Advanced School in Artificial Intelligence

Bertinoro (FC), 17-28 luglio 2023

Making Decisions with AI

Roberto Amadini

Choice vs. Decision

- What is the difference between **choice** and **decision**?

Choice vs. Decision

- What is the difference between **choice** and **decision**?

- Interchangeable notions, used often as *synonyms*
  - Choice ~ **option**, opportunity, "possible branches"
  - Decision ~ **act** of picking one (or more) *choice*(s) according to some criteria
- In natural language, no difference between "making a choice" or "taking a decision"
- From our perspective, the distinction will be more sharp
- Given a range of **(not) possible choices** how to make the **"right" decisions**?
  - https://www.moralmachine.net/

Decision Support Systems

- A **Decision Support System (DSS)** is a framework supporting the decision-making activities of people/organizations
- DSS focuses on **facilitating decision-making** rather than fully automating it
- E.g., **Clinical Decision Support Systems (CDSS)**
  - Many **ethical, security, transparency, legal issues** are involved
  - **AI** and **Machine Learning** in particular, play a big role here, e.g.
    - Predictions of risks
    - Artificial vision
    - Automated diagnosis

ChatGPT vs DSS

- **ChatGPT** is a chatbot launched by OpenAI in November 2022
  - Built on top of OpenAI's **GPT-3 language model** to predict what next word is
  - Tuned with both supervised/reinforcement learning (**175 billion** parameters)
  - Based on Transformer-based **deep neural networks** for NLP
  - Following the release of ChatGPT, OpenAI was valued at **$29 billion**

- It can provide **impressive** performance!
  - https://www.theguardian.com/australia-news/2023/jan/10/universities-to-return-to-pen-and-paper-exams-after-students-caught-using-ai-to-write-essays
- It may suffer from **hallucination** (incorrect/nonsense answers) and **algorithmic bias** (systematic errors "privileging unfair outcomes")

- Is ChatGPT a **DSS**?

ChatGPT vs DSS

ChatGPT, Guido Scorza al Wired Next Fest 2023: "Il Garante ha agito di urgenza? No, eravamo già in ritardo"

"Il Garante ha agito di urgenza su ChatGPT? Intervenire a valle dell'istruttoria ordinaria sarebbe stato ancora più tardivo,...

Moglie lascia il marito per l'amante su consiglio di una ChatGPT: «Mi ha dato la spinta di cui avevo bisogno»

La donna aveva una relazione extraconiugale da sei mesi mentre da cinque anni era sposata

Digital Life ChatGPT risponde ai pazienti con più tatto di un medico

ChatGPT risponde a chi chiede consigli sulla propria salute. E lo fa meglio dei medici. Occhio però: il rischio che dia informazioni sbagliate è dietro l'angolo.

Un caso esemplare è quello di un sindaco australiano che secondo il chatbot era stato condannato per corruzione; tutto falso, tanto che ha ottenuto che ChatGpt non risponda a chi chiede informazione su di lui.Insomma, se OpenAI non riesce a rettificare questi errori – e può essere difficile o impossibile, dato il funzionamento dell'algoritmo – almeno deve permettere a chiunque di essere esclusi dal sistema.

LA STORIA

«ChatGpt? Una droga: così l'intelligenza artificiale mi ha rubato la vita»

Il chatbot di OpenAi può creare dipendenza? Il racconto di Daniele Amadio al Corriere della Sera: "Restavo incollato al computer fino alle sei del mattino, ho abbandonato amici e fidanzate. Poi mi sono imposto uno stop..."

"Molla tua moglie e sposami!": Chat Gpt 'impazzisce' e risponde così ad un utente. Gli esperti: "Ha doppia personalità, sembra un adolescente con tendenze maniaco-depressive"

Wired Italia

ChatGPT, chi sono i lavoratori sottopagati che lo fanno funzionare

I trainer degli algoritmi sono impiegati da aziende terze e pagati cifre bassissime. Ma senza di loro i chatbot non potrebbero funzionare.

DSS, AI and Optimisation

- **...But AI is not just machine learning and robots!**

- Starting from the '90s, researchers started to apply **AI to optimisation** for having more **flexible** and **intelligible** techniques w.r.t. traditional **mathematical** approaches
  - Before that, mainly mathematical tools and methods **(Operations Research)**: dynamic programming, (non-)linear programming, …

- AI-based optimisation can improve the **decision support** in many areas
  - Planning factory assembly
  - Resource/task allocation
  - Roster scheduling
  - Budgeting
  - ….

Why not sub-symbolic?

- Why not using **sub-symbolic AI?**
  - It requires powerful **hardware**
  - It requires a lot of **data**
  - It does not guarantee a **sound** answer
  - It cannot **explain** an answer
  - It is often "**black-box**": we don't write a model, we prepare data and maybe tune parameters of a certain ML approach

Why ~~not sub-~~symbolic?

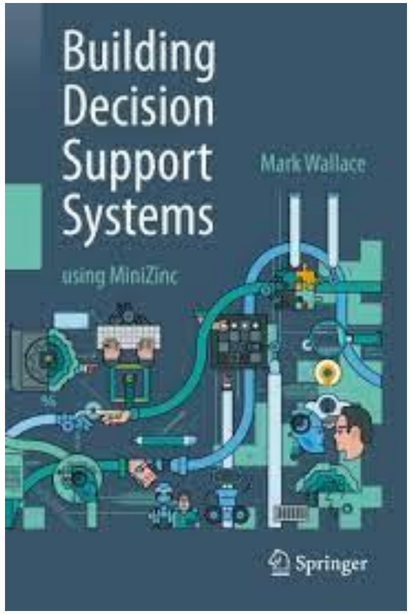
- Why ~~not~~ using ~~sub-~~**symbolic AI?**
  - It ~~requires~~ doesn't require powerful **hardware**
  - It ~~requires~~ doesn’t' require a lot of **data**
  - It does ~~not~~ guarantee a **sound** answer – possibly **optimal**
  - It can~~not~~ **explain** an answer (including failures)
  - It is often ~~"**black-box**"~~ "**gray-box**": we ~~don't~~ write a model, although we don't see how it is solved

DSS and Intelligence

- What makes a DSS for optimisation **intelligent**?

- What does it mean intelligent? From Wikipedia:
  “Intelligence has been defined in many ways: the capacity for logic, understanding, self-awareness, learning, emotional knowledge, reasoning, planning, creativity, critical thinking, and problem-solving […] ”

- We borrow the definition of **Intelligent DSS (IDS)** from the book
  Building Decision Support Systems: Using MiniZinc by M. Wallace (2020)



Intelligent DSS

- It defines a DSS **intelligent** if:

  - It can **do tasks** which would encourage us to call "intelligent" a **person** capable of doing such tasks
    - In this context, intelligence does not refer to systems behaving like humans, have consciousness or are able to think

  - It accomplishes tasks **complicated** enough that the present-day observer cannot see how they are done

  - It does things that are **currently clever** IT
    - E.g., in the 70s an electronic calculator was considered a clever gadget, while today is perceived as a very simple tool – probably a *useless* tool...

What decisions?

- IDS can help organisations to take better decisions
- The decisions made *by* and *for* an organisation have different types and **impact**
  - **Strategic** decisions
    - Long term (≥ 5 years), critical: **human expertise** needed
  - **Tactical** decisions
    - Mid-term (weeks/months), the **core application** of IDS
  - **Operational** decisions
    - Short term (up to few days), no complex IDS needed

Example: hospitality

- Strategic decisions
  - Opening / Moving / Closing a restaurant
  - Changing target customers
- Tactical decisions
  - **Rostering**
  - **Ordering supplies**
- Operational decisions
  - Replacing last minute sick employee
  - Food preparation/serving

Combination of choices

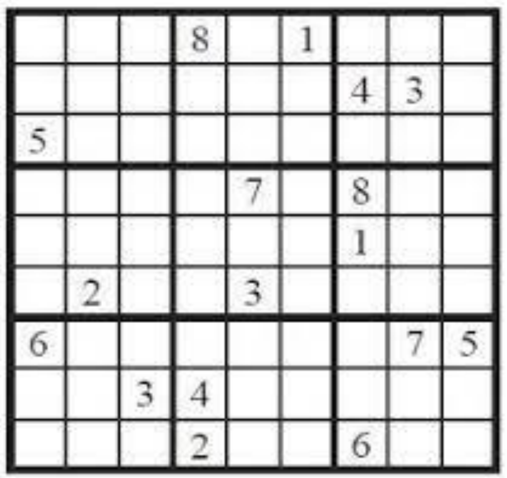
- An IDS enhances decision-making by enabling the **"best" choices** according to given **goal(s)**
    - **"Acting rationally"**

- IDS models and handles **combination of choices** to take "optimal" decisions

- The overall number of **possible combinations** can be **huge** even if the number of *individual* decisions and choices is limited!

Combination of choices

- E.g., **Sudoku** puzzle
  - **≤ 81 decisions** to take (the numbers to be filled in each cell)
  - **≤ 9 choices** for each decision (each cell is filled with a digit in 1..9)



- The number of **feasible combinations** for Sudoku is however much greater than 81×9 = 729. In fact, it is **6670903752021072936960 ≈ 6.671 × 10^{21}**
  - http://www.afjarvis.staff.shef.ac.uk/sudoku/sudoku.pdf

- Even when the no. of decisions and choices is relatively small, the no. of **combination of choices** can be huge!
  - Often, we have **constraints** restricting the no. of feasible combinations
  - Often, we also aim to find the **best feasible** combination!

Combination of choices

- How to solve Sudoku with **sub-symbolic AI**?

Combination of choices

- How to solve Sudoku with **sub-symbolic AI**? Let's ask ChatGPT

1. Data preparation: Prepare a dataset of Sudoku puzzles and their solutions for training and evaluation purposes.
2. Model training: Train a sub-symbolic AI model, such as a neural network or a genetic algorithm, to learn a policy or a heuristic that guides the search process. This training can involve techniques like reinforcement learning or evolutionary optimization.
3. Search and inference: Utilize the trained model to guide the search process. Start with an incomplete Sudoku puzzle and use the model to make predictions or generate candidate solutions. Apply search algorithms like backtracking or local search to explore the solution space based on the model's guidance.
4. Evaluation and refinement: Evaluate the generated solutions against the puzzle's constraints and scoring metrics. Refine the model and search algorithms based on the performance feedback to improve the solving accuracy.

Combination of choices

- How to solve Sudoku with **symbolic AI**? We shall see it!

```
Compiling sudoku.mzn
Running sudoku.mzn
```

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

```
----------
Finished in 130msec
```

Advanced School in Artificial Intelligence

Bertinoro (FC), 17-28 luglio 2023

Fundamentals of Modeling

Roberto Amadini

Selecting from a set

- **Example**: suppose we have a set **S** of numbers and consider the following problems:

- Are there **N** numbers in **S** with a total **over** a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - **N = 4, K = 50 → ?**

Selecting from a set

- **Example**: suppose we have a set **S** of numbers and consider the following problems:

- Are there **N** numbers in **S** with a total **over** a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → **YES:** {23, 16, 13, 4}, {23, 16, 13, 7}, …

Selecting from a set

- **Example**: suppose we have a set **S** of numbers and consider the following problems:

- Are there **N** numbers in **S** with a total **over** a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → **YES:** {23, 16, 13, 4}, {23, 16, 13, 7}, …
    - **N = 2, K = 50 → ?**

Selecting from a set

- **Example**: suppose we have a set **S** of numbers and consider the following problems:

- Are there **N** numbers in **S** with a total **over** a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → **YES:** {23, 16, 13, 4}, {23, 16, 13, 7}, …
    - N = 2, K = 50 → **NO!**

Selecting from a set

- **Example**: suppose we have a set **S** of numbers and consider the following problems:

- Are there **N** numbers in **S** with a total **over** a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → **YES:** {23, 16, 13, 4}, {23, 16, 13, 7}, …
    - N = 2, K = 50 → **NO!**

- Are there **N** numbers in **S** with a total **equal** to a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - **N = 4, K = 50 → ?**

Selecting from a set

- **Example**: suppose we have a set **S** of numbers and consider the following problems:

- Are there **N** numbers in **S** with a total **over** a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → **YES:** {23, 16, 13, 4}, {23, 16, 13, 7}, …
    - N = 2, K = 50 → **NO!**

- Are there **N** numbers in **S** with a total **equal** to a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → **YES:** {13, 23, 10, 4}

Selecting from a set

- **Example**: suppose we have a set **S** of numbers and consider the following problems:

- Are there **N** numbers in **S** with a total **over** a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → YES: {23, 16, 13, 4}, {23, 16, 13, 7}, …
    - N = 2, K = 50 → **NO!**

- Are there **N** numbers in **S** with a total **equal** to a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → YES: {13, 23, 10, 4}
    - N = 3, K = 50 → **NO!**

Selecting from a set

- **Example**: suppose we have a set **S** of numbers and consider the following problems:

- Are there **N** numbers in **S** with a total **over** a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → **YES:** {23, 16, 13, 4}, {23, 16, 13, 7}, …
    - N = 3, K = 50 → **NO!**

- Are there **N** numbers in **S** with a total **equal** to a number **K**?
  - E.g., if **S = {7, 10, 23, 13, 4, 16}**
    - N = 4, K = 50 → **YES:** {13, 23, 10, 4}
    - N = 3, K = 50 → **NO!**

- These are examples of **decision problems**: the answer is either YES or NO
  - Are they "easy" or "hard" to solve?

Selecting from a set

- **Example**: suppose we have a set **S** of numbers and consider the following problems:

- Are there **N** numbers in **S** with a total **over** a number **K**?
  - **Easy** to solve (*polynomial time* w.r.t. |S|)
  - Not very interesting...

- Are there **N** numbers in **S** with a total **equal** to a number **K**?
  - **Hard** to solve (*exponential time* w.r.t. *|S|*)
  - Variant of **subset-sum**: https://en.wikipedia.org/wiki/Subset_sum_problem

- Solving "efficiently" the subset-sum problem would imply **P = NP**
  - *And so, what?*

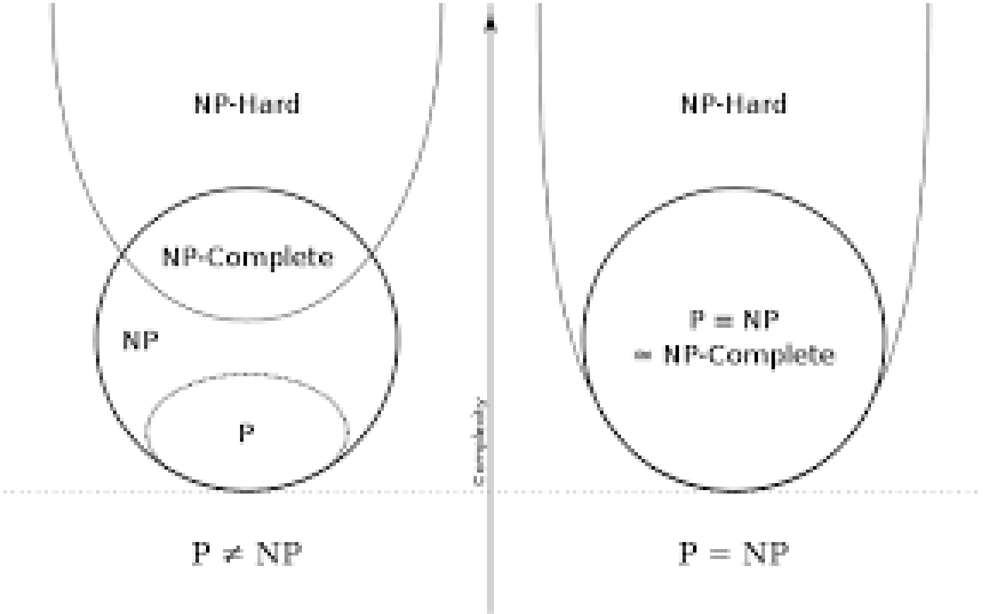
P vs NP dilemma

- **P:** class of problems that can be **solved** in polynomial time (~reasonable time)
- **NP**: class of all problems which solution can be **verified** in polynomial time
  - E.g., verify that {13, 23, 10, 4} has 4 elements and its sum is 50
- A problem in P is always in NP: **P ⊆ NP**
  - If we can find the solution of a problem in "reasonable" time, verifying its soundness cannot be slower…
- But what if **NP⊆P** ? Given a generic NP problem, which solution is verifiable in "reasonable" time, does it exist an algorithm **solving** it in "reasonable" time?
  - If so, **P = NP** (otherwise, P ≠NP)

P vs NP dilemma

- The P-NP dilemma is a **major open problem** in Computer Science
  - **$1.000.000 prize** for solving it

- Finding that P = NP could be **revolutionary**
  - Cryptosystems break down
    - HTTPS no more secure
  - Optimization becomes "easy"
    - And ML as well
  - …
  - The whole world could change!



P vs NP dilemma

- If a problem is “at least as hard as any other problem in NP” is called **NP-hard**
  - If it is also in NP, it is called NP-complete
- Why do we care? Because **several problems** faced by IDS are (NP-)hard!
  - Optimizing tasks/resource allocation, scheduling, planning, …
- A salesperson claiming that his DSS can **optimally solve** medium to large industrial problems (packing, delivering, scheduling, …) is either a charlatan or he doesn’t know what he’s selling!
  - We can be lucky on some *particular* instances
  - But at present no one could find a way to efficiently solve them in general
- The subset-sum problem seen above is NP-hard in general

P vs NP vs ChatGPT

RO

Can you solve a NP-hard problem?

As an AI language model, I can help you understand and explain NP-hard problems, but I cannot guarantee an efficient solution for all instances of NP-hard problems. NP-hard problems are a class of computational problems that are believed to require a significant amount of computational resources to solve optimally.

NP-hard stands for "nondeterministic polynomial-time hard." These problems are characterized by the property that if there exists an efficient algorithm to solve one of them, then there exists an efficient algorithm for solving all problems in the complexity class NP (nondeterministic polynomial-time). However, no efficient algorithm has been discovered yet for solving all instances of NP-hard problems.

Some well-known NP-hard problems include the traveling salesman problem, the knapsack problem, and the Boolean satisfiability problem. These problems have practical applications in various fields, such as logistics, optimization, and cryptography.

P vs NP vs ChatGPT

RO Can deep learning solve NP-hard problems?

Deep learning, as a subfield of machine learning, is primarily used for pattern recognition, feature extraction, and prediction tasks. While deep learning has achieved remarkable success in various domains such as image and speech recognition, it is not inherently designed to solve NP-hard problems efficiently.

NP-hard problems typically involve combinatorial or optimization aspects that require exploring a large search space. Deep learning models, such as neural networks, are not directly suited for solving such problems because they are not explicitly designed to handle discrete decision-making or combinatorial optimization.

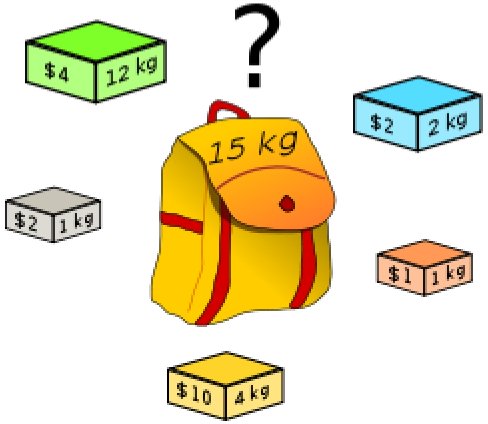
However, deep learning can still play a role in solving certain aspects of NP-hard problems. For example, it can be used for feature extraction or representation learning, which can then be fed into other algorithms or optimization techniques specifically designed for NP-hard problems.

Additionally, deep reinforcement learning, a combination of deep learning and reinforcement learning, has shown promise in tackling some combinatorial optimization problems. By using neural networks to approximate value functions or policies, deep reinforcement learning can guide the search or decision-making process in solving certain NP-hard problems.

In summary, while deep learning alone is not typically used as a direct solution method for NP-hard problems, it can be employed in conjunction with other techniques or algorithms to address specific aspects of these challenging problems.

Knapsack

- **Knapsack problem**: given a set of **items**, each with a **weight** and a **value**, determine the **number of each item** to put in a knapsack so that the **total weight** does not exceed a given **limit** and the total value is **maximal**
    - **Optimization** problem: we maximize the total value
    - **Generalization** of subset-sum problem



- Many **variants**: decision version, 0-1 knapsack, size/volume constraints, …

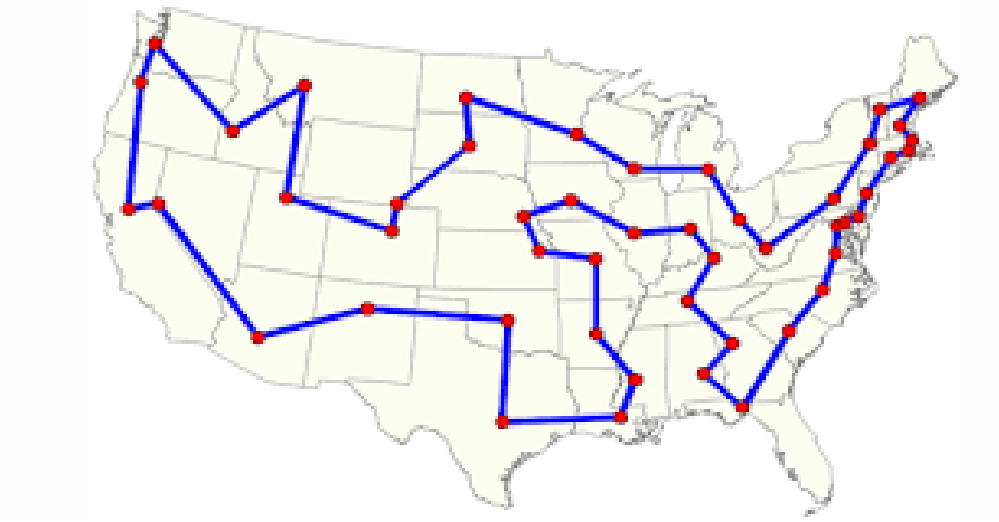
- In general knapsack is **(NP-)hard**
    - Some variants are easy, e.g., the **fractional knapsack** where we can break down items

Knapsack

- Ok, knapsack and subset-sum are hard to solve, and so what?
- Thinking to the above problems as some theoretical, useless stuff is **absolutely wrong**!
- The knapsack problem is **fundamental** for several logistics applications
  - loading ships, trucks, planes, space shuttles, …
- The key is to **model** a real-life problem into a “toy problem” which is both:
  - **simple** enough to be encoded and solved “efficiently”
  - **expressive** enough to properly approximate the original problem
  - possibly similar to other **well-studied** problems (e.g., knapsack, TSP, …)

Travelling Salesman Problem

- **Traveling Salesman problem**: given **N** cities and their distance, what is the **minimal-distance tour** visiting each city exactly once?
- TSP is a fundamental **routing** problem, with several variants and applications in **planning** and **logistics**
- As the knapsack problem, is **(NP-)hard** to solve
- But we can **model** it quite easily
  - Different models are possible!



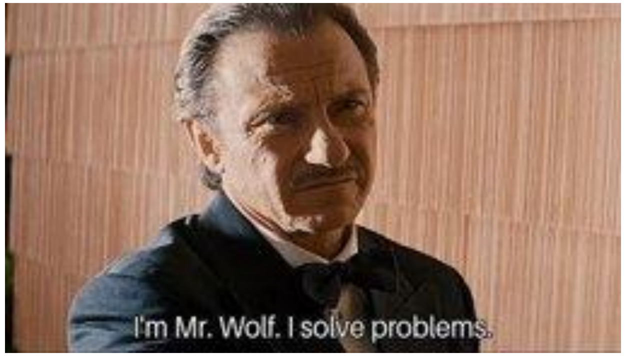
Modelling and solving

- So, how **AI** can be used to take **optimal decisions**?

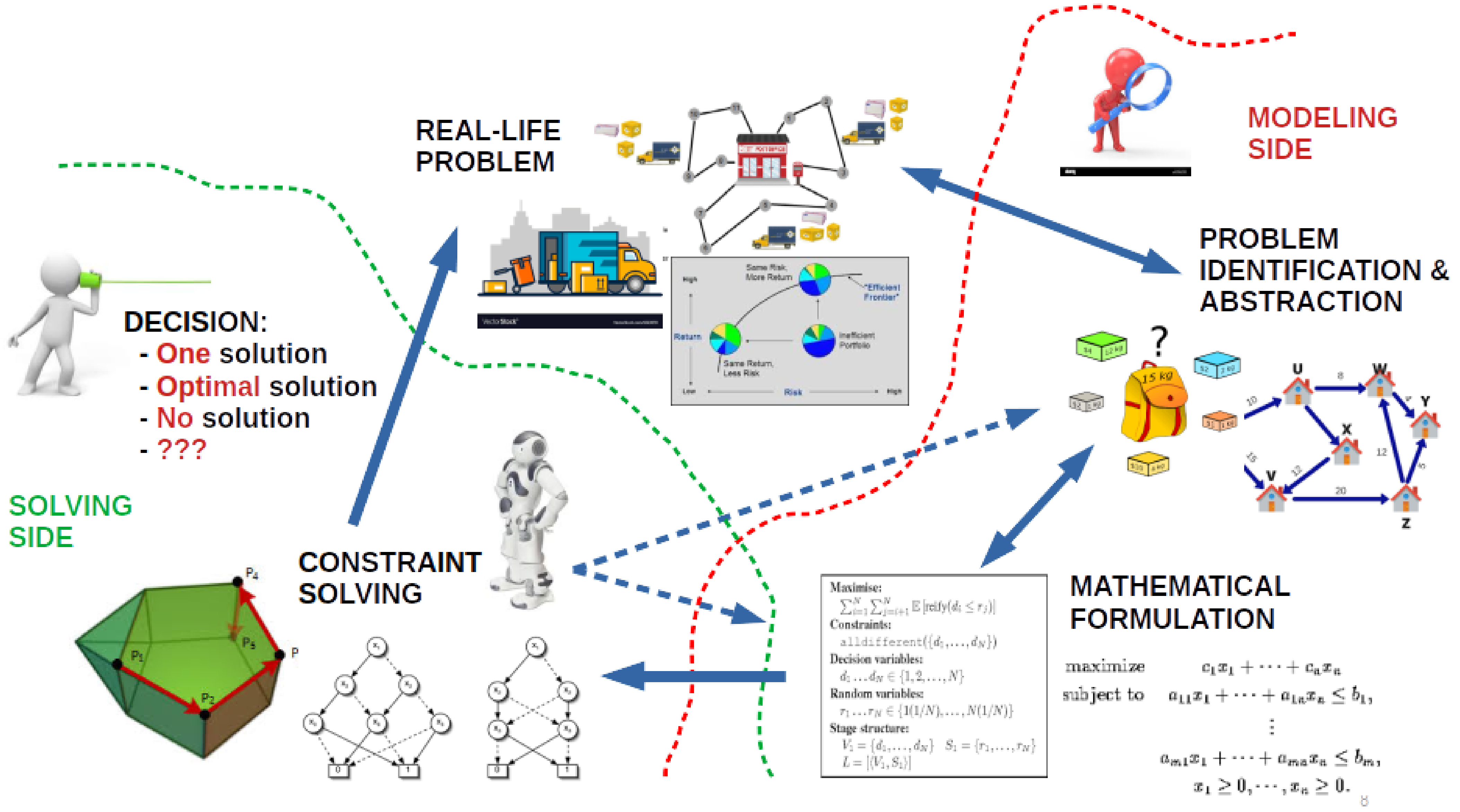
- The first step is **identifying** the **problem components** and their **interaction**
  - What should we decide and what we cannot? Which options do we have? What are the requirements? What is the goal? How to rank different solutions?

- This leads to a **model** of the problem: how to properly **abstract** these components?
  - What are the (not) relevant features of each component?

- Finally, a *"special agent"* (a.k.a. the **solver**) solves the problem for us
  - We specify **what** to solve rather than **how** to do it



Modelling and solving



Modeling

- To **model** combinatorial problems like subset-sum and knapsack we need to **identify**:

- **Variables:** model the possible **decisions**
  - A **value** assigned to a variable represents a **choice made** for that decision

- **Domains**: model the possible **choices** for a decision
  - Often a **finite set** of choices

- **Constraints**: define the **requirements** ruling out **incompatible choices**
  - Domains are a particular case of constraints defining the possible choices for a decision

- **Objective [optional]**: defines the **goal**, i.e., what we need to **maximize** or **minimize**

- **Parameters:** the input values **specifying** an instance of the model

Example: Sudoku

- E.g., a model for the **Sudoku** problem

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

- **Variables:** model the possible **decisions**
  - A **value** assigned to a variable represents a **choice made**
- **Domains:** model the possible **choices** for a decision
  - Often a **finite set** of choices
- **Constraints**: define the **requirements** ruling out **incompatible choices**
  - Domains are a particular case of constraints defining the possible choices for a decision
- **Objective [optional]**: something that we need to **maximize** or **minimize**
- **Parameters:** the input values **specifying** an instance of the model

Example: Sudoku

- E.g., a model for the **Sudoku** problem

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

- **Variables:** model the possible **decisions**
  - 9×9 = **81** variables, one for each cell of the table
- **Domains**: model the possible **choices** for a decision
  - For each variable, we can pick a value in **1..9** only
- **Constraints**: define the **requirements** ruling out **incompatible choices**
  - Values on the same row, column, and grid are **all different**
- **Objective [optional]**: nothing to minimize/maximize!
- **Parameters**: the initial values on the grid

Example: Subset-sum

- E.g., a model for **subset-sum** problem: are there **N** numbers in a set **S** adding up to **K**?

- **Variables:** model the possible **decisions**
  - N variables $X_1, X_2, \ldots, X_N$ corresponding to the N numbers

- **Domains:** model the possible **choices** for a decision
  - ?

Example: Subset-sum

- E.g., a model for **subset-sum** problem: are there **N** numbers in a set **S** adding up to **K**?

- **Variables:** model the possible **decisions**
  - N variables $\mathbf{X_1, X_2, \ldots, X_N}$ corresponding to the N numbers

- **Domains:** model the possible **choices** for a decision
  - For each variable, we can pick a value in **S** only

Example: Subset-sum

- E.g., a model for **subset-sum** problem: are there **N** numbers in a set **S** adding up to **K**?
- **Variables:** model the possible **decisions**
  - N variables $\mathbf{X_1, X_2, \ldots, X_N}$ corresponding to the N numbers
- **Domains:** model the possible **choices** for a decision
  - For each variable, we can pick a value in **S** only
- **Constraints:** rule out **incompatible choices**
  - **?**

Example: Subset-sum

- E.g., a model for **subset-sum** problem: are there **N** numbers in a set **S** adding up to **K**?

- **Variables:** model the possible **decisions**
  - N variables $\mathbf{X_1, X_2, \ldots, X_N}$ corresponding to the N numbers

- **Domains:** model the possible **choices** for a decision
  - For each variable, we can pick a value in **S** only

- **Constraints:** rule out **incompatible choices**
  - $X_1 + X_2 + \ldots + X_N = K$ and **all different** values for $X_1, \ldots, X_N$

Example: Subset-sum

- E.g., a model for **subset-sum** problem: are there **N** numbers in a set **S** adding up to **K**?

- **Variables:** model the possible **decisions**
  - N variables $\mathbf{X_1, X_2, \ldots, X_N}$ corresponding to the N numbers

- **Domains:** model the possible **choices** for a decision
  - For each variable, we can pick a value in **S** only

- **Constraints:** rule out **incompatible choices**
  - $\mathbf{X_1 + X_2 + \ldots + X_N = K}$ and **all different** values for $\mathbf{X_1, \ldots, X_N}$

- **Parameters:** the set **S** and numbers **N** and **K**