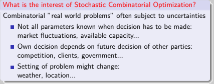


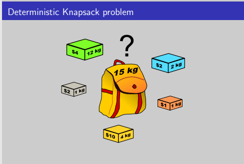
- Random parameters easy to implement $\rightarrow$ Random variables
- "Code" other uncertainties in parameters

Definition

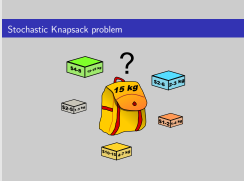
Stochastic Combinatorial Optimization concerns the study and resolution of Combinatorial Optimization problems that involve uncertainties.

Note that there is a slight difference between the usage of the term "Stochastic Optimisation" and "Stochastic Programming".
Stochastic Programming designs the modeling and study of optimization problems that involve uncertainties.
Stochastic Optimization addresses the study of optimization algorithms that are either randomized or created to solve stochastic programming problems.

However, these definitions are not always properly used and of course both fields intersect in a lot of aspects.



Deterministic knapsack problem: The problem consists in choosing a subset out of a given set of items such that the total weight (or size) of the subset does not exceed a given limit (the capacity of the knapsack) and the total benefit/reward of the subset is maximized.



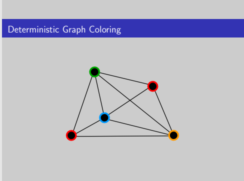
What happens if item rewards or weights are random? What is a feasible solution? For example, is it allowed to add all items apart from the green one although they *might* violate the capacity constraint? And what happens if they do?

Possible ways to handle capacity constraint

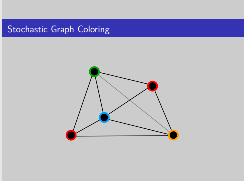
- knapsack constraint violated ⇒ penalty
- probability of capacity violation restricted
- decision can be corrected later (add. costs/reduced rewards)

In the first two examples violation was acceptable. But what if a violation is not allowed, in any case? Well, we could force our solution to *always* respect the knapsack constraint. In this case at most 3 items could be chosen, at a much lower reward.

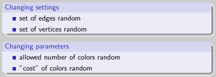
Or we could see, if a correction might be possible later (3rd example), i.e. we chose the 4 items (not the green one) and then, if their total weight exceeds the capacity, we reject one item.



Deterministic Graph Coloring: Color a graph such that no two adjacent vertices are colored in the same color and such that a minimum number of colors is used.
In this example: Use of 4 colors is optimal as graph contains complete graph with 4 vertices.
Application: Assignment problems. The vertices could represent university courses, two courses are linked iff there is at least one student that wants to attend both courses. Coloring the obtained modelgraph with the minimum number of colors tells you how many time slots you need to schedule these courses.

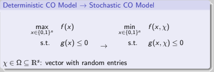


What if, at the moment where you have to create the schedule, you do not know the decision of the students yet? And what if there are two courses with a very low probability that a student wants to take both of them? And what if you are running out of time slots? You might consider coloring both vertices with the same color and reduce the number of used colors:



Vertex set random: You do not know which courses will take place in the end. For example a course might be cancelled due to lack of students.

Number of allowed colors random: The university might assign you a restricted number of time slots, that might change in the future due to changings in other programs.



If you have an SCO optimization with random parameters and you fix these parameters, you get a deterministic CO problem.
The other way round, if you have a deterministic CO problem and you assume some of the parameters to be ranom your problem gets stochastic. Question: Where does the randomness occurs? Only in the objective, only in the constraint, in both?

Minimize an expectation

$$\min_{x \in \{0,1\}^n} \quad \mathbb{E}\left[f(x,\chi)\right]$$
$$\text{s.t.} \quad g(x) \leq 0$$

Advantages:

- Good result "on average"
- Objective function can often be reformulated deterministically

Disadvantages:

- We might encounter very "bad cases"

- $\mathbb{E}[X]$: expectation of random variable X

Minimize variance

$$\begin{aligned} \min_{x \in \{0,1\}^n} \quad & Var\left[f(x, \chi)\right] \\ \text{s.t.} \quad & g(x) \leq 0 \end{aligned}$$

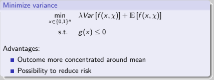
Advantages:

- Outcome more concentrated around mean
- Possibility to reduce risk

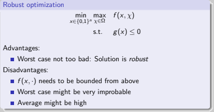
Disadvantages:

- Makes not much sense without benchmark for expected costs

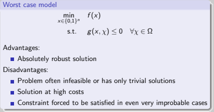
- $Var[X]$: variance of random variable X



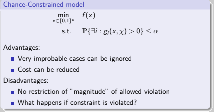
Role of λ: Control relative importance of expectation and variance in your model.



- Robust Optmization generally not considered as being part of Stochastic Optmization, e.g. as the solution algorithms and approaches are generally quite different.
- However, I think the presented worst case model can be of good use in many cases, at least as a subproblem.
- Most common assumed distribution: χ_i (uniformly) distributed over certain interval



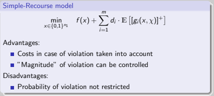
- consider a knapsack problem with no upper bound on random item weights: only feasible solution would be to add no item at all.
- Worst case problem considered as robust optmization problem.



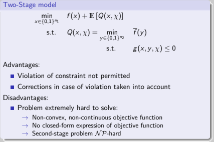
$\mathbb{P}\{A\}$: probability that event A occurs

$$\mathbb{P}\{\exists i : g_i(x,\chi) > 0\} = 1 - \mathbb{P}\{g_i(x,\chi) \leq 0 \quad \forall i\}$$

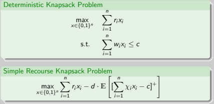
$$\mathbb{P}\{\exists i : g_i(x,\chi) > 0\} \leq \alpha \Leftrightarrow \mathbb{P}\{g_i(x,\chi) \leq 0 \quad \forall i\} \geq 1 - \alpha$$



- $d_i > 0$ penalty per "unit of expected amount of violation"
- $[x]^+ = max(0, x)$
- $[g_i(x,\chi)]^+ = 0$ iff constraint i satisfied, $[g_i(x,\chi)]^+$ gives "amount" of violation otherwise
- $\mathbb{E}\left[[g_i(x,\chi)]^+\right]$: expected amount of violation
- Combine Simple Recourse and Chance constraint in order to control both the magnitude and probability of violation



More general: First stage can of course have additional constraints.



- $[\sum_{i=1}^{n} \chi_i x_i - c]^+ =$ overweight
- $\mathbb{E}\left[[\sum_{i=1}^{n} \chi_i x_i - c]^+\right]$: expected overweight
- $d > 0$ penalty per overweight unit

Two-Stage Knapsack Problem

$$(TSKP) \quad \max_{x\in\{0,1\}^n} \quad \sum_{i=1}^{n} r_i x_i + \mathbb{E}[\mathcal{Q}(x,\chi)]$$

$$\text{s.t.} \quad \mathcal{Q}(x,\chi) = \max_{y^+,y^-\in\{0,1\}^n} \sum_{i=1}^{n} \overline{r}_i y_i^+ - \sum_{i=1}^{n} d_i y_i^-,$$

$$\text{s.t.} \quad y_j^+ \leq 1 - x_j, \quad j = 1,\dots,n,$$

$$y_j^- \leq x_j, \quad j = 1,\dots,n,$$

$$\sum_{i=1}^{n} (x_i + y_i^+ - y_i^-)\chi_i \leq c.$$

- Items can be added and/or removed in the second stage
- In the end remaining items need to respect knapsack capacity
- x: decision vector of 1^{st} stage
- y^+, y^-: decision vectors of 2^{nd} stage (recourse action)
- $\overline{r}_i < r_i$, $d_i > r_i$
- If $\overline{r}_i \geq r_i$: Add item i in 2. stage
- If $d_i \leq r_i$: Add item in 1. stage (removal is without cost or one even gains)

Static Stochastic Optimization problems

- Random parameters revealed after decision has been made.
- For decision maker parameters are revealed "once for all".
- No corrective decision can be made.

- Minimize expectation and/or variance
- Robust/Worst Case Optimization
- Chance-Constrained Optimization
- Simple Recourse Model

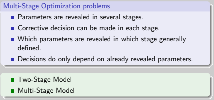
- Simple Recourse model: is paying a penalty a corrective decision? No.

Two-Stage Optimization problems

- For decision maker parameters are revealed "once for all".
- Random parameters revealed after first-stage decision has been made.
- Corrective decision can be made once the parameters are known.

- Two-Stage Model
- Simple Recourse Model

- Simple Recourse model: can be reformulated as Two-Stage decision model
- Continuous second stage decision variables y_i serve to "correct constraints"
- One variable for each constraint
- Second stage constraints: $g_i(x, \chi) \leq 0 + y_i$
- Optimal second-stage decision: $y_i = [g_i(x, \chi)]^+$



- Attention: In deterministic **multi-period** decision problems: future parameters all known
  decisions based on current parameters and future changings
  $\Rightarrow$ multi-period decision problems problems are deterministic problems

Structural Difficulties
- Non-convexity
- Non-continuous objective functions
- No closed-form expression for objective function

Two-Stage model

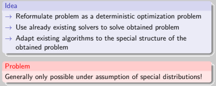
$$\begin{aligned} \min_{x \in \{0,1\}^{n_1}} \quad & f(x) + \mathbb{E}\left[Q(x,\chi)\right] \\ \text{s.t.} \quad & Q(x,\chi) = \min_{y \in \{0,1\}^{n_2}} \quad \overline{f}(y) \\ & \qquad\qquad \text{s.t.} \quad g(x,y,\chi) \le 0 \end{aligned}$$

The Two-Stage model is an extrem example for the structural difficulties as generally all of them are present (at least in case of second stage integer or binary decision variables)

Two-Stage Knapsack Problem

$$
\begin{aligned}
(TSKP) \quad & \max_{x \in \{0,1\}^n} \quad \sum_{i=1}^{n} r_i x_i + \mathbb{E}[\mathcal{Q}(x,\chi)] \\
\text{s.t.} \quad & \mathcal{Q}(x,\chi) = \max_{y^+, y^- \in \{0,1\}^n} \sum_{i=1}^{n} \bar{r}_i y_i^+ - \sum_{i=1}^{n} d_i y_i^-, \\
& \text{s.t. } y_j^+ \leq 1 - x_j, \quad j = 1, \ldots, n, \\
& \quad\;\; y_j^- \leq x_j, \quad j = 1, \ldots, n, \\
& \quad\;\; \sum_{i=1}^{n} (x_i + y_i^+ - y_i^-)\chi_i \leq c.
\end{aligned}
$$

An example for $\mathcal{NP}$-hardness of the second-stage problem is the Two-Stage Knapsack problem, as the second stage problem can be shown to be a "simple" knapsack problem.



Currently the most practiced approach to solve stochastic compbinatorial optimization problem. Unfortunately, as I think that if we tried to be a bit more innovative concerning the creation of special algorithms for SCO, we might advance faster.

Ex. 1: Chance-constrained Knapsack prob. (normal distribution)

$$\max_{x\in\{0,1\}^n} \quad \sum_{i=1}^{n} r_i x_i$$

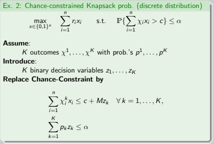
$$\text{s.t.} \quad \mathbb{P}\{\sum_{i=1}^{n} \chi_i x_i > c\} \leq \alpha$$

Assume:

$\chi \sim \mathcal{N}(\mu, \Sigma)$ and $\alpha < 0.5 \Rightarrow$

$$\mathbb{P}\{\sum_{i=1}^{n} \chi_i x_i > c\} \leq \alpha \Leftrightarrow \underbrace{\sum_{i=1}^{n} x_i \mu^i + \Phi^{-1}(1-\alpha)\|\Sigma^{1/2} x\| \leq c}_{\text{Second Order Cone Constraint!}}$$

- $\mathcal{N}(\mu, \Sigma)$: joint probability distribution for random vector χ
- $\mathcal{N}(\cdot, \cdot)$: (joint) normal distribution
- μ: vector of expectations of components of χ: $\mu_i = \mathbb{E}[\chi_i]$
- Σ: $n \times n$ covariance matrix
- here: Σ diagonal as weights assumed independently distributed
- Φ: cumulative distribution function of standard normal distribution
- values of Φ^{-1} can be looked up in tables
- $\alpha < 0.5$ needed for $\Phi^{-1}(1-\alpha)$ to be positive
- otherwise constraint not convex
- obtained constraint can be evaluated (one can check easily for feasibility)
- obtained problem type (Second Order Cone Problem) has been studied a lot and special algorithms have been proposed $\rightarrow$ no more "miracle" about how to solve the chance-constraint knapsack problem



- $M \geq \max_k(\sum_{i=1}^n \chi_i^k - c)$
- $z_k = 1$: scenario k is "ignored"
  corresponding constraint $\sum_{i=1}^n \chi_i^k x_i \leq c + M z_k$ always satisfied
- total probability of "ignored" scenarios must not exceed α

Idea of the SAA
→ Sample K outcomes for random parameters
→ Assign probability $1/K$ to each sample
→ Replace distribution by finite sample

Advantages
- *Approximate* information about underlying distribution ✓
- *Approximate* closed-form expression for objective function ✓
- *Approximate* deterministic reformulation of problem ✓
- *Smaller* number of scenarios ✓

Note that SAA methods are not usable for robust optimization as in robust optimization we are concerned about the worst case. This cannot be reflected by working only on a sample.

Concerning "Smaller number of scenarios": Of course sample average approximation can also be used in case of a discrete probability distribution with a huge number of scenarios. If an approximation of the solution is all we need, a SAA with smaller sample might be solved instead.

"Positive" Features

- Basically same as for deterministic comb. opt.
- Based on sample average approximations
- Increase sample size to obtain convergence.
- Additional diversification due to randomness of samples

Most metaheuristics for stochastic combinatorial optimisation work as follows: At the beginning of each iteration you draw a sample of the random parameters and create the corresponding SAA. The rest of the iteration this SAA is used to create new solutions, to compare their quality etc. In the next iteration, a new, slightly bigger sample is drawn etc..

Once more this approach is not possible for robust optimization (see remark in the SAA subsection).

Difficulties

- Evaluation of objective function expensive
- Comparison of quality of solutions difficult
- SCO problems generally have a lot of local optima
- Values of local optima can be close

In general you have to evaluate the objective function many times when applying a metaheuristic, in order to compare the quality of found solutions.