SO methods

Graham Burgess

February 2024

1 Introduction

How we categorise simulation optimisation (SO) methods:

- decision variable discrete or continuous. In the discrete case:
  - Finite or infinite number of feasible solutions
  - Ordered or unordered feasible solutions
  - Random search used or not. For random search methods:
    - neighbourhood structure (adaptive usually better than fixed)
- how the problem may be constrained:
  - unconstrained, partially constrained, fully constrained
  - stochastic constraints, deterministic constraints, box constraints
- dimensionality of the problem
- local or global solutions found
- convergence guarentee

The problem we face in the housing waiting-list problem has the following corresponding characteristics:

- In reality the decision variables (numbers of units of housing/shelter to build) are discrete, however we may work with a continuous approximation of the system. In the discrete case:
  - We have a finite but large number of feasible solutions
  - Within each dimension of the solution space, solutions are ordered, however ordering across dimensions is not obvious
- The problem we currently have in mind is:
  - fully constrained
  - deterministic constraints

- There is potentially a medium number (approx. 10) of dimensions (how much of housing/shelter to build each year over some reasonable time horizon for planning
- We would like a global solution, however we may still be interested in locally convergent methods provide we can add an element of random searching to escape any local optimum if necessary.

2 Integer ordered decision variables

This occurs when the decision variables are numerical in nature and only take integer values (e.g. how many staff to hire, how many houses to build) - can be seen as the integer points on a lattice.

2.1 Discrete Stochastic Approximation

The idea is that the objective function on the discrete space (say f) is extended to the continuous space (say $\tilde{f}$) and SA is performed on this extension. Therefore at each iteration, the step to a new solution is not restricted to an integer point, which means it is 'free' to quickly move towards the optimal solution. Finally, one rounds to the nearest integer point.

In the 1-D case, a simple linear interpolation between points in f is performed to construct $\tilde{f}$. Then, following an initial guess θ_0 of the minimiser of $\tilde{f}$, gradient descent steps are taken using a gradient estimator obtained via (multiple replications of) simulation of the integer points either side of θ_0. The step sizes are a sequence of positive numbers.

When $d > 1$, the extension from f to $\tilde{f}$ for $d > 1$ is done by partitioning the continuous space (using a colletion of convex hulls built on points in the integer space) so that every point in the continuous space is contained within one of the convex hulls in the collection. Then $\tilde{f}$ is constructed as a linear combination of f at all points on the surrounding convex hull. The weights in the linear combination depend on the proximity of the point in continuous space, to the corresponding points on the convex hull. Gradients are constructed using estimates (via multiple replications of simulation) of f at the points on the convex hull. At each iteration, once a gradient is constructed, one moves to a new point by taking a step along the gradient, with the step sizes a sequence of positive numbers. Now that the dimensionality of the problem is greater than 1, multimodularity of the objective function (or L^{sharp}-convex) (as defined in the paper) is required to guarantee the convergence of this algorithm, sinc for multimodular (or L^{sharp}-convex) functions, the extenstion to the continuous space gives a convex function.

Step-size at iteration n is of the form $a/(b+n)$ where a and b are positive constants. b is chosen as the total number of iterations and a is selected by the user based on what is most advantagous for an initial step-size.

2.2 R-SPLINE

R-SPLINE is "retrospective" in that it solves a series of sample-path problems, using the previous solution (i.e. retrospective) as a starting point for each new sample-path problem. Each sample-path problem is generated implicitly (i.e. sample-paths are collected as needed as the algorithm is progresses). Each

sample-path problem is solved using a repeating combination of a continuous line search, and a neighbourhood evaluation.

The line search moves along "phantom" gradients from integer point to integer point in search of a better solution. Each move starts with a random perturbation from the current best (integer) solution to a continuous point. Then the objective function value and gradient at this point is computed with a linear interpolation very similar to that in DSA. There is then "continuous" movement along this gradient and a jump to the nearest integer point, in search of an improved integer solution. The step-size at each continious movement is calculated to be as small as possible so that the jump to the nearest integer point will land on a new integer point. The line search stops when a move within the search is small (below some threshold). Once a line search is complete, a neighbourhood evaluation is performed to check that the current best is indeed a local solution (within all of its nearest neighbours, in every dimension). The combination of the line search and the neighbourhood evaluation is repeated until no improvement is found, or until the total simulation budget is used.

There is similarity between R-SPLINE and DSA in the way the objective function value and gradient in the continous space are calculated. However, there are differences: DSA, once transformed into the continous space, only moves in the continuous space, whereas R-SPLINE regularly jumps to the nearest integer point, before randomly perturbing away from that integer point. R-SPLINE involves repetitions of the line search and a neighbourhood evaluation, however DSA performs no neighbourhood evaluation as it moves towards the optimal solution. There is also a difference in how the step size is calculated as the search moves along the gradients. R-SPLINE has has an adaptive step-size (explained above), whereas DSA gets smaller and smaller as the algorithm progresses. Both algorithms increase the number of simulation replications from iteration to iteration.

More recently, a new algorithm called "ADALINE" incorporates many features of R-SPLINE but invovles an adaptive sample size at each iteration so as not to waste computation budget.

2.3 COMPASS

Convergent optimisation via most-promising-area stochastic search (COMPASS). Based on random search (like many other DOvS algs in literature) but offers a unique adaptive neighbourhood structure known as the "Most promising area" (MPA). For terminating and steady state simulation. For fully/partially/unconstrained problems. The fully-constrained setting is described as follows:

The algorithm starts by simulating an initial solution and the entire feasible region is in the MPA. Then, new solutions are sampled from the MPA and simulated. Once simulation has taken place, the "promising"-index of each

feasible solution is the sample mean performance of the visited solution which is closest to it. Therefore, the MPA is the set of feasible sols which have the current sample best solution as their closest visited solution. By design, the only previously-visted solution in the MPA is the current best. At each iteration, COMPASS updates the current best solution and the MPA and assigns simulation experiments (using a Simulation Allocation Rule) to candidate solutions sampled uniformly from the the MPA. It keeps information from previously-visited solutions over the course of the algorithm.

When the problem is partially- or un-constrained, a boundary around the current best solution is maintained and is widened as candidate solutions get close to the boundary. Within the boundary, the algorithm works as in the above fully-constrained description.

If the algorithm gets stuck in the wrong area (due to simulation noise) because all visited solutions are allocated further simulation budget, the algorithm will eventually move away from the sub-optimal area. The user must think carefully about how many solutions to sample from the MPA at each iteration. Fewer will reduce the time taken to find a local optimum, but the quality of this local optimum may not be good.

2.4 GMRF

2.5 RSM in a discrete setting

Response Surface Optimization with Discrete Variables (2004)

3 Continuous decision variables

3.1 Sample Average Approximation

"A Guide to Sample-Average Approximation" - Kim, Ragu, Shane Henderson Multi-dimensional newsvendor problem analysis shows that differentiability and expectation are interchangeable. The true function and the sample problem have the same nice properties of smoothness and concavity. So then - sufficiently large N -¿ effectively concave and deterministic opt can be used to find optimum.

We can say that SAA is appropriate for a problem if there is a structure to the sample average function which allows a nice deterministic optimisation solver to be used, and that the true function can be said to share that same structure.

3.1.1 Infinite dimension SO problems - (function decisions) - approx with SAA

Singham et al.

3.1.2 Retrospective framework

How this differs from SAA

3.2 Stochastic Approximation

3.3 Meta-modelling

3.3.1 Trust-region methods

3.3.2 Response Surface Methodology

STRONG

3.3.3 Global meta models

BO

3.3.4 Multi-fidelity models

A simulation-based optimization algorithm for dynamic large-scale urban transportation problems - Osorio and Chong 2018

Zirui Cao et al. CLUSTER-BASED SAMPLING ALLOCATION FOR MULTI-FIDELITY SIMULATION OPTIMIZATION (WSC 2023)