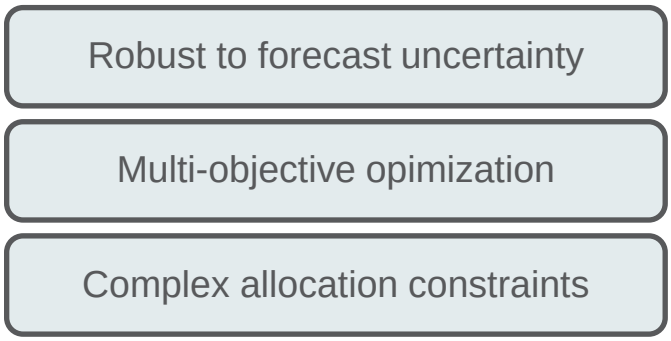
Now consider, instead of twins, you had triplets and each of them had a dinner preference that initially seemed even more incompatible. Finally, consider that you're all away from home in a city you've never been to before as nightfall approaches on a Sunday evening in the midst of foul weather… so time is of the essence. Where do you even begin to look for gnocchi now? Oh, and by the way, it better be delicious!

Likewise it is with asset allocation. For better or worse, there *will* be a policy portfolio, and there *will* be multiple stakeholders at the table–if not as physical persons, then at least in the form of multiple *mandate objectives*. Each objective is effectively a competing claim on an asset allocation that can only deliver one global optima at a time. Sometimes return will matter (or perhaps return above a threshold). Others will want yield (for income), or maybe *risk-adjusted* return. Still others will care about volatility, or drawdown, or Sortino ratio, or liquidity, or tax consequences. But most of all, they will likely care about sevreal of these things. Two, three, four or more objectives may really come to the forefront. How do we design a single portfolio that does *all* these jobs at least moderately well? And how do we do it when any forecasts of asset class behavior are uncertain at best, or, at worst, just plain wrong?

Problem Definition

Let's first look at defining the extents of the problem and the pitfalls of common asset allocation techniques that we have set out to address.

Figure 1: Three features of Robust SAA



To effectively serve clients in a solutions business, a firm like ours needs flexible tools and adaptive processes that are up to the range of challenges we're called to advise on in the marketplace. Thus, as we developed and evolved our technology for SAA design, it always had to deliver on three important features:

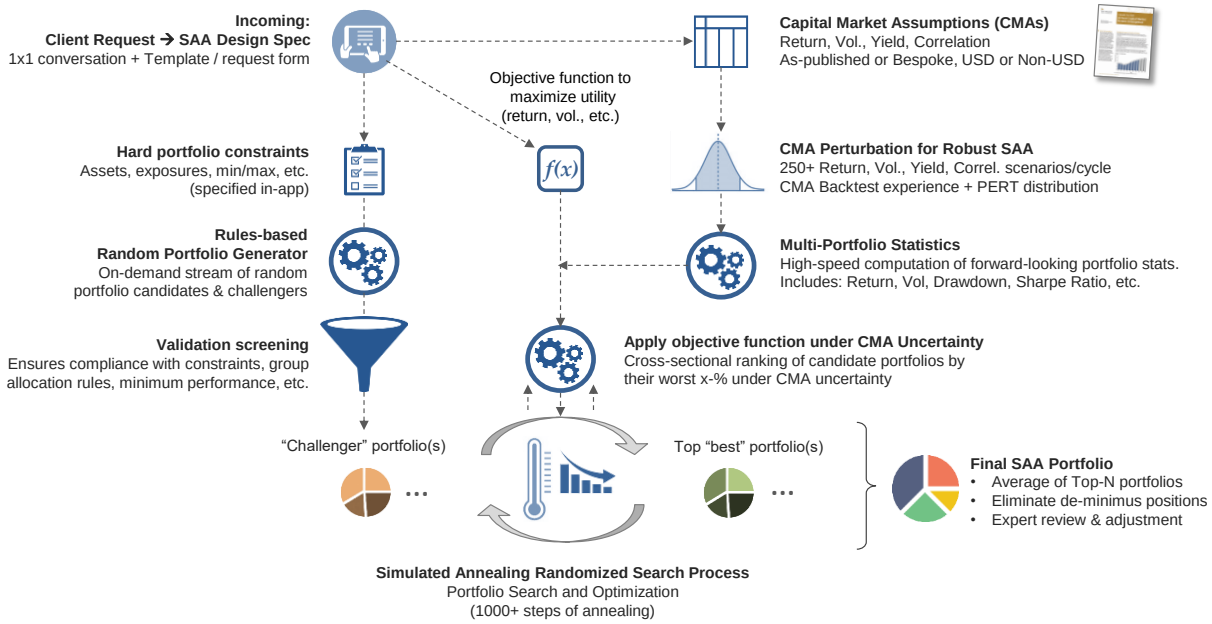
1. SAA portfolios that are robust to CMA forecast error and year-on-year changes in CMA outlook (instead of optimal to precise forecast assumptions)
2. The ability to optimize on multiple (three or more) portfolio performance objectives (such as return, drawdown, yield, etc.)
3. The ability to incorporate multiple absolute and relative constraints around portfolio allocation weights

The process we designed to address this is abstract, but not overly complex, and certainly not without prior art. If you're familiar with Monte Carlo simulation, you'll be able to follow how we've adapted elements of randomization and machine learning to build a powerful process to converge on the solution of robust portfolios.

Summary Methodology

The process map in Figure 2 below provides an overview of the entire SAA design process, from intake of client constraints to policy portfolio formation.

.**Figure 2:** Overview flowchart of the SAA design process



Starting from the upper left-hand corner, the mandate specifications and custom constraints are set by the client's circumstances and objectives, or collaboratively in consultation with the strategy team and subject-matter experts. These conversations and requirements are then construed into an asset class universe and a set of portfolio constraints. Based on the objectives to be optimized, a custom objective function is configured for use in portfolio evaluation. The asset class universe determines the relevant CMAs, and these are perturbed downstream to achieve a robust SAA design. The rest of the components in Figure 2 are the subject of the balance of this paper, but the summary box below provides a quick preview of how it all fits together.

Outline of BNY Mellon Investor Solutions' Robust SAA design process

Sequential steps:

1. Construct a range of Capital Market Assumptions that meet the asset class and currency needs of client portfolios. This is done on an annual basis, but can be revisited during the year if a material change in *long-term* capital market assumptions requires it.
2. Generate an initial pool of random portfolios that simply fulfill the required portfolio constraints (number of initial pool is arbitrary and can vary by project requirements). Though initially random, these portfolios will become increasingly refined/improved after each round of assessment. At each iteration, they represent the current state of our "best thinking" for the portfolio allocation (the "top" portfolios or "incumbents").
3. Begin an iterative loop (500+ iterations):
    a. For each member of this initial pool of "top" portfolios (the incumbents), we generate a set of "challenger" portfolios that differ slightly from their respective competitor.
    b. We compute a distribution of portfolio statistics to apply to the asset class components of all portfolios (incumbents and challengers) using perturbed CMA scenarios.
    c. We compare the performance of each "top" portfolio versus its associated "challenger" portfolio based on how each performs under their worst 20% of perturbed CMA scenarios.
    d. Using a Simulated Annealing process, we update/replace a "top" portfolio with its challenger portfolio if it delivers improvement that exceeds a hurdle rate that varies according the SA cooling process.
3. The final SAA portfolio is the equal-weighted average of the current "top" portfolios.

CMA Formation & Perturbation

For the foundation of any SAA design, practitioners will require capital market assumptions (CMAs), which represent the long-run expectations of asset class performance (return, risk, yield, correlation). In service of this important client need, here at BNY Mellon, we annually publish Capital Market Assumptions on over 50 global asset classes and sub-classes. SAA design is highly dependent on these capital market assumptions, but *any* forecasts from *any* investment manager or research institution–even our own–will likely not be perfect. CMA back testing (Figure 3) illustrates the extent of this reality (CMA forecast in blue; Realized ten-year performance in orange).

Figure 3: Ten-year forward return and volatility experience
(CMA forecast vs. actual) for U.S. Large Cap Equity and Aggregate Bond

Source: BNY Mellon Investor Solutions (as of January 2019). Charts are provided for illustrative purposes and are not indicative of the past or future performance of any BNY Mellon product. Projections or forecasts regarding future events, targets or expectations, are subject to change. There is no assurance that such events or expectations will be achieved, and actual results may be significantly different.

When used as point forecasts, CMAs can be quite precarious. Because they convey a sense of precision–without a corresponding indication of their uncertainty–it's all too easy to optimize an ideal solution based on a single, fragile expectation of the next ten years that may not actually occur. In all likelihood, it probably won't occur–at least not precisely–and the resulting performance shortfall experience will be an unforgiving judge of how robust any given SAA really is. In a world where *anything* can go wrong, but we don't know *what*, it's of great value to solve for acceptable solutions that are immune to the greatest number of scenarios that fall short of our expectations.

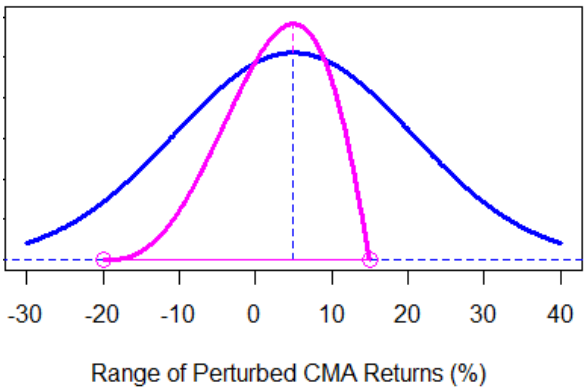
Recall that the motivation for this entire methodology was the need for "robust" SAA portfolios–portfolio allocations that would remain relatively stable from year-to-year as long-term capital market assumptions (CMAs) shift, and yet would deliver more reliable outcomes in the face of forecast error, versus other allocations that are more sensitive to forecast assumptions.

Oftentimes when modeling a variable process around some point of central tendency, the go-to distribution is the "normal", or Gaussian, distribution. While normality assumptions are familiar and a significant body of statistical theory exists to support them, they have a few critical properties that make them unsuited to modeling CMA uncertainty. Their symmetrical and unbounded nature–where variance or "spread" of the distribution of outcomes is expressed via standard deviation–are all rather unintuitive characteristics when applied to CMAs. In particular, the symmetry of the normal distribution

presents a significant problem for stress-test purposes: for as much downside as you prudently try to design in, you allow an equally large range of overly optimistic scenarios too. This obvious shortcoming is why asymmetrical probability distributions are essential for realistic and responsible portfolio stress testing.

To address the shortcomings and complicating factors of the normal distribution, we selected a lesser-known statistical distribution known as the PERT distribution. The motivation for PERT distributions originated in the 1950s to estimate uncertain project scheduling tasks around the U.S. Navy's Polaris missile program at the time. The name itself originated from an acronym: "Program Evaluation and Review Technique." The PERT distribution allows for asymmetric–but intuitive and finite–min-max bounds (see Figure 4). A PERT distribution is defined by three intuitive parameters: *minimum, maximum* and *most likely*. This is similar to a "triangle distribution" (also thee points) but smoother, with more of the distribution near the expected value (mode) and lighter in the tails.

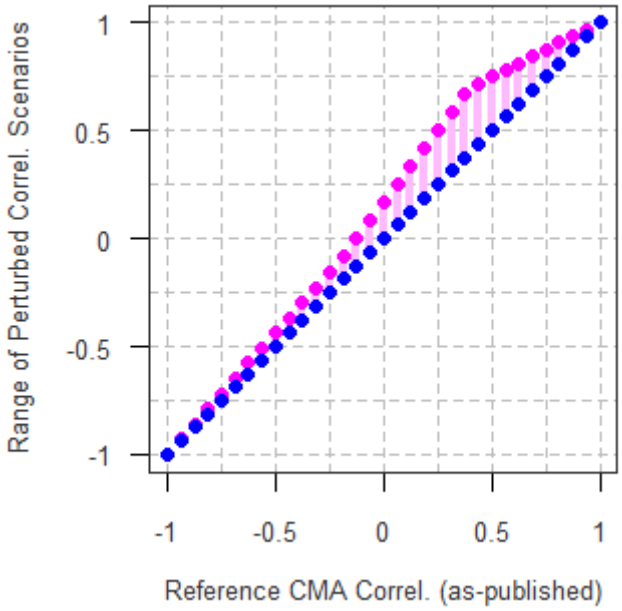
Figure 4: Example of PERT vs. Normal distribution



In the real world, things can go unexpectedly wrong much more easily than they can go unexpectedly right. For this reason, we skew our expected outcomes asymmetrically to the downside. The exact degree of perturbation is a matter of taste–or one's paranoia–but *some* is clearly better than *none*. To specify the min and max of our perturbation ranges, we rely on a combination of our empirical findings from CMA back testing and conservative assumptions applied to asset classes with similar risk. If an entirely systematic approach to setting min-max limits is preferred–and historical data is available–the observed historical upper and lower bound of each metric can serve as an objective standard for the desired forecast time horizon.

Similarly, expected correlations *between* asset classes are seldom stable, thus correlations must be stressed as well. This is a topic that's often overlooked–but critical for a meaningful analysis. We perturb correlations only to the upside, which means we're building in a belief that the risk reduction benefits from asset class diversification can only deteriorate. This is conservative by design, so that none of our scenarios enjoy a perceived benefit from false diversification that only arises from a sample episode of unusually low correlations. Since our downstream search process does not require inverting the covariance matrix, a positive semi-definite covariance matrix is not required. However, since we begin with our initial covariance matrix being positive semi-definite and our correlation perturbations are applied symmetrically across the diagonal, we can perform a test that the eigenvalues remain positive if positive semi-definite status would be required for other downstream processes.

Figure 5: Correlation perturbation range transformation



The methods above speak to the magnitude and range of perturbation performed however there are a few additional factors to consider. It's important to generate and explore a large number of scenarios to generate a uniform distribution for interaction with the other asset classes and eliminate the effect of outlier scenarios. Typically, we generate 250 scenarios, but this number is arbitrary, where *more scenarios* implies *greater stability*. This pool of scenarios is regenerated at each round of the search process, so the process does not become biased to exploit any one arbitrary distribution.

Candidate solutions under constraints: Portfolio generation

Unlike closed-form solutions, Monte Carlo (MC) and machine learning (ML) approaches to empirical optimization rely heavily upon a high-throughput source of candidate portfolios to efficiently explore the solution space. Typically, in asset allocation mandates, the solution space will be long-only (non-negative allocation weights only), and unlevered (all allocation weights sum to one). However, client requirements or commercial considerations nearly always constrain these generous initial boundaries much further. In practice, we've found that, even in the absence of explicitly-defined client constraints, many "common sense" constraints must still be applied to achieve portfolio options that are within the realm of acceptability. For instance, a client may have an objective for high income, yet it would be imprudent to deliver a solution comprised of two high income–but high risk–allocations, such as high-yield credit and emerging markets local debt. A palatable solution needs to have numerous constraints like these built in from the outset, and the portfolio search process needs to be flexible enough to accommodate these constraints. Examples of typical constraints on portfolio formation that are easily applied:

- Common sense rules & behavioral concerns (whether client beliefs are grounded in fact or fallacy, clients simply will not hold an asset allocation if they are not comfortable with it)
- Min & max constraints: Allocation to a specific asset must be at least (x)% but not more than (y)%

- Group constraints: The sum of allocations to a group of assets should not exceed (x)%
- Relative constraints: Allocation to asset “A” must be smaller than the allocation to asset “B”
- Minimum holdings: The portfolio must consist of more than (N) assets

The following table (Figure 6) illustrates a set of constraints that were applied to a recent project:

Figure 6: Example matrix of allocation rules and constraints

Traditional Asset Classes			Specific Rule(s)	Growth
1	USLC	U.S. Large Cap Equity	Maximum weight	35%
2	USMC	U.S. Mid Cap Equity	Maximum weight	25%
3	USSC	U.S. Small Cap Equity	Maximum weight	20%
4	DMLC	International Large Cap Equity	Maximum weight	25%
5	DMSC	International Small Cap Equity	Maximum weight	20%
6	EMEQ	Emerging Equity	Maximum weight	15%
7	UST	U.S. Treasury	Maximum weight	80%
8	USIG	U.S. Investment Grade Credit	Maximum weight	60%
9	USHY	U.S. High Yield	Maximum weight	25%
10	EMDUSD	Emerging Markets Sovereign USD	Maximum weight	25%
11	RAS	Real Assets	Maximum weight	15%
General Rule Type			**Specific Rule(s)**	**Growth**
1		**Stock vs. Bond Mix**	Sum(BONDS) <= 40%	
2			Sum(BONDS) >= 50%	
3			Sum(All EQTY) >= 10%	
4			Sum(All BONDs) >= 10%	
5		**Equity-related**	USLC >= 25% of sum(All EQTY)	
6			USSC <= 25% of sum(All EQTY)	
7			EMEQ <= 20% of sum(All EQTY)	
8			USLC > USMC	
9			USMC >= USSC	
10			USMC + USSC <= 50% of sum(All EQTY)	
11			Sum(DM + EM EQTY) <= Sum(US EQTY)	
12			DMSC <= USSC	
13			USLC > EMEQ	
14		**Bond-related**	UST >= 25%	
15			HY <= IG	
16			HY <= 1.2x IG	
17			IG + HY + EMD <= 40%	
18			EMD <= HY	

For any mandate we design, each of these constraints are codified as checks in an overall portfolio validation function (PVF). The PVF tests that a preliminary randomly generated portfolio passes all of the required constraints. If the portfolio is validated, it can then be used downstream in the process, as it will at least meet the client or mandate-specific constraints. However, if the preliminary portfolio *fails* to pass any of the underlying constraints, the portfolio is discarded, and another random portfolio is sampled. If the constraints are overly restrictive, or even unfeasible, the generation process may exhaust its iteration limit, and the user will be notified of the most onerous constraints.

With portfolio constraints explained and out of the way, it’s relatively straightforward to produce random portfolios that pass the test. Conveniently, the Random Portfolio Generator provides us with candidate portfolios “on tap” – with high throughput of hundreds or thousands per second. The details of this algorithm are explained, step-wise, in the following text box.

Random Portfolio Generation algorithm:

1. Fill “empty” portfolio to meet minimum required allocations
2. Randomly select an asset:
   a. Determine available allocation on this asset (current vs. max allowable)
   b. Random allocation between current and max allowable
3. If total portfolio allocation < 100%, repeat: randomly select another asset
4. Check that portfolio meets validation rules–if not, generate a new portfolio.

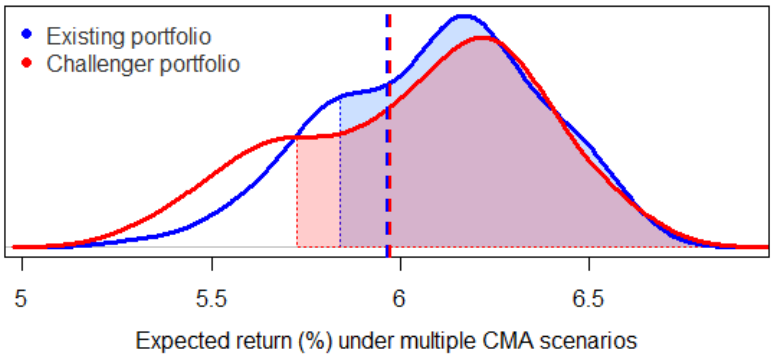
Since portfolio options can be generated ad infinitum, it’s usually not practical–and frequently not possible–to explore all or even most of them. Beyond a few assets, even with integer weights or a few coarse steps of allocation levels, the global range of potential allocation combinations grows super-exponentially and becomes an intractable search problem. This explosion in possible solution options is a well-studied problem in advanced mathematics, known as the “subset-sum” problem. Thus, intelligent search methods must be used to sample, evaluate and “steer” the eventual solution to areas where near-optimality exists.

Tail Scenario Robustness

“If your worst day is better than your neighbor’s best day, you’ve nothing to complain about!” …While the capital markets probably won’t ever offer up such an obviously advantageous binary choice, the spirit of that statement conveys a great deal about what we’re trying to evaluate and compare–the ideal we’re measuring from–between competing portfolio candidates in our effort to achieve robustness.

Recall that Capital Market Assumptions (CMAs) are *forward-looking* forecasts of how various asset classes are expected to perform. They *do not* represent backward looking historical experience. That said, Figure 7 shows the expected return outcomes of two portfolios across a range of 100 different perturbed CMA scenarios that could possibly come to pass–no one knows for sure *until it actually happens* (ten years from now). Note that the average return outcome across the 100 scenarios for both portfolios (as indicated by the heavy dashed lines near the center of the distributions) is nearly identical. In fact, the challenger portfolio (in red) is actually a few basis points higher than the existing portfolio. The mode, or most likely value, (shown by the peak of each distribution) is even more skewed in favor of the challenger portfolio. If we were to select portfolios simply on the basis of their *average* or *most likely* return, we would have to choose the challenger portfolio (in red) since the red dashed line is *ever so slightly* higher than the blue.

Figure 7: Distribution of scenario outcomes for two portfolios



However, doing this would set us up to fail if reality doesn't play out exactly as planned. The client does not get "100 chances" to experience the next ten years (over and over again) to realize that average. They get *one shot* at it–and the return will be what it will be. The only thing we can control in advance is our allocation. Therefore, our focus should be on improving the probability of a satisfactory result. Put another way, we want to ensure that a "bad" outcome (relative to everything else that *could have happened*) doesn't really hurt us so badly–and it's as painless as possible. That's one of the hallmarks of robust design. Conversely, if things go unexpectedly *better* than planned, that's a "good problem to have."

Note that we are not always evaluating *the same* Nth-percentile of tail scenarios. Because we have no forward knowledge or control over what scenarios will be dealt to us, we simply want to ensure that our preferred portfolio delivers better outcomes more often, regardless of what conditions deliver those outcomes. The reason you don't want to simply rank portfolio options based on their probability of outperforming an arbitrary threshold (i.e.: x% change of delivering a negative return) is because: A.) you run into "low count issues" when the number of evaluation scenarios is relatively small (arbitrarily less than 1000); and B.) the *performance threshold* becomes yet another unknown parameter to "tune," and vastly depends on the choices of assets & rules, whereas a *probability threshold* will "break-down" more gracefully.

Explained another way, If you rely on a performance threshold (to compare the probability of exceeding it), the threshold you choose can be either *very easy* or *very hard* for your candidate portfolios to beat. And if the performance of the candidate portfolios is very tightly clustered, it will be difficult to find the optimal split point for the threshold to yield a meaningful/interpretable distinction. Using a percentile-based threshold avoids these arbitrary decisions which are pathways for spurious conclusions. It acknowledges the fact that, yes, some otherwise great portfolios may still demonstrate 1% or 2% tail scenarios that are highly unfavorable–but so too might the challenger portfolios that are comparatively less desirable, even in normal scenarios.

Given the small sample size of tail events, it's ill-advised to quantitatively compare (or "split hairs") on what is already an unusual tail scenario. Suffice it to say, we're comparing occurrences under *stress scenarios*–and in stress scenarios, the quantitative precision of the specific outcomes observed are merely artifacts, rather than rational price discovery or reliable investment relationships. You don't want to engineer performance *within* tail scenarios–you want to *avoid* those murky scenarios altogether–and that's why using a 20% tail threshold is more conservative than using a 10% or 1%

threshold. Because they're so off-normal and extreme, "one percentile" performance estimates are not reliable enough to meticulously cross-compare, especially with a greedy algorithm. About the only conclusion we can say with certainty when comparing between a pair of one percentile statistics is *"yes, they are both very bad outcomes*." Because the distribution of estimated performance is typically unimodal (by construction, due to PERT), the closer one moves toward the center of the percentile spectrum (near maximum likelihood), the more consistent and reliable our shortfall estimates will be. It's a balance between worrying about how bad things can get in the tails (which argues for a *lower* percentile threshold) and making statistically valid portfolio selections (which argues for a *higher* percentile threshold). Both are valid intentions, and the practical compromise is found in the middle.

Iterating the Portfolio Search with Simulated Annealing (SA)

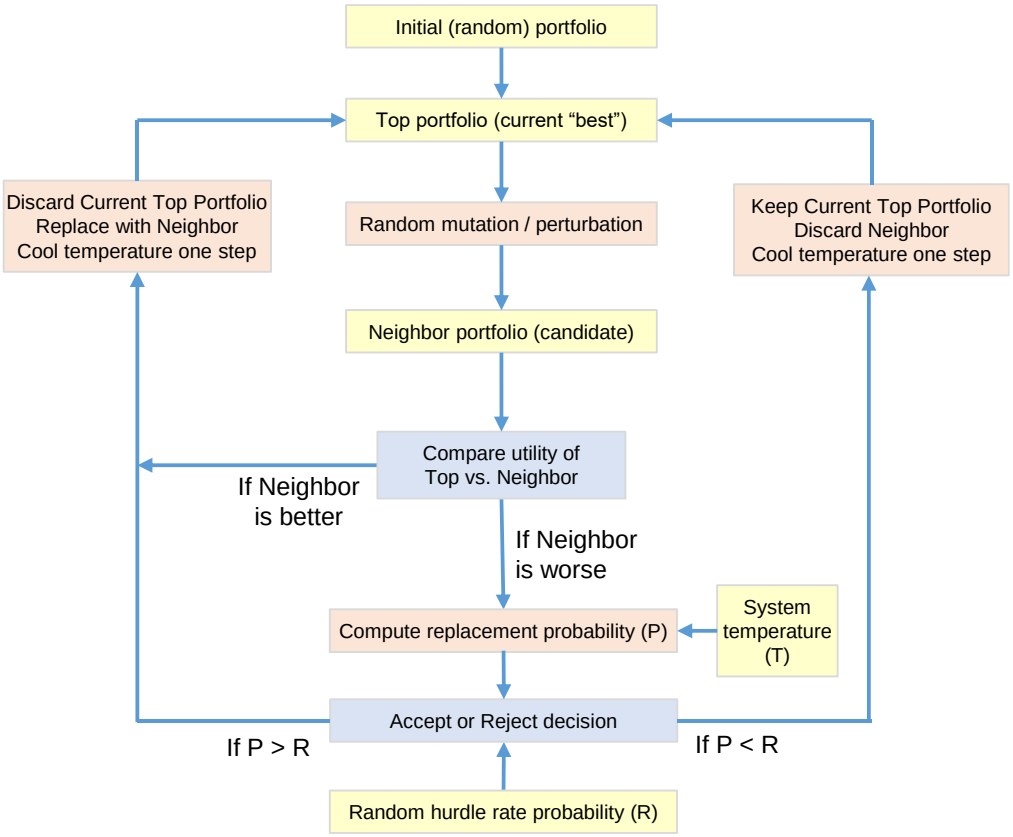
Consider this adventurous scenario (which is not unlike the challenge of conducting a portfolio search): You are a contestant in a new reality television program for geography nerds. You've been parachuted out of a helicopter *somewhere* in the Himalayas–home of Mount Everest (8,848m high)–and your mission is to find the tallest mountain in the land. As an added incentive for your quest, you'll be rewarded with a thousand dollars for *every meter* you can ascend *from the elevation where you were initially dropped at random!* Unfortunately, however, the region is clouded in dense fog and you have no knowledge of the local geography. The only measuring device you have is an altimeter to precisely determine your altitude at the point where you're standing. To make this analogy more accurate, we'll grant you the convenient luxury of a "jet-pack" that enables you to rocket yourself to any GPS coordinates you choose on an otherwise blank and unlabeled map. Upon landing in your new location, you can then measure your new altitude, mark it on your map, and reassess your options. How would you find the highest mountain–or at least higher than where you started–in a finite amount of time? Obviously, your viewing audience would quickly lose interest if you kept searching indefinitely! One option would be to compare the height of the four neighboring points immediately North, South, East, and West of you. Whichever one is higher, you could move that new point and re-assess. If none are higher, you could stay where you are and signal for the helicopter to save you from needless frostbite–you're likely as good as you can get. …*Or are you?* It turns out several heuristic-based machine-learning approaches exist to systematize precisely this sort of search problem. In prior decades, many of these methods were computationally expensive in terms of processing time, but today they are easily within the realm of practicality.

One such method that is particularly applicable to our problem is called Simulated Annealing (SA). SA is a probabilistic search technique from the field of machine learning for finding near-optimal solutions to multidimensional problems (or problems with many parameters). It was first proposed in the early 1980s by Kirkpatrick, Gelett, and Vecchi ["Optimization by simulated annealing," *Science* 220 (1983) 621-630.]. The name of the algorithm, and the underlying method itself, was inspired by the sciences of metallurgy and thermodynamics. The metallurgical process of "annealing" refers to the controlled cooling of molten metal. As we know, metals are very malleable and flexible at high temperatures (as the high-energy state keeps the molecules in rapid, chaotic motion); yet very hard and change-resistant at low temperatures (as regions of slowing molecules form structures and solidify). When metals are annealed in a systematic way, it's possible to enhance their inherent properties (such as hardness) beyond their initial, unconditioned levels. In a similar manner, simulated annealing allows us to produce a higher-utility solution from a crude initial starting point. It's called "simulated" because

we're not getting *anywhere near* molten metal here in the office! Instead, this real-world behavior provides a suitable analogy for a method of optimization, where vastly different portfolio options are explored randomly at the outset, with a gradual transition (as "pseudo cooling" occurs) to a progressively smaller search area with finer granularity in a localized region of allocations with higher utility.

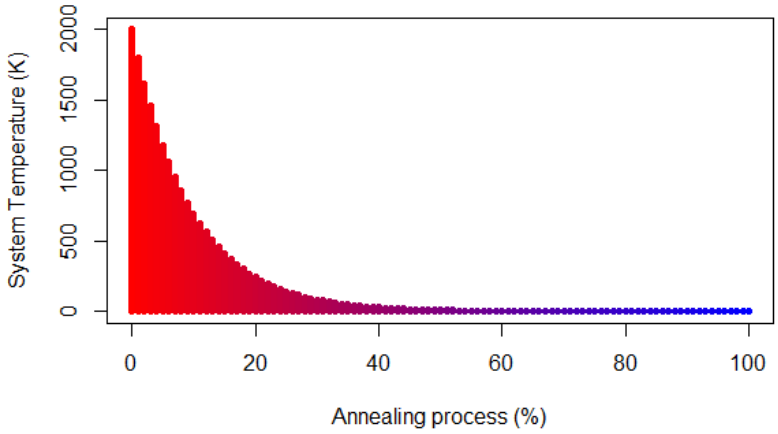
The SA process is one of many heuristic-based search algorithms, meaning search is achieved by way of a purely rule-based instruction set. Figure 8 below demonstrates exactly how simple the rules actually are.

Figure 8: Flowchart of simulated annealing process



As is evident, the algorithm is an iterative process, meaning it follows a repeated cycle for a number of times, or iterations. In keeping with the metallurgical analogy, SA uses the concept of "temperature" to keep track of the iterations and, more importantly, regulate how flexible the process is to exploring possibilities that appear suboptimal but might prove beneficial at later rounds. Temperature in SA begins at an arbitrary high level, and cools exponentially after each iteration, similar to a red-hot steel bar coming out of a metal forge.

Figure 9: The concept of annealing system virtual "temperature"



In our case, instead of performing this search and refinement using a single "best" portfolio, we run this process in parallel on a pool of 30 initial "best" portfolios–thus we have 30 separate hill-climbing "scouts," each looking for the highest mountain–with their findings compared at the end. This feature is borrowed from the setup of genetic algorithms (GA) which start with a pool of solutions ("genes") that mutate and recombine. In our approach here, we do not introduce recommendations or blending between the otherwise independent candidate solutions with the pool.

The search efficiency of Simulated Annealing comes about because it is a hybrid of two contrasting search strategies: stochastic search and gradient descent. Stochastic search is roughly analogous to trying multiple solutions at random and settling upon the most optimal solution encountered over a finite number of "hops." It's better than nothing, but it's apparent that there's little method to the madness. There is no discipline to focus or explore more in a region that appears to contain some "high-grade ore" for purposes of refining the current "best" solution. For problems we face in portfolio space, Simulated Annealing is preferable to elementary heuristic methods, primarily Stochastic Gradient Descent or "Hill Climbing." Hill climbing as the name implies, views the optimization space of utility as a landscape of hills and valleys to ascend rapidly in a relatively simplistic manner. Hill climbing is known as a "greedy" algorithm because it *only* accepts a new solution if the value of that new solution is *immediately greater* than the value of its current solution. It never "backtracks" or relinquishes any of its current value to pursue a different path that might lead to *even greater* or ultimate value. Thus, it's easy to see how the greedy approach can easily get stuck at a local optimum because its appetite for continuous improvement won't allow it to explore those choices that *appear to be* suboptimal at the outset. Hill climbing can only discover the optimal portfolio solution if it *just so happens* to begin its search from a particular allocation that sits along an *uninterrupted path of increasing improvement* to the global optimum, with no missteps or distractions along the way. Stochastic gradient descent (SGD) performs multiple searches from random starting points, but the algorithm itself is still greedy. Thus, the solution is sensitive to the initial random starting points, and the size of the parallel search pool takes on greater importance.

Simulated annealing also offers advantages over Genetic Algorithms (GA). Genetic algorithms construct alternative candidates by mutating and blending a large population of suboptimal initial solutions. Unfortunately for GA, however, many of these mutations are simply not useful, thus they can

expend a significant proportion of overall computing time in mutating and evaluating an evolutionary line of solutions that's not directly applicable to the objective sought.

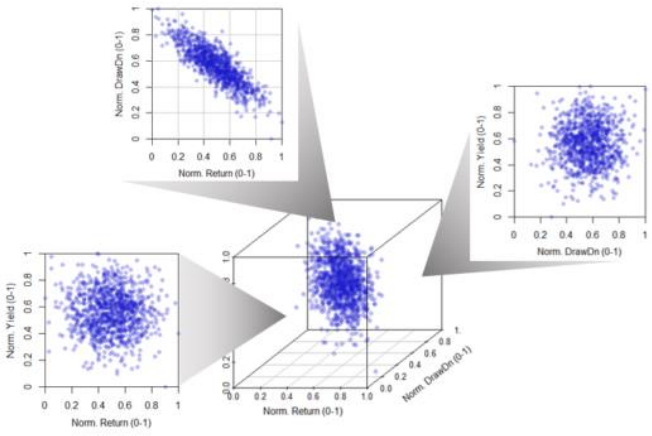
Balancing Multiple Objectives

Revisiting our dinnertime analogy if one child wants pasta, and the other wants potatoes, everyone's going to be happy with gnocchi. And the better that gnocchi is, the happier they *both* will be. The dish (gnocchi) is the *solution* to deliver, and the *quality* (frozen from the supermarket or fresh from Little Italy) is going to determine overall satisfaction or *utility*. Thus, the hapless parent must first realize that gnocchi is the ideal ingredient mix on the plate (like the asset allocation), out of all the thousand other foods to serve–and that in itself is no small feat. Then, once that's established, the parent can spend as much resources as are available in pursuit of quality (finding the best gnocchi in the city).

This analogy may actually work better in the investment space because, unlike food, we can *quantify* these multiple objectives with some modicum of precision. Still, there are always a few challenges. Often, when multiple objectives must be considered in combination, each objective may exist in a different numeric context–a different scale, or perhaps even different units. To rectify this condition, and put each dimension in a common scale, the data must be normalized. To achieve this, we use "high-low range normalization" (see Eqn. 1) to re-scale the data to a common range of zero to one. This method of normalization will faithfully represent the relative proximity of observations over the full span of the sample, including outliers which might contain clues of particular solutions to seek or avoid. Other types of data in other applications might require a different form of normalization, such as Z-scoring.

- $$x_{i_{NORM}} = \frac{x_i - x_{MIN}}{x_{MAX} - x_{MIN}}$$ (Eqn. 1)

Figure 10: Geometric cube analogy, showing normalized characteristics of 1000 portfolios in 2- and 3-dimensions

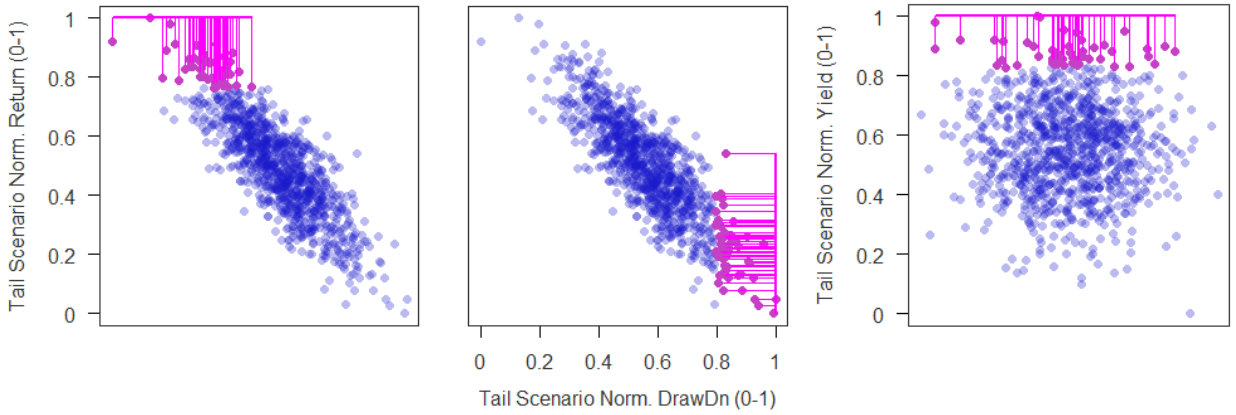


For use with Financial Professionals only. Not for distribution to the general public.

This is where the value–and necessity even–of having more than one set of existing and challenger portfolios comes to bear. More portfolio examples (in our case: 30 existing + 30 challengers = 60 portfolio examples) gives us *context* for how much portfolio differentiation *exists* and how much improvement is *possible* under each round of annealing, given a large, but finite, sample of CMA scenarios. If we only had a single existing portfolio and a single challenger portfolio, the single challenger would always be better or worse than the existing portfolio, but there would be *no context* for the degree of *how much* better or worse. This is especially true as the quantities for comparison become ever more abstract (such as: "*portfolio return at the 20th percentile tail of perturbation scenarios, range normalized from 0 to 1*").

Figure 11: Examples of 1-, 2-, and 3-dimensional distance measurement

Single objective:



Dual objective:

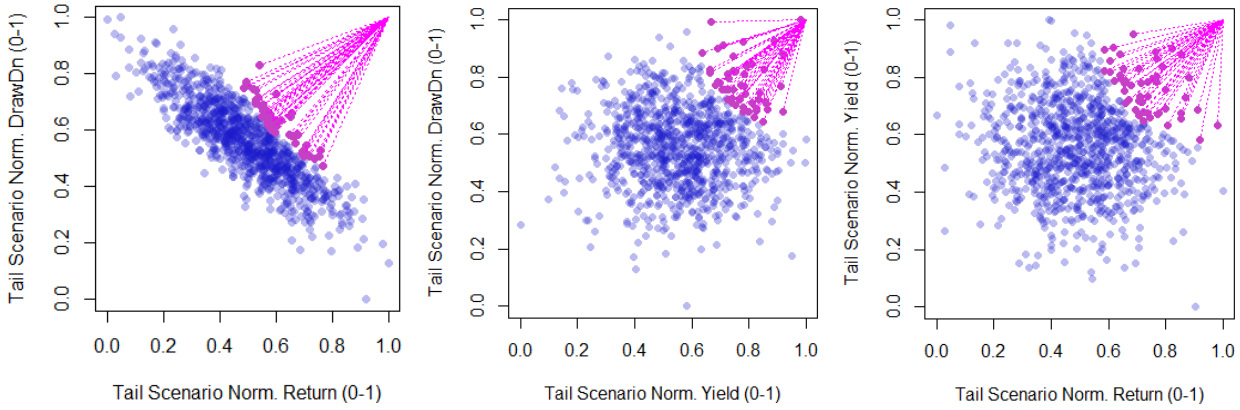
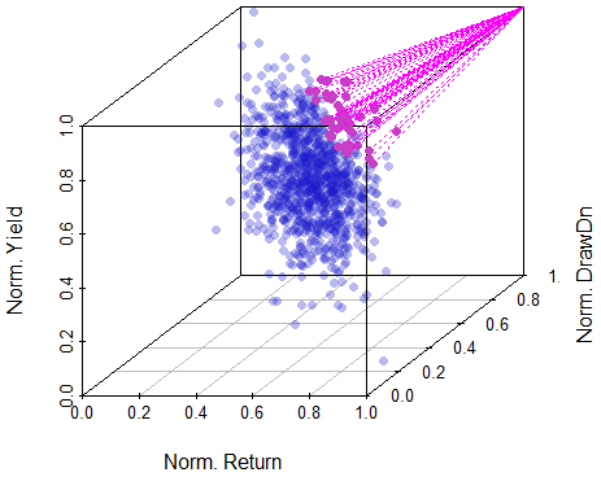


Figure 11 (cont.): Examples of 1-, 2-, and 3-dimensional distance measurement

Triple objective:



The multidimensional (n-dimensional) Euclidean distance function, equation (2), can accommodate any number of equally-important client objectives (return, drawdown minimization, yield, Sharpe ratio, etc.). Portfolios are evaluated by their Euclidean distance from the ideal/optimal point (q) in range-normalized space (an n-dimensional vector of 1s).

- $d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$

  $= \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$ (Eqn. 2)

  Where: $\mathbf{q} = (q_1, q_2, \cdots, q_n) = (1,1,\cdots,1)$

- $U_p = 1 - \frac{d(\mathbf{p},\mathbf{q})}{\sqrt{n}}$ (Eqn. 3)

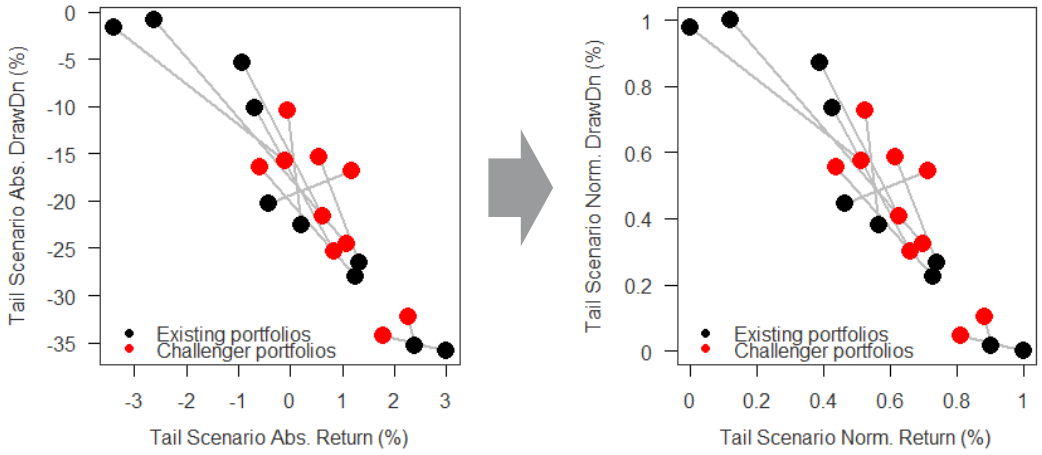
Equation (3) serves as our objective function to maximize, in terms of portfolio utility.

As apparent from the simple examples above, it soon becomes computationally inefficient to exhaustively analyze each portfolio in the feasible search space by brute-force. This is why optimization algorithms like Simulated Annealing are indispensable for traversing an otherwise intractable search space.

This distancing and utility formulas introduced above are applied repeatedly at every iteration of the annealing process, hundreds or potentially even thousands of times overall. The sequence of figures below illustrates how this works step-by-step within a pooled solution group for a two-objective search (here: maximizing return and minimizing drawdown severity). For the sake of simplicity, our example shows a pool of 10 parallel portfolio comparisons, rather than the 30+ we typically use in real-world SAA design projects. First the tail scenarios of return and drawdown of the existing and challenger portfolios (under CMA perturbation) are evaluated in absolute space. We plot these absolute levels in the left-hand side of figure 12. These absolute values are then easily rescaled using min-max

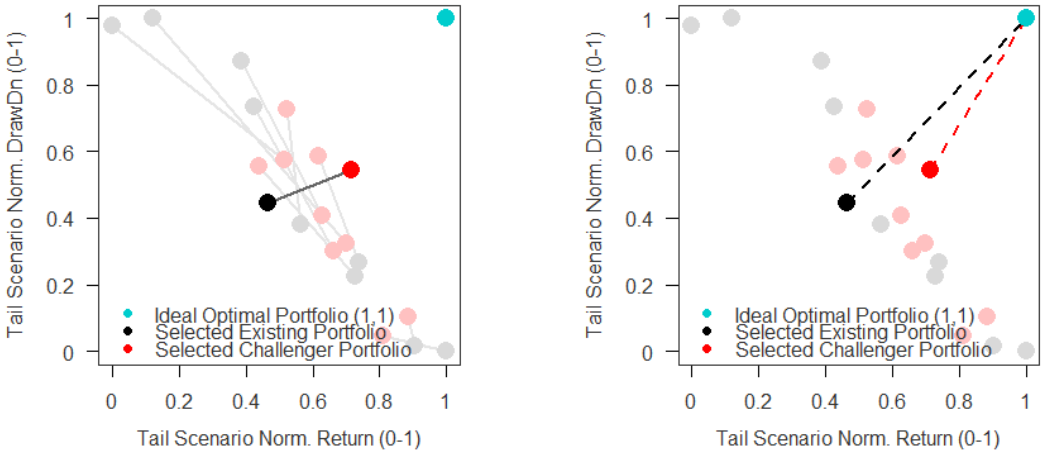
normalization, leaving the shape of the relative performance differentials unchanged (notice how the right-hand side of Figure 12 is spatially congruent to the left-hand side).

Figure 12: Transforming portfolios from absolute to normalized space for two objectives



Once all the portfolios' characteristics for this iteration have been rescaled in all associated dimensions, all corresponding *pairs* of portfolios (the existing top portfolios and their associated challengers) within the pool are evaluated in terms of their *distance* from relative optimality. In normalized space, optimality exists at the point where all objectives are at their theoretical maximum limit – the number "one" (1.0). For the case of multiple objectives, *multidimensional optimality* is represented by an N-dimensional point (**q**), where $(q_1, q_2, \ldots, q_N) = (1, 1, \ldots, 1)$.

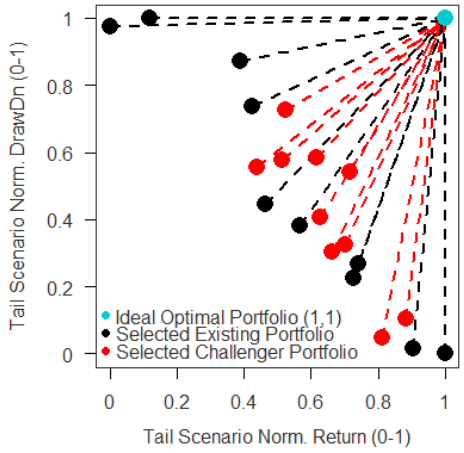
Figure 13: Comparing the overall utility between a pair of portfolios for two objectives



With this framework and Figure 13 in mind, we select a pair of portfolios to evaluate (the pair always consisting of an existing portfolio and its neighbor / "challenger" portfolio). We compute the Euclidean distance of each portfolio from optimality (using Eqn. 2). We then convert distance to overall utility (using Eqn. 3). This measure of utility attributed to each portfolio will be used prior to the end of each iteration to determine which of the two portfolios should be retained (i.e.: if the challenger portfolio should replace the existing). All corresponding pairs of portfolios (the existing, accepted "best" portfolio and its neighbor, the "challenger" portfolio) are evaluated in this way at each iteration of the annealing process before cooling the system by one step and advancing to the next iteration round.

The distance from optimality for all portfolios in the pool (existing and neighbors) is shown in Figure 14 simply for illustration purposes. Keep in mind that utility is the *complement* of distance, so a portfolio with *shorter distance* is also a *higher utility* portfolio.

Figure 14: Comparing the overall utility of all portfolios for two objectives

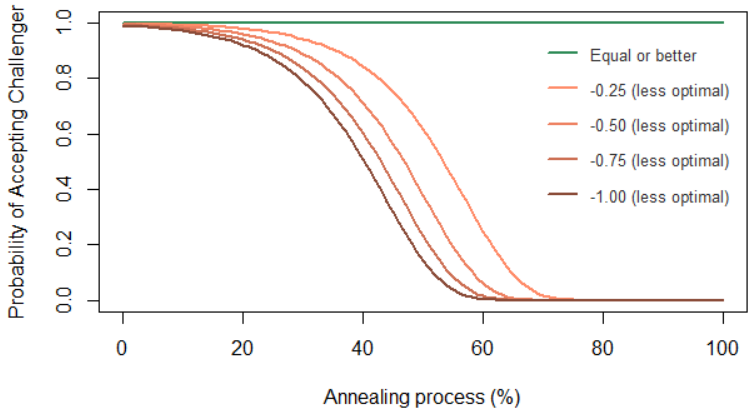


After the utilities have been computed for both the existing and challenger portfolios in a pair, the SA algorithm must decide which of the two portfolios to accept into the next round. If the challenger has higher utility than the existing portfolio by any amount, it's a no-brainer: the existing portfolio is discarded, and the challenger portfolio replaces it as the new "top" solution. However, if the utility of the challenger portfolio is *not* immediately greater than that of the existing counterpart, there's *still* a chance the challenger will be accepted in place of the existing portfolio. The *likelihood* of that chance, or the acceptance probability (P), is determined by two factors (see Eqn. 4): the current system temperature (T_i) and the magnitude of the utility shortfall exhibited ($U_C - U_E$).
Equation 4 introduces the formula for acceptance ("jump") probability:

$$P = min\left(\exp\left(\frac{U_N - U_T}{T_i}\right), 1\right) \qquad \text{(Eqn. 4)}$$

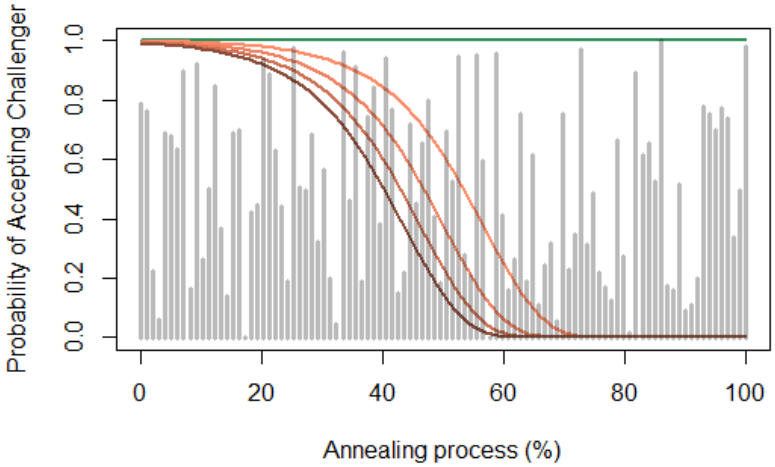
Now it's time for more randomness–and this part is a bit tricky–but it's what pushes SA to explore those counterintuitive solutions that don't seem so immediately great. If the acceptance probability (P) exceeds a random threshold (R), the neighbor solution is accepted and it replaces the current "top" solution (regardless of whether it is immediately of higher utility). That random threshold is re-randomized–so it's different at every instance it's called–but it effectively serves as a virtual wheel of chance, with an expected value of 0.5 (meaning 50% probability).

Figure 15: Probability of moving to an alternative portfolio of lower utility at various temperatures (annealing process %) and utility shortfalls:



As illustrated in Figure 15 above, the acceptance probabilities for different levels of *negative* excess utility over time (which is non-linearly related to temperature); the constant green line shows that challengers with *equal or better* utility are always accepted, no matter what. The front-end of the process is when the algorithm is more accepting of the suboptimal solutions, when it has more time to explore and make mistakes. By the time the annealing process is about 80% complete, it has moved almost entirely to the hill-climbing phase where *only* portfolios that improve utility will be accepted. Figure 16 shows the same acceptance probabilities for various negative utility differentials, but with a random threshold applied (gray bars). Acceptance probabilities above these gray bars will pass in this random scenario (and that challenger would replace the existing). Those below would not replace the existing. As the iterations reach 80% completion, virtually no suboptimal challengers would pass this screen and replace the existing high utility portfolios that have proven resilient.

Figure 16: Random threshold (gray bars) applied to Fig. 13 acceptance probabilities, showing pass rate

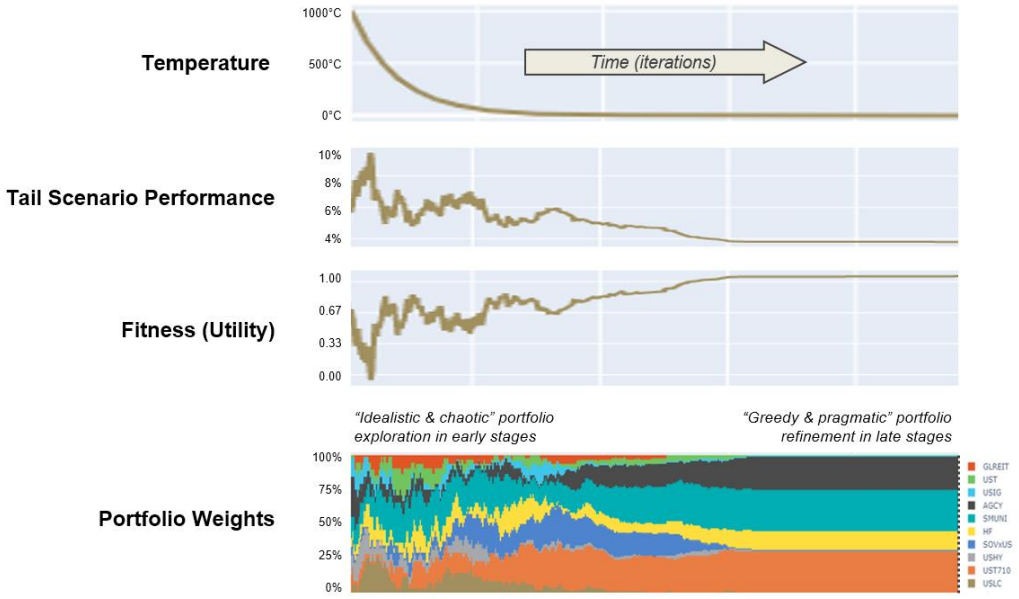


This process is performed at every iteration as the temperature continues to cool. If the pool contains ten portfolio *pairs* (20 portfolios), it must conduct this comparative decision independently ten times, once for each pair, per iteration.

Figure 17 below reveals the externally observable behavior of the annealing process over time. In this example, the evolution and improvement of utility (in this case, expected volatility) across the pool of

top portfolios over the course of the annealing process is clearly evident. For about the first 40% of iterations, the expected volatility appears chaotic and frequently jumps to lower utility states. No visible improvement is being made, but the process is learning useful information about the search space. In these early rounds at high temperature, the process is jumping to different "base-camps" of various "mountains" of utility, searching for allocations that offer both significant and permanent gains in improvement that are robust to challenger allocation alternatives. After 40% completion, it appears the process has found a region of allocations (a particularly high "mountain") that breaks out of the local utility maxima. From this point on, the algorithm is going to stay in that allocation region (that "mountain") and just keep searching and refining it ("climbing it"). The transition is gradual, but by the late stages (50% completion), it has become a completely greedy algorithm. Eventually, the pace of gains begins to slow as fewer moves exist to improve utility and only small moves can be made. In practice, we can devote as much computing time as we wish to search and refinement however, *most* of the achievable gains can often be realized after a few hundred iterations (about 500).

Figure 17: Improvement of expected drawdown over successive SA iterations for a single-objective mandate. In the case shown, the portfolio search is for a "Min Vol" mandate, intended to minimize the expected portfolio volatility (2nd panel).



Finalizing the SAA Portfolio Result

Before an SAA portfolio can be delivered, the raw findings from the pooled SA searches need to be aggregated and refined in preparation for implementation.

The first refinement is portfolio rationalization. While the simulated annealing process yields a pool of between 10 to 30 near-optimal solutions, the end-result must be a single portfolio recommendation, and a few options exist to arrive at this solution:

1. Select the single best portfolio in the pool of top portfolios based on the same utility measure we used throughout the process. This is acceptable in that it will definitely select the most-optimal of the near optimal solutions however it presents an unstable solution if it is an outlier among its peers. Selecting the single best portfolio undoes a small bit of the robustness that was built into the process all along, because a slightly different search path may alter this portfolio's repeatability.

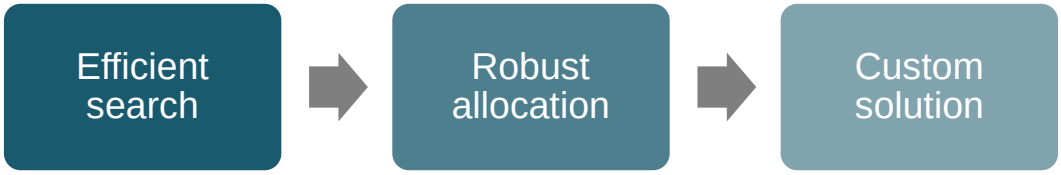
2. Use clustering to identify groups of similar solutions and perform two rounds of averaging. First, perform within-cluster averaging to determine each cluster's allocation, then average across the cluster averages. This is probably the most elegant solution, but it introduces additional explanatory complexity. Also, outlier allocations will have greater influence.

3. Average all of the solutions equally. For most problem types, this is the solution we typically favor. It builds in an additional layer of robustness against any aberrations in the randomization. By averaging *all* solutions, we achieve a single solution that was shaped by *all* the random starting points from different paths. If the optimization were to be re-run using a different random number seed, averaging across the pool of solutions reduces the influence of outliers and drives the final portfolio toward a repeatable point of central tendency. Additionally, for larger pool sizes, it becomes more difficult for outlier allocations to influence the average allocation.

After the portfolio is established, adjustments for simplification or commercial necessity can be applied. One such adjustment is rounding and reallocating the raw fractional portfolio allocations to integer weights or larger increments and eliminating *de minimus* positions (typically less than 2 or 3%). Finally, the portfolio is reviewed against the mandate's portfolio constraints to ensure any manual adjustments are still valid.

Solution Features and Benefits

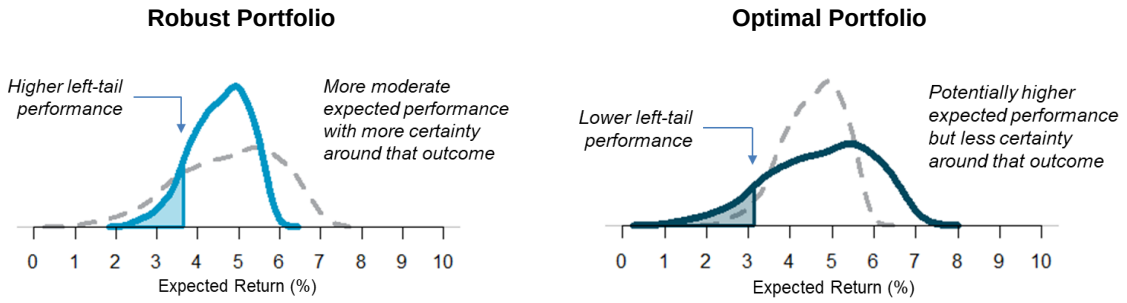
The approach we have presented achieves its objective of providing a portfolio that has less shortfall uncertainty, while also delivering several additional benefits:



- Our approach allows us to **efficiently explore a wide range of candidate portfolio solutions** and converge on a near optimally robust portfolio in a relatively small amount of time. This, in turn, allows analysts and strategists to consider options and make adjustments in near real-time.
- Secondly, the resulting portfolio solutions are **robust to errors in forecasts**. This is a key advantage over an optimal portfolio which is optimal *only under a single set of forecast assumptions* and where the consequences of failure are high. This distinction is illustrated in Figure 18 below where the *optimal* portfolio delivers a higher average return but with *less confidence*, and significant downside tail consequences if the forecast misses the mark.

- Finally, the process is practical and flexible, making it **ideally suited to delivering bespoke client solutions**. The code-based architecture can accommodate a wide spectrum of custom requests or constraints. The process can balance any number of performance objectives–beyond the commonplace risk and return of mean variance optimization. Additionally, the process requires only an intuitive and transparent set of data inputs, with conservative design principles built-in throughout, and few internal "meta-parameters" to complicate the operation or results.

Figure 18: Density plots of expected outcomes of robust and optimal portfolios under a range of CMA perturbations



Source: BNY Mellon Investor Solutions. Charts are provided for illustrative purposes and are not indicative of the past or future performance of any BNY Mellon product. Projections or forecasts regarding future events, targets or expectations, are subject to change. There is no assurance that such events or expectations will be achieved, and actual results may be significantly different.

Conclusion

Designing a truly "optimal" portfolio requires a level of clairvoyant certainty about the future that is most likely unachievable. It's wise to know what one doesn't know, and the only certainty in markets is uncertainty. Given this reality, investors should pursue *robustness*, rather than *optimality*, in their asset allocations–especially for the long term. By building a portfolio that is intended to withstand the test of time, while being robust to forecast error and intertemporal forecast noise, we seek to ensure that the investor has the highest probability of achieving their objectives.

If you'd like to learn more or discuss how our research can help you solve your investment challenges, please contact a BNY Mellon Investor Solutions representative.

Glossary of Terms

Capital Market Assumptions (CMAs): Long-term forecasts of average expected performance (return, volatility, yield, and correlation) for each asset class over a forward-looking time horizon (typically ten years or more, to average through *at least* one full market cycle).

Challenger portfolio: A portfolio derived from an existing portfolio but perturbed or altered in some way such that is *similar to but not a copy of* the existing portfolio. In a simulated annealing process, at least one challenger portfolio is produced in each iteration, and it may serve to replace the existing portfolio if its utility is higher or its acceptance probability exceeds a random threshold.

Genetic Algorithm (GA): A heuristic-based optimization method inspired from the biological process of gene evolution, whereby a large pool of random candidate solutions ("genes") is evaluated for fitness to an objective. Those candidates that are most "fit" survive into the next round, while those that are "below threshold" do not. At each round, the survivor candidates are randomly mutated or crossed with other successful survivors and the process is repeated to pursue an increasingly refined set of "near-optimal" solutions.

Neighbor portfolio: See "Challenger portfolio"

PERT Distribution: A bell-shaped probability distribution defined by a minimum and maximum boundary, and a mode ("most likely") value at some arbitrary point between the two bounds, which allows for an asymmetrically skewed distribution. This PERT distribution provides an intuitive alternative to the symmetric and unbounded normal probability distribution, when used in perturbation or Monte Carlo simulations.

Perturbation: A process of deliberately injecting multiple instances of statistical "noise" or error around a given single-point value, to generate a plausible distribution of uncertainty in the single-point estimate. Not directly related to or interchangeable with the similarly termed "PERT distribution".

Portfolio Validation Function (PVF): A test that includes a set of one or more sub-tests that must all pass as TRUE for a portfolio to be deemed "valid" and acceptable as a potential solution for further consideration or evaluation. The sub-tests typically involve allocation characteristics that are desired for commercial or behavioral reasons, such as "*equity allocation must be greater than alternatives allocation*", or "*domestic equity allocation must be greater than international equity allocation*".

Robust SAA Design: The process of determining a strategic policy portfolio allocation that will exhibit greater relative utility than conventional portfolio allocations, when subject to a range of CMA input assumptions or variations in forecast accuracy.

Simulated Annealing (SA): A probabilistic search technique from the field of machine learning for finding near-optimal solutions to multidimensional problems (or problems with many parameters). In proportion to the current level of *virtual temperature* in the simulation, this hybrid algorithm randomly explores suboptimal solutions early on at high temperature and gradually transitions to strict hill-climbing as system temperature cools.

Top portfolio: The current or existing solution accepted as "most-optimal" at each round of a SA process. Also known as the "incumbent" portfolio. It is not necessarily the highest utility solution, but it serves as the source for candidate or neighbor portfolios to be generated from. At the completion of the SA algorithm, the top portfolio(s) are accepted as the global near-optimal solution.

Disclosures

For Financial Professionals and Institutional Investors Only

All investments involve risk including loss of principal. Certain investments involve greater or unique risks that should be considered along with the objectives, fees, and expenses before investing.

Asset allocation and diversification cannot assure a profit or protect against loss.

This material has been distributed for informational purposes only. It is educational in nature and should not be considered as investment advice or a recommendation of any particular security, strategy or investment product and should not serve as a primary basis for investment decisions. Views expressed are those of the author stated and do not reflect views of other managers or the firm overall. Views are current as of the date of this communication and subject to change. Forecasts, estimates and certain information contained herein are based upon proprietary research and are subject to change without notice. Certain information has been obtained from sources believed to be reliable, but not guaranteed. Please consult a legal, tax or investment professional in order to determine whether an investment product or service is appropriate for a particular situation.

BNY Mellon Investment Management is one of the world's leading investment management organizations, encompassing BNY Mellon's affiliated investment management firms and global distribution companies. BNY Mellon is the corporate brand of The Bank of New York Mellon Corporation and may also be used as a generic term to reference the Corporation as a whole or its various subsidiaries generally. BNY Mellon Investor Solutions, LLC is an investment adviser registered as such with the U.S. Securities and Exchange Commission ("SEC") pursuant to the Investment Advisers Act of 1940, as amended. BNY Mellon Investor Solutions, LLC business is described in Form ADV, Part 1 and 2, which can be obtained from the SEC.gov website or obtained upon request. BNY Mellon Investor Solutions. LLC provides certain advisory services to BNY Mellon Securities Corporation. BNY Mellon Investor Solutions and BNY Mellon Securities Corporation are companies of BNY Mellon.

NOT FDIC INSURED | NO BANK GUARANTEE | MAY LOSE VALUE

WM-211751-2021-09-02