CO450 Combinatorial Optimization

Riley Jackson

September 28, 2020

Contents

1 Spanning Trees **1**
1.1 Overview 1
1.2 Basics 1
1.3 Minimum Spanning Tree Problem 2
1.4 Integer Programming Formulation 3

2 Greedy Algorithms and Matroids **4**
2.1 Maximum Weight Forest 4
2.2 Matroids 5

1 Spanning Trees

1.1 Overview

- Characterize spanning trees.
- Characterize minimum spanning trees (MSTs)
- Use characterizations to derive algorithms (and prove correctness)
- Prove correctness using linear programming

1.2 Basics

What is a spanning tree?

Definition Given a graph $G = (V, E)$, a subgraph T of G is a **spanning tree** of G if:

- $V(T) = V(G)$
- T is connected
- T is acyclic

Note Unless otherwise mentioned, all graphs are assumed to be simple (no self-loops, not multi-edges) and undirected.

We have from MATH 249 the following characterization of spanning tree:

Theorem *Let $G = (V, E)$ be a graph and let T be a subgraph of G with $V(T) = V(G)$. The following are equivalent:*

- *T is a spanning tree of G*
- *T is minimally connected (ie: removing any edge creates 2 connected components)*
- *T is maximally acyclic (ie: adding any edge creates a cycle)*
- *T is connected and has $|V| - 1$ edges*
- *T is acyclic and has $|V| - 1$ edges*

- $\forall_{u,v \in V}$ *there exists a unique* (u, v) *path in* T *(called* T_{uv}*)*

Notation Given a graph $G = (V, E)$ and some $A \subseteq V$, we denote the **cut** of A by $\delta(A)$. That is,

$$\delta(A) = \{e \in E : |e \cap A| = 1\}$$

Another useful theorem is the following

Theorem *A graph* $G = (V, E)$ *is connected if and only if for all* $\varnothing \subsetneq A \subsetneq V$ *we have* $\delta(A) \neq \varnothing$*.*

1.3 Minimum Spanning Tree Problem

Given these (rather brief) pre-requisites, we are now ready to define the minimum spanning tree problem. As **input** we take a connected graph $G = (V, E)$ as well as costs c_e for each edge $e \in E$. As **output** we produce a minimum spanning tree T of G.

Of course, we haven't actually defined a minimum spanning tree yet, but the definition is the obvious one:

Definition Let $G = (V, E)$ be a graph with edge weights $(c_e)_{e \in E}$. We define the **cost** of a subgraph H of G to be $c(H) \coloneqq \sum_{e \in E(H)} c_e$. Given a spanning tree T of G, we say that T is a **minimum spanning tree** if $c(T)$ is minimized amongst all spanning trees of G.

Theorem *Given a graph* $G = (V, E)$ *with edge weights* $(c_e)_{e \in E}$ *and a spanning tree* T *of* G*, the following are equivalent:*

- T *is a minimum spanning tree*
- $\forall_{uv \in E \setminus E(T)} \forall_{e \in E(T_{uv})}$ *we have* $c_e \leq c_{uv}$
- $\forall_{e \in E(T)}$ *if we let* T_1, T_2 *be the two connected components of* T *upon removing* e *then* e *is a minimum cost edge in* $\delta(T_1)$

Proof. We sketch a proof:

- Suppose T is a minimum spanning tree and there is some $uv \in E$ and some $e \in E(T_{uv})$ such that $c_e > c_{uv}$. Notice that adding the uv edge introduces a cycle containing e so the graph T' obtained by adding uv and removing e is still a tree (connected and $|V| - 1$ edges) yet it has strictly lower cost, a contradiction.
- Suppose the second property holds and let $e \in E(T)$ be such that there is some $f \in \delta(T_1)$ with $c_f < c_e$ (where T_1, T_2 are as defined above). Let u, v be the endpoints of f and notice that T_{uv} must contain e (for otherwise there is another (u, v) path), but this is a contradiction since $c_f < c_e$.
- Finally, suppose the third property holds for a spanning tree T and let T' be a minimum spanning tree such that $k \coloneqq E(T) \cap E(T')$ is maximized. If $k = |V| - 1$, then $T = T'$ and we're done, otherwise there is some $e \in E(T) \setminus E(T')$. Let T_1, T_2 be the connected components of T after removing e, then there is some $e' \in \delta(T_1) \cap E(T')$. By assumption, $c_e \leq c_{e'}$, but then the tree T with e' substituted for e has $k + 1$ edges in common and is minimum, contradicting the choice of T'.

□

With the above characterizations in hand, we are ready to describe Kruskal's algorithm for finding minimum spanning trees:

Algorithm 1 Kruskal's algorithm for finding minimum spanning trees

```
Initiation: H = (V, ∅)
while H is not a spanning tree do
    e = cheapest edge who's endpoints aren't in the same connected component of H
    add e to H
return H
```

We claim this algorithm is efficient in the computational complexity sense. To see this, notice that we can implement it by first sort the edges by weight ($O(|E| \log |E|)$) then add those edges with endpoints

in different components to H. Checking and updating connected component membership can be done in $O(|V|)$ time by mapping each vertex to a representative of its component. Since we do this for every edge, we conclude that Kruskals algorithm runs in $O(|V||E|) + O(|E|\log|E|) = O(|V||E|)$ time[1].

To see that this algorithm is correct we will first show that it returns a spanning tree, then we will show that spanning tree is minimal. Indeed, since G is connected, the minimal edge on line 3 always exists and inner loop is well defined. Since each iteration reduces the number of connected components by one, it follows that the loop terminates in $O(|V|)$ iterations. Suppose for contradiction that the H returned is not minimal, then there will be some $uv \in E \setminus E(H)$ and some $e \in H_{uv}$ such that $c_{uv} < c_e$. However, since our implementation iterates through edges in order, it follows that H_{uv} would not exist when uv is being tested, hence uv would have been added to H. Since it wasn't, this is a contradiction and H is minimal.

1.4 Integer Programming Formulation

Let $G = (V, E)$ be a graph and let $n = |V|, m = |E|$. Let $x_e \in \{0, 1\}$ be a variable that indicates if edge e is in the MST. Recall that spanning tree are acyclic and have $n - 1$ edges and consider the following formulation:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x(E) = n - 1 \\ & * \\ & x \in \{0, 1\}^m \end{aligned}$$

where we define $x(F) = \sum_{e \in F} x_e$ for $F \subseteq E$. This formulation finds the minimum weight subgraph with $n - 1$ vertices, so we need to choose the $*$ constraint to ensure that the subset of chosen edges forms an acyclic subgraph, hence a spanning tree.

So suppose that $F \subseteq E$. How many edges of F can a spanning tree have? Let $K(F)$ be the number of connected components of (V, F), we must have atmost $n - K(F)$ edges. It is clear that if we have $x(F) \leq n - K(F)$ for all $F \subseteq E$ then x represents an acyclic graph.

Hence we can fill in the $*$ constraint to get:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x(E) = n - 1 \\ & x(F) \leq n - K(F) \quad \forall F \subseteq E \\ & x \in \{0, 1\}^m \end{aligned}$$

and the above is an integer programming formulation of the minimum spanning tree problem.

Notice that when $F = \{e\}$ we have that $K(F) = n - 1$, thus the constraint for F states that $x(F) = x_e \leq n - K(F) = n - (n - 1) = 1 \implies x_e \leq 1$. It follows that the integer program can be rewritten as

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x(E) = n - 1 \\ & x(F) \leq n - K(F) \quad \forall F \subseteq E \\ & x \geq \mathbb{0} \\ & x \in \mathbb{Z}^m \end{aligned}$$

and the linear program relaxation is clearly seen to be

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x(E) = n - 1 \\ & x(F) \leq n - K(F) \quad \forall F \subseteq E \\ & x \geq \mathbb{0} \end{aligned} \tag{PST}$$

[1]This bound is not tight, Kruskal's can be implemented in $O(|E|\log|V|)$ time if we are more careful during the inner loop.

Thus, as an alternative proof of correctness of Kruskals we can show that the MST given is optimal in the LP (using complementary slackness). Let z^* be the optimal value of the LP above, notice that z^* exists because PST is feasible (since G is connected by assumption) and since PST is bounded (all variables are between 0 and 1).

Notice that any spanning tree T corresponds to a feasible solution to PST, hence $c(T) \geq z^*$, thus the spanning tree T' returned by Kruskals has $c(T') \geq z^*$.

Notice also that the constraint for $F = E$ is contained in the constraint $x(E) = n - 1$, so it follows that the following LP is equivalent to PST

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x(E) = n - 1 \\ & x(F) \leq n - K(F) \quad \forall F \subsetneq E \\ & x \geq \mathbb{0} \end{aligned} \tag{PST}$$

and taking the dual we get the dual problem is

$$\begin{aligned} \max \quad & \sum_{F \subseteq E} (n - K(F)) y_F \\ \text{s.t.} \quad & \sum_{F : e \in F} y_F \leq c_e \quad \forall e \in E \\ & y_F \leq 0 \quad \forall F \subsetneq E \end{aligned} \tag{DST}$$

Write $E = \{e_1, \ldots, e_m\}$ with $c(e_1) \leq \cdots \leq c(e_m)$ and let $E_i = \{e_1, \ldots, e_i\}$. Set $\bar{y}_{E_i} = c(e_i) - c(e_{i+1})$ for $1 \leq i < m$ and set $\bar{y}_E = c(e_m)$. For all other $F \subseteq E$, set $\bar{y}_F = 0$. Notice that all dual variables are non-negative by construction since the e_i's were sorted by cost (except for maybe y_E, but y_E is unconstrained). It is also clear that the sum in the constraint telescopes to leave exactly c_e and is thus met with equality for all $e \in E$.

Thus, to exhibit complementary slackness we must only show that the dual variable is zero or the corresponding primal constraint is tight. Let $\bar{x}$ be the incidence variable for T', the tree obtained from Kruskal's. Clearly the first primal constraint is met with equality (which is good because c_e is unconstrained in the dual). Let $T_i' = (V, E_i \cap E(T'))$ and notice that T_i' is a maximally acyclic subgraph of $H_i = (V, E_i)$ (if not we can join two connected components of T_i' with an edge e_j with $j \leq i$, but this contradicts Kruskals choice of cheapest connecting edges). It follows that $\bar{x}(E_i) = n - K(E_i)$ and all primal constraints are tight. Thus we get that $z^* = c^T \bar{x} = c(T')$ and the tree returned by Kruskals algorithm is minimal.

Remark An astute reader may notice that any choice of weights yields an integral optimal solution by the above proof, hence every vertex is integral and the above LP relaxation exactly solves the integer program (this is very very rare in general).

Remark Another way to formulate the acyclic constraint is to require for all $\varnothing \subsetneq S \subsetneq V$ that $x(E(S)) \leq |S| - 1$.

2 Greedy Algorithms and Matroids

Generally speaking, an algorithm is greedy if makes a sequence of locally optimal choices and obtains a globally optimal result. Notice that not every greedy algorithm works, our goal will be to try and characterize when they are guaranteed to succeed.

2.1 Maximum Weight Forest

Definition Given $G = (V, E)$, a **forest** is an acyclic subgraph.

The maximum cost forest is then the problem of finding some forest of G of maximum cost. One approach is to compute the MST with respect to negated weights, then simply remove all negative cost edges (if G is not connected, add missing edges with cost $-\ell_\infty(c) - 1$ where c is the cost vector). To see this is optimal (this is very handwavy btw and is essentially the raw argument I used to convince myself), let T be the maximum spanning tree, let T' be T with negative edges removed and let T^* be a max cost

forest. Let F be the set of negative edges in T. For $e \in F$, let T_1, T_2 be the two connected components of $T - e$, then e is the maximum cost in $\delta(T_1)$. In particular, every edge crossing this cut is negative and so T^* doesn't cross it. Clearly T_1 and T_2 are minimum spanning trees on their respective vertex sets. By repeating this process and subdividing again, we get down to components that are connected with positive edges and in this case T and T^* agree. Working back up, we see that $c(T) = c(T^*)$.

Of course, we can write this idea in pseudocode and get

Algorithm 2 Modified Kruskal's algorithm for finding maximum cost forests

```
Initiation: H = (V, ∅)
while ∃e ∈ E : c_e > 0 and the endpoints of e are in different connected components of H do
    e = highest cost such edge
    if c_e > 0 then
        add e to H
return H
```

2.2 Matroids

Let's re-examine the problem of maximum weight forest, if we let F denote the current edge set and we let $\mathcal{I}$ denote the set of all forests of G then we can rewrite the above pseudocode as:

Algorithm 3 Modified Kruskal's algorithm for finding maximum cost forests (rewrite)

```
Initiation: F = ∅
while ∃e ∈ E : F ∪ {e} ∈ I and c_e > 0 do
    e = highest cost such edge
    F = F ∪ {e}
return H
```

We can notice and abstract away several key properties of $\mathcal{I}$, the "universe" we are optimizing over. In particular we have $\mathcal{I} \subseteq 2^E$ and

- $\varnothing \in \mathcal{I}$
- $F' \subseteq F \in \mathcal{I} \implies F' \in \mathcal{I}$
- $\forall A \subseteq E$, every inclusion-wise maximal element of $\mathcal{I}$ contained in A has the same cardinality.

Definition Let S be a finite set, called a **ground set**, and let $\mathcal{I} \subseteq 2^S$. The set $M = (S, \mathcal{I})$ is called a **matroid** if

(M1) $\varnothing \in \mathcal{I}$

(M2) $F' \subseteq F \in \mathcal{I} \implies F' \in \mathcal{I}$

(M3) $\forall A \subseteq S$, every basis of A has the same cardinality

where a **basis** of $A \subseteq S$ is an inclusion wise maximal element of $\mathcal{I}$ contained in A

Example

- Let $G = (V, E)$, let $S = E$, and let $\mathcal{I}$ be the set of all forests of G, this is called the **Graphical/Forest Matroid**.
- Let $S = \{1, 2, \ldots, n\}$, let $r \in \{0, 1, \ldots, n\}$, and let $\mathcal{I}$ be the subsets of S with atmost r elements. $U_n^r = (S, \mathcal{I})$ is called the **Uniform Matroid of Rank** r.
- Let N be an $m \times n$ matrix of real numbers, let $S = \{1, 2, \ldots, n\}$, and let $\mathcal{I}$ be the set of all $A \subseteq S$ such that the columns indexed by A are linearly independent. This is called the **Linear Matroid** (notice in this case that a basis of A in the matroid sense corresponds to a basis of the span of the columns indexed by A in the linear algebra sense).

Elements of $\mathcal{I}$ are called **independent sets**. Similarly, subsets of S that are no in $\mathcal{I}$ are called **dependent sets**. Minimal dependent sets are called **circuits**[2]. If $M = (S, \mathcal{I})$ satisfies (M1) and (M2) then is is called an **independence system**[3]. Given an independence system A, we say that $r(A) \coloneqq \max\{|B| : B \subseteq A, B \in \mathcal{I}\}$ is the **rank of** A. We say that $r(S)$ is the **rank of** M, ie: the rank of the matroid or independence system. Finally, we define $\rho(A) \coloneqq \min\{|B| : B \text{ is a basis of } A\}$. Clearly we have that M is a matroid if and only if $\rho(A) = r(A)$ for all $A \subseteq S$.

Given the above, we can present an algorithm[4] for optimization over an arbitrary independence system $M = (S, \mathcal{I})$, ie: we find $A \in \mathcal{I}$ maximizing $c(A) = \sum_{e \in A} c_e$

Algorithm 4 Generic Greedy Algorithm

```
Initiation: F = ∅
while ∃e ∈ E : F ∪ {e} ∈ I and c_e > 0 do
    e = highest cost such edge
    F = F ∪ {e}
return H
```

Theorem (Rado '57, Edmonds '71) *Let M be a matroid and $c \in \mathbb{R}_+^S$, then a greedy algorithm finds the maximum weight independent set.*

Proof. This will follow from a later result. □

Theorem (Rado, Edmonds) *Let $M = (S, \mathcal{I})$ be an independence system, then greedy finds an optimal independent set for all $c \in \mathbb{R}_+^S$ if and only if M is a matroid.*

Proof. The reverse implication follows from the theorem above. Suppose that M is not a matroid, then there is some $A \subseteq S$ and some bases A_1, A_2 of A such that $|A_1| < |A_2|$. Define c by

$$c_e = \begin{cases} 1 & e \in A_1 \\ \frac{|A_1|}{|A_2|} + \varepsilon & e \in A_2 \setminus A_1 \\ 0 & \text{else} \end{cases}$$

for sufficiently small ε. Greedy will find some independent set with cost $c(A_1) = |A_1|$ but the optimal solution has cost atleast

$$c(A_2) = |A_1 \cap A_2| + |A_2 \setminus A_1| \cdot \left(\frac{|A_1|}{|A_2|} + \varepsilon \right) = |A_1 \cap A_2| + |A_1| - |A_2 \cap A_1| \cdot \frac{|A_1|}{|A_2|} + \varepsilon \left(|A_2 \setminus A_1| \right) > |A_1|$$

□

So clearly to guarantee optimality in all cases, the independence system over which we are optimizing must actually be a matroid. What if it's not, how bad can we do?

Theorem (Jenkyns '76) *Let $(S, \mathcal{I})$ be an independence system, let $g_{S,\mathcal{I}}$ be the total weight of the independent set found by the greedy algorithm, and let $OPT_{S,\mathcal{I}}$ be the optimal solution weight. Then*

$$g_{S,\mathcal{I}} \geq q_{S,\mathcal{I}} \cdot OPT_{S,\mathcal{I}}$$

where

$$q_{S,\mathcal{I}} \coloneqq \min_{\substack{A \subseteq S \\ r(A) \neq 0}} \frac{\rho(A)}{r(A)}$$

*The quantity $q_{S,\mathcal{I}}$ is called the **rank quotient**.*

[2] Circuits in graphical matroids are cycles.
[3] This is quite common, for example: stable sets
[4] This is exactly the re-written algorithm for max cost forest.

Proof. Let $S = \{e_1, \ldots, e_n\}$ where $c_1 \geq \cdots \geq c_n$ and let $S_j = \{e_1, \ldots, e_j\}$. Let $G \in \mathcal{I}$ be a solution obtained by the greedy algorithm and let $\sigma \in \mathcal{I}$ be an optimal solution. Define $G_j \coloneqq S_j \cap G$ and $\sigma_j \coloneqq S_j \cap \sigma$. Then we have (defining $c_{e_{j+1}} = 0$) that

$$c(G) = \sum_{j \in G} c_j = \sum_{j=1}^{n} c_{e_j} \left(|G_j| - |G_{j-1}|\right) = \sum_{j=1}^{n} |G_j| \left(c_{e_j} - c_{e_{j+1}}\right)$$

Since greedy computes a maximal independent set, it follows that G_j is a basis of S_j and therefore $|G_j| \geq \rho(S_j$ thus we get (defining $\delta_j = c_{e_j} - c_{e_{j+1}}$) that

$$c(G) \geq \sum_{j=1}^{n} |G_j| \delta_j \geq \sum_{j=1}^{n} \rho(S_j)\, \delta_j \geq \sum_{j=1}^{n} q_{S,\mathcal{I}} r(S_j)\, \delta_j \geq \sum_{j=1}^{n} q_{S,\mathcal{I}} |\sigma_j| \delta_j = c(\sigma)$$

□

Notice that if M is a matroid in the above theorem then $q_{S,\mathcal{I}} = 1$ and it follows that the greedy solution is optimal, proving the omitted direction in the result of Rado and Edmonds.

How fast are greedy algorithms? Assuming we define the size of the input[5] to be the size of S (ie: we implicitely define $\mathcal{I}$) the runtime is polynomial if and only if we can check $F \cup \{e\} \in \mathcal{I}$ in polynomial time.

Now that we understand matroids, we realize that checking if an independence system is a matroid usually comes down to verify (M3), which can sometimes be quite tough. We will now work towards several alternative characterizations of matroids that may be easier to verify in some settings.

Theorem *Let $M = (S, \mathcal{I})$ be an independence system, (M3) holds if and only if for all $X, Y \in \mathcal{I}$ with $|X| > |Y|$ there exists some $x \in X \setminus Y$ such that $Y \cup \{x\} \in \mathcal{I}$.*

Proof. The reverse direction is clear for if two bases have different sizes we could simply grow one. To see the forwards direction suppose (M3) holds and let $A = X \cup Y$, then clearly Y is not a basis of A (since $|Y| < |X|$). Thus there is some $x \in (X \cup Y) \setminus Y = X \setminus Y$ such that $Y \cup \{x\} \in \mathcal{I}$. □

Example As an example of how the above characterization can be useful, let $G = (V, E)$ be a graph and $W \subseteq V$ a stable set. Let $k_v \in \mathbb{Z}_+$ for all $v \in W$, let $S = E$ and let

$$\mathcal{I} = \{F \subseteq E : |\delta(v) \cap W| \leq k_v \text{ forall } v \in W\}$$

Clearly $(S, \mathcal{I})$ is an independence system. To see that it is a matroid, let $X, Y \subseteq E$ with $|X| > |Y|$ and let $W_y = \{v \in W : |\delta(v) \cap Y| = k_v\}$. Notice that

$$2|X| = \sum_{v \in W_y} |X \cap \delta(v)| + \sum_{v \in W \setminus W_y} |\delta(v) \cap X| + \sum_{v \notin W} |X \cap \delta(v)|$$

and

$$2|Y| = \sum_{v \in W_y} |Y \cap \delta(v)| + \sum_{v \in W \setminus W_y} |\delta(v) \cap Y| + \sum_{v \notin W} |Y \cap \delta(v)|$$

Since the first term in $2|Y|$ is $\sum_{v \in W_y} k_v$, it follows that atleast one of the second and third terms of $2|X|$ is greater then the corresponding term in $2|Y|$. That is, we can find some $x \in X \setminus Y$ such that $Y \cup \{x\} \in \mathcal{I}$ and it follows that $(S, \mathcal{I})$ is a matroid.

Notice that, if we don't necessarily know $\mathcal{I}$ but we know all the circuits of $\mathcal{I}$, it is possible to tell whether some set is an independent set (just verify it contains no circuits). A natural question to then ask is, when is $\mathcal{C} \subseteq 2^S$ the set of circuits of some matroid $M = (S, \mathcal{I})$?

Lemma *Let $M = (S, \mathcal{I})$ be a matroid, then for all $A \in \mathcal{I}$ and forall $e \in S$, the set $A \cup \{e\}$ contains at most one circuit.*

[5] If the size of the input is the size of $(S, \mathcal{I})$ then clearly every greedy algorithm is polynomial time and the question is not interesting.

Proof. Let A be the smallest set such that $A \in \mathcal{I}$ and there is some e such that $A \cup \{e\}$ has two distinct circuits. Clearly in this case we have $A \cup \{e\} = C_1 \cup C_2$. Since neither $C_1 \subseteq C_2$ or $C_2 \subseteq C_1$ we can pick $e_1 \in C_1 \setminus C_2$ and $e_2 \in C_2 \setminus C_1$. Consider $A' = (C_1 \cup C_2) \setminus \{e_1, e_2\}$. If A' has a circuit C then we must have $C \neq C_1, C_2$. It is also obvious that $(A \setminus \{e_1\}) \cup \{e\}$ contains both C_2 and C, hence A is not minimal and the result follows. □

Theorem *Let $\mathcal{C} \subseteq 2^S$, then $\mathcal{C}$ is the set of circuits of some matroid if and only if*

(Ci) $\varnothing \notin \mathcal{C}$

(Cii) *If $C_1, C_2 \in \mathcal{C}$ and $C_1 \subseteq C_2$ then $C_1 = C_2$*

(Ciii) *If $C_1, C_2 \in \mathcal{C}$ and $C_1 \neq C_2$ and $e \in C_1 \cap C_2$ then there is some $C \in \mathcal{C}$ with $C \subseteq (C_1 \cup C_2) \setminus \{e\}$.*

Proof.

(i) Suppose first that $\mathcal{C}$ is the set of circuits of a matroid. Clearly (C1) and (C2) hold, so suppose that (C3) is violated. Then there are circuits distinct C_1, C_2 and some edge $e \in S$ such that $(C_1 \cup C_2) \setminus \{e\} \in \mathcal{I}$. This contradicts the prior lemma though, because adding e introduces two circuits.

(ii) Suppose now that (C1), (C2), and (C3) hold. Define $\mathcal{I} = \{A \subseteq S : \nexists_{C \in \mathcal{C}} : C \subseteq A\}$ and let $M = (S, \mathcal{I})$. Clearly (M1) and (M2) are satisfied. Suppose that (M3) is false, then there are bases A_1, A_2 of $A \subseteq S$ with $|A_1| < |A_2|$. Pick A_1, A_2 such that $|A_1 \cap A_2|$ is maximized and let $e \in A_1 \setminus A_2$. Then $A_2 \cup \{e\}$ contains a unique circuit C. Choose $f \in C \setminus A_1$, then $A_3 \coloneqq (A_2 \cup \{e\}) \setminus \{f\} \in \mathcal{I}$. But this is a contradiction since $|A_3 \cap A_1| > |A_2 \cap A_1|$.

□

Alternatively, we can instead specify a matroid by its bases. If we know the set $\mathcal{B}$ of bases then we have that $A \in \mathcal{I} \iff A \subseteq B$ for some $B \in \mathcal{B}$. This begs the similar question, when is $\mathcal{B} \subseteq 2^S$ the set of bases of some matroid.

Theorem *Let $\mathcal{B} \subseteq 2^S$, then $\mathcal{B}$ is the set of bases of some matroid $(S, \mathcal{I})$ if and only if*

(i) $\mathcal{B} \neq \varnothing$

(ii) *For any $B_1, B_2 \in \mathcal{B}$ and $x \in B_1 \setminus B_2$ there is some $y \in B_2 \setminus B_1$ such that $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$.*

Theorem *Let $\mathcal{B} \subseteq 2^S$, then $\mathcal{B}$ is the set of bases of some matroid $(S, \mathcal{I})$ if and only if*

(i) $\mathcal{B} \neq \varnothing$

(ii) *For any $B_1, B_2 \in \mathcal{B}$ and $y \in B_2 \setminus B_1$ there is some $x \in B_1 \setminus B_2$ such that $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$.*

Notice that these are indeed different since in the first we choose an element to remove first and in the second we choose an element to add first.