# The Maximum Traveling Salesman Problem with Submodular Rewards

Syed Talha Jawaid    Stephen L. Smith

***Abstract*— In this paper we extend the classic problem of finding the maximum weight Hamiltonian cycle in a graph to the case where the reward is a submodular function of the edges. We propose a greedy algorithm and a 2-matching based algorithm, and we show that they have approximation factors of $\frac{1}{2+\kappa}$ and $\max\left\{\frac{2}{3(2+\kappa)}, \frac{2}{3}(1-\kappa)\right\}$ respectively, where $\kappa$ is the curvature of the function. Both algorithms run in time cubic to the number of vertices. We provide simulation results to empirically evaluate the performance of the algorithms.**

## I. Introduction

The maximum weight Hamiltonian cycle is a classic problem in combinatorial optimization. It consists of finding a cycle in a graph that visits all the vertices and maximizes the sum of the weights on the edges traversed. Also referred to as the max-TSP, the problem is NP-hard; however, a number of approximation schemes have been developed. In [1] four simple approximation algorithms are analyzed. The authors show that greedy, best-neighbour, and 2-interchange heuristics all give a $\frac{1}{2}$ approximation to the optimal tour. They also show that a 2-matching based heuristic, gives a $\frac{2}{3}$ approximation. Serdyukov's algorithm [2] — which combines a cycle cover and a matching to compute a tour — can give a $\frac{3}{4}$ approximation. In this paper we look at extending the max-TSP problem to the case of submodular rewards.

The main property of a submodular function is that of decreasing marginal value, i.e., adding an element to a set will result in a larger reward than adding it to a superset. One application in which submodular functions appear is in making sensor measurements in an environment. For example, in [3] the authors consider the problem of placing static sensors over a region for optimal sensing. This can be represented quantitatively by using the concept of mutual information of a set of sensors, which is a submodular function. Other areas where submodular functions come up include viral marketing, active learning [4] and AdWords assignment [5]. A different form of sensing involves using mobile sensors (robots) for persistent monitoring of a large environment [6]. The metric used to determine the quality of the sensing is usually submodular in nature. Due to the persistent operation, it is desirable to have a closed walk or a tour over which the sensing robot travels. This motivates the problem of finding a tour that has the maximum reward.

Maximizing a monotone submodular function over an independence system constraint is known to be NP-hard. Approximation bounds exist for optimizing over a matroid [7], and more generally, $p$-systems [8] as well as for the class of $k$-exchange systems [9]. Some bounds that include the dependence on curvature are evaluated in [10].

*Contributions:* The contributions of this paper are to present and analyze two simple algorithms for constructing a maximum reward tour on a graph. The metric used in evaluating the "reward" of a particular tour is a positive monotone submodular function of the edges. We frame this problem as an optimization over an independence system constraint and present two approximation algorithms. The first method is greedy and gives a $\frac{1}{2+\kappa}$ approximation. The second method creates a 2-matching and then turns it into a tour. This gives a $\max\left\{\frac{2}{3(2+\kappa)}, \frac{2}{3}(1-\kappa)\right\}$ worst case approximation where $\kappa$ is the curvature of the submodular function. Both techniques require $O(|V|^3)$ value oracle calls to the submodular function. The algorithms are also extended to directed graphs. To obtain these results, we present a new bound for the greedy algorithm as a function of curvature.

In an extended version of this paper [11] we present some preliminary results for the case of a multi-objective optimization consisting of submodular (sensing) rewards along with modular (travel) costs.

*Organization:* The organization of this paper is as follows. In Section II we review independence systems and submodularity. In Section III we formalize our problem. In Section IV we analyze a simple greedy strategy. In Section V we present and analyze a strategy to construct a solution using a matching. In Section VI we look at how the presented algorithms extend to the case where the graph is directed. Finally, simulation results are provided in Section VII.

## II. Preliminaries

Here we present preliminary concepts and give a brief summary of results on combinatorial optimization problems over independence systems.

### A. Independence systems

Combinatorial optimization problems can often be formulated as the maximization or minimization over a set system $(E, \mathcal{F})$ of a cost function $f : \mathcal{F} \to \mathbb{R}$, where $E$ is the base set of all elements and $\mathcal{F} \subseteq 2^E$. An **independence system** is a set system that is closed under subsets (i.e., if $A \in \mathcal{F}$, then $B \subseteq A \implies B \in \mathcal{F}$). Sets in $\mathcal{F}$ are referred to as "independent sets". Maximal independent sets (i.e., $\{A \in \mathcal{F} | A \cup \{x\} \notin \mathcal{F}, \forall x \in E \setminus A\}$) are the **bases**.

**Definition II.1** ($p$-system)**.** Given an independence system $S = (E, \mathcal{F})$. For any $A \subseteq E$, let $U(A)$ and $L(A)$ be the sizes of the maximum and minimum cardinality bases of $A$ respectively. $S$ is a $p$-system if $U(A) \leq pL(A), \forall A \subseteq E$.

This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (stjawaid@uwaterloo.ca; stephen.smith@uwaterloo.ca)

Two classes of $p$-systems are called $p$-extendible systems and matroids. A matroid is a 1-extendible system and any $p$-extendible system is a $p$-system.

### B. Submodularity

Without any additional structure on the set-function $f$, the optimization problem is generally intractable. However, a fairly general class of cost functions for which approximation algorithms exist is the class of submodular set functions.

**Definition II.2** (Submodularity)**.** Let $N$ be a finite set. A function $f : 2^N \to \mathbb{R}$ is submodular if

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T), \quad \forall S, T \subseteq N.$$

Submodular functions satisfy the property of *diminishing marginal returns*. That is, the contribution of any element $x$ to the total value of a set decreases as the set gets bigger. More formally, let $\Delta_A(B) := f(A \cup B) - f(A)$. Then,

$$\Delta_A(x) \geq \Delta_B(x), \quad \forall A \subseteq B \subseteq N.$$

Since the domain of $f$ is $2^N$, there are an exponential number of possible values for the set function. We will assume that $f(S)$ for any $S \subseteq N$ is determined by a black box function. This *value oracle* is assumed to run in polynomial time in the size of the input set.

The class of submodular functions is fairly broad and includes linear functions. One way to measure the degree of submodularity is the *curvature*. A submodular function has a curvature of $\kappa \in [0, 1]$ if for any $A \subset N$ and $e \in N \setminus A$

$$\Delta_A(e) \geq (1 - \kappa) f(e). \tag{1}$$

In other words, the minimum possible marginal benefit of any element $e$ is within a factor of $(1 - \kappa)$ of its maximum possible benefit.

We formulate a slightly stronger notion of the curvature—the independence system curvature, $\kappa_I$—by taking the independence system into account. In this case, (1) need only be satisfied for any $A \in \mathcal{F}$ and $e \in N \setminus A, A \cup \{e\} \in \mathcal{F}$. This value of curvature will be lower than the one obtained by the standard definition given above.

### C. Approximations

Here we consider some useful results for optimization over independence systems. Combining results from [12], [8] and [7], we have the following.

**Lemma II.3.** *For the problem of approximating the maximum valued basis of a $p$-system,*

(i) *using a non-negative linear objective function, the greedy algorithm gives a $\frac{1}{p}$ approximation.*
(ii) *using a monotone non-negative submodular function, the greedy algorithm gives a $\frac{1}{p+1}$ approximation.*

Linear functions are a special case of submodular functions (curvature is 0) so it is reasonable to expect the greedy bound to be a continuous function of the curvature. In [10], for a system that is the intersection of $p$ matroids the greedy bound is shown to be $\frac{1}{p+\kappa}$. We extend this result to $p$-systems. The proof is omitted here and can be found in [11].

**Theorem II.4.** *Consider the problem of maximizing a non-negative monotone submodular function $f$ with curvature $\kappa$, over a $p$-system. Then, the greedy algorithm gives an approximation factor of $\frac{1}{p+\kappa}$.*

### D. Set Systems on Graphs

In this section we introduce some graph constructs and then give some results relating them to $p$-systems. We are given a graph $G = (V, E, w)$ where $V$ is the set of vertices and $E$ is the set of edges. A set of edges has a reward or utility associated with it given by the function $w : 2^E \to \mathbb{R}_{\geq 0}$. Let $\delta(v)$ denote the set of edges that are incident to $v$.

We follow the standard definitions for *simple path*, *simple cycle* and *Hamiltonian cycle* (cf. [13]). We refer to a Hamiltonian cycle as a **tour**, and a simple cycle that is not a tour as a **subtour**. Let $\mathcal{H} \subset 2^E$ be the set of all tours in $G$.

**Definition II.5** (Simple $b$-matching)**.** Given vertex capacities $b : V[G] \to \mathbb{N}$. A **simple b-matching** is an assignment to the edges $f \in \{0, 1\}^E$ such that $\sum_{e \in \delta(v)} f_e \leq b(v), \forall v \in V[G]$. If $\sum_{e \in \delta(v)} f_e = b(v), \forall v$, the b-matching is **perfect**.

For the rest of this paper, any reference to a b-matching will always refer to a simple b-matching.

**Theorem II.6** (Mestre, [14])**.** *A simple $b$-matching is a 2-extendible system.*

Given a complete graph, the classical Traveling Salesman Problem (TSP) is to find a minimum cost tour. On a directed graph, the TSP can be divided into two variants: the Asymmetric TSP (where for two vertices $u$ and $v$, $c(u, v) \neq c(v, u)$) and the Symmetric TSP (where $c(u, v) = c(v, u)$, which is the case if the graph is undirected).

The set of possible solutions for the TSP can be defined using an independence system. A set of edges is independent if they form a collection of vertex disjoint paths, or a complete tour. Although the STSP is just a special case of the ATSP, directly defining an independence system for an undirected TSP can lead to better approximations. A set of edges is independent if the induced graph satisfies the two conditions (i) each vertex has degree at most 2, and (ii) there are no subtours.

**Lemma II.7** (Jenkyns, [15])**.** *On a graph with $n$ vertices the undirected TSP is a $p$-system with $p = 2 - \left\lfloor \frac{n+1}{2} \right\rfloor^{-1} < 2$.*

## III. Problem Formulation

Given a complete graph $G = (V, E, w)$, where $w$ is a submodular rewards function that has a curvature of $\kappa$, we are interested in analysing simple algorithms to find a Hamiltonian tour $S$ that has the maximum reward:

$$\max_{S \in \mathcal{H}} w(S). \tag{2}$$

In [11] we also consider the problem of where costs are incorporated into the optimization problem.

In the following sections, we look at two methods of approximately finding the optimal tour according to (2).

## IV. A Simple Greedy Strategy

A greedy algorithm to construct the TSP is given in Algorithm 1. The idea is to pick the edge that will give the largest marginal benefit at each iteration. The selected edge cannot cause the degree of any vertex to be more than 2 nor create any subtours.

```
Algorithm 1: GREEDYTOUR
  Input: Graph $G = (V, E)$. Function oracle
         $w : 2^E \to \mathbb{R}_{\geq 0}$
  Output: Edge set $M$ corresponding to a tour.
1 $M \leftarrow \emptyset$
2 while $E \neq \emptyset$ and $|M| < |V|$ do
3     if $M$ was updated then recalculate $\rho_e, \forall e \in E$
4     $e_m \leftarrow \mathrm{argmax}_{e \in E}\, \rho_e$
5     Determine if $e_m$ is valid by checking vertex degrees
      and checking for loops
6     if $e_m$ is valid then $M \leftarrow M \cup \{e_m\}$
7     $E \leftarrow E \setminus \{e_m\}$
8 return M
```

**Theorem IV.1.** *The complexity of the greedy tour algorithm (Alg. 1) is $O(|V|^3(f + \log|V|))$, where $f$ is the runtime of the oracle, and is a $\frac{1}{2+\kappa}$ approximation.*

*Proof.* By Lemma II.7 and Theorem II.4, Algorithm 1 is a $\frac{1}{2+\kappa}$-approximation of (2). At each iteration, the marginal benefit of edges not yet selected are recalculated and sorted. This dominates the runtime and has a complexity of $O(|V|(|E|f + |E|\log|E|))$. □

Motivated by the reliance of the bound on the curvature, in the next section we will consider a method to obtain improved bounds for functions with a lower curvature.

## V. 2-Matching based tour

Another approach to finding the optimal basis of an undirected TSP set system is to first relax the "no subtours" condition. The set system defined by the independence condition that each vertex can have a degree at most 2 is in fact just a simple 2-matching. As before, finding the optimal 2-matching for a submodular function is a NP-hard problem. We discuss two methods to approximate a solution. The first is a greedy approach and the second is by using a linear relaxation of the submodular function. We will see that the bounds with linear relaxation will be better than the greedy approach for lower values of curvature.

### A. Greedy 2-Matching

One way to find an approximate maximum 2-matching is to use a greedy approach similar to GREEDYTOUR, except there is no need to check for subtours. We call this the GREEDYMATCHING algorithm and present details in [11].

**Theorem V.1.** *The complexity of the greedy matching algorithm is $O(|V|^3(f + \log|V|))$, where $f$ is the runtime of the oracle. The greedy approach is a $\frac{1}{2+\kappa}$-approximation.*

*Proof.* A simple 2-matching is a 2-extendible system, so the greedy solution will be within $\frac{1}{2+\kappa}$ of the optimal (Lemma 2). The runtime analysis is similar to that of GREEDYTOUR. □

### B. Maximum 2-Matching Linear Relaxation

Looking at the case of a linear objective function, the problem of finding the maximum weight 2-matching can be found in $O(|V|^3)$ time [13] via an extension of Edmonds' Maximum Weighted Matching algorithm. For our original problem with (2) as the objective function for the maximization, this method can obviously not be applied directly. Therefore, we define a linear relaxation $\tilde{w}$ of the submodular function $w$ as follows,

$$\tilde{w}(S) = \sum_{e \in S} w(e) = \sum_{e \in S} \Delta_{\emptyset}(e), \quad \forall S \subseteq E. \tag{3}$$

In other words, we assign to each edge its maximum possible marginal benefit. Using this relaxation, the optimal 2-matching based on the weights $\tilde{w}$ can be calculated.

**Theorem V.2.** *Let $M_1$ be the maximum 2-matching for a submodular rewards function $w$ and let $M_2$ be the maximum weight 2-matching using $\tilde{w}$ as the edge weights. If $w$ has an independence system curvature of $\kappa_I$, then*

$$w(M_2) \geq (1 - \kappa_I) w(M_1).$$

*Proof.* The definition of curvature states that $\Delta_S(e) \geq (1 - \kappa_I)w(e)$ for any independent subset $S$ of the edges and $e$ such that $S \cup \{e\}$ is also independent. Using this and the definition of submodularity,

$$\begin{aligned} w(M_2) &= \Delta_{\emptyset}(e_1) + \Delta_{\{e_1\}}(e_2) + \Delta_{\{e_1,e_2\}}(e_3) + \ldots \\ &\geq (1 - \kappa_I) \sum_{e \in M_2} w(e) = (1 - \kappa_I)\tilde{w}(M_2). \end{aligned}$$

Since $\tilde{w}(M_2)$ is maximum, $\tilde{w}(M_2) \geq \tilde{w}(M_1)$. Therefore, $w(M_2) \geq (1 - \kappa_I)\tilde{w}(M_1) \geq (1 - \kappa_I)w(M_1)$, due to decreasing marginal benefits. Note that this bound also holds using the standard definition of curvature. □

### C. Reduced 2-Matching

The output of either of the two algorithms described will be a basis of the 2-matching system. Once a maximal 2-matching has been obtained, it needs to be converted into a tour. The edge set corresponding to the 2-matching can be divided into a collection of disjoint sets. At most one of these will consist of a simple path containing at most two vertices (i.e., one edge); the rest will be subtours. If any simple path consisted of more than one edge, its endpoints could be joined together contradicting the maximality of the 2-matching.

In order to convert the maximal 2-matching to a tour, the subtours are broken by removing an edge from each one. The remaining set of simple paths are then connected up.

We first give a result on efficiently finding a subset of edges to remove from a set while maintaining $\frac{2}{3}$ of the value. We then give an algorithm to reduce a set of subtours starting from a maximal 2-matching.

*1) Removing elements from a set:* Given a set $S$ and a $m$-partition of the set $\{A^i\}_{i=1}^m$, i.e. $S = \bigcup_{i=1}^m A^i$ and that $A^i \cap A^j = \emptyset$ for all $i \neq j$. Set $A^i$ contains $n_i$ elements, $A^i = \bigcup_{j=1}^{n_i} a_j^i$, such that $3 \leq n_i \leq N$. Let $k = \min_i n_i$ and let $\bar{n} = (n_1, \ldots, n_m)$. Also given a monotone non-decreasing submodular function $f : 2^S \to \mathbb{R}_{\geq 0}$. Let $A_{-j}^i := A^i \setminus a_j^i$.

**Theorem V.3.** *Given set $S$ and disjoint subsets $A^i, i = 1, \ldots, m$, where $k$ is the size of the smallest subset $A^i$, as defined above. There exists a set $R$ of $m$ elements (to which each set $A^i$ contributes exactly one element) such that $f(R) \geq (1 - \frac{1}{k})f(S) \geq \frac{2}{3}f(S)$.*

*Proof.* A selection of one element from each set can be given by the vector $p \in \mathbb{Z}_+^m, p \leq \bar{n}$. Let $b_p = f(S) - f(\bigcup_{i=1}^m A_{-p_i}^i)$ be the unique contribution of the selected elements to the total reward of $S$. For convenience of notation, define $\mathbf{i} := (i, i, \ldots, i) \in \mathbb{R}^m$.

The following lemma will help show the desired result.

**Lemma V.4.** $\sum_{i=1}^k b_{\mathbf{i}} \leq f(S)$.

*Proof.* The basic argument is that $b_{\mathbf{i}}$ is the minimum possible contribution of the set $\bigcup_j a_i^j$. So the total contribution over different $i$ will be less than (or equal to) the sum of their minimum contributions.

Let $B_i = \bigcup_{j=1}^m a_i^j$ be the set obtained by selecting the $i$th element from each set and let $T = \bigcup_{i=1}^k B_i$. Using the definition of marginal benefit $\Delta_S(X)$, we can write $f(T) = \Delta_\emptyset(B_1) + \ldots + \Delta_{T \setminus B_k}(B_k)$. However, by submodularity, $b_{\mathbf{i}} = \Delta_{S \setminus B_i}(B_i) \leq \Delta_X(B_i)$, $\forall X \subseteq S \setminus B_i$. Therefore, $f(T) \geq \sum_{i=1}^k \Delta_{S \setminus B_i}(B_i)$.

Combining this with the fact that $T \subseteq S \implies f(T) \leq f(S)$ by monotonicity, we get the desired result. □

To prove the theorem statement, assume that there does not exist any set with the desired property, i.e. $\forall p \in \mathbb{Z}_+^m, p \leq \bar{n}$ we have $f(\bigcup_{i=1}^m A_{-p_i}^i) < (1 - \frac{1}{k})f(S)$. From this assumption, we can see that $b_p > \frac{1}{k}f(S)$. Therefore, $\sum_{i=1}^k b_{\mathbf{i}} > f(S)$. With Lemma V.4 the desired result is obtained by contradiction.

For the second part of the inequality, since $k = \min n_i \geq 3$, so $(1 - \frac{1}{k}) \geq \frac{2}{3}$. □

**Corollary V.5.** *Given a set $S$ such that $|S| \geq mk$, then there exists a subset $T$ of $m$ elements such that $f(S \setminus T) \geq (1 - \frac{1}{k})f(S)$.*

*2) Algorithm to remove one element per set:* Based on the above results for existence of a set that can be removed while maintaining at least $\frac{2}{3}$ of the original value, we introduce a simple technique (given in Algorithm 2) to find such a set by searching over a finite number of disjoint sets.

**Theorem V.6.** *Algorithm 2 is correct and its complexity is $O(kf) = O(|S|f)$.*

The proof (see [11]) follows from the following lemma.

**Lemma V.7.** *Consider a set $S$ composed of $m$ disjoint subsets as defined above. Then there exists $i \in \{1, \ldots, k\}$ such that $f(\bigcup_{j=1}^m A_{-i}^j) \geq (1 - \frac{1}{k})f(S)$.*

**Algorithm 2:** REDUCESET($S, A^1, \ldots, A^m$)

```
Input: $S = \bigcup_{i=1}^m A^i$ where $A^i = \bigcup_{j=1}^{n_i} a_j^i$
Output: $U \subset S$ s.t. $U \cap A^i = 1$ for all $i = 1 \ldots m$
1 $i \leftarrow 1$; $k \leftarrow \min_j |A^j|$;
2 $U := \bigcup_{j=1}^m a_i^j$;
3 while $f(S \setminus U) < \frac{k-1}{k} f(S)$ do
4     $i \leftarrow i + 1$;
5     $U := \bigcup_{j=1}^m a_i^j$;
6 return $U$;
```

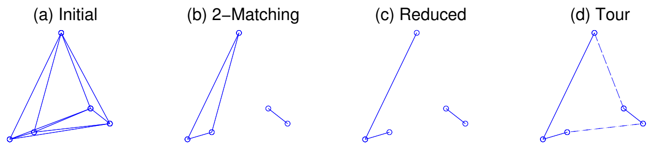


Fig. 1. The steps in the 2-matching based tour algorithm

*3) Algorithm to delete edges:* We can now use Algorithm 2 to remove one edge from each subtour in a matching.

The subtours, $T^i$, can be considered as a collection of disjoint edge sets. Each subtour will consist of atleast three edges. Since we want to remove one element from each set, the results of Theorem V.3 apply and so we know that there will exist a solution such that at most $\frac{1}{3}$ of the value of the objective is lost. If there exists an extra edge not part of any subtour, it does not affect the result since, following from Corollary V.5, it can just be considered as part of one of the other subtours. Outlined in Algorithm 3 is a method that will find a good set of edges to remove.

**Algorithm 3:** REDUCEMATCHING

```
Input: A 2-matching $G_M = (V, M)$ where
       $M = \bigcup_{i=1}^m T^i$ and the sets $T^i$ are the subtours.
1 Ignore all sets $T^i$ such that $|T^i| = 1$. Label the
  remaining $n \leq m$ sets $A^1, \ldots, A^n$.
2 return REDUCESET($M, A^1, \ldots, A^n$)
```

**Theorem V.8.** *Algorithm 3 correctly reduces the matching while maintaining $\frac{2}{3}$ of the original value. The complexity of the algorithm is $O(kf) = O(|V|f)$*

## D. Tour using matching algorithm

We now present an outline of the complete 2-matching tour algorithm. The steps are illustrated in Figure 1.

(i) Run GREEDYMATCHING to get a simple 2-matching, $M_1$. Using the linear relaxation $\tilde{w}$ of $w$, solve for the maximum weight 2-matching, $M_2$. From $M_1$ and $M_2$, choose the 2-matching that has a higher reward.
(ii) Determine all sets of subtours.
(iii) Run Algorithm 3 to select edges to remove.
(iv) Connect up the reduced subtours into a tour.

**Theorem V.9.** *The 2-matching tour algorithm gives a $\max\{\frac{2}{3(2+\kappa)}, \frac{2}{3}(1-\kappa)\}$-approximation in $O\big((|V|^3 + |V|)f + |V|^3 \log |V|\big)$ time.*

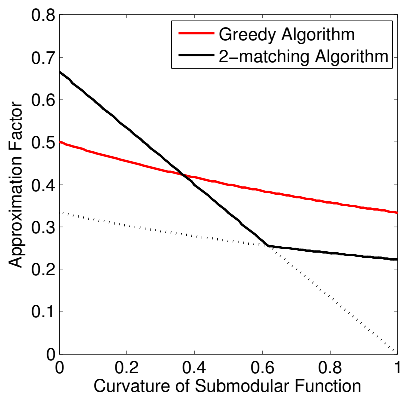


Fig. 2. Comparison of bounds for the two algorithms.

*Proof.* Note that the optimal tour has a value less than or equal to the optimal 2-matching. The 2-matching is a $\max\{\frac{1}{2+\kappa}, 1-\kappa\}$-approximation from Theorems V.1 and V.2. Removing edges from the 2-matching retains atleast $\frac{2}{3}$ of the original value of the 2-matching (Theorem V.8). $\square$

A similar method of using a matching is used in [1] for a linear reward function. In that case, the amount loss in the reduction step is shown to be $\frac{2}{3}$ of the optimal matching. We showed that a similar bound limiting the loss can be obtained for the submodular case; however, using the greedy approximation for the 2-matching, we see a further loss in value resulting in the final tour being within $\frac{2}{3}\frac{1}{2} = \frac{1}{3}$ of the optimal. By also using the second method of finding the 2-matching, our resulting bound for the final tour in the case of a linear function improves to $\frac{2}{3}$.

*Remark* V.10. For any value of $\kappa < \frac{1}{2}(\sqrt{3}-1) \approx 0.366$, constructing a 2-matching and then converting it into a tour, gives a better bound with respect to the optimal tour than by using the greedy tour approach (cf. Figure 2). •

## VI. Extension to Directed Graphs

The algorithms can also be applied to directed graphs, yielding approximations for the submodular ATSP.

### A. Greedy Tour

For the greedy tour algorithm, a slight modification needs to be made to check that the in-degree and out-degree of the vertices are less than 1 instead of checking for the degree being less than 2. Since the ATSP is a 3-extendible system [14], the approximation of the greedy algorithm changes to $\frac{1}{3+\kappa}$ instead of $\frac{1}{2+\kappa}$ as in the undirected case.

### B. Tour using Matching

Instead of working with a 2-matching, the system can be modeled as the intersection of two partition matroids defined as edge sets such that the indegree (outdegree) of each vertex $\leq 1$. This system is still 2-extendible and so the approximation for the greedy 2-matching does not change. For the second approximation, the Maximum Assignment Problem can be solved optimally by representing the weights in (3) as a weight matrix $\tilde{W}$ where we set $\tilde{W}_{ii} = -\infty$, then applying the Hungarian algorithm, which has a complexity of $O(|V|^3)$. Therefore, the result of Theorem V.2 still applies.

The result of the greedy algorithm or the solution to the assignment problem will be a set of edges that together form a set of cycles, with the possibility of a lone vertex. Note that a "cycle" could potentially consist of just two vertices. Therefore, removing one edge from each cycle will result in a loss of at most $\frac{1}{2}$ instead of $\frac{1}{3}$. This follows from Theorem V.3, setting $k = 2$. The final bound for the algorithm is therefore $\max\left\{\frac{1}{2(2+\kappa)}, \frac{1}{2}(1-\kappa)\right\}$.

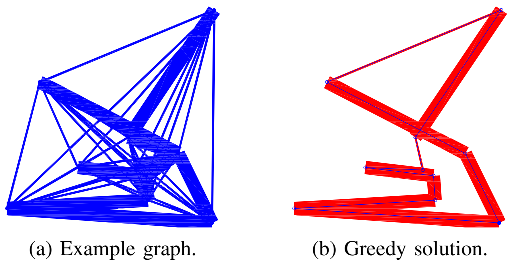


Fig. 3. A ten vertex graph and example solution.

## VII. Simulations

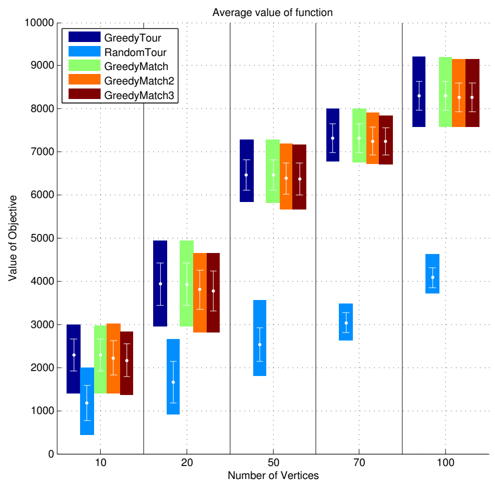
In order to empirically compare our algorithms, we have run simulations for a function that represents coverage of an environment. A complete graph is generated by uniformly placing vertices over a rectangular region. Each edge in the graph is associated with a rectangle and each rectangle is assigned a width to represent different amounts of coverage. An example of a ten vertex graph is given in Figure 3a. Here we see the complete graph as well as a representation of the value of each edge given by the area of the rectangle the edge corresponds to. Running the greedy tour algorithm, we get the tour given in Figure 3b.

We used a slightly different implementation for the greedy algorithm in our tests. For a submodular objective function, the marginal value of each element in the base set changes every iteration and so has to be recalculated. In [16], the author presents an accelerated greedy algorithm that uses the properties of submodularity to reduce the number of recalculations and thus improve the efficiency of the simple greedy approach. The accelerated greedy algorithm has the same worst case bound as the naive version; however, empirical, it can achieve significant speed-up factors [16],[4].

The simulations were performed on a quad-core machine with a 3.10 GHz CPU and 6GB RAM.

### A. Algorithm Comparison

All algorithms were run on 30 randomly generated graphs for five different graph sizes. The resulting value of the objective function was recorded and averaged over all 30 instances. The vertices were distributed randomly over a 100x100 region. The edge thickness was assigned a value of 7 with probability $2/\sqrt{|V|}$, or 1 otherwise (so $O(|V|)$ of the edges had a high reward). The results are shown in Figures 4 and 5.



| | GT | RT | GM | GM2 | GM3 |
|---|---|---|---|---|---|
| 10 | 27 (3) | 0 (0) | 25 (1) | 16 (1) | 2 (0) |
| 20 | 23 (8) | 0 (0) | 21 (5) | 8 (0) | 1 (0) |
| 50 | 26 (16) | 0 (0) | 14 (4) | 5 (0) | 0 (0) |
| 70 | 27 (18) | 0 (0) | 12 (3) | 3 (0) | 1 (0) |
| 100 | 27 (16) | 0 (0) | 14 (3) | 2 (0) | 1 (0) |

Fig. 4. (Top) The bars give the range of results. The white markers inside the bars show the mean and standard deviation. (Bottom) Number of wins for each algorithm. Wins include ties (unique wins specified in parens).

| | GT | RT | GM | GM2 | GM3 |
|---|---|---|---|---|---|
| 10 | 0.2 | 0 | 0.4 | 0.3 | 0.2 |
| 20 | 0.8 | 0.1 | 2.2 | 1.4 | 1.1 |
| 50 | 13.3 | 0.8 | 27.1 | 15.9 | 14.4 |
| 70 | 37.4 | 2.2 | 70.4 | 40.7 | 39.0 |
| 100 | 109.7 | 5.4 | 189.9 | 116.5 | 113.1 |

Fig. 5. Average runtime (seconds) of algorithms.

The algorithms compared are now described. **GreedyTour (GT):** The greedy algorithm for constructing a tour. **RandomTour (RT):** Randomly select edges to construct a tour. For the 2-matching based algorithm, three possibilities are considered. All three start off by greedily constructing a 2-matching. **GreedyMatching (GM):** Remove from each subtour the element that will result in the least loss to the total value and greedily connect up the complete tour. **GreedyMatching2 (GM2):** Use Algorithm 3 to reduce the matching and greedily connect up the complete tour. **GreedyMatching3 (GM3):** Use Algorithm 3 to reduce the matching and arbitrarily connect up the complete tour.

While the performance of GT and GM are similar, GM takes considerably longer to run, due to the oracle calls needed to determine which edge to remove from each subtour.

Note that in this particular set up, there were only a small number of subtours (compared to the number of vertices), so few calculations were needed to construct the final tour from the reduced subtours in GM2. It is however possible for the number of subtours to be $\Theta(|V|)$ and in those cases GM2 would be much slower compared to GM3 as the problem size would not be significantly reduced by first coming up with a 2-matching.

## VIII. Conclusions and Future Directions

In this paper, we extended the max-TSP problem to submodular rewards. We presented two algorithms; a greedy algorithm which achieves a $\frac{1}{2+\kappa}$ approximation, and a 2-matching-based algorithm, which achieves a $\max\{\frac{2}{3(2+\kappa)}, \frac{2}{3}(1-\kappa)\}$ approximation (where $\kappa$ is the curvature of the function). Both algorithms have a complexity of $O(|V|^3)$ in terms of number of oracle calls. We extended these results to directed graphs and presented simulation results to compare performance.

There are several directions for future work. First, we would like to determine the tightness of our bounds. Also, there are other strategies, such as best neighbour, insertion heuristics or Serdyukov's algorithm, that could be extended to the submodular case. One other possible extension would be to consider the case where multiple tours are needed.

## References

[1] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for finding a maximum weight hamiltonian circuit," *Operations Research*, vol. 27, no. 4, pp. pp. 799–809, 1979.

[2] A. I. Serdyukov, "An algorithm with an estimate for the traveling salesman problem of the maximum," *Upravlyaemye Sistemy*, vol. 25, pp. 80–86, 1984.

[3] C. Guestrin, A. Krause, and A. Singh, "Near-optimal sensor placements in Gaussian processes," in *Int. Conf. on Machine Learning*, Bonn, Germany, Aug. 2005.

[4] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Articial Intelligence Research*, vol. 42, pp. 427–486, 2011.

[5] P. R. Goundan and A. S. Schulz, "Revisiting the greedy approach to submodular set function maximization," 2007, Working Paper, Massachusetts Institute of Technology.

[6] A. Singh, A. Krause, C. Guestrin, and W. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[7] G. Calinescu, C. Chekuri, M. Pl, and J. Vondrk, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[8] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions - II," in *Polyhedral Combinatorics*, ser. Mathematical Programming Studies, 1978, vol. 8, pp. 73–87.

[9] J. Ward, "A (k+3)/2-approximation algorithm for monotone submodular k-set packing and general k-exchange systems," in *29th International Symposium on Theoretical Aspects of Computer Science*, vol. 14, Dagstuhl, Germany, 2012, pp. 42–53.

[10] M. Conforti and G. Cornuejols, "Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem," *Discrete Applied Mathematics*, vol. 7, no. 3, pp. 251 – 274, 1984.

[11] S. T. Jawaid and S. L. Smith, "The maximum traveling salesman problem with submodular rewards," Sep. 2012, available at http://arxiv.org/abs/1209.3759.

[12] D. Hausmann, B. Korte, and T. A. Jenkyns, "Worst case analysis of greedy type algorithms for independence systems," in *Combinatorial Optimization*, ser. Mathematical Programming Studies. Springer Berlin Heidelberg, 1980, vol. 12, pp. 120–131.

[13] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 4th ed., ser. Algorithmics and Combinatorics. Springer, 2007, vol. 21.

[14] J. Mestre, "Greedy in approximation algorithms," in *Algorithms ESA 2006*, Y. Azar and T. Erlebach, Eds. Springer Berlin / Heidelberg, 2006, vol. 4168, pp. 528–539.

[15] T. A. Jenkyns, "The greedy travelling salesman's problem," *Networks*, vol. 9, no. 4, pp. 363–373, 1979.

[16] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization Techniques*, ser. Lecture Notes in Control and Information Sciences, J. Stoer, Ed. Springer Berlin / Heidelberg, 1978, vol. 7, pp. 234–243.