Feasibility in stochastic programming

An argument against constraints

Stein W. Wallace

Department of Business and Management Science
Norwegian School of Economics

May 10, 2013

Types of feasibility constraints

Feasibility comes naturally in two forms:

- Book-keeping constraints, typically inventory and conservation-of-flow
- Resource constraints

The first set is very useful, the second set a source of trouble.

The knapsack problem

$$
\begin{array}{ll}
\text{maximize} & \displaystyle\sum_{i=1}^{n} c_i x_i \\
\text{such that} & \left\{ \begin{array}{lcl} \sum_{i=1}^{n} w_i x_i & \leq & b \\ x_i \in \{0, 1\} & , & i = 1, \ldots, n \end{array} \right.
\end{array}
\tag{1}
$$

where:

c_i is the value of item i,

w_i its weight, and

b is the capacity of the knapsack.

Two types of models

Inherently two-stage (invest-and-use) models are models where the first stage is a major long-term decision, expensive or irreversible (or both). Examples are major investments (such a buildings and ships) and airline schedules.

Two types of models

Inherently two-stage (invest-and-use) models are models where the first stage is a major long-term decision, expensive or irreversible (or both). Examples are major investments (such a buildings and ships) and airline schedules.

Inherently multi-stage (operational) models are models where all stages in principle are of the same type. Examples are production-inventory models, financial portfolio models and project scheduling.

Random weights and stages

What if the weights are random?

Random weights and stages

What if the weights are random?

- What is the inherent stage structure and how many stages are there?

Random weights and stages

What if the weights are random?

- What is the inherent stage structure and how many stages are there?
- Major question: When will we learn the weights?

Random weights and stages

What if the weights are random?

- What is the inherent stage structure and how many stages are there?
- Major question: When will we learn the weights?
  - We learn the weight *of each item just before* we decide whether or not to put it into the knapsack.

Random weights and stages

What if the weights are random?

- What is the inherent stage structure and how many stages are there?
- Major question: When will we learn the weights?
  - We learn the weight *of each item just before* we decide whether or not to put it into the knapsack.
  - We learn the weight *of each item just after* putting it in.

Random weights and stages

What if the weights are random?

- What is the inherent stage structure and how many stages are there?
- Major question: When will we learn the weights?
  - We learn the weight *of each item just before* we decide whether or not to put it into the knapsack.
  - We learn the weight *of each item just after* putting it in.
  - We learn the weight *of the full set of items just after* we decide what items to put in.

Random weights and stages

What if the weights are random?

- What is the inherent stage structure and how many stages are there?
- Major question: When will we learn the weights?
  - We learn the weight *of each item just before* we decide whether or not to put it into the knapsack.
  - We learn the weight *of each item just after* putting it in.
  - We learn the weight *of the full set of items just after* we decide what items to put in.
- The first two will normally lead to inherently multi-stage models, the third to inherently two-stage models.

Inherently two-stage models

1. We may require that the chosen set of items always fits into the knapsack.

Inherently two-stage models

1. We may require that the chosen set of items always fits into the knapsack.
2. We may list the items in a certain order and pick them up until we come to one that does not fit. Then we stop. (So the decision is the list.)

Inherently two-stage models

1. We may require that the chosen set of items always fits into the knapsack.
2. We may list the items in a certain order and pick them up until we come to one that does not fit. Then we stop. (So the decision is the list.)
3. We may do as above, but if a later item fits (as it is light enough) we take it.

Inherently two-stage models

1. We may require that the chosen set of items always fits into the knapsack.
2. We may list the items in a certain order and pick them up until we come to one that does not fit. Then we stop. (So the decision is the list.)
3. We may do as above, but if a later item fits (as it is light enough) we take it.
4. We may list the items, and keep adding items until we have added an item that did not fit. We then pay a penalty for the "overweight".

Inherently two-stage models

1. We may require that the chosen set of items always fits into the knapsack.
2. We may list the items in a certain order and pick them up until we come to one that does not fit. Then we stop. (So the decision is the list.)
3. We may do as above, but if a later item fits (as it is light enough) we take it.
4. We may list the items, and keep adding items until we have added an item that did not fit. We then pay a penalty for the "overweight".
5. We may pick a set of items, such that if the items do not fit into the knapsack after we have learned their weights, we pay a penalty for the total overweight.

Inherently two-stage models

1. We may require that the chosen set of items always fits into the knapsack.
2. We may list the items in a certain order and pick them up until we come to one that does not fit. Then we stop. (So the decision is the list.)
3. We may do as above, but if a later item fits (as it is light enough) we take it.
4. We may list the items, and keep adding items until we have added an item that did not fit. We then pay a penalty for the "overweight".
5. We may pick a set of items, such that if the items do not fit into the knapsack after we have learned their weights, we pay a penalty for the total overweight.
6. We may pick a set of items of maximal value so that the probability that the items will not fit is below a certain level.

How many stages in the model?

If we need a set, the model is probably two-stage. If we need a list, it is probably multi-stage. So most likely, setting up a list is much harder than finding a set.

How many stages in the model?

If we need a set, the model is probably two-stage. If we need a list, it is probably multi-stage. So most likely, setting up a list is much harder than finding a set.

But what if we want to pick the items one by one and then continue as long as there is still room?

How many stages in the model?

If we need a set, the model is probably two-stage. If we need a list, it is probably multi-stage. So most likely, setting up a list is much harder than finding a set.

But what if we want to pick the items one by one and then continue as long as there is still room?

- This will most likely lead to a very hard multi-stage model.

They must all fit ...

This is a worst-case setting. Is this really what you want? What can happen with such an approach?

They must all fit ...

This is a worst-case setting. Is this really what you want? What can happen with such an approach?

- First think about the problem itself: Is this is really required? This is a *modeling* question.

They must all fit ...

This is a worst-case setting. Is this really what you want? What can happen with such an approach?

- First think about the problem itself: Is this is really required? This is a *modeling* question.
- But what if an item does not fit? Is there nothing we can do?

They must all fit ...

This is a worst-case setting. Is this really what you want? What can happen with such an approach?

- First think about the problem itself: Is this is really required? This is a *modeling* question.
- But what if an item does not fit? Is there nothing we can do?
- Send it the next day? With another mode? By mail? Put the last box in the passenger seat?

They must all fit ...

This is a worst-case setting. Is this really what you want? What can happen with such an approach?

- First think about the problem itself: Is this is really required? This is a *modeling* question.
- But what if an item does not fit? Is there nothing we can do?
- Send it the next day? With another mode? By mail? Put the last box in the passenger seat?
- Saying that all items must be delivered is *not* equivalent to a saying that all items must fit in the knapsack!

They must all fit ...

This is a worst-case setting. Is this really what you want? What can happen with such an approach?

- First think about the problem itself: Is this is really required? This is a *modeling* question.
- But what if an item does not fit? Is there nothing we can do?
- Send it the next day? With another mode? By mail? Put the last box in the passenger seat?
- Saying that all items must be delivered is *not* equivalent to a saying that all items must fit in the knapsack!
- What if the size is not really known? What could then happen?

They must all fit ...

This is a worst-case setting. Is this really what you want? What can happen with such an approach?

- First think about the problem itself: Is this is really required? This is a *modeling* question.
- But what if an item does not fit? Is there nothing we can do?
- Send it the next day? With another mode? By mail? Put the last box in the passenger seat?
- Saying that all items must be delivered is *not* equivalent to a saying that all items must fit in the knapsack!
- What if the size is not really known? What could then happen?
- A model which requires all items to fit, but where it is still very likely that they don't.

Do we really want hard constraints?

Unless the constraint is really (really) hard, I would say no! Can you give an example of a genuinely hard constraint?

Do we really want hard constraints?

Unless the constraint is really (really) hard, I would say no! Can you give an example of a genuinely hard constraint?

Instead, use penalties to represent resource constraints, if necessary with very high penalties, but not ∞.

$$\max \quad \sum_{i=1}^{n} c_i x_i - d \sum_{s \in \mathcal{S}} p^s z^s$$

$$\text{such that} \quad \begin{cases} \sum_{i=1}^{n} w_i^s x_i - z^s \leq b & \forall s \in \mathcal{S} \\ z^s \geq 0 & \forall s \in \mathcal{S} \\ x_i \in \{0,1\} & 1 = 1, \ldots, n \end{cases} \tag{2}$$

where d is the unit penalty for overweight. We call this a *penalty formulation* since it can be replaced by

$$\max_{x_i \in \{0,1\}} \quad \sum_{i=1}^{n} c_i x_i - d \sum_{s \in \mathcal{S}} p^s \left[\sum_{i=1}^{n} w_i^s x_i - b \right]_+ \tag{3}$$

where $[x]_+$ is equal to x if $x \geq 0$, zero otherwise.

Chance constrained formulation

$$
\max_{x_i \in \{0,1\}} \quad \sum_{i=1}^{n} c_i x_i
$$
$$
\text{such that} \quad \begin{cases} \sum_{s \in W(x)} p^s & \geq \alpha \\ W(x) = \{s : \sum_{i=1}^{n} w_i^s x_i & \leq b\} \end{cases} \tag{4}
$$

where α is the required probability of feasibility.

Note that this model (as the worst-case model) is sensitive to parameters. It depends seriously on the worst-case weights.

Conclusion on feasibility

- It is very rare that hard resource constraints exist. Use them only if absolutely sure.

Conclusion on feasibility

- It is very rare that hard resource constraints exist. Use them only if absolutely sure.
- Use penalties instead, very high ones if need be.

Conclusion on feasibility

- It is very rare that hard resource constraints exist. Use them only if absolutely sure.
- Use penalties instead, very high ones if need be.
- Remember the difference between "must be done" and "must be done within the model".

Conclusion on feasibility

- It is very rare that hard resource constraints exist. Use them only if absolutely sure.
- Use penalties instead, very high ones if need be.
- Remember the difference between "must be done" and "must be done within the model".
- Be particularly careful when "worst-case" is not well understood.

Conclusion on feasibility

- It is very rare that hard resource constraints exist. Use them only if absolutely sure.
- Use penalties instead, very high ones if need be.
- Remember the difference between "must be done" and "must be done within the model".
- Be particularly careful when "worst-case" is not well understood.
- Remember that a hard constraint means: I am willing to pay *any finite amount* to make this true.