Faculty of Computer
and Information Sciences
Ain Shams University

كلية الحاسبات و المعلومات
جامعة عين شمس

Data Science

Code:

Instructor

Prof.Dr. Abeer M. Mahmoud

Professor of Computer Science
-faculty of Computer and Information Sciences- Ain Shams University

Data Science and Big Data Analytics v2

Topics : Data Science and Big Data Analytics Course

Introduction to Big Data Analytics + Data Analytics Lifecycle	Review of Basic Data Analytic Methods Using R	Advanced Analytics – Theory and Methods	Advanced Analytics - Technology and Tools	The Endgame, or Putting it All Together + Final Lab on Big Data Analytics
Big Data Overview State of the Practice in Analytics The Data Scientist Big Data Analytics in Industry Verticals Data Analytics Lifecycle	Using R to Look at Data - Introduction to R Analyzing and Exploring the Data Statistics for Model Building and Evaluation	K-means Clustering Association Rules Linear Regression Logistic Regression Naive Bayesian Classifier Decision Trees Time Series Analysis Text Analysis	Analytics for Unstructured Data (MapReduce and Hadoop) The Hadoop Ecosystem In-database Analytics – SQL Essentials Advanced SQL and MADlib for In-database Analytics	Operationalizing an Analytics Project Creating the Final Deliverables Data Visualization Techniques + Final Lab – Application of the Data Analytics Lifecycle to a Big Data Analytics Challenge

Lesson: Time series analysis

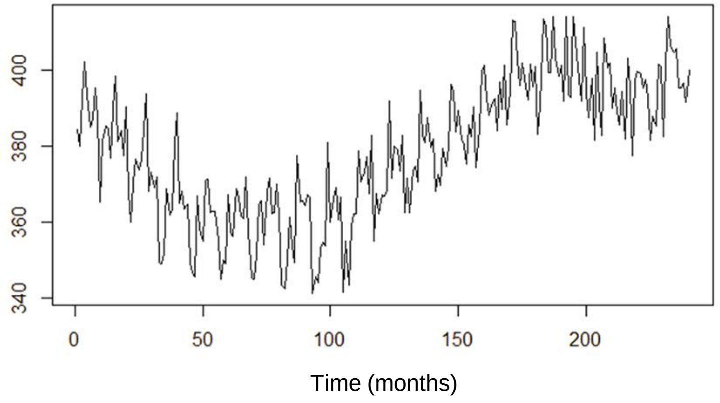
Time Series Analysis

During this lesson, the following topics are covered:

- Time Series Analysis and its applications in forecasting
- Autoregressive Moving Averages (ARMA) and ARIMA Models
- Implementing the Box-Jenkins Methodology using R
- Reasons to choose (+) and cautions (-) with Time Series Analysis

Time Series Analysis, cont.

- Time Series: Ordered sequence of equally spaced values over time
- Analysis Goals
  - To identify the internal structure of the time series
  - To forecast future events
    - Example: Based on sales history, what will next December's sales be?

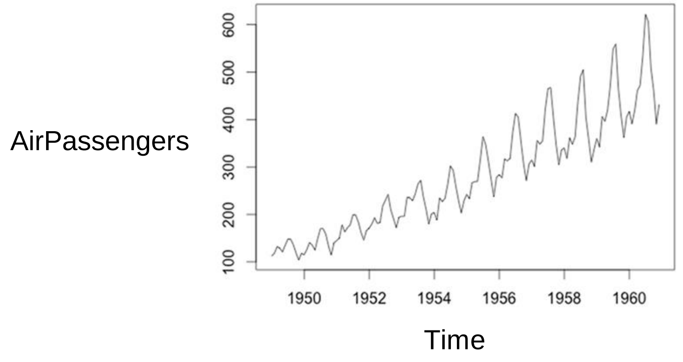


Components of Time Series Analysis

- A time series may involve the following components:
  - Trend
  - Seasonality
  - Cycles
  - Random
- **Method: Box-Jenkins—ARMA**

Box-Jenkins method—what is it?

Models historical behavior to forecast the future



Applies ARMA

- Accounting for the Trends and Seasonality components
- Input: Time Series
- Output: Expected future value of the time series

AR and MA models

- Autoregressive process
  - AR(p): Current value of the series depends on it own p previous values.
  - p is the order of the AR process.
- Moving Average process
  - MA(q): Current deviation from mean depends on q previous deviations.
  - q is the order of the MA process.

Time series—use cases

Forecast:

- Employment Trend
- Next month's sales
- Tomorrow's stock price
- Hourly power demand

Modeling time series

Model the time series as

$$Y_t = T_t + S_t + R_t, \qquad t=1,\dots,n.$$

T_t Trend term

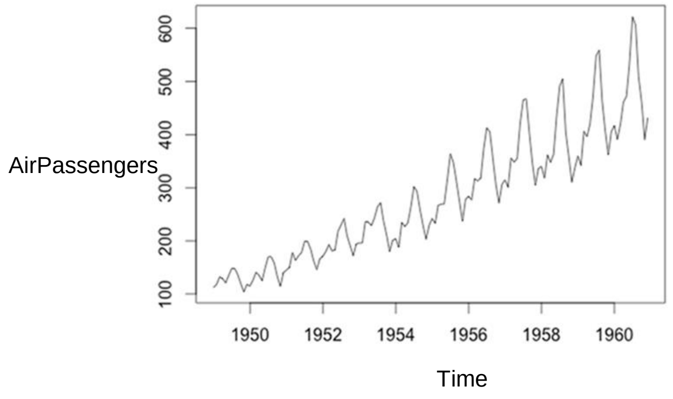
Air travel steadily increased over the last few years

S_t The seasonal term

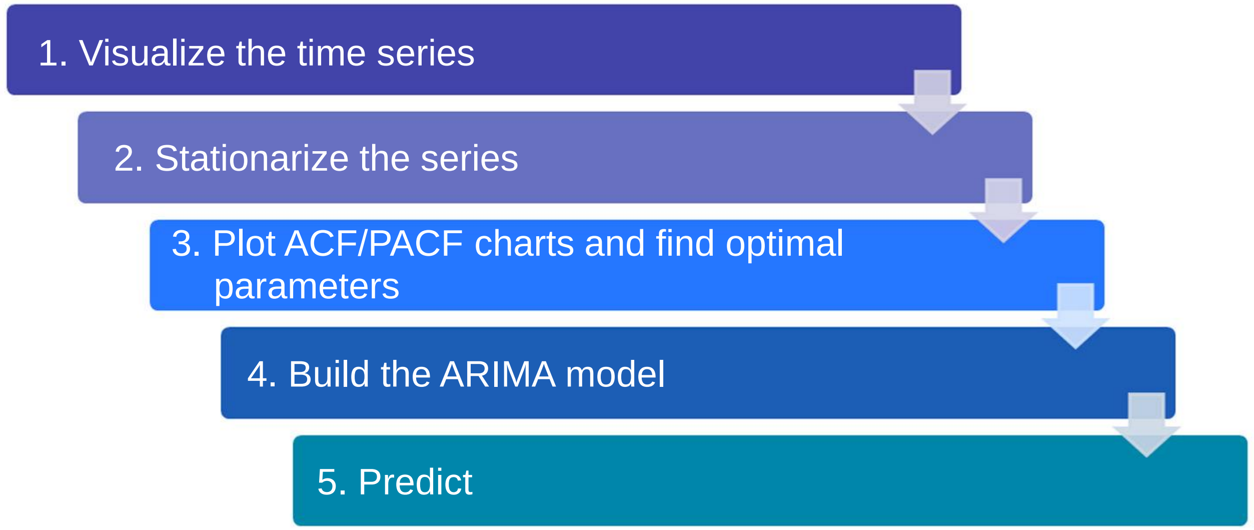
Air travel fluctuates in a regular pattern over the course of a year

R_t Random component

To be modeled with ARMA

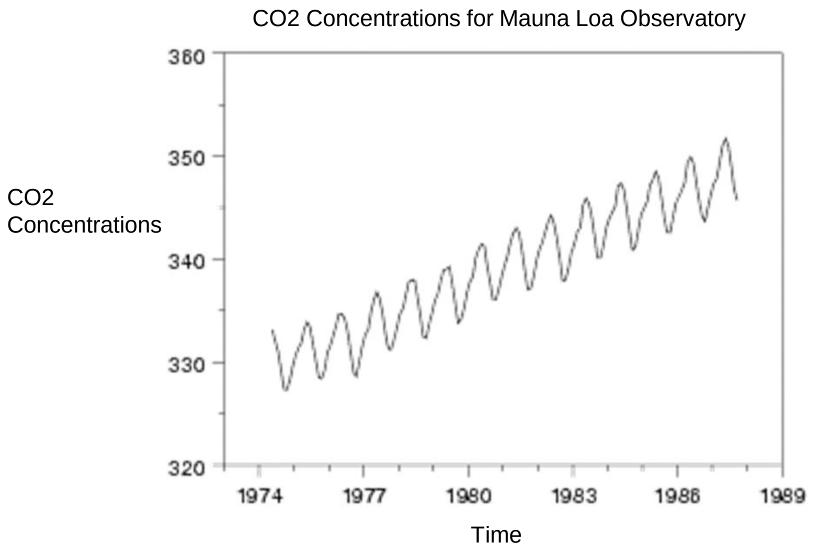


Framework for ARIMA Time Series Modeling


1. Visualize the time series
2. Stationarize the series
3. Plot ACF/PACF charts and find optimal parameters
4. Build the ARIMA model
5. Predict

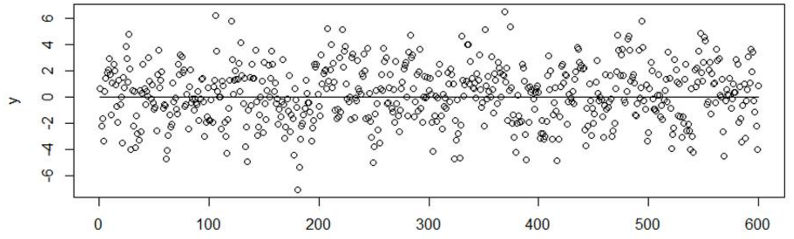
Step 1—visualizing time series

- Analyze the trend before building any time series model.
- Details of interest pertain to any patterns.
  - Trend
  - Seasonality
  - Random behavior
- This graph is an upward trending time series with seasonality every year.

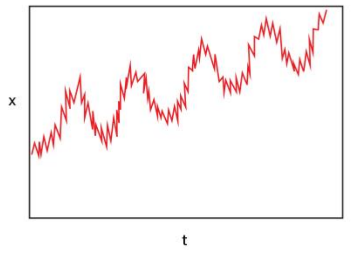


Step 2—stationarize series

- Box-Jenkins methodology assumes that the random component is a stationary sequence.
  - Constant mean
  - Constant variance
  - Autocorrelation does not change over time.
    - Constant correlation of a variable with itself at different times
- In practice, to obtain a stationary sequence, the data must be:
  - Detrended.
  - Seasonally adjusted.



Stationary Series



Nonstationary Series

Step 2—stationarize series—detrending

In this example, there is a linear trend, so fit a linear model:

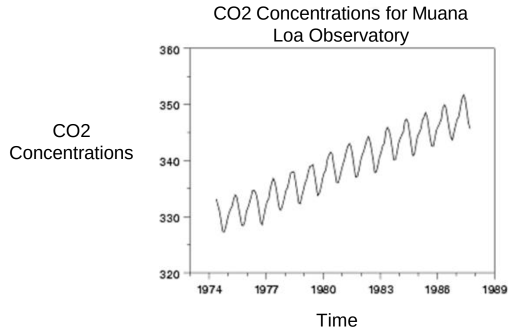
$T_t = m \cdot t + b$

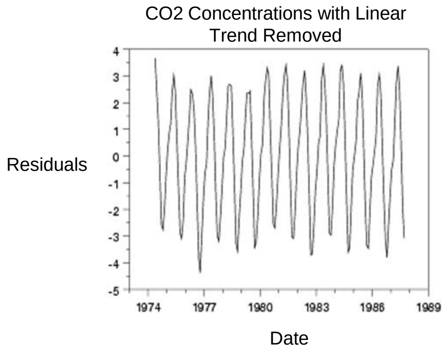
The detrended series is then:

$Y^1_t = Y_t - T_t$

Sometimes, you may have to fit a nonlinear model:

- Quadratic
- Exponential

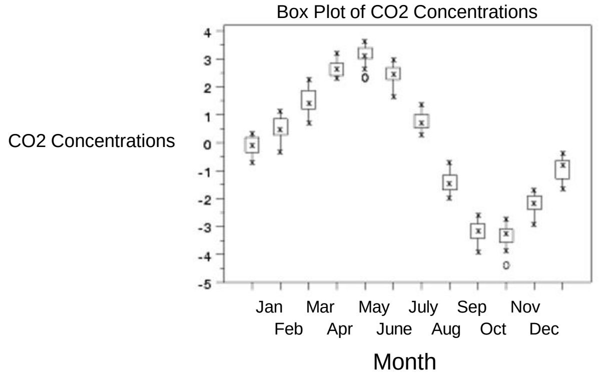




Step 2—stationarize series—seasonal adjustment

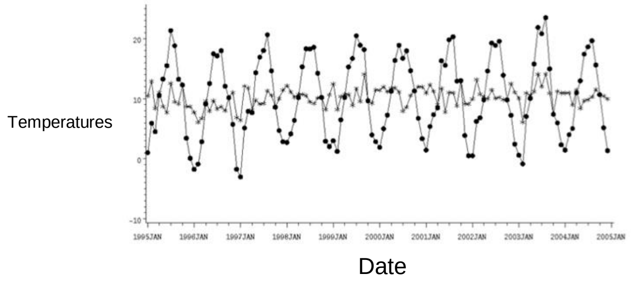
Plotting the detrended series identifies seasons

For CO2 concentration, you can model the period as being a year, with variation at the month level



Simple improvised adjustment: take several years of data, calculate the average value for each month, and subtract that from Y^1_t

$$Y^2_t = Y^1_t - S_t$$



Step 3—plot ACF and PACF to identify optimal parameters

- Auto Correlation Function (ACF)
  - Correlation of the values of the time series with itself
  - Autocorrelation "carries over"
  - Helps to determine the order, q, of a MA model
    - Where does ACF go to zero?
- Partial Auto Correlation Function (PACF)
  - An autocorrelation calculated after removing the linear dependence of the previous terms
  - Helps to determine the order, p, of an AR model
    - Where does PACF go to zero?

Step 4—build model—ARMA (p, q)

$$\begin{aligned} Y_t = &\boxed{\delta + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p}} \\ &+ \boxed{\varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}} \end{aligned}$$

- Y_t is detrended and seasonally adjusted.
- Combination of two process models
  - **Autoregressive**: Y_t is a linear combination of its last *p* values.
  - **Moving average**: Y_t is a constant value plus the effects of a dampened white noise process over the last *q* time values—lags.

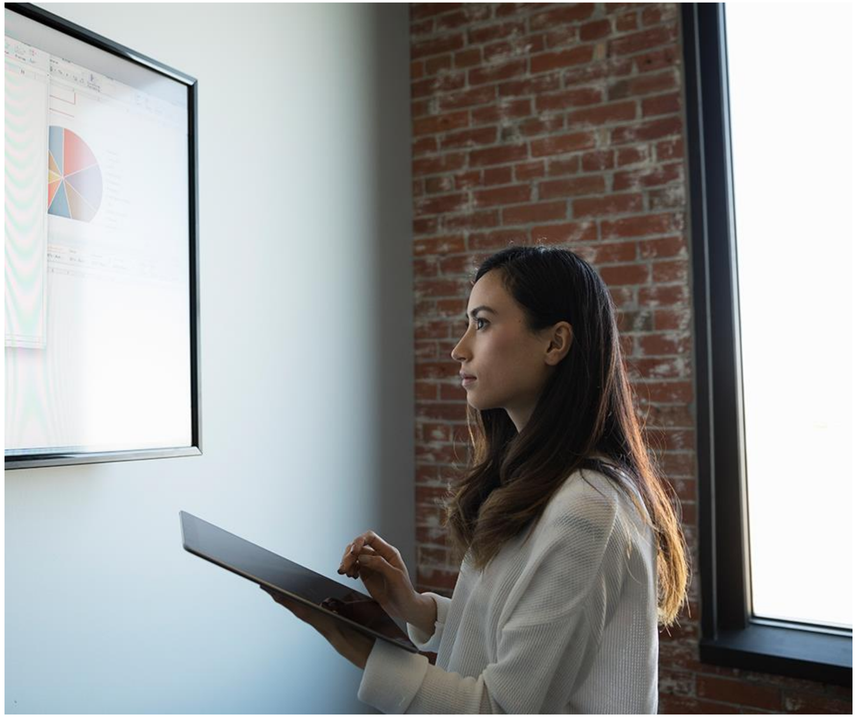
Step 4—build model—ARIMA (p, d, q)

- A stochastic modeling approach that can be used to calculate the probability of future value lying between two specified limits
- **ARIMA** adds a differencing term, *d*, to the **ARMA** model
  - Autoregressive <u>Integrated</u> Moving Average
  - Includes the detrending as part of the model:
    - Linear trend can be removed by *d*=1
    - Quadratic trend can be removed by *d*=2
    - And so on, for higher-order trends
- The general nonseasonal model is known as ARIMA (*p, d, q*):
  - *p* is the number of autoregressive terms.
  - *d* is the number of differences.
  - *q* is the number of moving average terms.

Step 4—build model—model selection

- Based on the data, the Data Scientist selects *p*, *d*, and *q*:
  - An "art form" that requires domain knowledge, modeling experience, and a few iterations
  - Use a simple model when possible:
    - AR model (q = 0)
    - MA model (p = 0)

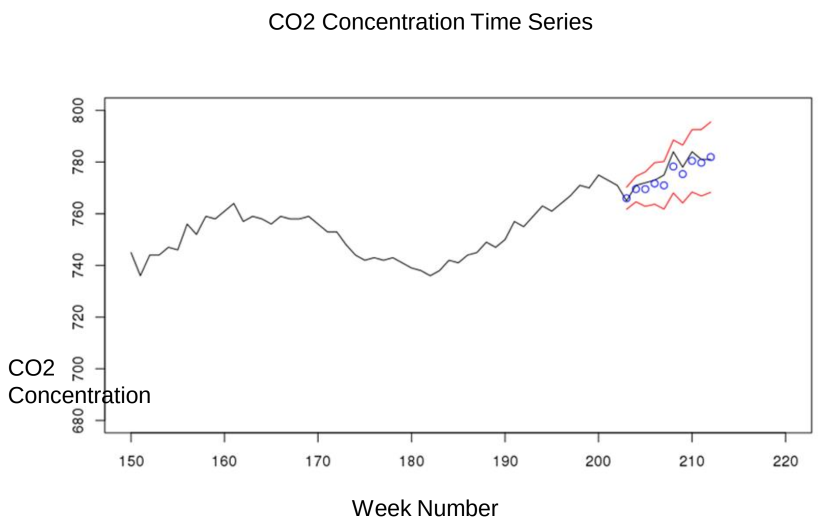
- Multiple models must be built and compared, using:
  - ACF and PACF.
  - p-values for the model parameter estimates
  - Information criteria metrics.
    - Imposes a penalty based on number of model parameters
    - Example: Akaike information Criterion (AIC)



Step 5—predict

After you have the final ARIMA model, you can use this model to make future predictions with the time series.

- Here is an example output of prediction using ARIMA model. You are predicting CO2 concentrations for the next four weeks, based on historical time series. The blue dots to the right of the graph indicate the prediction, and the two red lines indicate the prediction interval.



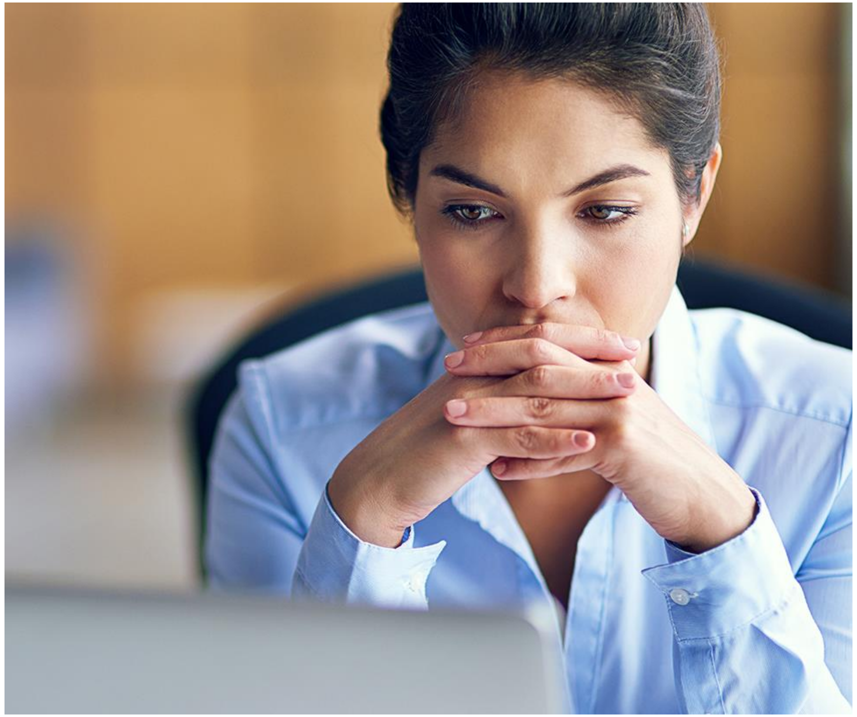
DELLTechnologies

Time Series Analysis—reasons to choose (+) and cautions (-)

Reasons to choose (+)	Cautions (-)
Minimal data collection Collect the series itself. It is not necessary to input drivers.	No meaningful drivers: prediction based only on past performance No explanatory value Cannot do "what-if" scenarios Cannot stress test
Designed to handle the inherent autocorrelation of lagged time series	It is an "art form" to select appropriate parameters
Accounts for trends and seasonality	Only suitable for short-term predictions

Check your knowledge

1. What is a time series, and what are the key components of a time series?
2. How do you detrend time series data?
3. What makes data stationary?
4. How is seasonality removed from the data?
5. What are the modeling parameters in ARIMA?
6. How do you use ACF and PACF to determine the stationarity of time series data?



Time series analysis—summary

During this lesson, the following topics were covered:

- Time series analysis and its applications in forecasting
- ARMA and ARIMA Models
- Implementing the Box-Jenkins methodology using R
- Reasons to choose (+) and cautions (-) with time series analysis

Module summary

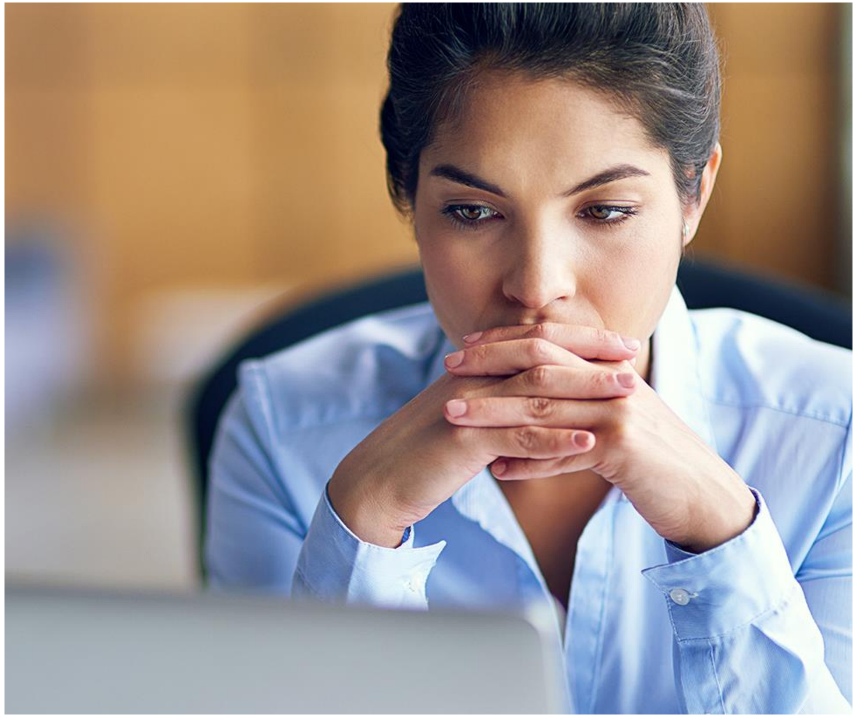
Key topics covered in this module	Methods covered in this module
Algorithms and technical foundations	Categorization, unsupervised: K-means clustering Association rules
Key use cases	Regression Linear Logistic
Diagnostics and validation of the model	Classification, supervised: Naïve Bayesian classifier Decision trees
Reasons to choose (+) and cautions (-) of the model	Time series analysis
Fitting, scoring, and validating model in R and in-db functions	Text analysis

Lesson: Improving model performance

Evaluating and improving model performance

Some common situations that data scientists may encounter when it comes to model performance:

- Underfitting: The model poorly explains the training dataset
- Overfitting: The model fits the training dataset very well (almost too well), but the predictive power of the model is poor
- Once in production, the predictive power of a model degrades over time



Underfitting

In the Model Building phase of an analytic project

- Underfitting is essentially the starting state
  - One or two obvious variables are added to the model
  - But the dataset is so poorly explained by the fitted model, there is no point to use the testing dataset
- To improve the model performance
  - Include additional features/variables to the model
  - Account for any correlated input variables
  - Apply necessary transformations to the input or outcome variables
  - Consider splitting one massive model into several smaller models
  - Try a different analytical model or an ensemble of analytical techniques
- Evaluate the fitted model against a testing dataset

Overfitting

- Overfitting is often observed when
  - The fitted model explains the training dataset very well
  - But the model fails to accurately predict the outcomes when presented with a new dataset
- Possible causes
  - Too many unimportant variables are added to the model
  - Fine tuning (training) the model based on the testing dataset
- Possible remediation steps
  - Revisit the variable/feature selection process
  - Ensure that any unimportant variables are removed from the model
  - Look for outliers that may be unduly influencing the modeling process

Other reasons that a model may not perform well

- The datasets were not properly obtained or built
  - Proper randomization techniques need to be applied
  - oThe data is not representative of the population of interest
- Predicting outcomes for inputs outside the range of the training dataset
- The operating environment has changed since the model went into production
  - External change examples
    - New competitors enter the market
    - New product alternatives are available
    - Macroeconomic changes occur
  - Internal change examples
    - Corporate strategy changes
    - Improvement initiatives implemented
    - New marketing or sales incentives

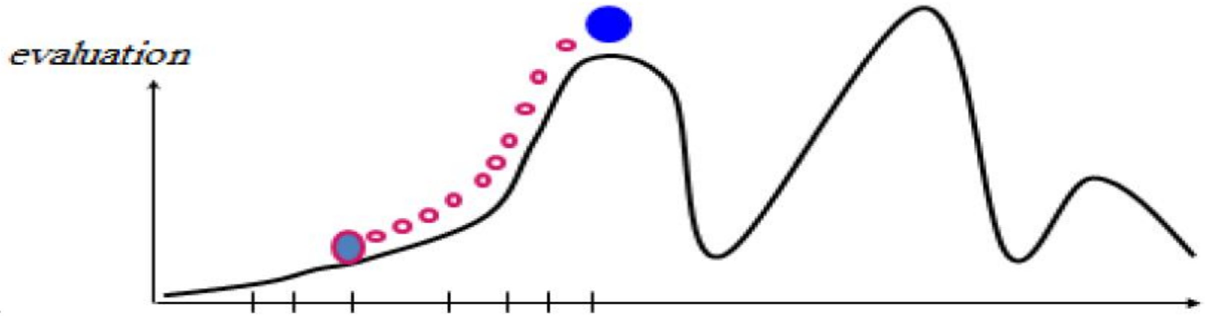
SUPPORT STRATEGY

- Hill-Climbing Search.
- Simulated Annealing Search.
- Local Beam Search.
- Genetic Algorithms.

Hill-Climbing Search

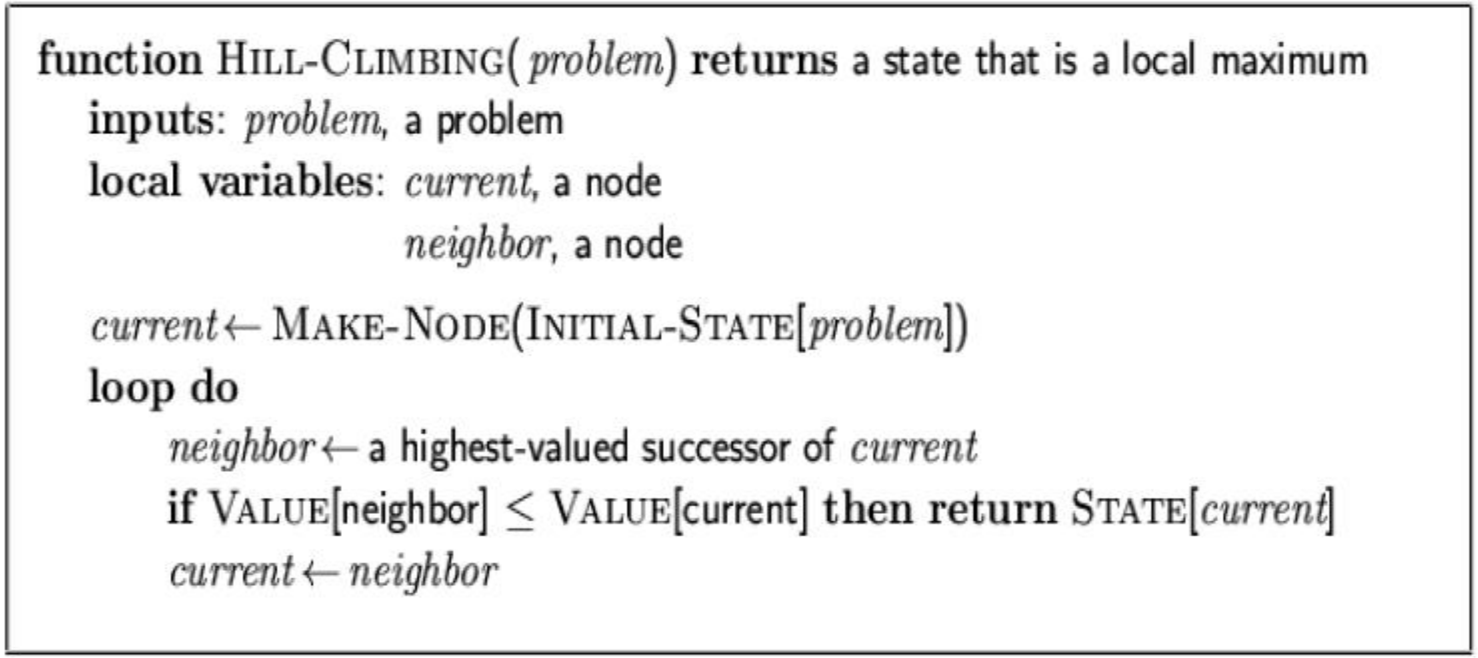
Hill-Climbing Search

- **Main Idea**: Keep a single current node and move to a neighboring state to improve it.
- Uses a loop that continuously moves in the direction of increasing value (**uphill**):
- Choose the best successor, choose randomly if there is more than one.
- Terminate when a peak reached where no neighbor has a higher value.
- It also called **greedy local search**, steepest ascent/descent.



Dell Technologies

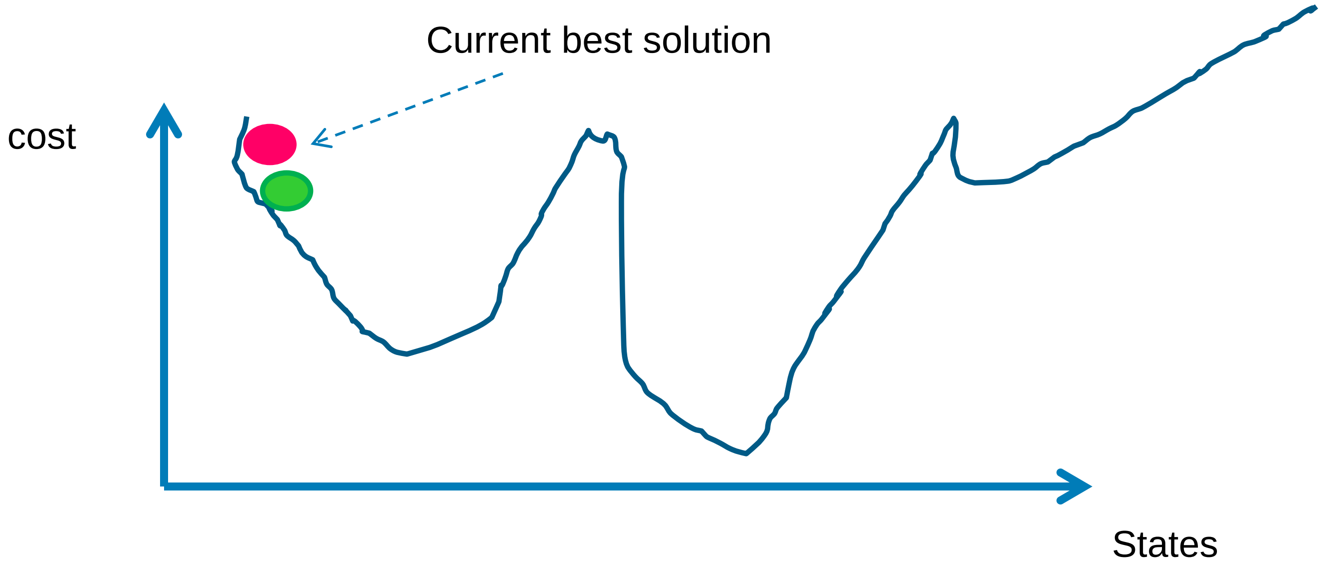
Hill-Climbing Search



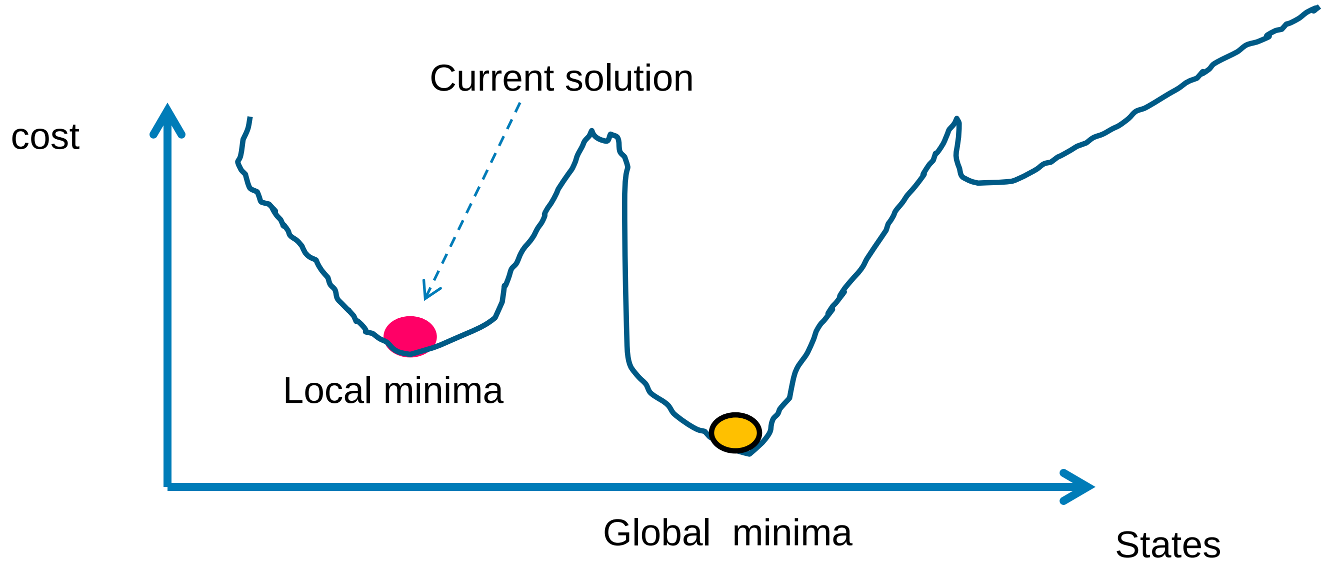
```
function HILL-CLIMBING( problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
```

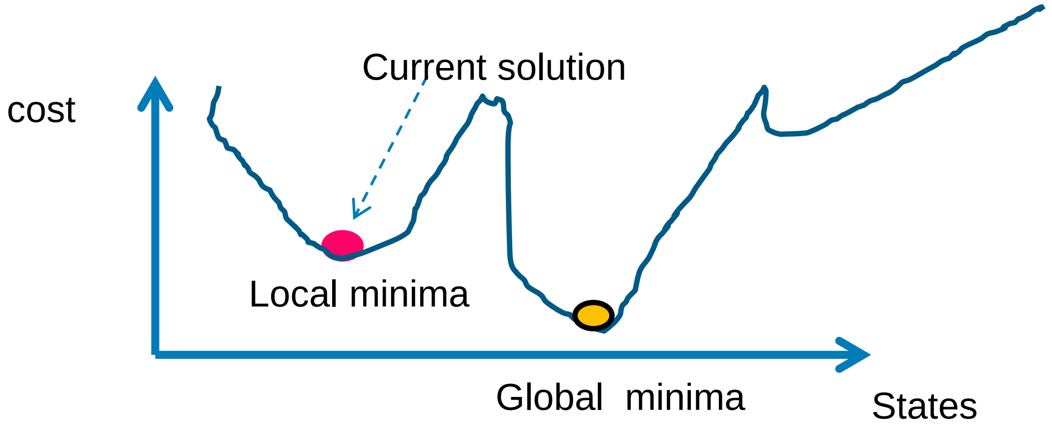
Hill-Climbing in Action …



Hill-Climbing in Action …



Hill-Climbing in Action …



Drawback: Depending on initial state, it can get stuck in local maxima/minimum or flat local maximum and not find the solution.

Solution : Random restart.

Simulated Annealing Search

The Problem

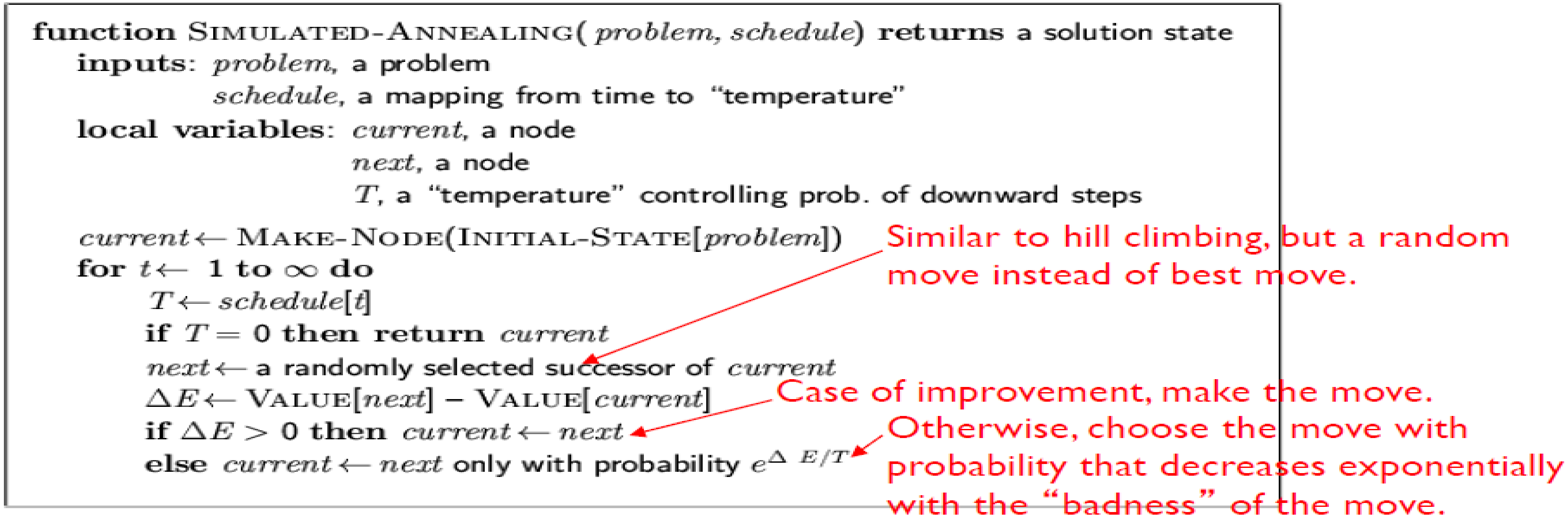
- ❑ Most minimization strategies find the *nearest* local minimum
- ❑ Standard strategy
  - ✓ Generate trial point based on current estimates
  - ✓ Evaluate function at proposed location
  - ✓ Accept new value if it improves solution
- ❑ We need a strategy to find other minima
- ❑ This means, we must sometimes select new points that do not improve solution How?

Simulated annealing Search

- **Main Idea**: escape local maxima by allowing some "bad" moves but gradually **decrease their frequency**.

- Instead of picking the **best** move, it picks a **random** move..

Simulated annealing Search



```
function SIMULATED-ANNEALING( problem, schedule) returns a solution state
    inputs: problem, a problem
            schedule, a mapping from time to "temperature"
    local variables: current, a node
                     next, a node
                     T, a "temperature" controlling prob. of downward steps

    current ← MAKE-NODE(INITIAL-STATE[problem])
    for t ← 1 to ∞ do
        T ← schedule[t]
        if T = 0 then return current
        next ← a randomly selected successor of current
        ΔE ← VALUE[next] – VALUE[current]
        if ΔE > 0 then current ← next
        else current ← next only with probability $e^{\Delta E/T}$
```

Similar to hill climbing, but a random move instead of best move.

Case of improvement, make the move.

Otherwise, choose the move with probability that decreases exponentially with the "badness" of the move.

- ➢ say the change in objective function is δ
- ➢ **if** δ is positive, then move to that state
- ➢ **otherwise**:
    - move to this state with probability proportional to δ
    - thus: worse moves (very large negative δ) are executed less often

Simulated annealing Search

- Lets say there are 3 moves available, with changes in the objective function of d1 = -0.1, d2 = 0.5, d3 = -5. (Let T = 1).
- pick a move randomly:
  - if d2 is picked, move there. (don't calculate just accept cause it is better)
  - if d1 or d3 are picked, probability of move = exp(d/T)
  - move 1: prob1 = exp(-0.1) = 0.9,
    - i.e., 90% of the time we will accept this move
  - move 3: prob3 = exp(-5) = 0.0067
    - i.e., 0.6% of the time we will accept this move

Test yourself

if the <u>T=3</u> , calculate the probability of move to the current state given the change in objective function in the following table

ΔE	Returned Solution	
	Next	Current
d1 = 0.4	_ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _
d2 = - 0.2	_ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _
d3= -3	_ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _

Properties of Simulated Annealing

- Cooling Schedule: determines rate at which the temperature T is lowered.

Local Beam Search

Local Beam Search

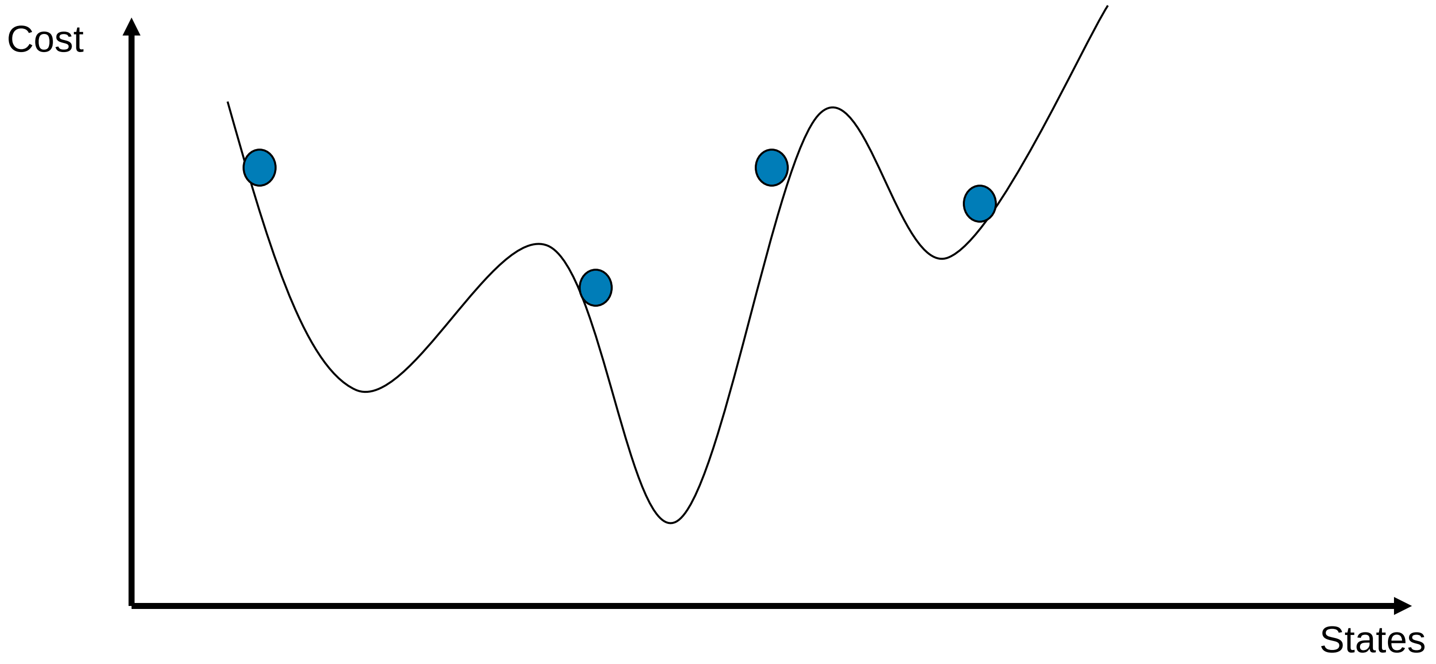
- **Main Idea**: Keep track of **k** states rather than just one.
- Start with **k** randomly generated states.

- At each iteration, all the successors of all **k** states are generated.

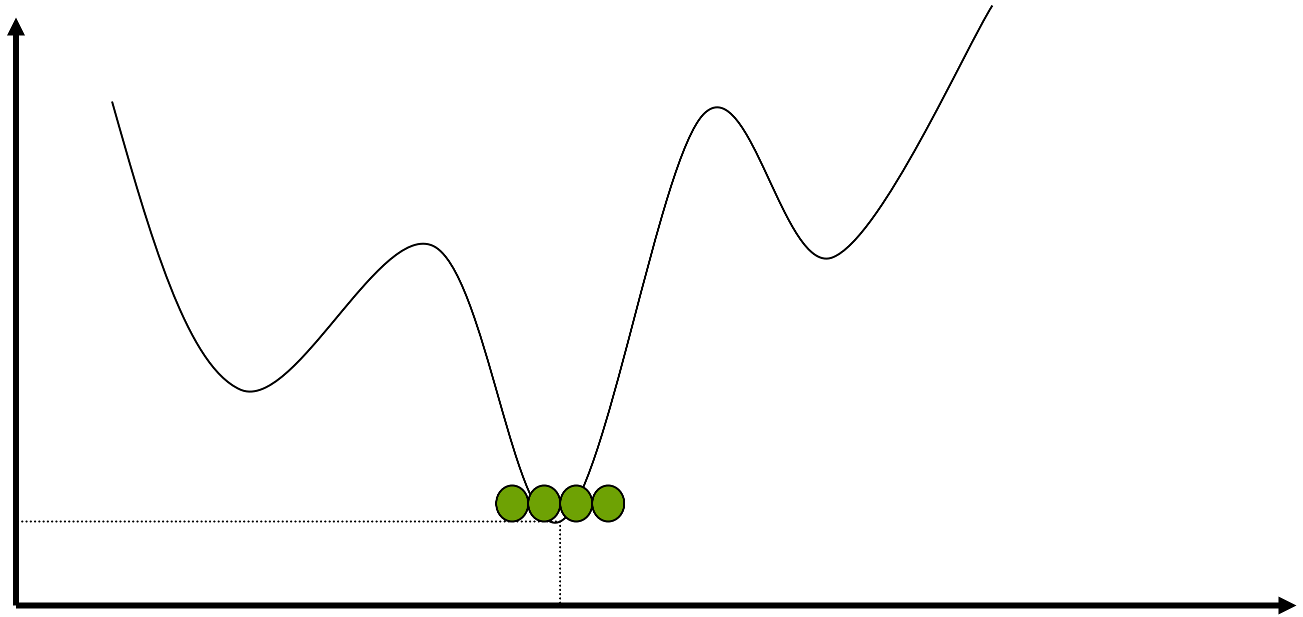
- If any one is a goal state, stop; else select the k best successors from the complete list and repeat.

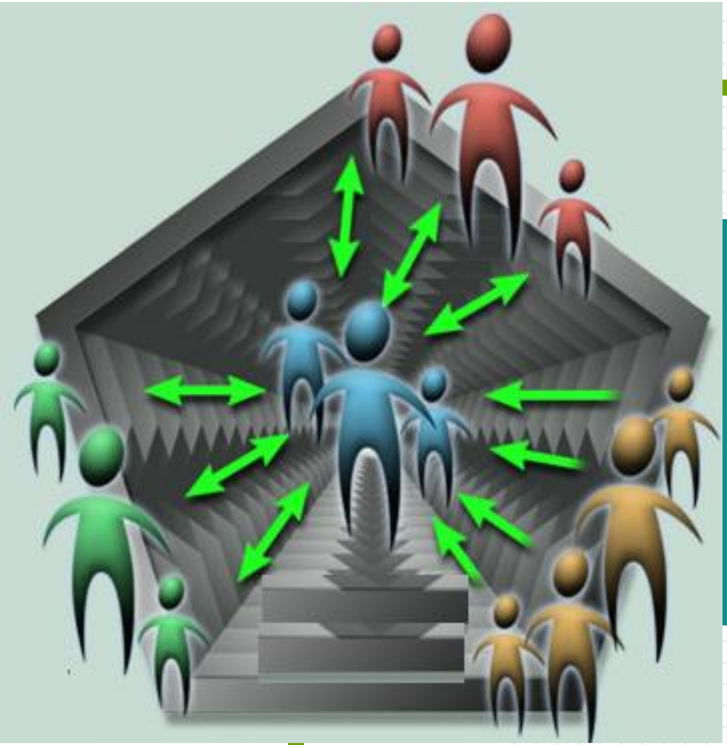
- Drawback: the k states tend to regroup very quickly in the same region → lack of diversity.

Local Beam Search



Local Beam Search





Thank you !