To be Greedy or not to be Greedy?

- Student's name: Tal Mizrahi

- Instructor's name: Zeev Nutov
- The Open University

Summary

- This lecture provides characterizations of the cases where the greedy algorithm produces the optimal solution.

- We'll also see that in general the use of a greedy algorithm might yield the unique worst possible solution.

- We'll demonstrate the above with the help of independent systems, matroids, and some algorithmic problems.

- Based on paper " When the greedy algorithm fails " by Jorgen Bang-Jensen.

The Greedy Algorithm

- The greedy algorithm is used in optimization problems.
- Its core principle: at each step, choose the local most optimal choice available.
- Its implementation is simple, and often has an efficient time complexity.

Why do we use the Greedy algorithm?

- There’s a common belief that the Greedy often produces approximated solutions significantly better than the worst solution.

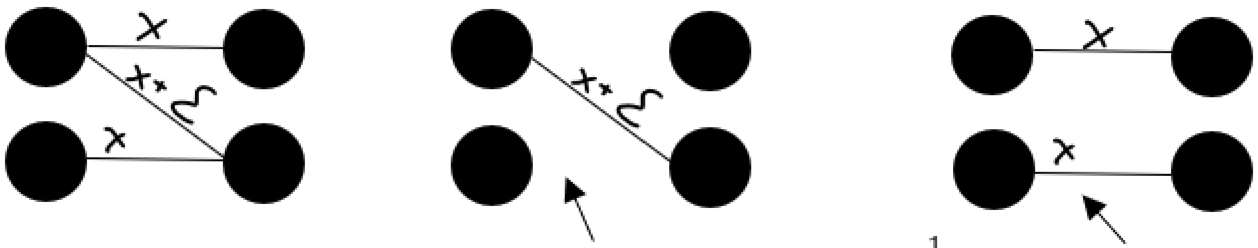


Illustration – a bipartite-graph maximization, $\varepsilon \to 0$.

The Greedy matching is at least $\frac{1}{2}$ as good as the Optimal.

- Even when not optimal, since it’s simple and efficient, it’s widely used as a trade-off to an optimal, but a less simple and less efficient solution.

The Problem – a unique worst solution

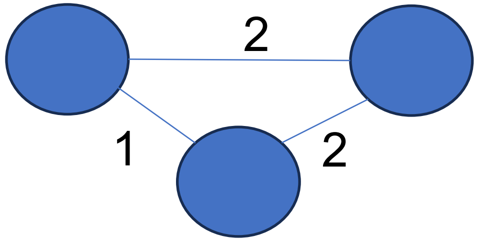
- However, certain experimental results and theories cast doubt on this assumption and shows that the Greedy might yield the unique worst solution or somewhat close to it.

- We’ll start by characterizing the occasions where the Greedy yields the optimal solution.

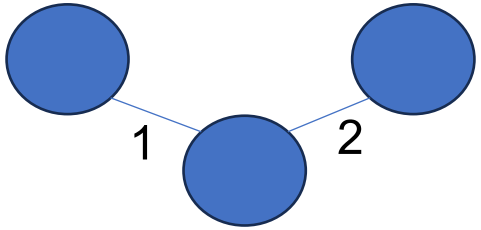
Minimum Spanning Tree (MST)

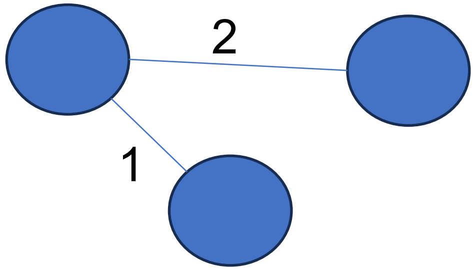
- Given a connected and weighted graph, find a minimal tree between all the nodes.
- Example:

Original:

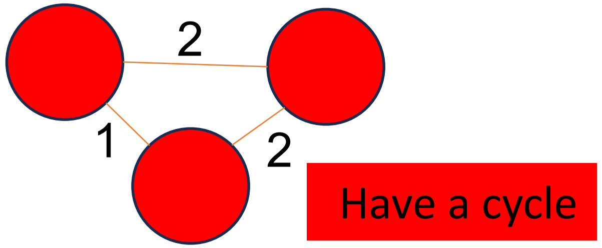


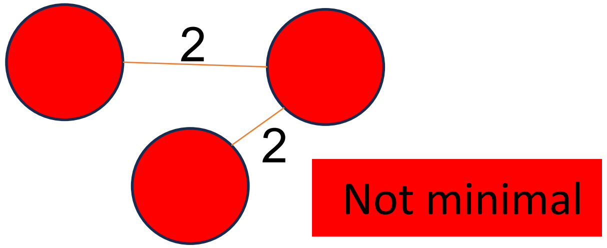
MST:





Not MST:

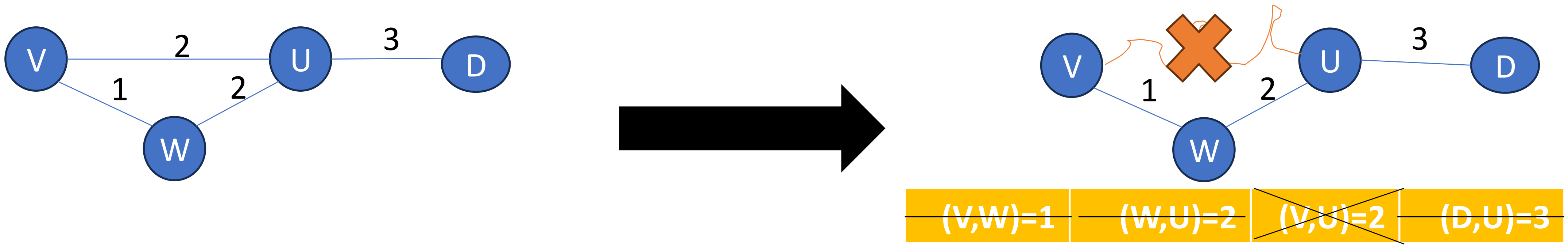




The Greedy in MST

- **Stage1:** add (V,W).
- **Stage2:** add (W,U).
- **Stage3:** eliminate (V,U), the cycle creator.
- **Stage4:** add (U,D).
- **Terminate** since there's no edges left.

A cycle creator is eliminated from the solution!



The best environment for the greediness

- Greedy algorithm works in MST, can we characterize problems in which greediness will succeed?
- Let's define a few terms that'll answer this question.

An independent system

- An independent system (E,F) is characterized by a ground set E and independent sets F of subsets of E, $\mathsf{F} \subseteq P(E) = 2^E$.

- Independence have two conditions:

  1) $\phi \in \mathsf{F}$.
  2) **Hereditary property**: If a set S $\in \mathsf{F}$, then $\forall A \subseteq$ S, A$\in \mathsf{F}$.

Independent system structure in MST problem

1) **The Ground Set:** E , is all the edges in the graph.

2) **Independent Sets:** $\mathsf{F} \subseteq 2^{E}$, where each F$\in$ F is a forest.

3)**Empty Set:** ϕ $\in$ F(empty forest).

4)**Heredity Property:** Sub-forest of a forest is a forest.

Thus, it's an independent system.

Matroid

- A Matroid is an independent system with one additional property.

- **Exchange property** - if A,B∈ F and |A| < |B| , then $\exists x \in B \setminus A$ such that
$A \cup \{x\} \in$ F.

Matroid structure in MST problem

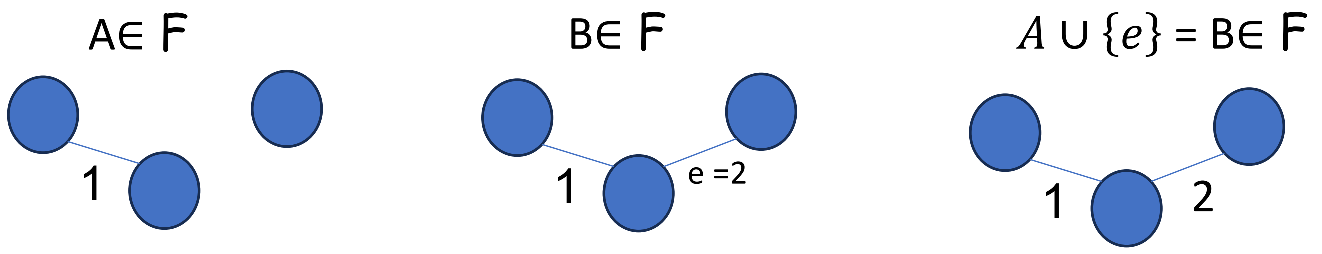
1) **The Ground Set:** E , is all the edges in the graph.
2) **Independent Sets:** F $\subseteq 2^{E}$, where each F∈ F is a forest.
3)**Empty Set:** ϕ ∈ F(empty forest).
4)**Heredity Property:** Sub-forest of a forest is a forest.

1)**It's An Independent System:** as shown before.

2)**Exchange Property:** For any A,B∈ F with |A|<|B| , ∃e ∈ $B\backslash A$ such That $A \cup \{e\}$ ∈ F.

Proof(2): assume negatively $A \cup \{e\}$ is cyclic, then A would connect all vertices, contradicting the fact that B is a forest with more edges.

Thus, it's a matroid.



Matroid maximization

- **Consider a matroid** - $M \; = \; (E, \mathsf{F})$.
- **Having a weight function** - $w: E \, \rightarrow \, R$.
- **For a subset A:** w(A) = $\sum_{a \in A} w(a)$.
- **Objective** – finding a maximum weight independent set.

Greedy algorithm for Matroid maximization

- **Initialization** –
  - start with $S = \phi \in \mathsf{F}$.

- **Iteratively adding elements** -
  - Going through all the elements in descending order .
  - If $S \cup \{element\} \in \mathsf{F} \rightarrow S \coloneqq S \cup \{element\}$

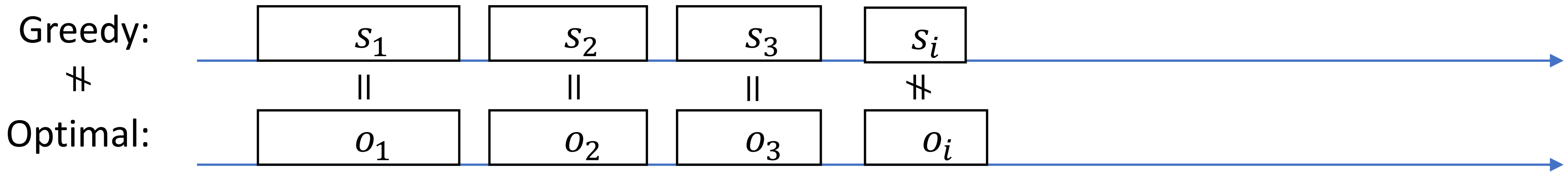
- Claim – the greedy is optimal and S now is the maximum weight independent set.

Claim proof

- For contradiction, say that $\exists OPT \in$ F such that $w(OPT) > w(S)$.
- Let $s_1, s_2, \dots, s_k$ be S's elements and $o_1, \dots, o_m$ be OPT's elements.
- **Cardinality argument** $(m = k)$ –
  - if $k > m$, then $\exists x \in S \backslash OPT$, that by the **exchange property** can be added to OPT, contradicting O as an optimal solution. Therefore, $k \ngtr m$.
  - If $m > k$,then $\exists x \in OPT \backslash S$, that by the **exchange property** can be added to S, but the greedy algorithm should've considered such an element, therefore adding it to S will violate S independency. Therefore, $m \ngtr k$.
- Thus, $m = k$.

Claim proof continuation

- Let $OPT = \{o_1, o_2, \dots, o_m\}$ and $S = \{s_1, s_2, \dots, s_m\}$
- Assume that both are sorted in a decreasing order of weights, $w(o_i) \geq w(o_{i+1})$ and $w(g_i) \geq w(g_{i+1})$.
- Since $S \neq OPT$ and OPT is optimal, there must $\exists i$ such that $s_i \neq o_i$ and $w(s_i) < w(o_i)$. Find the smallest such i.

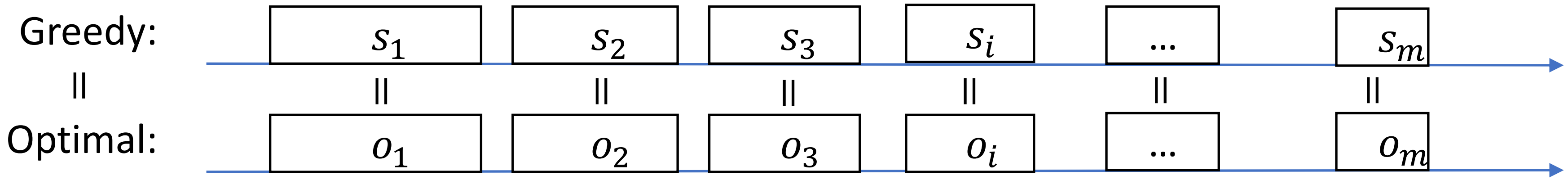


Claim proof continuation

- However, Since $(s_1, \dots, s_{i-1}) = (o_1, \dots, o_{i-1})$, the greedy algorithm would locally choose the highest weighted element for the ith element, hence $w(o_i) \ngtr w(s_i)$.

- Since OPT is the optimal solution, $w(s_i) \ngtr w(o_i)$, thus $w(s_i) = w(o_i)$.

- Repeat for every i and get $S = OPT. \blacksquare$



Matroid as a proving tool

- As proven, if a system is a Matroid $\rightarrow$ the Greedy is optimal.
- Based on the previous, a way to prove that a greedy algorithm outputs the optimal solution on some problems requires showing:
  - That the problem satisfy the properties of a matroid.
- Let’s demonstrate that in the MST problem.

The Greedy preserves the independent set

- As already shown, the MST is a matroid maximization problem.
- **Greedy Choice:** At each step, the algorithm chooses the smallest edge that doesn’t form a cycle with the already included edges.
- By that, the algorithm ensures that the growing set of chosen edges always remains an independent set in the matroid.
- Therefore, the greedy solution indeed is the optimal solution in this case.■

Job scheduling problem

- The Job scheduling problem is another matroid maximization problem.
- Consider a set of jobs, each with its own deadline and profit.
- We have one machine that completes one job in one time unit.
- The objective is to schedule the jobs in a way that maximizes total profit, with the constraint that each job must be completed by its deadline.

Job scheduling problem – A Matroid

- **Independent sets** – a subset of jobs is independent if all jobs can be scheduled by their deadlines ($\phi \in F$).

- **Heredity property** satisfaction – removing jobs doesn't create collisions of time units.

- **Exchange property** – if an independent set A is larger than B, there must be at least one job in A that can be scheduled in the slots available for B.

Job scheduling problem

- Suppose we have the following jobs sorted by profit in descending order:
  - J_1: Deadline = 2, profit = 20.
  - J_2: Deadline = 2, profit = 15.
  - J_3: Deadline = 1, profit = 10.
  - J_4: Deadline = 3, profit = 5.
  - J_5: Deadline = 3, profit = 1.

- We also have 3 time slots in accordance with the deadlines.

Timeslots:

The greedy in Job scheduling problem

- The greedy algorithm will place the jobs in the time slots from the highest profit to the lowest if the time slot is available.

J_1: Deadline = 2, profit = 20.
J_2: Deadline = 2, profit = 15.
J_3: Deadline = 1, profit = 10.
J_4: Deadline = 3, profit = 5.
J_5: Deadline = 3, profit = 1.

Timeslots:	J_3	J_1	J_4
	1	2	3

Matroids - a Rich World

- **Graphic Matroids –** such as the MST.

- **Transversal Matroids –** can be associated as bipartite graph matching. Such as the Job scheduling problem.

- There are plenty more types of matroids to explore:

Uniform matroids, Cographic matroids, Binary Matroids, Algebraic Matroids, Partition Matroids, Lattice Path Matroids, Regular Matroids, Discrete Matroids, and more …

Some more matroid optimization problems

- **Maximum Weight Forest** –
  - The goal is to maximize the total weight of the selected edges while ensuring the subset of edges forms a forest.(generalize the MST problem)

- **Basis of Vector Space – algebraic matroid**
  - Find a base for a R^n vector space.
  - **Greedy -** Adding vectors to the basis, checking that they're not linear dependent.

What about non-Matroid structures?

- As we’ve seen, a matroid system ensures us an optimal solution for the greedy algorithm.

- In the followings, we’ll examine non-Matroid structures such as the Symmetric Traveling Salesman Problem and the Minimum Coin Problem.

Symmetric Traveling Salesman Problem(STSP)

- Given a complete graph of cities(vertices) and distances(weights) between each pair of cities.

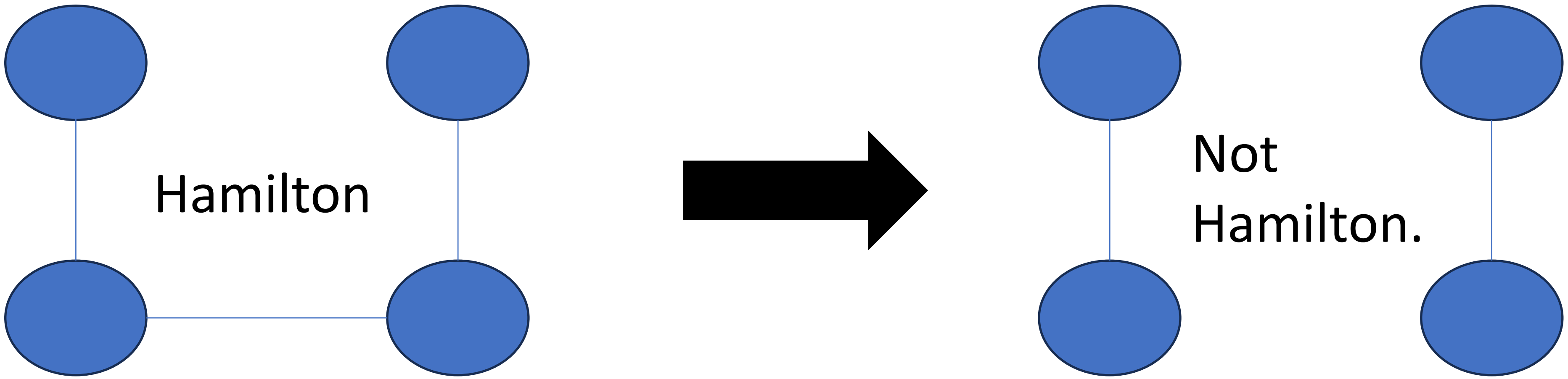
- The task is to find the shortest possible route that visits each city exactly once and returns to the origin city(in the **S**TSP, the distance A→B is the same as B→A).

- The STSP is a Minimization Problem.

STSP - not an independent system

- Consider independent sets as Hamilton cycles.
- ~~**Heredity Property:**~~ The subsets of a Hamilton cycle do not necessarily form Hamilton cycles themselves.

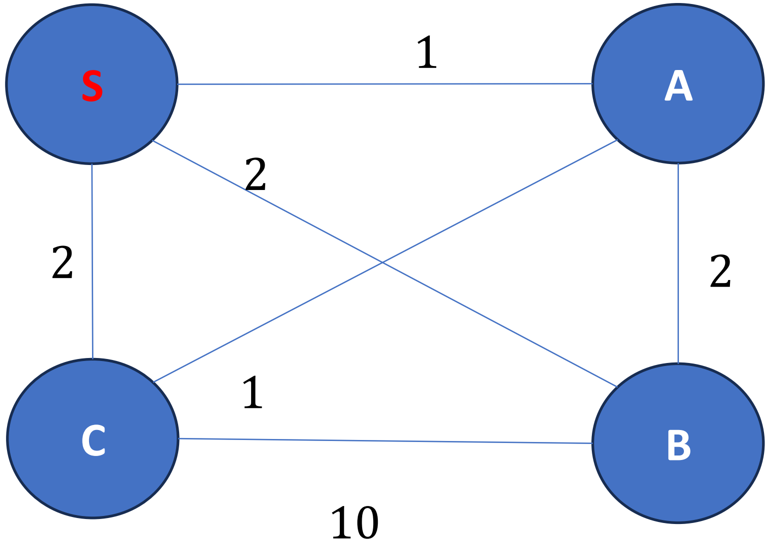
- Since the STSP is not an independent system , it is also not a matroid.



Choosing the greedy route - initialization

Each iteration, the greedy algorithm will choose the nearest city from the current position.

Initializing ‘S’ as the source node



Node/ Distance to	S(start)	A	B	C	S(end)
S(end)	∞	1	2	2	–
A	1	∞	2	1	–
B	2	2	∞	10	–
C	2	1	10	∞	–

$$R = 0$$

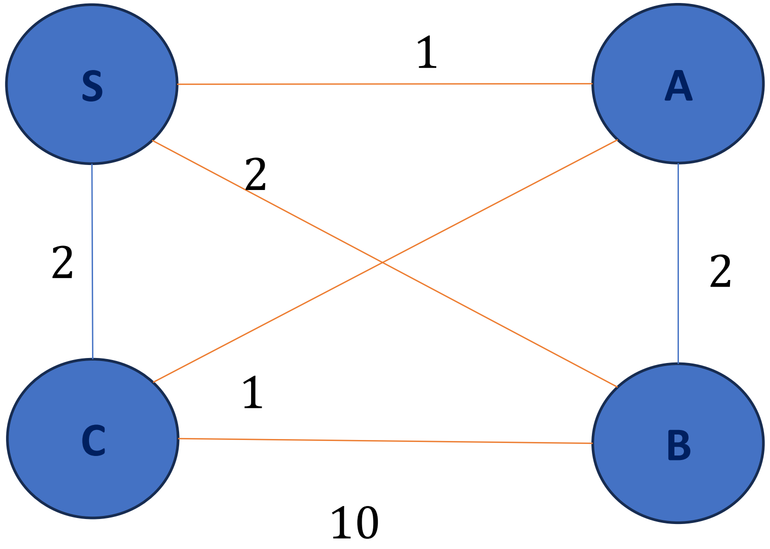
$$I(R) = \{u(S,A), u(S,B), u(S,C), u(A,B)$$
$$u(A,C), u(B,C)\}$$

Choosing the greedy route - iterations

$d(S \to A) = u(S,A) = 1$
$d(A \to C) = u(A,C) = 1$
$d(C \to B) = u(C,B) = 10$
$d(B \to S) = u(B,S) = 2$

Terminate.

Node/ Distance to	S(start)	A	B	C	S(end)
S(end)	∞	1	2	2	–
A	1	∞	2	1	–
B	2	2	∞	10	–
C	2	1	10	∞	–

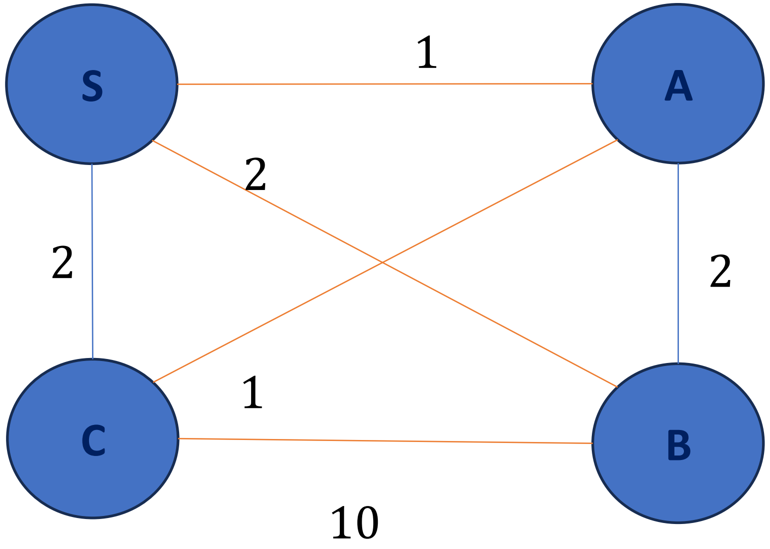


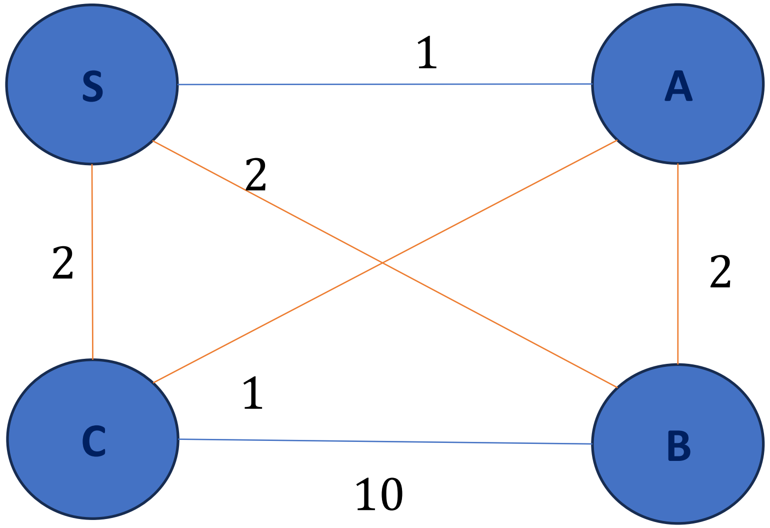
$$R = u(S,A) + u(A,C) + u(C,B) + u(B,S).$$
$$I(R) = \{u(S,C), u(A,B)\}.$$

Greedy Vs. Optimal

The Greedy total route is
$u(S,A) + u(A,C) + u(C,B) + u(B,S)$
$= 1 + 1 + 10 + 2 = 14.$
Can we find a better route?

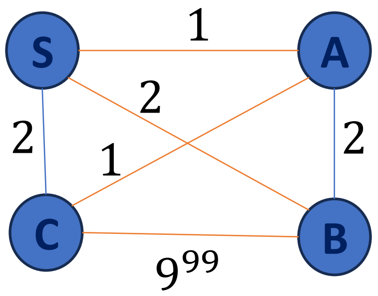
The Optimal total route is
$2 + 2 + 1 + 2 = 7.$





Conclusions of the executions

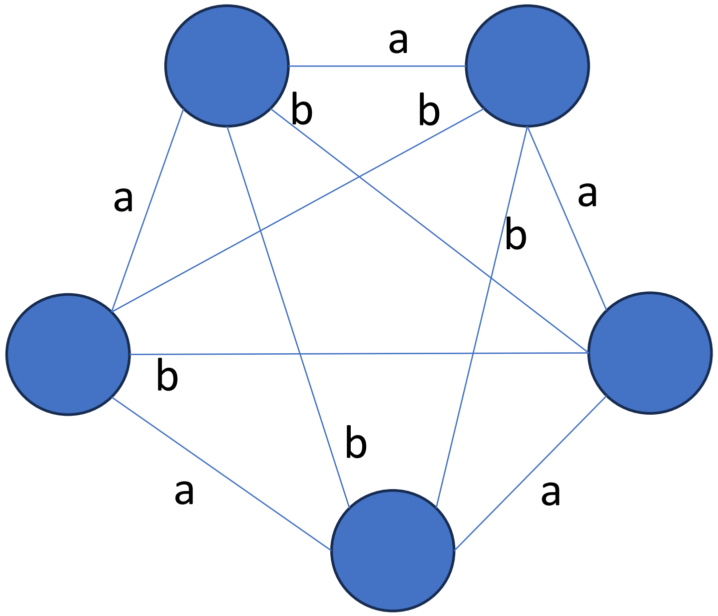
- The greedy algorithm provided the unique worst solution.
- Therefore, in the STSP, the greedy algorithm may yield the unique worst solution.
- What would've happened if the bottom edge will be replaced with a weight of 9^{99} instead of 10?



Theorem 1:

- When will the greedy algorithm never find the worst solution in STSP?

- Conditions: n $\geq$ 4 and $|W| \leq \left\lfloor \frac{n-1}{2} \right\rfloor$.



Illustration

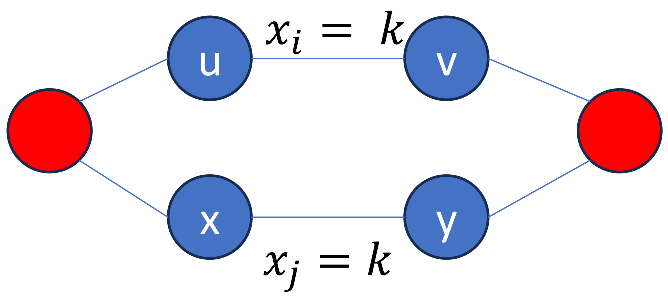
Start of Proof

- Assume $B = \{x_1, \ldots, x_n\}$ is the unique worst solution of edges chosen by the greedy algorithm in a sequence given by their indices.

- Since there's only $\left\lfloor \frac{n-1}{2} \right\rfloor$ different weights, by induction, there most exist two edges in B, x_i, x_j, weighted as 'k', without any common nodes.

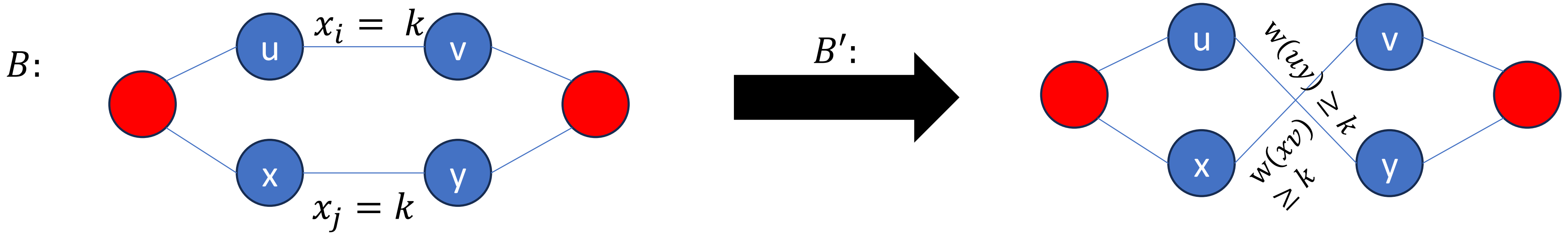
- Let x_i be the edge $\{uv\}$, and x_j be the edge $\{xy\}$.

B:



Continuation of proof

- Since the graph is a clique, let another solution $B' = \mathrm{B} \cup \{uy, vx\} - \{uv, xy\}$.
- Both other edges must have a weight greater than 'k' since otherwise, the greedy algorithm would have chosen one of these edges instead of x_i or x_j in the step just before the first pick of x_i, x_j.



Conclusion of Proof

- Thus, $w(B') \geq w(B)$, a contradiction. Therefore, the greedy algorithm will never produce the unique worst possible solution under the restricted conditions.■

- To conclude, these conditions ensure that there is always at least one locally optimal choice available, preventing the greedy from being trapped in the unique worst solution.

The coin problem – multiplication factor

- We already know that if a system is a matroid $\rightarrow$ the greedy algorithm is optimal. But is the other direction also true?

- The goal is to pay for an item with a minimum number of coins possible. The ground set is the values of the coins.

- **Multiplication Factor** in a coin system says that each denomination is a multiple of the previous.

- The Greedy (chooses the largest coin every iteration) works optimally with a set of coins which applies the **Multiplication Factor**, such as $\{1,5,10,25\}$.

The coin problem – multiplication factor

- However, for a set of coins without the multiplication factor such as {1,3,4}, the greedy solution will fail.

- For example, summing 6 with a greedy would use $|\{4,1,1\}| = 3$ while the optimal solution is $|\{3,3\}| = 2$.

- If this was a matroid maximization problem, the greedy solution would’ve been optimal for every possible ground set independently on other factors.

- Therefore, the greedy algorithm is optimal $\not\rightarrow$ a system is a matroid.

Based on the paper “Canonical Coin Systems For Change-Making Problems” by Xuan Cai.

Conclusions and open problems

- We introduced the notion of the greedy algorithm on non-independent systems, and Matroids.

- Matroids are a rich world, and as much as it’s rich, the much problems that can be solved optimally with a greedy algorithm.

- It’s still investigated when the greedy algorithm will succeed or might produce the unique worst solution for 100% certainty.

- If a system isn’t a matroid, we should take extra caution implementing the greedy algorithm.