DECISION ANALYSIS

Integer Programming, Part 1

Tony Starfield

Recorded: March, 2011

You have in front of you a problem called "The Endangered Species of Int." I'm assuming you've read it. This is a useful sort of problem. It's an example of what are generally known as portfolio problems. In this case, you're trying to collect a portfolio of properties that will enable you to preserve a whole set of species.

PAUSE the video: Read the Endangered Species of Int.

Just to explain one or two things in the description of the problem, the table is an example of what's known as a **Presence-Absence Table**. So down the left-hand side, you have the sites. Across the top, you have the species. A 0 means that species is not found at that site and a 1 means that the species has been found at that site.

In real-world problems, very often this table is much, much larger. Here you have 14 species in 12 sites. People might look at, for example, how to choose conservation areas in the whole of Australia. And if they were to do that, they might have 1,000 sites and a few thousand species.

So when we think about solving this problem, any technique that we come up with must be one that would work equally well on a very large table. You can't look for shortcuts.

The idea here is that, associated with each site, there is a price. And you want to come up with a portfolio that is cheap, but preserves or provides habitat in at least one of the sites, for every single species on the list.

Now I'm going to give you a hint before I let you try and think about how you might do this, and that hint would be to use salami tactics on this problem. And think a little bit about what we mean by salami tactics here, because so far, we've seen salami tactics in the context of coming up with a way of getting from one time step to the next in a model. Here, salami tactics would probably mean choose or develop your portfolio one property at a time. In other words, come up with a way of deciding which property you're going to buy first and then, on the basis of that, how you would use that same approach to decide which property you could buy second, third, fourth, until you've developed your whole portfolio.

I'd like you to spend maybe five or ten minutes, see if you can come up with a salami-tactic-type solution to this problem. Doesn't matter if you don't work it all the way through as long as you have it quite clear in your own mind how it would work and whether this would work equally well on a much larger table.

<ins>PAUSE the video</ins>: Work on the salami tactic problem.

What we have here is just the top left-hand corner of the Presence-Absence Table, but it's enough for us to be able to see how our salami tactics might work. As you probably realized when you tried to grapple with this problem, you have two things to deal with. The one is the cost of each site. So there's the question of cost. And the other is the presence or absence of the species.

So how do you balance, when you're choosing a site, the question of how many species are on that site and how much it costs? And the answer is that what you need to do for every single site is, first, total up the number of species that occur going all the way across the row, and then either divide the number of species into the cost to get the cost per species, or divide the number of species by the cost to get the species per dollar.

What you want to do, if you're going to choose just one site, is to choose the site that gives you the maximum number of species per dollar. Now that would be the first slice of the salami. What do you do after you've done that?

Well, for example, let's suppose, for just the sake of argument, that it turned out that Site 4 was the site that had the maximum number of species per dollar. What you would then do is remove Site 4 from your table. You would also know that you no longer have to save the species that occur on Site 4. So you could remove the column wherever there is a 1 in the site you've chosen. And if you did that for the entire table, you would then end up with a table with one less site and a number of fewer species. And you could then compact that table, recalculate the number of species remaining at each site. You've got the cost of the site. So you could go on to the next slice of the salami until there are no more species to save.

That will be quite easy to do, and it would be quite easy to do; although, it might take the computer a little longer, on a very large Presence-Absence Table. And, in fact, people have developed computer programs that work pretty much in that way to help you choose your portfolio.

But at the end of the day, once you've got an answer, once you've selected what you think is your best portfolio, how do you know that it is really your best? In other words, the question that I'm asking is, "Does the salami-tactic paradigm lead to a heuristic or an algorithm?" In other words, does it or does it not come with a guarantee that it is going to come up with a portfolio that really is the cheapest?

And the answer, alas, is that it's a heuristic. In other words, you might get a pretty good portfolio, in the sense that it saves all the species and is relatively inexpensive, but you might not come up with the best. And the question then arises, "Is there a way to get to the very best?" In other words, if this is a heuristic that we've developed, is there an algorithm for solving the problem, even if it's a very big problem? And the answer is, "Yes, there is." And I'm going to lead you through it.

What one should start off with, as one does in most of these problems, is thinking up a notation. And, again, we get to the nub of what are we trying to do here? What are the questions we're asking? And the question we're really asking about each site is do we buy it, or don't we buy it?

So a suitable notation might be to introduce, say, a variable X, and the variable X for site number K we could call X subscript K. And we could say that variable is either 0 or 1. It can't be anything else. It's 1 if we buy the site, and it's 0 if we don't buy the site. And since we have 12 sites, we could have an X1 and X2, all the way up to X12. And all of those are 0s or 1s, and the combinations of 1s and 0s make up a portfolio.

Well if we go along with that notation, can we write down the cost of that particular portfolio in terms of the Xs if we don't know beforehand which of those we're actually going to buy? And the answer is, "Yes." Here is a formula for the price of the purchase. What we do is we multiply each of the Xs by the cost of that particular site.

So, for example, for the first site, the cost is 10 times X1 in thousands of dollars. Does that work? Yes it does because, remember, X1 is 0 or 1. If it's 0, which means we don't buy the site, 10 times 0 is 0, and we don't pay anything for it. If X1 is 1, then we are paying 10 thousands of dollars for it.

So we now have an expression that represents the cost of a purchase plan or portfolio, and we want to minimize this. But we want to minimize it subject to a set of conditions. And, remember, those conditions are that we have to make sure that we have saved each species in at least one of the sites.

So the question we have to ask next is, “How, for example, do we ensure that we save species number 1?” And if we could go back to our Presence-Absence Table, you will see that species number 1 occurs in sites 3, 5, and 6. And, in fact, if we had the whole table, we would also see it in 8, 11, and 12.

So if you wrote down X3, X5, X6, X8, X11 and X12, in other words, all the sites where it occurs, and suppose you added those together. Well, if it turned out that you hadn’t bought any of those sites, then adding it all together--all those Xs together would just give you 0. But if you had bought one of those sites, then adding the Xs will give you 1. So your condition is X3 plus X5 plus X6 plus X8 plus X11 plus X12 is greater than or equal to 1.

And you can do something similar for all the other 13 species--all the way to species number 14, where you would have X3 plus X4 plus X9 plus X11 has to be greater than or equaled to 1. And since you already have this table, if you had it in a computer with 0s and 1s, I think you could see how you could automate that set of conditions pretty easily.

Okay, so what have we got here? We have a problem where we have 12 variables, X1 through X12. We have 14 conditions, which are inequalities to make sure that we save every single species in at least one site. And then we have an expression that we want to minimize. What’s more, both the expression we want to minimize and the inequalities are, mathematically speaking, linear. This is beginning to look awfully like a linear programming problem, right?

Remember what makes a linear programming problem. First of all, you have to have variables. Those variables have to be greater than or equal to 0. You’ve got that. You’ve got your 12 variables. Secondly, you have to have an objective function that you maximize or minimize. You’ve got that, and it’s linear. And then you have a set of constraints, and lo and behold, you have 14 constraints here, and they are linear.

But there’s a catch. The catch is that your variables aren’t just real numbers as they are in linear programming. They have to, in the first place, be integers. And, secondly, they can’t just be any integers. They’re either 0 or 1.

Well it turns out that somewhere along the line, somebody said, “You know there are problems, a lot of problems out there that look rather like a linear programming problem. But the variables have to be integers that are 0 or positive.” And somebody came up with an algorithm for solving that kind of problem. And that is known as an **integer programming algorithm**.

If you were to use an integer programming algorithm for this problem, you’d have to somehow either tell the computer if you took a code off the Web, for example, that your variables can’t be 2, 3, 4, or 5. So you’d have to add another 12 constraints, which basically said, “X1 is less than 2. X2 is less than 2. X3 is less than 2,” and so on. And if it’s less than 2, and it’s an integer, it’s 0 or 1.

Somebody else then came along and said, “You know, in a lot of integer programming solutions, we actually have integers that have to be 0 or 1. We don’t worry about 2, 3, 4 or 5.” And they came up with an algorithm for solving what is called a **‘binary programming’** problem. And that is a problem where your integers have to be, ahead of time, 0 or 1. And in that case, you don’t need those extra constraints because the algorithm already knows that your variables are 0 or 1.

So what we have here is a problem where there’s an intuitively obvious heuristic solution, which is, perhaps, very easy to explain to a group of stakeholders or people who might be involved in trying to come up with a solution. And you also have the rather esoteric algorithmic solution that guarantees that it’s really going to get you the best answer, but might be a little bit more of a black box to the people that are dealing with the problem.

Before we discuss this duality or this conflict between easy to explain and guarantees, let’s see how you might be able to solve a problem like this on a spreadsheet. Terri’s going to show you how to do that now.

< 00:17:17 END >