On the Characterization of Combinatorial Optimization Problems Soluble Using Greedy Strategies

Siddharth Mahendraker
siddharth.mahen@gmail.com

Candidate Number: 000652 – 0016
School: International School of Helsinki
Word Count: 3900

November 14, 2013

Abstract

In this paper, we examine the necessary and sufficient conditions required for a greedy algorithm strategy to be successful for an arbitrary combinatorial optimization problem. This is done by studying matroids and their properties. We show that matroids are intimately connected to the solution of combinatorial optimization problems using greedy strategies. Finally, we conclude that a greedy algorithm is successful for a combinatorial optimization problem if and only if the optimization problem has matroid structure.

Key terms: greedy algorithms, matroid theory, combinatorial optimization.

Contents

1 Introduction

Combinatorial optimization is a mathematical field concerned with efficiently finding certain optimal objects among a large set of objects. A famous example of a combinatorial optimization problem is the travelling salesman problem, or TSP. Assume a salesman has a list of cities he wants to visit, and knows the distance between various pairs of cities. The TSP asks: What is the shortest path which takes the salesman through each city once and back to his starting point? Here, the optimal object is a shortest path which passes through each city once and returns the salesman to his point of departure, and the set of objects is the set of all paths which can be taken. To see why this problem could be difficult, assume that every city is connected to every other city and we want to check all the paths to find the shortest one. If we fix a city of departure, there are $n-1$ cities left to go to, where n is the total number of cities. Each of these choices represents a different path the salesman could take. In the next city, there are $n-2$ cities left to go to, and so on. It turns out, given n cities there are $(n-1)!/2$ unique paths which reach every city and return the salesman to his point of departure. Even for relatively small values of n, the number of possible paths is enormous; far larger than anything even a computer could hope to check. Consider for example the choice of $n = 64$. In this case, there are over 10^{86} possible paths to check! More paths than there are atoms in the universe! Using combinatorial optimization techniques, however, we can significantly reduce the amount of work needed to find an optimal path. Indeed, to date, instances of the TSP have been solved with tens of thousands of cities.

One particular strategy often used in combinatorial optimization is the greedy optimization strategy, or greedy algorithm. The greedy algorithm attempts to find an optimal object by making successive locally optimal decisions, in the hopes that this will lead to a globally optimal solution. For example, a greedy algorithm for the TSP could work as follows: Fix the city of departure. To reach another city, pick the shortest route out of the current city which leads to an unvisited city. Repeat until you return to the city of departure. Unfortunately, greedy strategies are often non-optimal. In the case of the TSP, this particular greedy algorithm will not always yield the optimal path, and in certain situations, can in fact give the worst possible solution. More generally, it is known that greedy algorithms are far from optimal for most instances of the TSP [3].

However, for certain combinatorial optimization problems, greedy algorithms can be very useful. If they can be proven to be correct, greedy algorithms are often far simpler to implement than other combinatorial optimization strategies, such as dynamic programming (see [1, pp. 378–387] for more information). Furthermore, they often consume less resources when implemented on a computer and thus run faster and more efficiently. Therefore, knowing when a greedy algorithm can be applied is very useful.

In this paper, we examine the necessary and sufficient conditions required for a greedy algorithm strategy to be successful for a given combinatorial optimization problem. This is done through the study of matroids and their properties, which as we shall soon see, lead naturally to greedy algorithm strategies.

The remainder of this paper will proceed as follows. First, we give an example of a successful greedy algorithm by examining the minimum spanning tree problem and Kruskal's algorithm. Next we introduce matroids and discuss their various forms and properties. Then, we link certain properties of matroids to solutions to a general class of combinatorial optimization problems. Finally, we show that these solutions can be obtained through greedy strategies.

This paper assumes some familiarity with graph theory and basic linear algebra. Throughout the paper, we use standard notation from graph theory. For details, see [4, pp. 1–6]. Note that unless otherwise indicated, all proofs in this paper are original.

2 The MSTP and Kruskal's Algorithm

Recall that a *spanning tree* T of a graph G is a connected, acyclic subgraph of G which contains all the vertices of G. Given a weighted graph $G = (V, E, w)$, the minimum spanning tree problem (MSTP) asks us to find a spanning tree T of G, such that T has minimal weight.

Unlike the TSP, the MSTP can be solved with a simple and elegant algorithm, due to Kruskal [1, pp. 631–633].

Definition 2.1 (Kruskal's Algorithm). Given a connected graph $G = (V, E, w)$, the minimum spanning tree T^* of G can be computed as follows:

1. Begin with an empty set $T := \emptyset$.
2. Until $|T| = |V| - 1$, repeat steps 3 and 4 below.
3. Select a minimum weight edge e in $E - T$, breaking ties arbitrarily.
4. If $T \cup e$ is acyclic, set $T := T \cup e$. Otherwise discard the edge e and never consider it again.
5. Output a minimum spanning tree $T^* = (V, T, w)$ and stop.

Note that due to our frequent addition and removal of single elements from sets, we abbreviate $X \cup \{e\}$ to $X \cup e$ and $X - \{e\}$ to $X - e$.

Clearly, Kruskal's algorithm is greedy. At every step, the algorithm selects a minimum weight edge in G, and attempts to add it to the edge set of the output spanning tree.

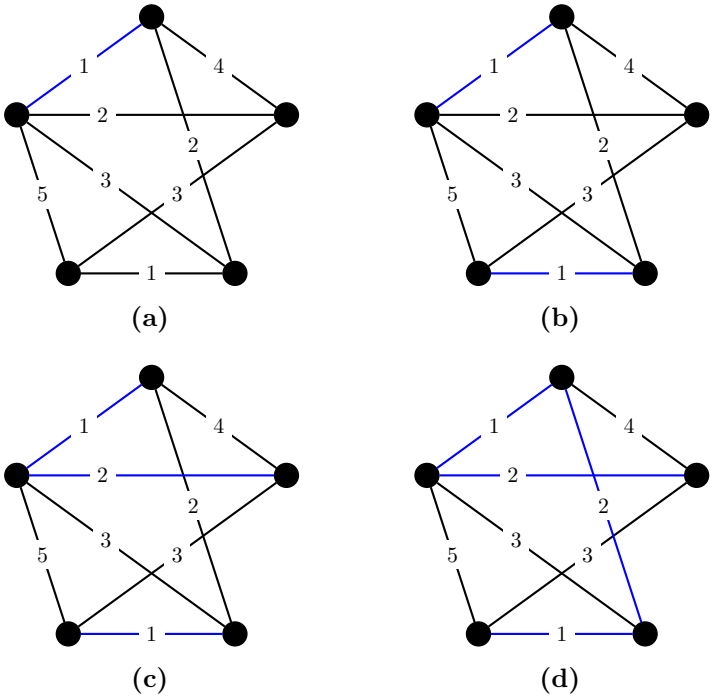

Figure 1: The execution of Kruskal's algorithm on a weighted graph G. Blue edges are part of the final MST of G generated by the algorithm.

Definition 2.2. An algorithm is called a *greedy algorithm* if it makes a locally optimal decision at each step.

Kruskal's algorithm shows how efficient greedy algorithms can be, given the correct problem structure. The algorithm works because the MSTP has two properties which immediately allow us to identify edges which are part of an MST of G, and edges which are not [2, pp. 10–11]. Let us call a partition of G into two disjoint sets A and B a cut (A, B) of G. The first property says that an edge $e \in E$ is part of the MST of G if and only e is the lightest edge across some cut of G. This is called the cut property. The second property says that an edge $e \in E$ is not part of the MST of G if and only if e is the heaviest edge in some cycle of G. This is called the cycle property. Because Kruskal's algorithm adds edges from lightest to heaviest, if an edge $e \in E - T$ is being considered by the algorithm it must be the lightest edge across some cut, unless it creates a cycle in $T^* = (V, T, w)$, in which case it is the heaviest edge in some cycle. Since one of these possibilities always holds, Kruskal's algorithm always returns a MST of G. See Figure 1 for an example of Kruskal's algorithm in action.

In the next section we introduce matroids, which allow us to reason about a large class of optimization problems with useful properties similar to those of the MSTP mentioned above. We will revisit Kruskal's algorithm

in subsequent sections and give a more rigorous proof of correctness using matroid theory.

3 Matroid Theory

Intuitively, matroids consolidate various ideas of dependence which arises naturally in certain fields of mathematics, particularly linear algebra and graph theory. In linear algebra, a set of vectors U of a vector space V is called dependent if there is a linear of combination of these vectors which sum to 0. In graph theory, a set of edges S of a graph G is called dependent if the edges of S contain a cycle.

An interesting property of matroids is that there are several equivalent axiom systems which can be used to reason about them. In this essay, we will only explore two such axiom systems: the independent set system, and the bases system. For a more thorough overview of matroid theory, see [4].

Definition 3.1. A *matroid* M is an ordered pair $(E, \mathcal{I})$, where E is a finite set and $\mathcal{I}$ is a collection of subsets of E satisfying the following three conditions:

(I1) $\emptyset$ is in $\mathcal{I}$.

(I2) If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$.

(I3) If I_1 and I_2 are in $\mathcal{I}$ and $|I_1| < |I_2|$, then there exists an element $e \in I_2 - I_1$ such that $I_1 \cup e \in \mathcal{I}$.

We call E the *ground set* of a matroid $M = (E, \mathcal{I})$ and we call the members of $\mathcal{I}$ the *independent sets* of M. A subset S of E which is not in $\mathcal{I}$ is called a *dependent set*. We also call (I2) the *hereditary property* and (I3) the *exchange property*.

From this definition, we can already intuitively see that matroids encapsulate lots of structure which is important when solving combinatorial optimization problems. Specifically, (I2) and (I3) are reminiscent of the overlapping subproblem structure and optimal substructure exploited by dynamic programming algorithms [1]. These conditions imply that we are guaranteed that all subsets of an independent set are themselves independent, and that we can always build larger independent sets from smaller ones[1].

Proposition 3.2. *Let $G = (V, E)$ be a graph and let $\mathcal{I}$ be the set of subsets of E which do not contain a cycle. Then $M = (E, \mathcal{I})$ is a matroid.*

Proof. By definition, $\emptyset \in \mathcal{I}$ and thus $\mathcal{I}$ satisfies (I1). It is evident that if $H \in \mathcal{I}$ and $K \subseteq H$, then $K \in \mathcal{I}$. Thus $\mathcal{I}$ satisfies (I2). We show that $\mathcal{I}$

[1] Until we build a maximal independent set, at which point no more elements can be exchanged via (I3).

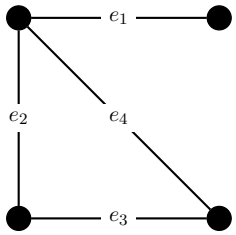

Figure 2: Consider the cycle matroid $M(G)$ of the graph above. The collection of independent sets of $M(G)$ are exactly the subsets of the edge set of G which do not contain any cycles. Thus $\mathcal{I} = \{e_1, e_2, e_3, e_4, e_5, \{e_1, e_2\}, \{e_1, e_3\}, \{e_1, e_4\}, \{e_2, e_3\}, \{e_2, e_4\}, \{e_3, e_4\}, \{e_1, e_2, e_3\}, \{e_1, e_2, e_4\}, \{e_1, e_3, e_4\}\}$, where $\mathcal{I}$ is the independent set of $M(G)$. Note that $\{e_2, e_3, e_4\} \notin \mathcal{I}$ as the edges e_2, e_3 and e_4 form a cycle in G.

satisfies (I3) by contradiction. Suppose I_1 and I_2 are elements of $\mathcal{I}$, and $|I_1| < |I_2|$. Assume that for all $e \in I_2 - I_1$, the set $I_1 \cup e \notin \mathcal{I}$. This means that the set $I_1 \cup e$ contains a cycle for all e. Since I_1 is itself acyclic, there is only one cycle in $I_1 \cup e$, and this cycle must contain e. Now, consider $I_1 \cup I_2$. This set has exactly $|I_2 - I_1|$ cycles, one for each $e \in I_2 - I_1$. Let I_3 be a maximal independent set in $I_1 \cup I_2$, obtained by removing an edge from each of the $|I_2 - I_1|$ cycles in $I_1 \cup I_2$. Now,

$$\begin{aligned} |I_3| &= |I_1 \cup I_2| - |I_2 - I_1| \\ &= |I_1| + |I_2| - |I_1 \cap I_2| - |I_2 - I_1| \\ &= |I_1| + |I_2| - |I_1 \cap I_2| - (|I_2| - |I_1 \cap I_2|) \\ &= |I_1|. \end{aligned}$$

It follows that $|I_3| < |I_2|$, contradicting the maximality of I_3. Therefore, (I3) holds, and M is indeed a matroid. □

A matroid derived from a graph G is called a *cycle matroid* and is denoted $M(G)$. See Figure 2 for an example of a cycle matroid.

Consider a collection $\mathcal{B}$ of maximal independent sets of a matroid M. Given the ground set E of M we can easily reconstruct the independent sets of M as follows: The set X is in $\mathcal{I}$ if and only if $X \subseteq B$ for some $B \in \mathcal{B}$. We call a maximal independent set of a matroid a *base* or *basis*.

From our discussion of MSTs and Proposition 3.2 we can deduce that any spanning tree of a graph G is a basis of the cycle matroid $M(G)$. Notice that this idea of a basis coincides with our idea of a basis from linear algebra. Like bases of a vector space, all spanning trees are equicardinal. Furthermore, if one "fills in the cycles" we can imagine that a spanning tree "spans" the graph, just as the span of a basis spans a vector space.

Proposition 3.3. *Let $\mathcal{B}$ be a collection of bases of a matroid $M = (E, \mathcal{I})$. Then the elements of $\mathcal{B}$ are equicardinal.*

Proof. Let B_1 and B_2 be elements of $\mathcal{B}$. Suppose $|B_1| < |B_2|$. Then by (I3), there exists a element $e \in B_2 - B_1$ such that $B_1 \cup e \in \mathcal{I}$. However, this contradicts the maximality of B_1. Therefore, $|B_1| \geq |B_2|$. Similarly, $|B_2| \geq |B_1|$. Thus $|B_1| = |B_2|$. Since B_1 and B_2 were chosen arbitrarily, all elements of $\mathcal{B}$ are equicardinal. □

The bases of a matroid M also have a very attractive property, similar to the exchange property for independent sets, called the *basis exchange property.*

Proposition 3.4. *Let $\mathcal{B}$ be a collection of bases of a matroid $M = (E, \mathcal{I})$. If B_1 and B_2 are elements of $\mathcal{B}$ and $x \in B_1 - B_2$, then there exists an element $y \in B_2 - B_1$ such that $(B_1 - x) \cup y \in \mathcal{B}$.*

Proof. Consider $B_1 - x \subset B_1$. Then $B_1 - x$ is independent by (I2). Now, $|B_1 - x| < |B_2|$ by Proposition 3.3. Thus, there exists an element $y \in B_2 - B_1$ such that $(B_1 - x) \cup y \in \mathcal{I}$ by (I3). Since $|(B_1 - x) \cup y| = |B_1|$ and all bases are equicardinal, there cannot exist an independent set containing $(B_1 - x) \cup y$. Thus $(B_1 - x) \cup y \in \mathcal{B}$. □

In the next section, we link the bases of matroids to solutions to a general class of combinatorial optimization problems. We then explore solutions to to these problems using greedy strategies.

4 Greedy Algorithms and Matroids

Let us now define more precisely what it means for a problem to be a combinatorial optimization problem. As we mentioned earlier, combinatorial optimization problems deal with finding an optimal object among a large set of objects.

Definition 4.1. Let $\mathcal{I}$ be a collection of subsets of a set E. Define w as a function from E to $\mathbb{R}^{\geq 0}$, where

$$w(X) = \sum_{x \in X} w(x)$$

for all $X \subseteq E$. We call $w(X)$ the *weight* of X.

An *optimization problem* for a triple $(E, \mathcal{I}, w)$ is the problem of finding a maximal element S of $\mathcal{I}$ such that S has minimal weight.

We call a maximal element with minimum weight a *solution* to an optimization problem. If an algorithm always finds a solution to a combinatorial optimization problem P, we say the algorithm solves P.

It is easy to see how this definition encapsulates our intuition about combinatorial optimization problems. Because $\mathcal{I}$ is a collection of subsets

of a set, it is plausible that $\mathcal{I}$ could become exponential in E, and thus very large. Specifically, in the worst case if $\mathcal{I} = \mathcal{P}(E)$, the power set of E, then $|\mathcal{I}| = 2^{|E|}$, which grows exponentially in the size of E. This means that finding a maximal element S of I with minimal weight could become exponentially difficult.

For example, the TSP corresponds to the optimization problem where $\mathcal{I}$ is a collection of subsets of the edge set of a graph $G = (V, E, w)$ which do not form cycles or are Hamiltonian cycles[2], and w is simply the weight function of G. In this optimization problem, a maximal element of $\mathcal{I}$ would be a Hamiltonian cycle, and a solution to $(E, \mathcal{I}, w)$ would be Hamiltonian cycle of minimal weight.

[3]

Given this definition of a combinatorial optimization problem, we can now define a general greedy algorithm which attempts to find a solution to arbitrary combinatorial optimization problems.

Definition 4.2 (Generalized Greedy Algorithm). Given a combinatorial optimization problem $P = (E, \mathcal{I}, w)$, a potential solution to P can be computed as follows:

1. Begin with an empty set $X := \emptyset$.
2. Select a minimum weight element $e \in E - X$, such that $X \cup e \in \mathcal{I}$. If such an element exists, set $X := X \cup e$. Otherwise, output X and stop.
3. Repeat the previous step until the algorithm stops.

We abbreviate the generalized greedy algorithm to GGA.

Clearly, the GGA is greedy. At every step, the algorithm makes a locally optimal decision by selecting an element of minimum weight. Since E is finite and the algorithm adds one element in $E - X$ to X at every step, the algorithm must stop in a finite number of steps. Furthermore, because X is only augmented when $X \cup e$ is an element of $\mathcal{I}$, the output X is also an element of $\mathcal{I}$. Thus, it is plausible that the GGA solves some combinatorial optimization problems.

Incredibly, it can be shown that if $(E, \mathcal{I})$ is a matroid, then the solution to the combinatorial problem $(E, \mathcal{I}, w)$ is given by the output of the generalized greedy algorithm! This follows from the fact that X can always be augmented using the exchange property until it becomes a basis in $\mathcal{I}$, and that the algorithm only selects minimum weight elements to add to X.

Theorem 4.3. *Let $M = (E, \mathcal{I})$ be a matroid, and let $P = (E, \mathcal{I}, w)$ be a combinatorial optimization problem. Then the generalized greedy algorithm solves P.*

[3] A Hamiltonian cycle of a graph G is a cycle of G which touches every node in the graph exactly once.

Proof. Let X be the output of the GGA given P. We show that X is a maximal element of $\mathcal{I}$. At every step of the algorithm, $|X| < |E|$. If $|X| = |E|$, then the algorithm must terminate, as $E - X$ would be empty. Furthermore, at every step of the algorithm, X is independent. This is because X is initialized to the empty set, and at every subsequent step X is augmented such that X remains in $\mathcal{I}$. Therefore, by (I3), we can always increase the cardinality of X by one in each step of the algorithm. Suppose the bases of M have cardinality r. Since the cardinality of X is initially 0, and at every step we increase the cardinality of X by one, X will be a basis of M after r steps. Thus, the output of the GGA is a maximal element of $\mathcal{I}$.

Now, let Y be a maximal element of $\mathcal{I}$ with minimal weight, i.e. a solution to P. We show that X has minimal weight. Since both X and Y are bases of $\mathcal{I}$, we can exchange an element $x \in X - Y$ with an element $y \in Y - X$ to form a new basis $(X - x) \cup y$ by Proposition 3.4. This new basis has weight lower than the weight of X if and only if $w(y) < w(x)$. We show this to be false. Let $X' \subset X$ be the elements of X chosen before x was chosen. Because the GGA chose x, this implies x had lower weight than any other element in $E - X'$. However, $Y - X \subset Y - X' \subset E - X'$. Thus, as $y \in Y - X$ it follows that $w(x) \leq w(y)$. Therefore, $w(X) \leq w((X - x) \cup y)$. Let $X_1 = (X - x) \cup y$. We can form a chain of alternate bases X_{i+1} of $\mathcal{I}$ by exchanging elements between X_i and Y, where $i \in \{1, 2, \ldots, k\}$. After every exchange, we can use a similar argument to show that $w(X_i) \leq w(X_{i+1})$. This process will stop when we have exchanged all the elements of $X - Y$ and $Y - X$, at which point $X_k = Y$. Now, since $w(X_i) \leq w(X_{i+1})$ at every link in the chain, this implies

$$w(X) \leq w(X_1) \leq w(X_2) \leq \cdots \leq w(X_{k-1}) \leq w(Y).$$

However, this contradicts the minimality of Y, and thus an equality must hold throughout. Therefore $w(X) = w(Y)$, and X is indeed of minimal weight. Since X is a maximal element of $\mathcal{I}$ with minimal weight it is a solution to P, and the GGA solves P. □

Let us now reconsider Kruskal's algorithm and the MSTP in the context of combinatorial optimization problems. Fix a connected graph $G = (V, E, w)$. We can think of the MSTP as an optimization problem $(E, \mathcal{I}, w)$, where $\mathcal{I}$ is a collection of acyclic subsets of the edge set E of G, and w is simply the weight function of G. This problem also has a natural matroid structure, given by the cycle matroid $M(G)$ of G. Let us now compare Kruskal's algorithm to the GGA. If we can show that Kruskal's algorithm is somehow a special case of the GGA, its correctness comes for free by the previous theorem. For every edge e added to the edge set T, Kruskal's algorithm checks to make sure $T \cup e$ is acyclic. This corresponds exactly to the independence check, $X \cup e \in \mathcal{I}$, executed by the GGA, where $\mathcal{I}$ is the independent set of $M(G)$. It is simple to see that the other aspects of the algorithm are also essentially equivalent.

Thus, we can view Kruskal's algorithm as a special case of the GGA for cycle matroids. It follows from Theorem 4.3 that Kruskal's algorithm solves the MSTP for G.

Corollary 4.4. *Let $G = (V, E, w)$ be a connected graph. Then Kruskal's algorithm solves the MSTP for G.*

Definition 4.5. We say a combinatorial optimization problem $P = (E, \mathcal{I}, w)$ has *matroid structure* if $M = (E, \mathcal{I})$ is a matroid.

So far, we have shown that the GGA solves all combinatorial optimization problems with an underlying matroid structure. It is natural to now ask what other combinatorial optimization problems, if any, the GGA is capable of solving. Perhaps counterintuitively, it can be shown that the GGA solves a combinatorial optimization problem if and only if the problem has a matroid structure.

Theorem 4.6. *Let $P = (E, \mathcal{I}, w)$ be a combinatorial optimization problem. Then P has matroid structure if and only if P satisfies the following conditions:*

(I1) *$\emptyset$ is in $\mathcal{I}$.*

(I2) *If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$.*

(G) *The generalized greedy algorithm solves P for all weight functions $w : E \to \mathbb{R}^{\geq 0}$.*

Proof. Assume P has matroid structure. Then P satisfies (I1) and (I2) by definition. By Theorem 4.3, P also satisfies (G).

Conversely, assume P satisfies (I1), (I2) and (G). We show by contradiction that P satisfies (I3) also, and thus has matroid structure. Let I_1 and I_2 be elements of $\mathcal{I}$ such that $|I_1| < |I_2|$ and for all x in $I_2 - I_1$, $I_1 \cup x \notin \mathcal{I}$. We proceed by constructing a weight function $w : E \to \mathbb{R}^{\geq 0}$ under which the GGA fails to solve P.

Let r be the cardinality of a maximal element in $\mathcal{I}$ and let $w : E \to \mathbb{R}^{\geq 0}$ be a weight function defined by

$$w(x) = \begin{cases} 0 & \text{if } x \in I_1 \\ \dfrac{1}{2|I_2 - I_1|} & \text{if } x \in I_2 - I_1 \\ 1 & \text{if } x \in E - (I_1 \cup I_2) \end{cases}$$

Now, let S be the solution to P returned by the GGA. Given the weight function w above, the GGA would first add all of I_1 to S. Next, the GAA would attempt to add the elements of I_2 to S, however, because $I_1 \cup x$ is dependent for all x in I_2, the GGA would be forced to add the elements

of $E - (I_1 \cup I_2)$ to S instead. The solution returned by the GGA would therefore have weight

$$w(S) = 0(|I_1|) + 1(r - |I_1|) = r - |I_1|.$$

We now show that there exists a maximal element S^* of $\mathcal{I}$ with weight lower than that of S. Let $S^* = (I_1 \cap I_2) \cup (I_2 - I_1) \cup X$, where X is an arbitrary subset of $E - (I_1 \cup I_2)$ and $|X| = r - |I_2|$. The potential solution S^* has weight

$$\begin{aligned} w(S^*) &= 0(|I_1 \cap I_2|) + \left(\frac{1}{2|I_2 - I_1|}\right)|I_2 - I_1| + 1(r - |I_2|) \\ &= 1/2 + r - |I_2| \end{aligned}$$

Observe that because $|I_1|$ and $|I_2|$ can only take discrete values, $|I_2| - |I_1| \geq 1$. Now, we find that the difference $w(S) - w(S^*)$,

$$\begin{aligned} w(S) - w(S^*) &= (r - |I_1|) - (1/2 + r - |I_2|) \\ &= |I_2| - |I_1| - 1/2 \\ &\geq 1/2 \end{aligned}$$

is bound from below by $1/2$. Thus $w(S^*) < w(S)$.

However, this contradicts the minimality of S as per condition (G). Therefore, (I3) holds and P has matroid structure. □

This result is interesting for several reasons.

First, it tells us something important about how much structure a black-box combinatorial optimization problems must have to be soluble. If we think of the generalized greedy algorithm as the simplest general method of solving combinatorial optimization problems, we see informally that it is in general very difficult to solve combinatorial optimization problems without knowing more about their internal structure. Indeed, unless we require that the problems have matroid structure, even the simplest general method will not be able to solve them.

Second, this result allows us to quickly identify combinatorial optimization problems which can be solved using greedy strategies. When faced with a novel combinatorial optimization problem, this allows us to quickly determine whether it is worth pursuing more complex computational solutions or not.

From an applied perspective, this result is important because it establishes that the generalized greedy algorithm solves every combinatorial optimization problem with matroid structure, and thus, one does not need to optimize several algorithms to solve different kinds of combinatorial optimization problems quickly.

Finally, we see that Theorem 4.6 provides a complete characterization of all combinatorial optimization problems soluble using greedy strategies. Namely, a combinatorial optimization problem is soluble if and only if it has a matroid structure.

5 Conclusions and Further Discussion

In conclusion, we have completely characterized the combinatorial optimization problems which are soluble using greedy algorithm strategies. We have proven that greedy algorithm strategies solve combinatorial optimization problems if and only if these problems possess a matroid structure.

Of course, there are many more features of greedy strategies that we could explore in combinatorial optimization which are interesting from a combinatorial perspective as well as an applied perspective.

One important question we can ask, for example, is how well the generalized greedy algorithm approximates the true solution to combinatorial optimization problems without matroid structure. For example, it is known that the greedy algorithm provides a fairly good solution to the knapsack problem in the general case.

The generalized greedy algorithm is also often used to establish good initial conditions before a more specific algorithm is used to solve a combinatorial optimization problem. For example in graph colouring, a greedy strategy is often used to establish initial conditions before local search techniques are used.

For a more in depth overview of greedy algorithm theory and matroid theory, we refer the reader to [1] and [4].

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 55 Hayward Street, Cambridge, Massachusetts, third edition, 2009.

[2] Jason Eisner. State-of-the-Art Algorithms for Minimum Spanning Trees. Technical report, University of Pennsylvania, 1997.

[3] G. Gutin, A. Yeo, and A. Zverovich. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for TSP. *Discrete Applied Mathematics*, (117):81–86, 2002.

[4] James G. Oxley. *Matroid Theory*. Oxford University Press, New York, 1992.