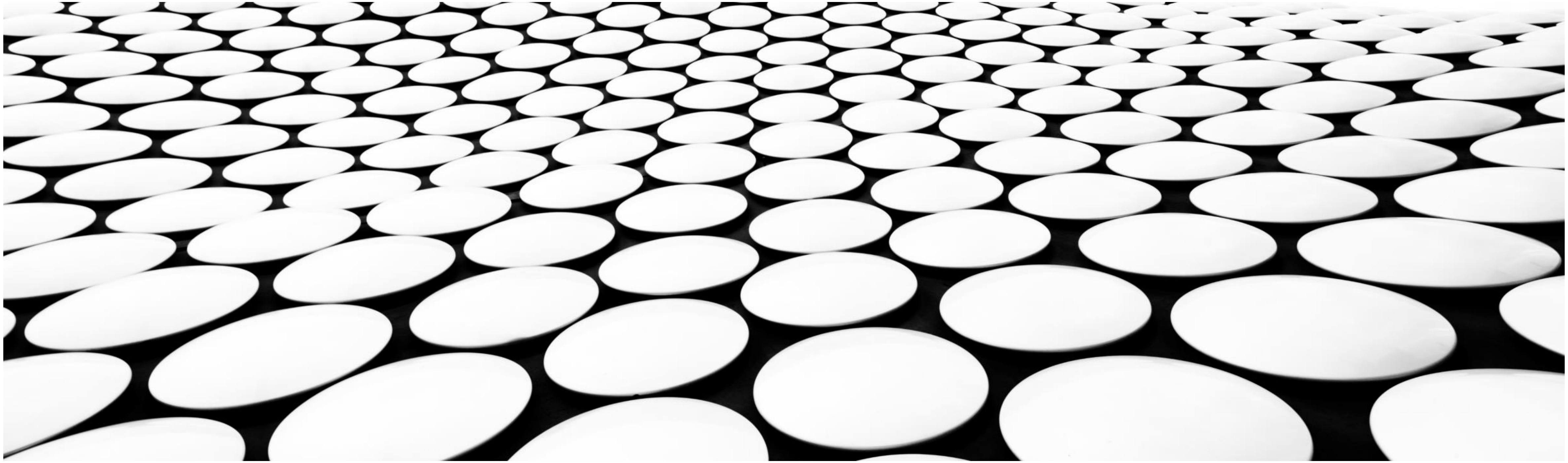
CONSTRAINT-SATISFACTION PROBLEMS

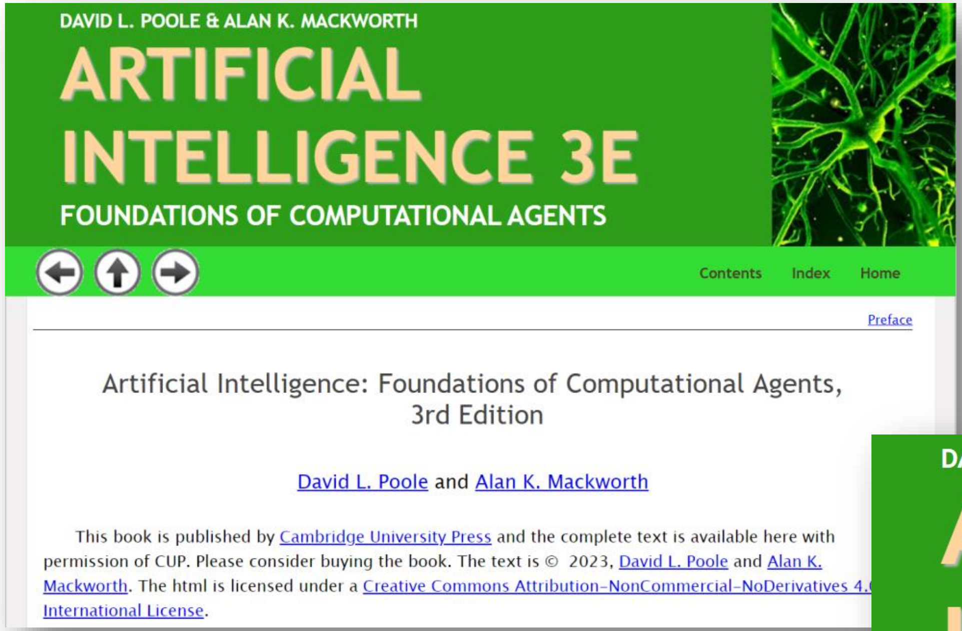
GREEDY AND RANDOMIZED SEARCHES



CONSTRAINT SATISFACTION PROBLEMS

- Part 1 – Introduction
- Part 2 – Greedy Searches
- Part 3 – Randomized Searches

Part 1
Introduction



DAVID L. POOLE & ALAN K. MACKWORTH

ARTIFICIAL INTELLIGENCE 3E

FOUNDATIONS OF COMPUTATIONAL AGENTS

Contents Index Home

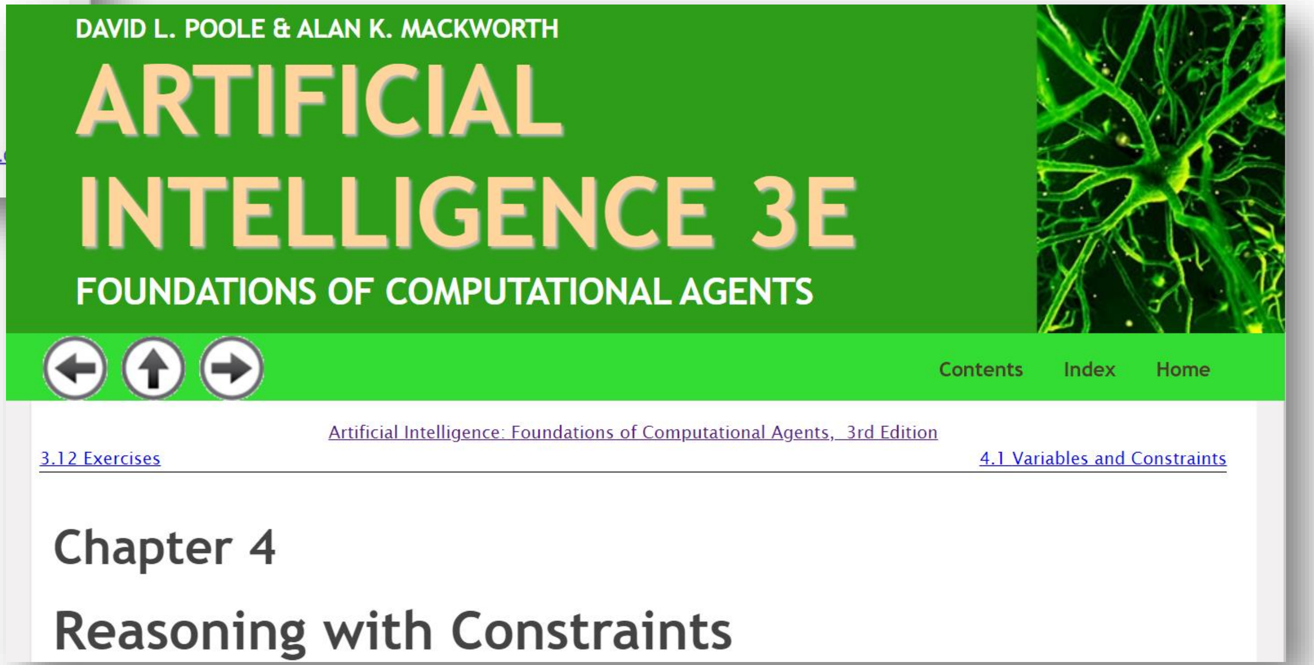
Preface

Artificial Intelligence: Foundations of Computational Agents, 3rd Edition

David L. Poole and Alan K. Mackworth

This book is published by Cambridge University Press and the complete text is available here with permission of CUP. Please consider buying the book. The text is © 2023, David L. Poole and Alan K. Mackworth. The html is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

There is actually a new version! 2023.



DAVID L. POOLE & ALAN K. MACKWORTH

ARTIFICIAL INTELLIGENCE 3E

FOUNDATIONS OF COMPUTATIONAL AGENTS

Contents Index Home

Artificial Intelligence: Foundations of Computational Agents, 3rd Edition

3.12 Exercises 4.1 Variables and Constraints

Chapter 4

Reasoning with Constraints

Source

A CSP is formulated with:

- A set of **variables** V_1, V_2, ... V_n
- Each variable has a **domain** D_{Vi} of possible values
- There are **hard constraints** on various subsets of the variables which specify legal combinations of values for these variables.
- A **solution** to the CSP is an assignment of a value to each variable that satisfies all the constraints.

EXAMPLE OF A CSP

- Variables:

  A, B, C, D, E represent activities

- Domains:
  - $D_A = \{1, 2, 3, 4\}$
  - $D_B = \{1, 2, 3, 4\}$
  - $D_C = \{1, 2, 3, 4\}$
  - $D_D = \{1, 2, 3, 4\}$
  - $D_E = \{1, 2, 3, 4\}$

- Hard constraints:

  $(B \neq 3)$ and $(C \neq 2)$ and $(A \neq B)$ and $(B \neq C)$ and $(C < D)$ and $(A = D)$ and $(E < A)$ and $(E < B)$ and $(E < C)$ and $(E < D)$ and $(B \neq D)$

- Soft constraints:

  B and C should be as small as possible.

EXAMPLE OF A CSP

- Variables:

  A, B, C, D, E represent activities

- Domains:
  - $D_A = \{1, 2, 3, 4\}$
  - $D_B = \{1, 2, 3, 4\}$
  - $D_C = \{1, 2, 3, 4\}$
  - $D_D = \{1, 2, 3, 4\}$
  - $D_E = \{1, 2, 3, 4\}$

- Hard constraints:

  $(B \neq 3)$ and $(C \neq 2)$ and $(A \neq B)$ and $(B \neq C)$ and
  $(C < D)$ and $(A = D)$ and $(E < A)$ and $(E < B)$ and
  $(E < C)$ and $(E < D)$ and $(B \neq D)$

- Soft constraints:

  B and C should be as small as possible.

Solution space (A,B,C,D,E)

(1,1,3,4,1)
(1,1,1,1,1)
(1,1,3,1,1)
(2,3,3,4,1)
(2,1,3,4,1)
(2,1,3,4,4)
(5,1,3,4,1)
(2,5,3,4,3)
(5,1,4,4,2)
(4,1,3,4,2)

Find the solution(s)

Given a set of variables, assign a value to each variable for:

Satisfiability:

Satisfy a set of hard constraints.

Optimization:

Minimize the value of a cost function. There is a cost associated with each assignment of a value to a variable. We are talking about soft constraints.

Constrained optimization problem

Mix of hard and soft constraints. We must satisfy a set of hard constraints, but we also try to optimize a cost function.

EXAMPLES OF PROBLEMS AS CSP

1. Simple scheduling
2. Knapsack
3. Travelling Salesman Problem (TSP)
4. Sudoku

SCHEDULING PROBLEM

Determine the time at which each professor must teach. The classes are all 1 hour long, and there is a single room.

teacher	min	max
Peter	3	6
Jane	3	4
Anne	2	5
Yan	2	4
Dave	3	4
Mary	1	6

Variables?
Domains?
Constraints?

SCHEDULING PROBLEM

teacher	min	max
Peter	3	6
Jane	3	4
Anne	2	5
Yan	2	4
Dave	3	4
Mary	1	6

Variables: P, J, A, Y, D, M

Domains: D_P=3..6, D_j=3..4, D_A=2..5, …

Hard constraints: $P \neq J \neq A \neq Y \neq D \neq M$

SCHEDULING PROBLEM

teacher	min	max
Peter	3	6
Jane	3	4
Anne	2	5
Yan	2	4
Dave	3	4
Mary	1	6

Oh… and also… Jane, Anne and Dave want to teach as early as possible.

Cost function

SCHEDULING PROBLEM

teacher	min	max
Peter	3	6
Jane	3	4
Anne	2	5
Yan	2	4
Dave	3	4
Mary	1	6

Variables: P, J, A, Y, D, M

Domains: D_P=3..6, D_j=3..4, D_A=2..5, …

Hard constraints: $P \neq J \neq A \neq Y \neq D \neq M$

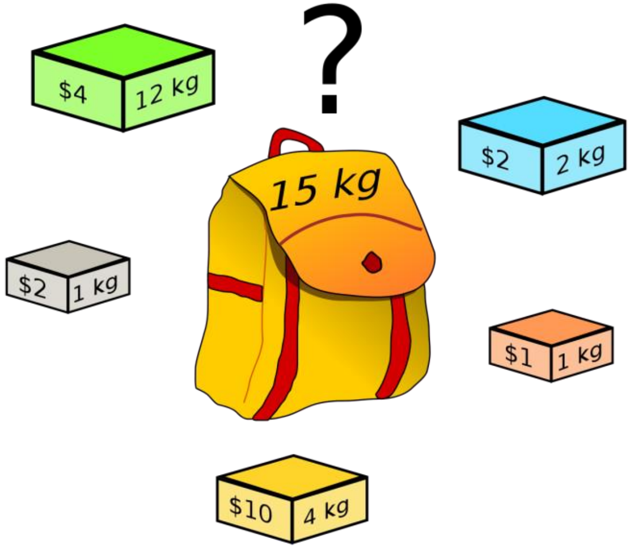
Cost function: F = J + A + D

Goal for optimization: Minimize(F)

KNAPSACK PROBLEM

There is a knapsack of capacity M, and there are N articles of weight S_i, and value/cost C_i.

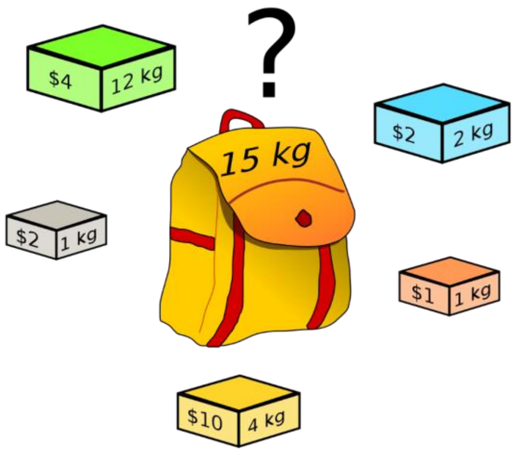
Choose the articles to place in the knapsack to maximize the cost of the knapsack.



Variables?
Domains?
Constraints?

Cost function?

KNAPSACK PROBLEM



Variables: $X_1 \ldots X_n$

Domains: $D(X_i)$ in [0,1] (item included or not)

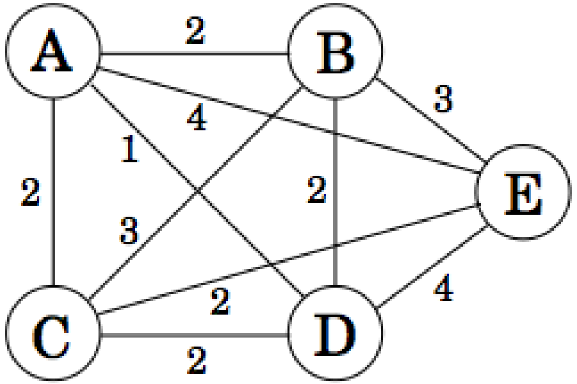
Hard constraint: $S_1 * X_1 + S_2 * X_2 + \ldots + S_n * X_n < M$

Cost function: $F = C_1 * X_1 + C_2 * X_2 + \ldots + C_n * X_n$

Goal for optimization: Maximize(F)

TSP

Try to visit all cities in the minimum amount of time, returning to the starting city.



Variables: V1, V2, V3, V4, V5

Domains: D_{Vi}=A,B,C,D,E

Cost function: F = C(V1,V2) + C(V2,V3) + C(V3, V4) + C(V4, V5) + C(V5,V1)

Goal: Minimize F

Hard constraints could be added if a partial ordering was required, or a specific starting city, etc.

SUDOKU

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

Variables: X_{11}... X_{99} (indices are identifiers of row and column)

Domains: $D(X_{ij})$ in [1,9]

Constraints: $X_{ij} \neq X_{ik}$ pour $j \neq k$

$X_{ij} \neq X_{kj}$ pour $i \neq k$

$X_{ij} \neq X_{kw}$ pour $1 <= i,k <= 3$ && $1 <= j,w <= 3$

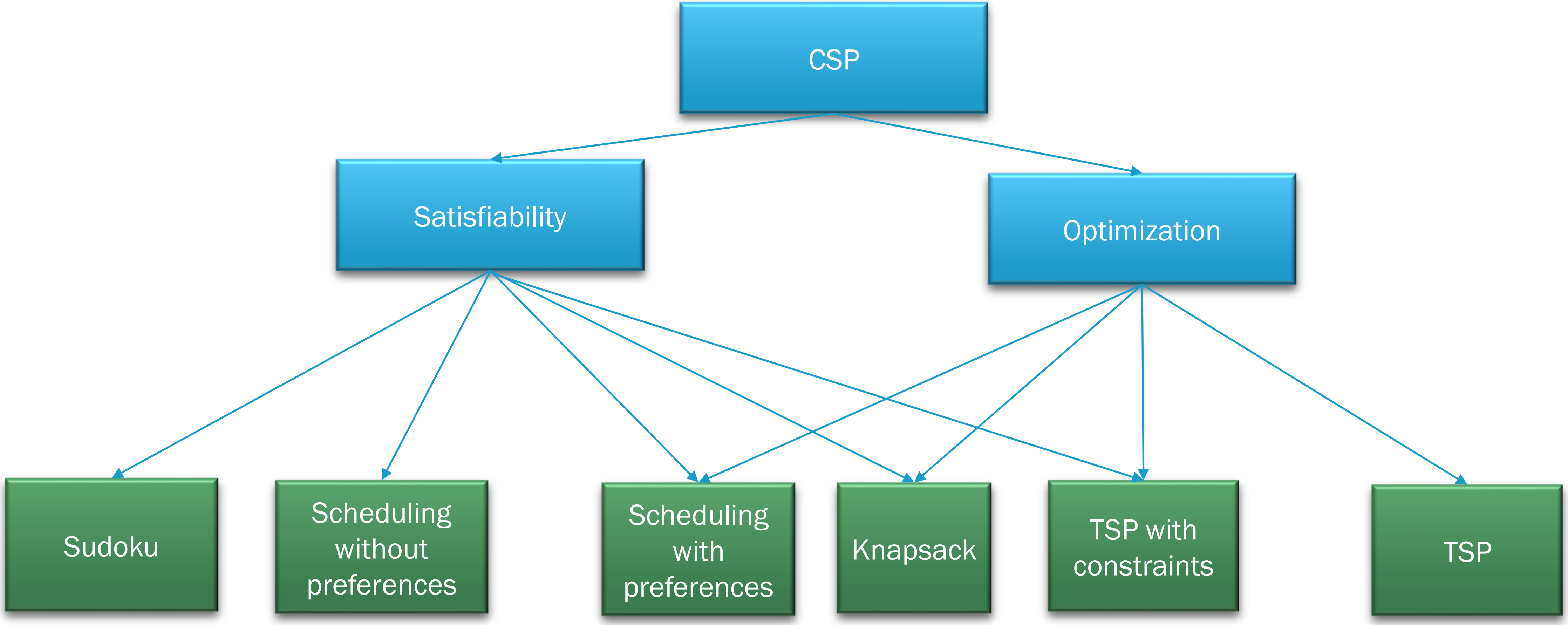
$X_{ij} \neq X_{kw}$ pour $1 <= i,k <= 3$ && $4 <= j,w <= 6$

$X_{ij} \neq X_{kw}$ pour $1 <= i,k <= 3$ && $7 <= j,w <= 9$

....

Cost function: None

Goal : Satisfy all hard constraints


CSP
Satisfiability
Optimization
Sudoku
Scheduling without preferences
Scheduling with preferences
Knapsack
TSP with constraints
TSP

IN SUMMARY

- How to formulate problems as Constraint Satisfaction Problems
- Presentation of four classical problems that can be formulated as CSP:
  - Scheduling
  - Knapsak
  - TSP
  - Sudoku
- Difference between satisfiability and optimisation

Part 2
Greedy Search

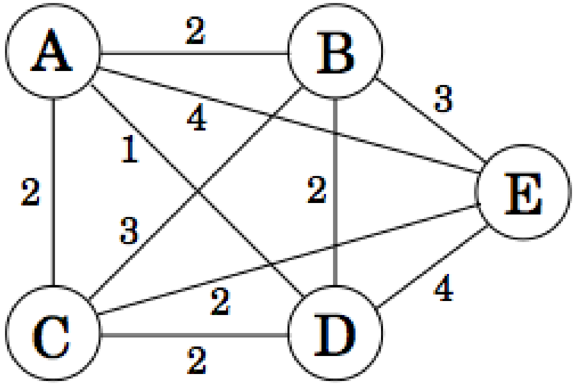
Greedy search follows a STRATEGY to move forward systematically.

The strategy determines the next action to take.

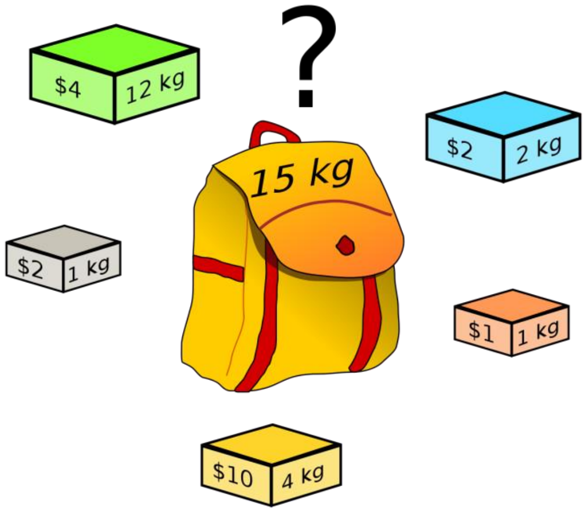
A greedy search is considered a heuristic search (and not a blind search) because domain knowledge can be included in the strategies.

The algorithm will go back (backtrack) when it reaches a dead end.

CLASSIC PROBLEMS



teacher	min	max
Peter	3	6
Jane	3	4
Anne	2	5
Yan	2	4
Dave	3	4
Mary	1	6



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

Source Image Knapsack

GREEDY SEARCH TO SOLVE A SUDOKU

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

Strategy 1:

- Fill in numbers from row 1 to 9.
- Go from left to right for each row.
- Use the smallest number that satisfies the constraints.

GREEDY SEARCH TO SOLVE A SUDOKU

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

Strategy 2:

- Explore columns in order of most constrained to less constrained.
- Fill in the columns from top to bottom.
- Use a number at random that satisfies the constraints.

GREEDY SEARCH FOR PLANNING

teacher	min	max
Peter	3	6
Jane	3	4
Anne	2	5
Yan	2	4
Dave	3	4
Mary	1	6

Greedy strategy #1:

Alphabetical order, smaller non-conflicting value

A	D	J	M	P	Y
2	3	4	1	5	?

GREEDY SEARCH FOR PLANNING

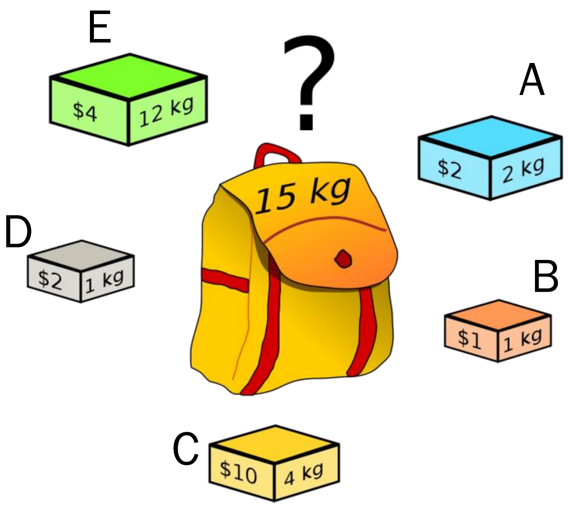
teacher	min	max
Peter	3	6
Jane	3	4
Anne	2	5
Yan	2	4
Dave	3	4
Mary	1	6

Greedy strategy #2:

Most contrained to least constrained domain

J	D	Y	A	P	M
3	4	2	5	6	1

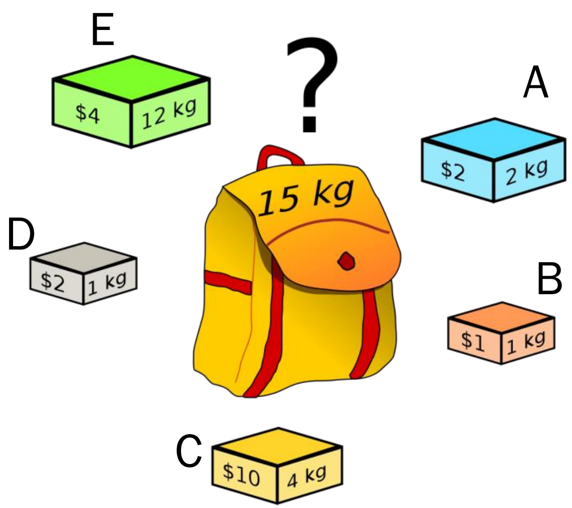
GREEDY SEARCH FOR KNAPSACK



Greedy Strategy #1:
Choose the highest value first. For two equal values, choose the least heavy.

Now we are in front of a constrained optimization problem

C	D	A	B	Total
10$	2$	2$	1$	15$
4kg	1kg	2kg	1kg	8kg

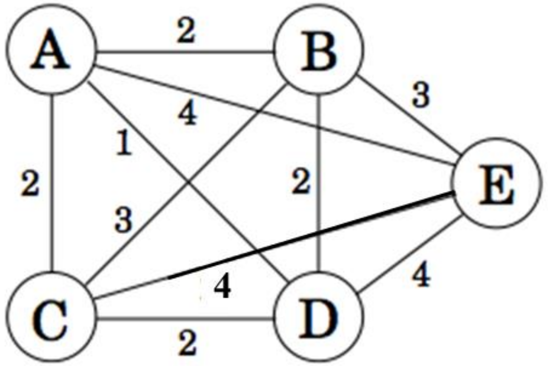


Greedy Strategy #2:

Choose heaviest first. If two equal weights, choose highest value.

E	A	D	Total
4$	2$	2$	8$
12kg	2kg	1kg	15kg

GREEDY SEARCH FOR TSP



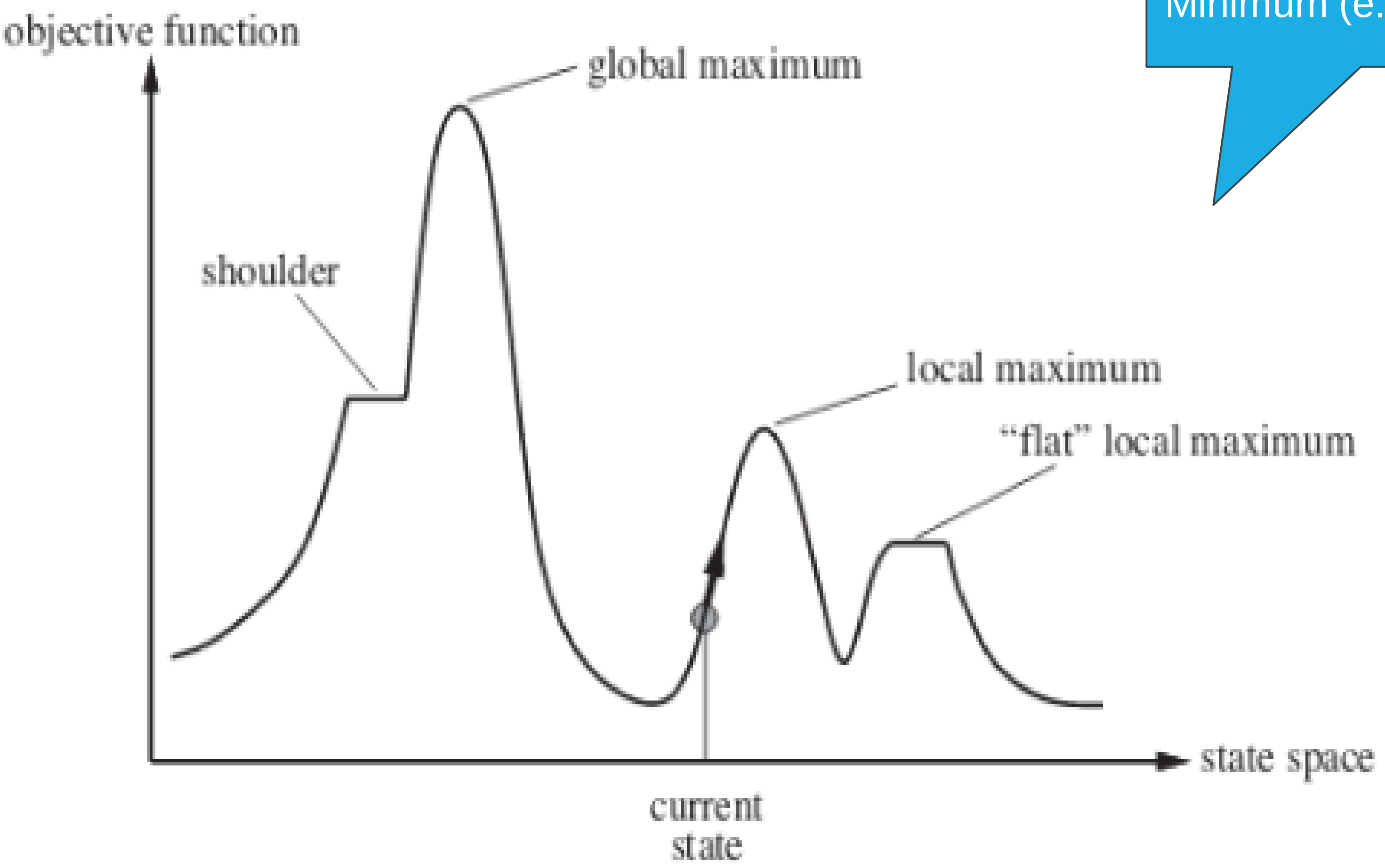
It is a pure optimisation problem (if starting city is not constrained)

Greedy strategy:
Go to the closest city. If 2 equal choices, select in descending alphabetical order (Z..A).

T1	T2	T3	T4	T5	Back	Cost
D	A	C	B	E	D	
	1	2	3	3	4	13

OPTIMIZATION FUNCTION

Maximum (e.g. Knapsack)
Minimum (e.g. TSP)



Source

OPTIMIZATION FUNCTION

Local maximum (or minimum): There exists a better solution that cannot be reached by making local moves according to the cost function.

Global maximum (or minimum): Where the optimal solution is found. Where the cost function is minimized (or maximized).

Plateau: An area of the search space which provides no clue as to where to go since all neighbors seem locally equal in their evaluation of the cost function.

IN SUMMARY

- Greedy search definition
- Examples of using strategies on 4 classic examples
- Limitations of greedy search

Part 3
Randomized Searches

THE ROLE OF RANDOMNESS

RANDOMNESS - At the heart of the idea of exploitation vs exploration.

Exploitation:

- Following a strategy (such as greedy)

Exploration:

- Make "random" choices (at least other than what is guided by the strategy)

RANDOMIZED SEARCHES

Random restart

- We try to create a first complete solution.
- We restart from the beginning with a new starting point.

Random step

- We introduce random steps within the construction of a solution. Sometimes we take a path that is not locally the best.

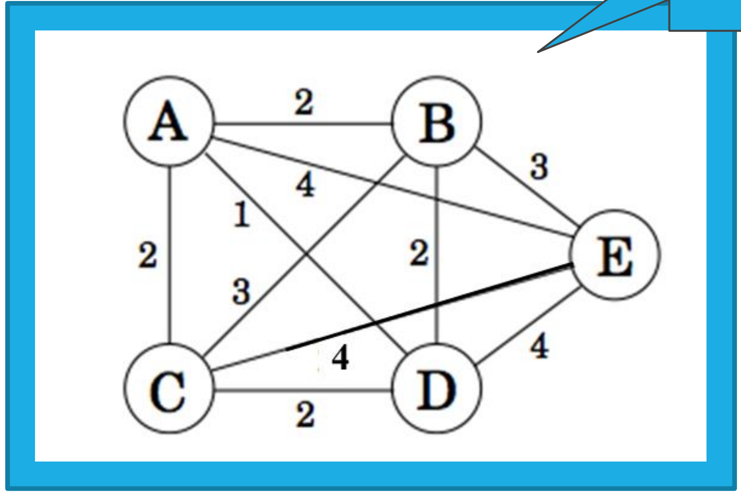
Random modification

- We create a first complete solution.
- We then modify parts of the solution following some random strategy.

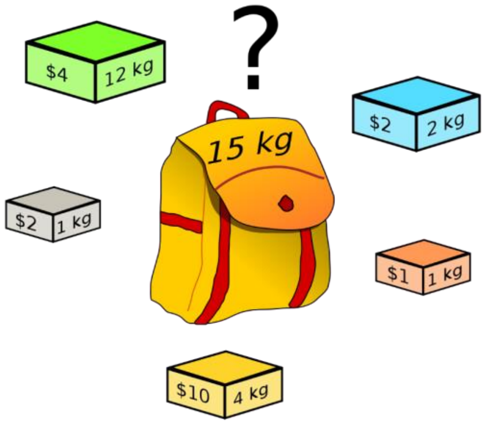
CLASSIC PROBLEMS

teacher	min	max
Peter	3	6
Jane	3	4
Anne	2	5
Yan	2	4
Dave	3	4
Mary	1	6

Focus on this problem.



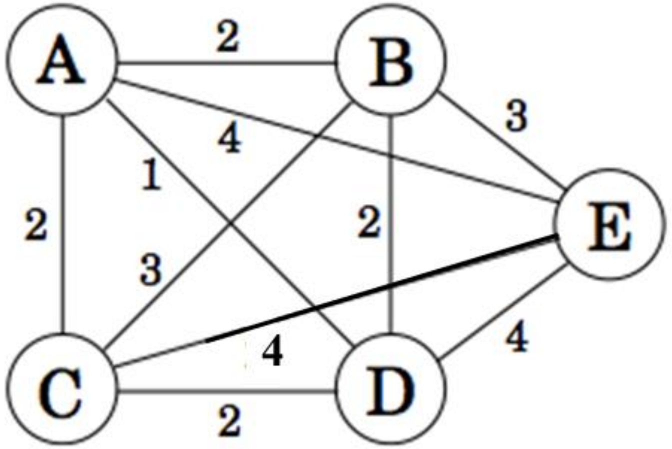
		3		2		6		
9			3		5			1
		1	8		6	4		
		8	1		2	9		
7								8
		6	7		8	2		
		2	6		9	5		
8			2		3			9
		5		1		3		



Source Image Knapsack

GREEDY SEARCH FOR TSP

Greedy strategy:
Go to the closest city. If 2 equal choices, select in descending alphabetical order (Z..A).



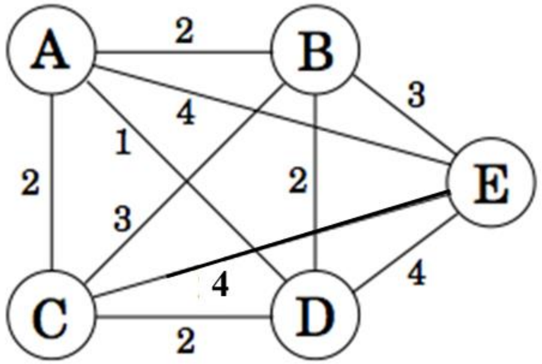
T1	T2	T3	T4	T5	Back	Cost
D	A	C	B	E	D	
	1	2	3	3	4	13

RANDOM RESTART

Algorithm for search with random restart

- We try to create a first complete solution.
- We restart from the beginning with a new starting point.

RANDOM RESTART



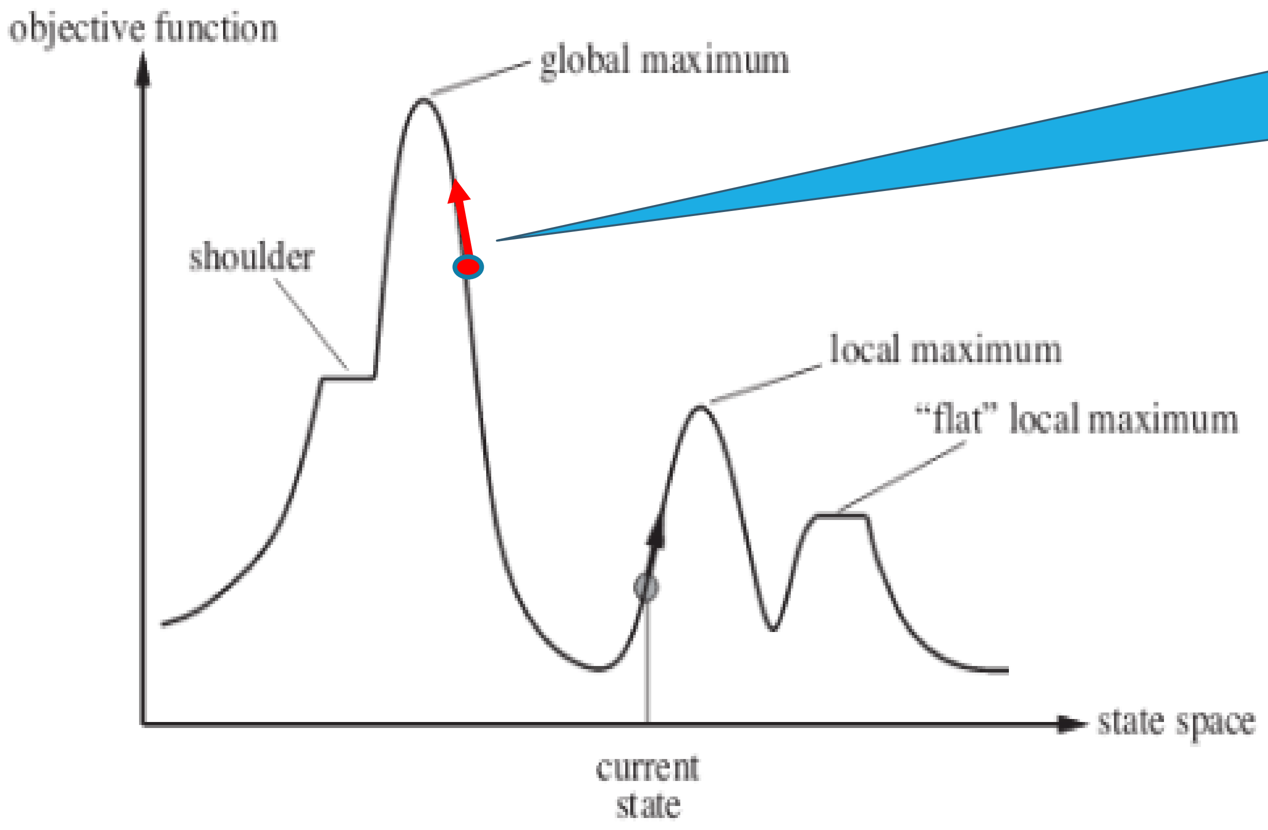
First solution...

T1	T2	T3	T4	T5	back	Cost
D	A	C	B	E	D	
	1	2	3	3	4	13

We start over...

T1	T2	T3	T4	T5	T1	Cost
C	D	A	B	E	C	
	2	1	2	3	4	12

RANDOM RESTART



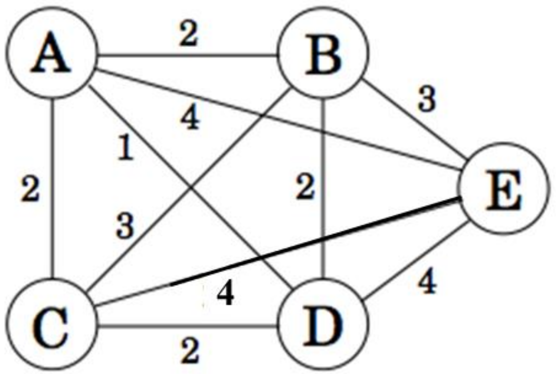
If we restart here, we are in better position to reach the global maximum.

RANDOM STEP

Algorithm for search with random step

- We introduce random steps within the construction of a solution. Sometimes we take a path that is not locally the best.

RANDOM STEP



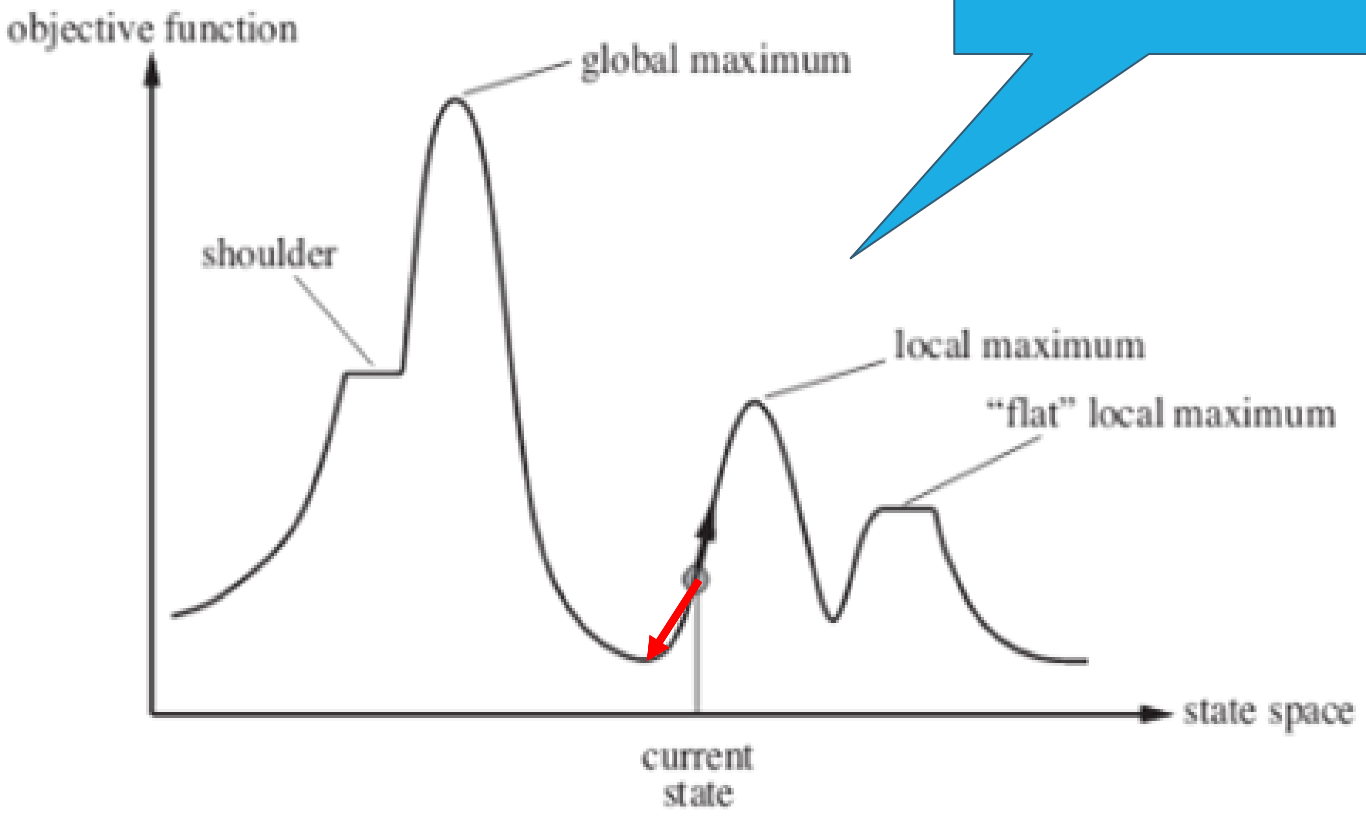
Choose B even if it has a highest cost.

T1	T2	T3	T4	T5	Coût
C	D	B			
	2				

Use probabilities to manage this choice.

RANDOM STEP

Sometimes we take a chance to go in another direction.

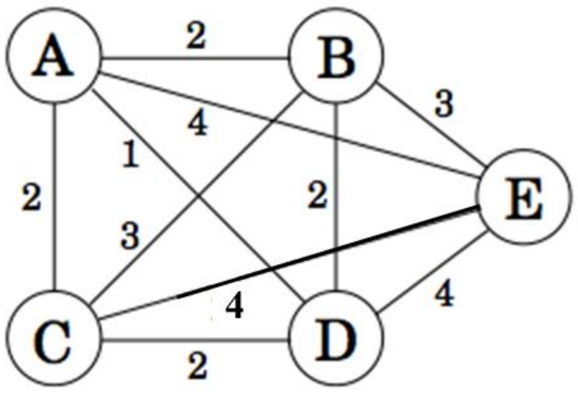


RANDOM MODIFICATION

Algorithm for search with random modification

- We create a first complete solution.
- We then modify parts of the solution following some random strategy.

RANDOM MODIFICATION



First solution…

T1	T2	T3	T4	T5	retour	Coût
D	A	C	B	E	D	
	1	2	3	3	4	13

We exchange cities (modification)…

T1	T2	T3	T4	T5	retour	Coût
D	A	C	E	B	D	
	1	2	4	3	2	12

RANDOM MODIFICATION

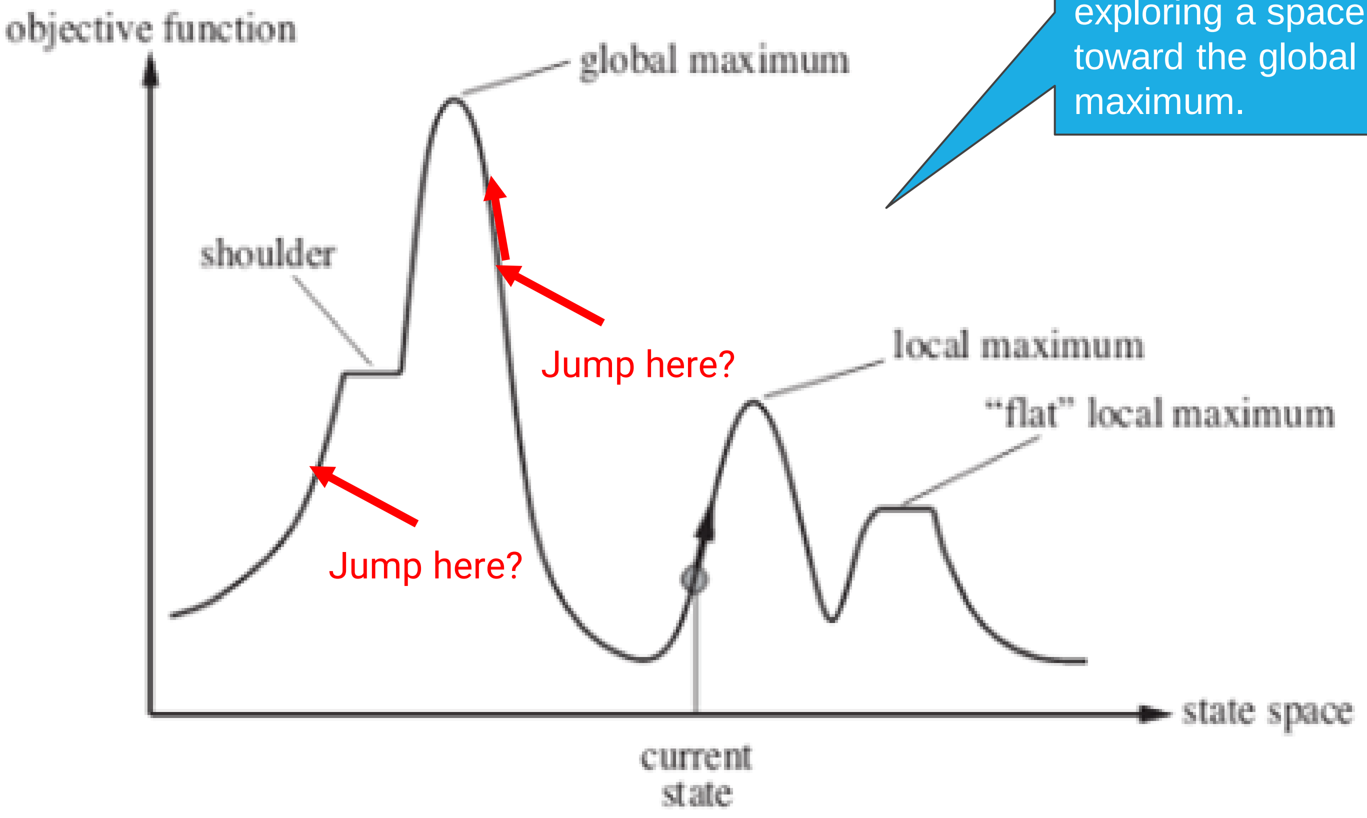
Generic algorithm:

- Start with an initial (random / greedy) solution
- Repeat for N iterations:
  - Make local changes
  - If better (according to cost function):
    - Keep change
  - Else
    - Keep change (according to some probability)

Variations exists on how to calculate/adjust such probability

The algorithm also keeps track of the best solution so far

RANDOM MODIFICATION



Jumping in different places sometimes might lead to exploring a space toward the global maximum.

RANDOMIZED SEARCHES

Random restart

- We try to create a first complete solution.
- We restart from the beginning with a new starting point.

Random step

- We introduce random steps within the construction of a solution. Sometimes we take a path that is not locally the best.

Random modification

- We create a first complete solution.
- We then modify parts of the solution following some random strategy.

EVALUATION OF RANDOMIZED SEARCHES

How to evaluate?

If we introduce random in the algorithm, we must repeat it many times to measure an average of its performance.

The performance could be in terms of runtime and/or quality of the obtained solution(s).

IN SUMMARY

- Randomized searches
- Algorithms:
  - Random restart
  - Random step
  - Random modification

CONSTRAINT SATISFACTION PROBLEMS

- Part 1 – Introduction
- Part 2 – Greedy Searches
- Part 3 – Randomized Searches