Integer Programming, Part 2

Tony Starfield

Recorded: March, 2011

Terri's shown you now how to solve a binary integer programming problem on a spreadsheet. She hasn't necessarily used the kind of algorithm you can get by going to the Web and downloading a integer programming algorithm. But, nevertheless, she's come up with a fairly effective solution, or a very effective solution using Solver.

You have to be a little bit careful when you use Solver to solve integer programming or binary integer programming problems, because different versions of software sometimes do different things. So, for example, it might be very essential to tell a computer ahead of time that the integers have to be greater than or equal to 0. Others will solve a linear programming problem and then round the answers up or down, which is, in fact, the wrong solution. So just be a little bit careful with the version of Solver you might be using. If you have a very large problem, it probably pays to go and download a binary or an integer programming algorithm from the Web.

Well as we were saying before Terri showed you what to do on the spreadsheet, we have two solutions here: a heuristic solution and an algorithmic solution to the same problem. And I'm going to ask what might seem to be a bit of a stupid question, and that is, "Which one is correct?" And you're probably saying, "Obviously, the algorithmic solution is correct because it gives you the guaranteed optimum solution."

One has to be learn to be a little careful when one's using the word 'optimum.' People love to use it. They love to say, "We are going to optimize. We're going to give you the optimum solution." But the optimum depends on precisely how you define the problem. As you saw in the linear programming exercises, if you change one of the constraints a little bit, you could get a completely different solution. So optimum depends precisely on the definition of the problem.

How precisely are we define the problem in this case? Well if you actually look at the handout, you'll realize that what you were given was a model-world problem, not a real-world problem because we'd left a whole lot of things out. We had already gone through the process of reducing the real world to a model world. And it's very useful before we get picky about whether we are using the right technique for solving the problem to make sure that we're solving the right one.

Let’s look at some of the things we’ve left out. We haven’t asked how well a site suits a species. We’ve just asked, “Is it present or absent?” Some sites might be ideal habitat. Other sites could actually be sinks for the species where the population occurs but can’t breed effectively.

We haven’t asked about the size of the site. Is the site large or small? If it’s large, there are going to be fewer edge effects.

We haven’t asked about the shape, which could also affect edge effects.

We haven’t asked, “Is the site close to a city or close to a large road, or are there roads through it?”

There is all sorts of information about each site that we’ve just totally ignored. And we’ve also ignored what could be important political aspects. For example, which state is the site in? If one were trying to solve a problem like this for the whole of the United States of America, including Hawaii, and you just got the computer to come up with a solution, you’d spend most of your money probably in Hawaii and Southern California, because those are your species--your biodiversity hotspots. How would you get Congress to produce funds if most of those funds are going to be spent in only two states? So there are political issues as well.

Now a purist might say, “Just tell me all these things, and I’ll come up with suitable constraints, and we’ll put it all into an algorithmic solution.” But I don’t think that is the way to go here.

And what I want to introduce to you is two new terms and the way in which they’re used. The one word is “**hard**,” and the other word is “**soft**.” And one talks about hard or soft solutions to problems.

What one means by “hard” is not difficult. One means hard in being a very clear-cut, hard-edged technical problem. In other words, if I ask, “Which is the better solution,” and one is an algorithm, and the other is a heuristic, the hard answer is - without any doubt - the algorithm is better, on purely technical grounds, even though technical--you’re getting a wonderful solution to the wrong problem.

A soft solution looks at the bigger picture. It would ask questions like, “What about the things we haven’t put into our definition of the problem? How do we bring those in?” It would also ask questions like, “We’ve come up with the technically best solution that is the optimum, in terms of how we define the problem, but how likely are people to implement it?” Is there another solution which isn’t quite as good but which people will really buy into and will implement? In other words, would you rather have the perfect answer that people are not going to implement versus a slightly less-good answer that people are going to implement? Those are the issues that soft people deal with.

And at a conference once--there was somebody who made the famous statement, “It is the soft problems that are hard. The hard problems are easy.” Think about it.

So this sort of gets one to the point of thinking about how would one really deal with a very, very large problem like this, given that you might have politicians and stakeholders and other people whom you want to get to buy into the solution, again, irrespective of whether you use the algorithmic or the heuristic computer program to solve the problem. How do you get people to buy into an answer?

In other words, how, for example, would you make good use of a committee? If you have a committee that’s supposed to help you make a decision, and you run an optimization program and give them the answer and say, “Okay, everybody, here’s the best answer, rubber stamp it.” You’re not going to get buy-in. Worse, you’re not going to draw on the expertise of the people on that committee.

On the other hand, if you came up with a Presence-Absence table with maybe 1,000 sites and 2,000 species, and you gave a committee--each member of the committee this huge table with a whole lot of 0s and 1s and said, “Okay, committee, come up with an answer.” Well obviously, you wouldn’t get anywhere either.

So the question is how do you make the best use of the computer and the best use of committees and stakeholders? And you could rephrase that question by saying, “How much do you need to chew on, or analyze the information before you pass it on to committees to reach a solution?” And you want to make sure that you try and get the computer to do what it does best and the committee to do what they do best.

So, for example, if I were looking to a good approach to a problem like this, I would want to reduce all the information in that huge Presence-Absence table, down to maybe five or six different portfolios. And each of those portfolios, I would want to be a pretty good portfolio. What do I mean by a pretty good portfolio? They do a great job of saving species, and they are not too expensive. And then let people choose any one of those five. From my perspective, as an analyst, I don't care much which they choose because I've done a pretty good job of giving them five defensible solutions. They care about which they choose. And the way in which they choose there might determine whether or not they buy into the solution and actually implement it.

So if I take that kind of approach, what I have to think about as an analyst or modeler, is how could I produce five or six pretty good solutions. And maybe how I could use a mix of the heuristic and algorithmic approaches. So maybe I could get one solution where I have to save each species in at least one site. Maybe I could get another portfolio by changing that set of constraints so that I save each species in at least two sites. That would be a clever thing to do. And notice, by the way, if that is what one was trying to do. The whole heuristic-salami-tactics approach we developed goes right out the window. It's just not going to work at all. And then maybe I'll try and develop some very different solutions using the heuristics.

So to sum up, then, for a problem like this, one wants to be imaginative. And let's look at what we've learned and think a bit about how one uses what we've done, imaginatively.

First of all, we've come up with a distinction between algorithms and heuristics. Algorithms come with guarantees. Heuristics might sometimes fail you. Algorithms are, perhaps, technically harder to explain. Heuristics might be more appealing to people.

From an algorithmic point of view, we've learned two new techniques. If we're keeping a toolkit of techniques that we might want to use as modelers, we can now add to that toolkit integer programming and binary programming.

And then we've had this very important discussion about hard versus soft solutions and how important it is to bring people and the people who are going to implement the solution and their ideas into a problem. One's got to remember that every time you optimize, you are optimizing a model-world problem, and maybe people need to think about the real world.

And, finally, we've talked about how modelers and analysts and facilitators can work together to try and get stakeholders, groups of people, committees, and so on to think intelligently about how to solve problems.

What can you do from a technical point of view that will make it easier for people to reach a good solution without necessarily coming through with one solution and saying, “That’s the only solution one could use.”

So we’re beginning to think of our models and computer software as facilitation tools, where we are not using them to find the answer. We are using them to help other people discuss a problem and find an answer that they are comfortable with, find a good answer, if not the best answer that they are comfortable with.

Terri’s now going to show you how you might be able to use your software in this kind of way.

< 00:14:08 END >