Informative Path Planning as a Maximum Traveling Salesman Problem with Submodular Rewards[☆]

Syed Talha Jawaid[a], Stephen L. Smith[a]

[a] *Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada*

Abstract

In this paper we extend the classic problem of finding the maximum weight Hamiltonian cycle in a graph to the case where the objective is a submodular function of the edges. We consider a greedy algorithm and a 2-matching based algorithm, and we show that they have approximation factors of $\frac{1}{2+\kappa}$ and $\max\{\frac{2}{3(2+\kappa)}, \frac{2}{3}(1-\kappa)\}$ respectively, where κ is the curvature of the submodular function. Both algorithms require a number of calls to the submodular function that is cubic to the number of vertices in the graph. We then present a method to solve a multi-objective optimization consisting of both additive edge costs and submodular edge rewards. We provide simulation results to empirically evaluate the performance of the algorithms. Finally, we demonstrate an application in monitoring an environment using an autonomous mobile sensor, where the sensing reward is related to the entropy reduction of a given a set of measurements.

Keywords: submodular function maximization, traveling salesman problem, informative path planning

1. Introduction

The maximum weight Hamiltonian cycle is a classic problem in combinatorial optimization. It consists of finding a path in a graph that starts and ends at the same vertex and visits all other vertices exactly once while maximizing the sum of the weights (i.e., the reward) on the edges traversed. Also referred to as the max-TSP, the problem is NP-hard; however, a number of approximation algorithms have been developed. In [1] four simple approximation algorithms are analyzed. The authors show that greedy, best-neighbor, and 2-interchange heuristics all give a $\frac{1}{2}$ approximation to the optimal tour. They also show that a 2-matching based heuristic, which first finds a perfect 2-matching and then converts that to a tour, gives a $\frac{2}{3}$ approximation. The simple and elegant Serdyukov's algorithm [2] — which combines a perfect 2-matching and a 1-matching to compute a tour — gives a $\frac{3}{4}$ approximation. The best known deterministic algorithm is given in [3] and it achieves a $\frac{7}{9}$ approximation in $O(n^3)$ time. A number of randomized algorithms also exist such as the one given in [4] that achieves a $\frac{25}{33}$ approximation ratio. In [5] it was shown that this algorithm can be derandomized while maintaining its approximation factor. In this paper we look at extending the max-TSP problem to the case of submodular rewards.

This extension is motivated, in part, by the application of mobile sensing robots to persistently monitor a large environment [6] such as for obtaining data on natural phenomenon or for security.

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

Email addresses: stjawaid@uwaterloo.ca (Syed Talha Jawaid), stephen.smith@uwaterloo.ca (Stephen L. Smith)

It is desirable to have a closed walk or a tour over which the sensing robot travels to minimize re-planning during long term operation. Applications include tasks such as monitoring oil spills [7], forest fires [8] or underwater ocean monitoring [9].

Informative path planning involves using preexisting information about the environment (such as a probability distribution) to plan a path that maximizes the information gained. It is a topic that has been researched with various different approaches. For example, in [10] the authors look into creating trajectories to best estimate a Gaussian random field by intelligently generating rapidly-exploring random cycles. One common way to measure information content is using mutual information. In [11], this metric is used to place static sensors in a Gaussian field. Other papers investigate maximizing the knowledge at specific points by planning a path for a sensing robot while also taking into account budget constraints [12, 13]. The metrics used to determine the quality of the sensing, such as mutual information, are usually submodular in nature. The defining property of a submodular function is that of decreasing marginal value, i.e., adding an element to a set will result in a larger increase in value than adding it to a superset of that set. For example, if a sensor is placed close to another, then the benefit gained by the second sensor will be less than if the first sensor had not already been placed. Other areas where submodular functions arise include viral marketing, active learning [14] and AdWords assignment [15].

Our problem can be stated as maximizing a submodular function over the edges of a graph subject to the constraint that the selected edges form a tour. Generally, one way to represent constraints in a combinatorial optimization problem is through the concept of independence systems or its many specializations, including p-systems and matroids. Although unconstrained minimization of a submodular function can be achieved in polynomial time [16, 17], maximizing a non-decreasing submodular function over an independence system constraint is known to be NP-hard. For a monotone submodular function, a number of approximation algorithms exist for optimizing over multiple matroid constraints [18]. Some bounds that include the dependence on curvature are presented in [19]. Local search methods have been found to be particularly useful [20, 21] for both monotone and non-monotone functions. Various results exist for non-monotone submodular function maximization for both the unconstrained case [22] and for the case where constraints exist, such as multiple matroid and knapsack constraints [23, 24] or p-system constraints [25]. The use of continuous relaxations of the submodular function have also lead to optimal approximation bounds [26] for the case of a single matroid as well as improved bounds for a combination of various constraints [27].

Contributions: The contributions of this paper are to present and analyze two algorithms for constructing a maximum reward tour on a graph. The metric used in maximizing 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 $\max\left\{\frac{2}{3(2+\kappa)}, \frac{2}{3}(1-\kappa)\right\}$ approximation where κ 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 establish a few new properties of submodular maximization, including the generalization of a known bound on the performance of the greedy algorithm to functions with curvature κ (Theorem 2.9), and a result on the minimum possible value of disjoint subsets within a function's base set (Theorem 3.5).

We also present an approach for the case of a multi-objective optimization consisting of submodular (sensing) rewards on the edges along with modular (travel) costs. We incorporate these two objectives into a single function which is no longer monotone nor positive. We provide bounds

on the performance of our algorithms in this case, which depend on the relative weight of the rewards.

A preliminary version of this work was presented in [28]. This paper substantially improves upon those results by providing complete proofs of the approximation ratios, a discussion on extending Serdyukov's algorithm, an approach for multi-objective optimization with weights and costs, simulations demonstrating dependence on curvature as well as an application in informative path planning.

Organization: The organization of this paper is as follows. In Section 2 we review independence systems and submodularity and we formalize our problem. In Section 3 we analyze two different strategies for approximating a solution; namely, a greedy approach and a 2-matching based tour. We also discuss extending Serdyukov's algorithm. In Section 4 we extend our algorithms to the case where the graph is directed. In Section 5 we discuss a method to incorporate costs into the optimization. Finally, we provide some simulation results in Section 6 and demonstrate an example application in monitoring.

2. Preliminaries and Problem Formulation

Here we present preliminary concepts and formulate the maximum submodular TSP problem. We give a brief summary of results on combinatorial optimization over independence systems and we extend a known result on optimization over p-systems to incorporate curvature.

2.1. Independence systems

Combinatorial optimization problems can often be formulated as the maximization or minimization of an objective function $f : \mathcal{F} \to \mathbb{R}$ over a set system $(E, \mathcal{F})$, 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., all $A \in \mathcal{F}$ such that $A \cup \{x\} \notin \mathcal{F}, \forall x \in E \setminus A$) are the **bases**.

Definition 2.1 (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. Then, S is a p-system if $U(A) \leq pL(A), \forall A \subseteq E$.

Definition 2.2 (p-extendible system)**.** An independence system $(E, \mathcal{F})$ is p-extendible if given any independent set $B \in \mathcal{F}$, for every subset A of B and for every $x \notin A$ such that $A \cup \{x\} \in \mathcal{F}$, there exists $C \subseteq B \setminus A$ such that $|C| \leq p$ and for which $(B \setminus C) \cup \{x\} \in \mathcal{F}$.

Definition 2.3 (Matroid)**.** An independence system $(E, \mathcal{F})$ is a matroid if it satisfies the additional property that if $X, Y \in \mathcal{F}$ such that $|X| > |Y|$, then $\exists x \in X \backslash Y$ with $Y \cup \{x\} \in \mathcal{F}$.

Remark 2.4. These specific types of independence systems are intricately related. A matroid is a 1-extendible system and any p-extendible system is a p-system. In addition, any independence system can be represented as the intersection of a finite number of matriods [29]. •

An example of a matroid is the *partition matroid*. The base set is composed of n disjoint sets, $\{E_i\}_{i=1}^n$. Given $k \in \mathbb{Z}_+^n$, the matroid is defined by the collection $\mathcal{F} := \{A \subseteq E : |A \cap E_i| \leq k_i, \forall i = 1 \dots n\}$.

2.2. Submodularity

Without any additional structure on the set-function f, optimizing f subject to any constraints is generally intractable and inapproximable. However, a fairly general class of objective functions for which approximation algorithms exist is the class of submodular set functions.

Definition 2.5 (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 $\Delta_f(B|A) := f(A \cup B) - f(A)$. Then,

$$\Delta_f(x|A) \geq \Delta_f(x|B), \quad \forall A \subseteq B \subseteq N.$$

Henceforth, the subscript f will be omitted unless there is ambiguity.

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 of a function is the *curvature*. A submodular function has a curvature of $\kappa \in [0, 1]$ if for any $A \subset N$ and $x \in N \setminus A$,

$$\Delta(x|A) \geq (1 - \kappa) f(x). \tag{1}$$

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

Remark 2.6 (Stronger Notion of Curvature). We can formulate a slightly stronger notion of the curvature, which we call the independence system curvature κ_I by taking the independence system in to account. In this case, condition (1) need only be satisfied for any $A \in \mathcal{F}$ and $x \in N \setminus A$, such that $A \cup \{x\} \in \mathcal{F}$. This notion of curvature is stronger in that its value is lower than that obtained by the standard definition. All results presented in this paper hold for this stronger notion of curvature, thus giving better performance bounds. However, the disadvantage of κ_I is that it is computationally intensive to compute: One would have to compute all maximal basis of the independence system. •

2.3. Approximations

Here we present some results for optimization over independence systems. Throughout this paper, we will assume that the objective function is **normalized**, i.e., the empty set has a value of 0. Combining results from [30], [18] and [26], we have the following.

Lemma 2.7. *Consider the problem of approximating the maximum valued basis of a p-system.*

(i) *If the objective function is linear and non-negative, then the greedy algorithm gives a* $\frac{1}{p}$ *approximation.*

(ii) *If the objective function is submodular, non-negative and monotone non-decreasing, then the greedy algorithm gives a* $\frac{1}{p+1}$ *approximation.*

In [31] the authors look at maximizing a submodular function over a uniform matroid (i.e., selecting k elements from a set). They show that the greedy algorithm gives an approximation of $1 - \frac{1}{e}$. This is the best factor that can be achieved, unless P = NP [32].

In [18], the optimization problem is generalized to an independence system represented as the intersection of p matroids. The authors state that the result can be extended to p-systems. A complete proof for this generalization is given in [26]. For a single matroid constraint, an algorithm to obtain a $(1 - 1/e)$ approximation is also given in [26].

Linear functions are a special case of submodular functions (curvature is 0) so it is reasonable to expect the bound for the greedy algorithm to be a continuous function of the curvature. In [19], 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.

Definition 2.8 (α-approximate greedy)**.** Given an objective function f defined over an independence system $(E, \mathcal{I})$. For $\alpha \in (0, 1]$, an α-approximate greedy algorithm greedily constructs an approximation to the maximum value basis by selecting at each iteration i an element g_i such that

$$\Delta(g_i|G_{i-1}) \geq \alpha \max_{\substack{e \in E \setminus G_{i-1} \\ G_{i-1} \cup \{e\} \in \mathcal{I}}} \Delta(e|G_{i-1}),$$

where $G_i = \bigcup_{j=0}^{i} g_j$ and $g_0 = \emptyset$.

Theorem 2.9. *Consider the problem of maximizing a monotone submodular function f with curvature κ, over a p-system. Then, the α-approximate greedy algorithm gives an approximation factor of $\frac{\alpha}{p+\alpha\kappa}$.*

Proof. The proof is inspired from the proof where the system is the intersection of p integral polymatroids [19, Theorem 6.1]. Let W be the optimal set. Let $S = \{s_1, \ldots, s_k\}$ be the result of the greedy algorithm where the elements are enumerated in the order in which they were chosen. For $t = 1, \ldots, k$, let $S_t := \bigcup_{i=1}^{t} s_i$ and $\rho_t := \Delta(s_t|S_{t-1}) = f(S_t) - f(S_{t-1})$. Therefore, $f(S) = \sum_{t=1}^{k} \rho_t$. So,

$$f(W \cup S) \geq f(W) + \sum_{e \in S} \Delta(e|(W \cup S) \setminus e) \geq f(W) + (1 - \kappa) \sum_{t=1}^{k} \rho_t, \tag{2}$$

since $\Delta(e|(W \cup S) \setminus e) \geq (1-\kappa)f(e)$ by the definition of curvature, and the rest of the inequalities hold due to submodularity. Also,

$$f(W \cup S) \leq f(S) + \sum_{e \in W \setminus S} \Delta(e|S). \tag{3}$$

Following from the analysis of the greedy algorithm for p-systems in [26, Appendix B], a k-partition $W_1, \ldots, W_k$ of W, can be constructed such that $|W_t| \leq p$ and $\rho_t \geq \alpha\Delta(e|S_{t-1})$ for all $e \in W_t$. Therefore,

$$\sum_{e \in W \setminus S} \Delta(e|S) = \sum_{i=1}^{k} \sum_{e \in W_i \setminus S} \Delta(e|S) \leq \sum_{i=1}^{k} |W_i \setminus S| \frac{\rho_t}{\alpha} \leq \frac{p}{\alpha} \sum_{i=1}^{k} \rho_t, \tag{4}$$

since $\Delta(e|S) \leq \Delta(e|S_{t-1})$ for all t by submodularity.

Combining (3) and (4) with (2), the desired result can be derived as follows,

$$f(W) + (1 - \kappa)f(S) \leq f(S) + \frac{p}{\alpha} f(S) \implies f(W) \leq \frac{(p + \alpha\kappa)}{\alpha} f(S).$$

□

2.4. Set Systems on Graphs

In this section we introduce some graph constructs and relate them to p-systems. Given a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. We follow the standard definitions for *simple path*, *simple cycle* and *Hamiltonian cycle* (cf. [29]). 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. Let $\delta(v)$ denote the set of edges that are incident to v.

Definition 2.10 (Simple b-matching)**.** Given vertex capacities $b : V \to \mathbb{N}$. A **simple b-matching** is an assignment $f \in \{0, 1\}^E$ to the edges such that $\sum_{e \in \delta(v)} f(e) \leq b(v)$, $\forall v \in V$, If equality holds for all v, then 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 2.11 (Mestre, [33])**.** *A simple b-matching is a 2-extendible system.*

Given a complete graph, the classic Maxmimum Traveling Salesman Problem (Max-TSP) is to find a maximum 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 feasible solutions for the directed TSP can be defined using a directed (Hamiltonian) tour independence system. A set of edges is independent if they form a collection of vertex disjoint simple paths, or a complete tour. The directed tour system can be formulated as the intersection of three matroids. These are:

(i) Partition matroid: Edge sets such that the in-degree of each vertex ≤ 1,
(ii) Partition matroid: Edge sets such that the out-degree of each vertex ≤ 1,
(iii) The 1-graphic matroid: the set of edges that form a forest with at most one simple cycle.

Theorem 2.12 (Mestre, [33])**.** *The directed tour independence system is 3-extendible.*

For an undirected graph, one option is to double the edges and formulate the problem as a STSP. Since the STSP is just a special case of the ATSP, the above formulation applies. Instead, directly defining an undirected tour independence system for an undirected TSP can lead to a stronger classification. In this case, a set of edges is independent if the induced graph satisfies the two conditions:

(i) each vertex has degree at most 2,
(ii) there are no subtours.

Lemma 2.13 (Jenkyns, [34])**.** *On a graph with n vertices, the undirected tour is a p-system with $p = 2 - \left\lfloor \frac{n+1}{2} \right\rfloor^{-1} < 2$.*

2.5. Problem Formulation

Given a complete graph $G = (V, E, w)$, where a set of edges has a reward or utility given by the submodular rewards function $w : 2^E \to \mathbb{R}_{\geq 0}$ that has a curvature of κ. We are interested in analyzing simple algorithms to find a Hamiltonian tour that has the maximum reward:

$$\max_{S \in \mathcal{H}} w(S). \tag{5}$$

In Section 5, we will briefly discuss the problem where costs are incorporated into the optimization problem; that is, where the graph contains both edge weights, representing travel costs, and edge rewards, representing information gain.

3. Algorithms for the Submodular Max-TSP

In this section we present some algorithms for approximating the submodular Max-TSP, i.e., approximating (5). We first describe a technique to generalize any known approximation algorithm, that uses a modular objective function, by performing a linear relaxation of the submodular function. Next, we analyze a greedy algorithm and a 2-matching based algorithm and provide complexity and approximation bounds for each. Finally, we discuss extending Serdyukov's algorithm to the submodular setting.

3.1. Linear Relaxation

One possibility for constructing an algorithm for a submodular objective function is to just use any algorithm that is for a modular objective. In order use this approach, the submodular function needs to be approximated by a modular function.

We define a linear relaxation $\tilde{w}$ of the submodular function w as follows,

$$\tilde{w}(S) = \sum_{e \in S} w(e) = \sum_{e \in S} \Delta(e|\emptyset), \quad \forall S \subseteq E. \tag{6}$$

In other words, each edge is assigned its maximum possible marginal benefit. Using this relaxation, any known algorithm to approximate the maximum tour can be applied. The question arises as to what the bound is for the value of the final tour.

Theorem 3.1. *Consider an independence system $(E, \mathcal{I})$ over which a submodular rewards function, w, is defined, with a curvature of κ. Denote the linear relaxation of w as $\tilde{w}$. Let M_O^s and M_O^r be the maximum value bases with respect to w and $\tilde{w}$ respectively. For some set M_1, if $\tilde{w}(M_1) \geq \alpha \tilde{w}(M_O^r)$ for $\alpha \in [0, 1]$, then*

$$w(M_1) \geq (1 - \kappa)\alpha w(M_O^s).$$

Proof. The definition of curvature states that $\Delta(e|S) \geq (1 - \kappa)w(e)$ for $S \subset E$ and $e \in E \setminus S$. Using this and the definition of submodularity,

$$\begin{aligned} w(M_1) &= \Delta(e_1|\emptyset) + \Delta(e_2|\{e_1\}) + \Delta(e_3|\{e_1, e_2\}) + \cdots \\ &\geq (1 - \kappa) \sum_{e \in M_1} w(e) = (1 - \kappa)\tilde{w}(M_1) \\ &\geq (1 - \kappa)\alpha \tilde{w}(M_O^r) \end{aligned}$$

Since $\tilde{w}(M_O^r)$ is maximum, $\tilde{w}(M_O^r) \geq \tilde{w}(M_O^s)$. Therefore,

$$w(M_1) \geq (1 - \kappa)\alpha \tilde{w}(M_O^s) \geq (1 - \kappa)\alpha w(M_O^s),$$

where the last inequality holds due to decreasing marginal benefits. □

Although the maximum value of the element is easy to evaluate, it could be a gross overestimate of the actual contribution of the element to a set. As a result, this method works better for lower values of curvature.

3.2. A Simple Greedy Strategy

A greedy algorithm to construct the TSP tour 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.

Theorem 3.2. GREEDYTOUR *(Algorithm 1) gives a* $\frac{1}{2+\kappa}$ *approximation of the optimal tour and has a complexity of* $O(|V|^3(f + \log|V|))$*, where* f *is the runtime of the function oracle.*

Proof. By Lemma 2.13 and Theorem 2.9, Algorithm 1 is a $\frac{1}{2+\kappa}$-approximation of (5).

At each iteration, the marginal benefit of edges not yet selected are recalculated and sorted (line 3). Since recalculation need only be done when the set M changes, and only one edge is added to the partial tour M at each iteration, recalculation only needs to take place a total of $|V|$ times. This dominates the runtime and has a complexity of $O(|V|(|E|f + |E|\log|E|))$. □

Remark 3.3. The detection of subtours can be done using disjoint-sets for the vertices where each set represents a group of vertices that are in the same subtour. The overall runtime for detecting subtours adds up to $|V|^2 \log|V|$ following the analysis in [35, Ch. 21,23]. The "recalculation" part dominates the total runtime and so the exact method used to check for validity of edges does not have a significant effect on the runtime. •

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 $\Delta(e|M)$, $\forall e \in E$
4 $e_m \leftarrow \text{argmax}_{e \in E} \Delta(e|M)$
5 Determine if e_m is valid by checking vertex degrees and checking for subtours
6 **if** e_m *is valid* **then** $M \leftarrow M \cup \{e_m\}$
7 $E \leftarrow E \setminus \{e_m\}$
8 **return** M

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

3.3. 2-Matching based tour

Another approach to finding the optimal basis of an undirected tour 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 certain values of curvature.

3.3.1. 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 refer to this as the GREEDY-MATCHING algorithm. The algorithm is exactly the same as Algorithm 1 except for a single change:

5 Determine if e_m is valid by checking vertex degrees.

For the GREEDYMATCHING algorithm we have the following result.

Theorem 3.4. *The* GREEDYMATCHING *algorithm gives a* $\frac{1}{2+\kappa}$*-approximation of the optimal 2-matching and has a complexity of* $O(|V|^3(f + \log|V|))$*, where* f *is the runtime of the function oracle.*

Proof. A simple 2-matching is a 2-extendible system (Theorem 2.11), so the greedy solution will be within $\frac{1}{2+\kappa}$ of the optimal (Theorem 2.9). The runtime analysis is similar to that of GREEDYTOUR. □

3.3.2. Maximum 2-Matching Linear Relaxation

For a linear objective function, the problem of finding a maximum weight 2-matching can be formulated as a binary integer program. Let $x = \{x_{ij}\}$ where $1 \leq i < j \leq |V|$ and let each edge be assigned a real positive weight given by $\tilde{w}_{ij}$. Define $E(x)$ as the set of edges for which $x_{ij} = 1$. Then the maximum weight 2-matching, $(V, E(x))$, can be obtained by solving

$$\begin{aligned} \max \quad & \sum_{i=1}^{|V|-1} \sum_{j>i} \tilde{w}_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} = 2, \quad \forall i \in \{1, \ldots, |V|\} \\ & x_{ij} \in \{0, 1\}, \quad 1 \leq i < j \leq |V|. \end{aligned}$$

Alternatively, for a weighted graph the maximum weight 2-matching can be found in $O(|V|^3)$ time [29] via an extension of Edmonds' Maximum Weighted Matching algorithm.

For our original problem with (5) as the objective function for the maximization, the two methods for constructing an optimal 2-matching described here can obviously not be applied directly. Therefore, we use the linear relaxation $\tilde{w}$ of the submodular function w. The optimal 2-matching based on the weights $\tilde{w}$ can be calculated and this 2-matching will be within $(1 - \kappa)$ of the optimal 2-matching based on w by Theorem 3.1.

3.3.3. 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 vertex-disjoint sets of edges. At most one of these will consist of a simple path, which will contain 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.

Theorem 3.5. *Consider a submodular function* f *defined over a set* E*, and* k *disjoint subsets* $\{E_i\}_{i=1}^k$ *of* E*. Then, at least one of these subsets satisfies* $f(E \setminus E_i) \geq \left(1 - \frac{1}{k}\right) f(E)$*.*

Proof. Consider the marginal value of each set E_i. At least one of these sets will have a marginal value that is less than the average marginal value. Without loss of generality, let E_1 be such that

$$\Delta(E_1 | E \setminus E_1) \leq \frac{1}{k} \sum_{i=1}^{k} \Delta(E_i | E \setminus E_i).$$

Also, letting $E_0 = \emptyset$,

$$\sum_{i=1}^{k} \Delta(E_i | E \setminus E_i) \leq \sum_{i=1}^{k} \Delta(E_i | \bigcup_{j=1}^{i-1} E_j) = f\left(\bigcup_{i=1}^{k} E_i\right) \leq f(E).$$

Combining these two inequalities,

$$f(E) - f(E \setminus E_1) = \Delta(E_1 | E \setminus E_1) \leq \frac{1}{k} f(E),$$

and the desired result follows. □

Using this theorem, an algorithm to reduce a 2-matching can be constructed. The REDUCE-MATCHING algorithm is outlined in Algorithm 2.

Theorem 3.6. *Algorithm 2 reduces the 2-matching while maintaining at least $\frac{2}{3}$ of the original value of the 2-matching. The complexity of the algorithm is $O(|V|f)$*

Proof. Note that each subtour will consist of at least three edges and so $k \geq 3$. Assume the edges within each subtour are arbitrarily indexed. Construct a collection of sets $\{E_i\}_{i=1}^{k}$ such that E_i contains edge i from each of the subtours. These sets will be disjoint. Therefore, by Theorem 3.5, at least one of these sets can be removed from the 2-matching while maintaining $(1 - \frac{1}{k}) \geq \frac{2}{3}$ of the value of the 2-matching.

As the existence of such a set is guaranteed, the algorithm merely cycles through each of the sets E_i until an appropriate one is found. The number of possible sets is bounded by the size of the smallest subtour, which is at most $|V|$ (consider the case where the output of the 2-matching is a tour). □

Algorithm 2: 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.
Output: A set of edges to remove from the 2-matching.

Ignore all sets T^i such that $|T^i| \leq 1$. Label the remaining $n \leq m$ sets $A^1, \ldots, A^n$.
`// Let` $\{a_j^i\}_j$ `be the edges in subtour` A^i
$j \leftarrow 1$; $k \leftarrow \min_i |A^i|$
$R := \bigcup_{i=1}^{n} a_j^i$
while $w(M \setminus R) < \frac{k-1}{k} w(M)$ **do**
    $j \leftarrow j + 1$
    $R := \bigcup_{i=1}^{n} a_j^i$
return R

3.3.4. Tour using matching algorithm

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

(a) 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 value.
(b) Determine all sets of subtours.

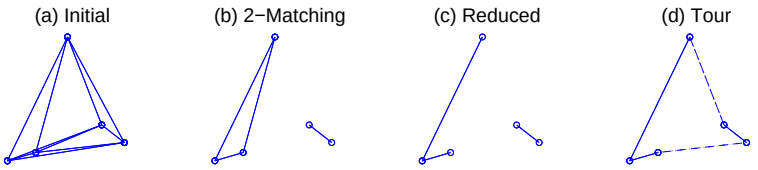


Figure 1: The steps in the 2-matching based tour algorithm.

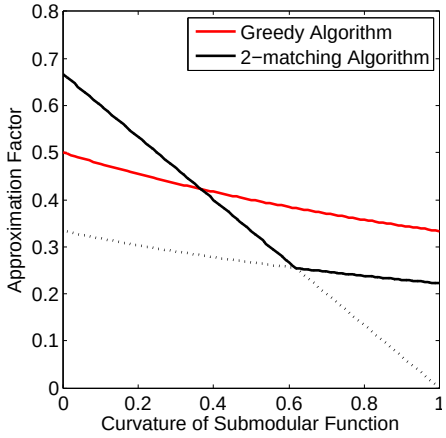


Figure 2: Comparison of bounds for the two algorithms.

(c) Run Algorithm 2 to select edges to remove.
(d) Connect up the reduced subtours into a tour.

Theorem 3.7. *The 2-matching tour algorithm gives a* $\max\left\{\frac{2}{3(2+\kappa)}, \frac{2}{3}(1-\kappa)\right\}$*-approximation in* $O((|V|^3+|V|)f + |V|^3 \log|V|)$ *time.*

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 3.4 and 3.1. Removing edges from the 2-matching retains atleast $\frac{2}{3}$ of the original value of the 2-matching (Theorem 3.6) and adding edges can only increase the value (i.e., no loss).

The greedy 2-matching takes $O(|V|^3(f+\log|V|))$ time and the linear relaxation approximation takes $O(|V|^3)$ time. Finding the edges in all the subtours is $O(|V|)$ since the number of edges in a 2-matching is at most $|V|$. Reducing the 2-matching is $O(|V|f)$. Connecting up the final graph is $O(m) = O(|V|)$ where m is the number of subtours and ranges from 1 to $\left\lfloor \frac{|V|}{3} \right\rfloor$. Therefore, the total runtime is $O(|V|^3(f+\log|V|) + |V|(2+f))$. □

A similar method of using a 2-matching is used in [1] for a linear objective function. In that case, the amount loss in the reduction step is shown to be at most $\frac{1}{3}$ of the optimal 2-matching. We showed that a similar bound limiting the loss can be obtained for the submodular case; however, we see a further loss in value since the 2-matching was constructed greedily, resulting in the final tour being within $\frac{2}{3}\frac{1}{2} = \frac{1}{3}$ of the optimal. By also using the relaxation 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 3.8 (Comparison of bounds). 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). •

Remarks 3.9 (Heuristic improvements). i) The $\frac{2}{3}$ loss is actually a worst case bound where the smallest subtour is composed of three edges. For a given problem instance, an improved bound of $\frac{k-1}{k}$ can be obtained, where k is the size of the smallest subtour in the 2-matching.

ii) In removing the edges we used an algorithm to quickly find a "good" set of edges to remove but made no effort to look for the "best" set. Although the $\frac{2}{3}$ bound is tight, using a different heuristic we could get a better result on some problem instances (of course at the cost of a longer runtime). For instance, instead of just constructing one group of disjoint sets, a few other groups can also be randomly constructed (by labeling the edges differently) and then the best result of all the iterations can be selected. If each subtour contains three edges, then there are $3^{O(|V|)}$ possible groups. As a result, there are numerous possibilities to choose from.

iii) In the last step, instead of arbitrarily connecting up the components, completing the tour can be achieved using various different techniques. A greedy approach could be used (so running Algorithm 1 except with an initial state); this would not change the worst case runtime given in Theorem 3.7. Alternatively, if the number of subtours is small, an exhaustive search could be performed. •

3.4. Discussion on Serdyukov's Algorithm

In the case that f is a modular objective function, the 2-matching algorithm presented in [1] constructs a perfect 2-matching then reduces it by removing an edge from each subtour. In doing so, the entire value of the removed edges is lost. Serdyukov's algorithm [2] improves upon this by trying to use the edges that are removed. A brief description of the algorithm is now given for the case in which the number of vertices is even. The algorithm can be extended to the case where the graph has an odd number of vertices but that is slightly more involved.

Given a graph, an optimal perfect 2-matching, F, as well as an optimal perfect 1-matching, M, are created. Denote the optimal tour as H. The edges to be removed from the subtours of F, denoted E, are chosen in a way that they can be added to M without creating any subtours. Since $f(F) \geq f(H)$ and $f(M) \geq \frac{1}{2}f(H)$, one of the two edge sets constructed is a partial tour that is at least $\frac{3}{4}$ the value of the optimal tour.

Can this be extended to the case with submodular functions? Using a linear relaxation of the objective function results in a bound of $\frac{3}{4}(1-\kappa)$ by Theorem 3.1. The downside of this method is that for high curvature the bound is close to 0. As a result, we now attempt to extend the algorithm to objective functions with a high degree of submodularity.

The main idea in Serdyukov's algorithm was that the edges removed from the 2-matching do not lose their value since they are transfered over to the 1-matching. In the case that the objective function is submodular, the marginal increase in the value of the 1-matching due to the addition of these edges is unknown. Finding the best set of edges to transfer can be phrased as the constrained maximization of a non-monotone submodular function. Specifically,

$$\max_{E \subset F} f(F \setminus E) + f(M \cup E),$$

such that E contains exactly one edge from each subtour of F. Note that the constraint is not an independence system; only sets E that satisfy $|E| =$ (number of subtours) are valid, which violates the subset property of independence systems. As a result any existing approximation results do not apply. Additionally, it appears quite challenging to relate the value of resulting partial tours back to the value of the optimum tour. Therefore, an extension for high values of curvature is left as a question for future research.

4. Extension to directed graphs

The algorithms described can also be applied to directed graphs yielding approximations for the Max-ATSP.

4.1. 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 or equal to 1 instead of checking for the degree being less than or equal to 2. Since the directed tour is a 3-extendible system (Theorem 2.12), the approximation of the greedy algorithm changes to $\frac{1}{3+\kappa}$ instead of $\frac{1}{2+\kappa}$ for the undirected case.

4.2. Tour using Matching

Instead of working with a 2-matching, the system can be modelled as the intersection of two partition matroids:

- Edge sets such that the indegree of each vertex ≤ 1.
- Edge sets such that the outdegree of each vertex ≤ 1.

This system is still 2-extendible (Remark 2.4) and so the approximation for the greedy 2-matching does not change. For the second approximation using the linear relaxation, the problem becomes the Maximum Assignment Problem (max AP). It can be solved optimally by representing the edge weights as a weight matrix $\tilde{W}$, where we set $\tilde{W}_{ii} = -\infty$, then applying the Hungarian algorithm, which has a complexity of $O(|V|^3)$ [29].

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 subtours, with the possibility of a lone vertex. Note that a subtour could potentially consist of just two vertices. Therefore, removing one edge from each subtour will result in a loss of at most $\frac{1}{2}$ instead of $\frac{1}{3}$. This follows from Theorem 3.5, setting $k = 2$. The final bound for the algorithm is therefore $\max\left\{\frac{1}{2(2+\kappa)}, \frac{1}{2}(1-\kappa)\right\}$.

5. Incorporating Costs

Consider a graph $G = (V, E, w, c)$ with edge rewards w defined as previously. Each edge has a cost given by $c : E \to \mathbb{R}_{\geq 0}$ and the cost of a set of edges is the sum of the cost of each edge in the set. We can consider the tradeoff between the reward of a set and its associated cost. A number of algorithms presented in literature seek to maximize the reward given a "budget" on the cost, i.e. find a tour T such that

$$T \in \operatorname*{argmax}_{S \in \mathcal{H}} w(S) \text{ s.t. } c(S) \leq k.$$

This involves maximizing a monotone non-decreasing submodular function over a knapsack constraint as well as an independence system constraint. However, the result of these "budgeted" solutions may not be a tour.

We are interested in finding tours that balance the reward obtained with travel cost. Therefore, we will work with a different form of the objective function defined by a weighted combination of the reward and cost. For a given value of $\beta \in [0, 1]$, solve

$$T \in \operatorname*{argmax}_{S \in \mathcal{H}} f(S, \beta) \tag{7}$$

$$f(S, \beta) = (1 - \beta)w(S) - \beta c(S). \tag{8}$$

An advantage of having this form for the objective is that the "cost trade-off" is being incorporated directly into the value being optimized. Since the cost function is modular, maximizing the negative of the cost is equivalent to minimizing the cost. So the combined objective seeks to simultaneously maximise the reward and minimize the cost.

The parameter β is used as a weighting mechanism. The case of $\beta = 0$ corresponds to ignoring costs and that of $\beta = 1$ corresponds to ignoring rewards and just minimizing the cost (i.e., the classic TSP).

Finally, we normalize w and c so they share a common range of values. Therefore, the final objective function is:

$$f(S, \beta) = \frac{1-\beta}{M_w} w(S) - \frac{\beta}{M_c} c(S) \tag{9}$$

where M_w and M_c are estimates of the maximum reward and cost, respectively.

The objective (9) is non-monotone and may be negative. To address this, consider the alternative modified cost function

$$c'(S) = |S|M - c(S), \quad M = \max_{e \in E} c(e).$$

This gives the following form for the objective function,

$$f'(S, \beta) = (1-\beta)w(S) + \beta c'(S) = f(S, \beta) + \beta |S| M, \tag{10}$$

which is a monotone non-decreasing non-negative submodular function; hence, has the advantage of offering known approximation bounds. The costs have in a sense been "inverted" and so maximizing c' still corresponds to minimizing the cost c.

Remark 5.1. Instead of using $|S|M$ as the offset, the sum of the $|S|$ largest costs in E could be used. This would not improve the worst case bound but may help to improve results in practice. •

Lemma 5.2. *For any two sets S_1 and S_2, if $f'(S_1, \beta) \geq \alpha f'(S_2, \beta)$, then $f(S_1, \beta) \geq \alpha f(S_2, \beta) + \beta M(\alpha |S_2| - |S_1|)$.*

Proof. If $f'(S_1, \beta) \geq \alpha f'(S_2, \beta)$, then by definition we have $f(S_1, \beta) + \beta |S_1| M \geq \alpha (f(S_2, \beta) + \beta |S_2| M)$. This implies that $f(S_1, \beta) \geq \alpha f(S_2, \beta) + \beta M(\alpha |S_2| - |S_1|)$, completing the proof. □

Remark 5.3. As a special case, if $|S_1| = |S_2|$, then $f(S_1, \beta) > f(S_2, \beta)$ if and only if $f'(S_1, \beta) > f'(S_2, \beta)$. Therefore, it can be deduced that over all sets of the same size, the one that maximizes (8) is the same one that maximizes (10). •

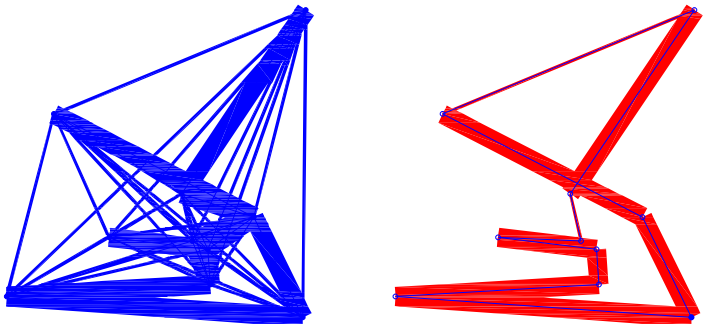
Using this result, any α-approximate algorithm maximizing f' can be related back to a bound on maximizing f. This result is given in the appendix.

5.1. Performance Bounds with Costs

Using the proposed modification, new bounds can be derived for the algorithms discussed in this paper. One thing to note is that in the case of the tour, all tours will have length $|V|$, even though the tour is not a 1-system. Therefore, we can apply Lemma 5.2 directly.

Theorem 5.4. *Using* (8) *as the objective function,*

- *Algorithm 1 outputs a tour that has a value of at least $\frac{1}{3} OPT - \frac{2}{3} \beta M |V|$,*
- *the tour based on a matching algorithm outputs a tour that has a value at least $\frac{2}{9} OPT - \frac{7}{9} \beta M |V|$,*



(a) Example graph. (b) Greedy solution.

Figure 3: An example ten vertex graph with the edges representing area cover, and the corresponding greedy approximation for the max value tour.

where M is the maximum cost of any edge.

Note that if objective (10) has a curvature of κ, then from Theorem 3.2, Algorithm 1 gives a $1/(2+\kappa)$ approximation to maximizing objective (10). Thus, the result for Algorithm 1 in Theorem 5.4 becomes $\frac{1}{2+\kappa}OPT - \frac{1+\kappa}{2+\kappa}\beta M|V|$. For the special case of $\beta = 1$, the objective function (8) is modular, and thus so is objective (10), implying that $\kappa = 0$. This is the minimum TSP. From Theorem 5.4, the greedy algorithm produces a TSP tour with length at most $\frac{1}{2}(\text{MIN-TSP} + M|V|)$, which is identical to the bound derived in [34].

6. Simulations and Applications

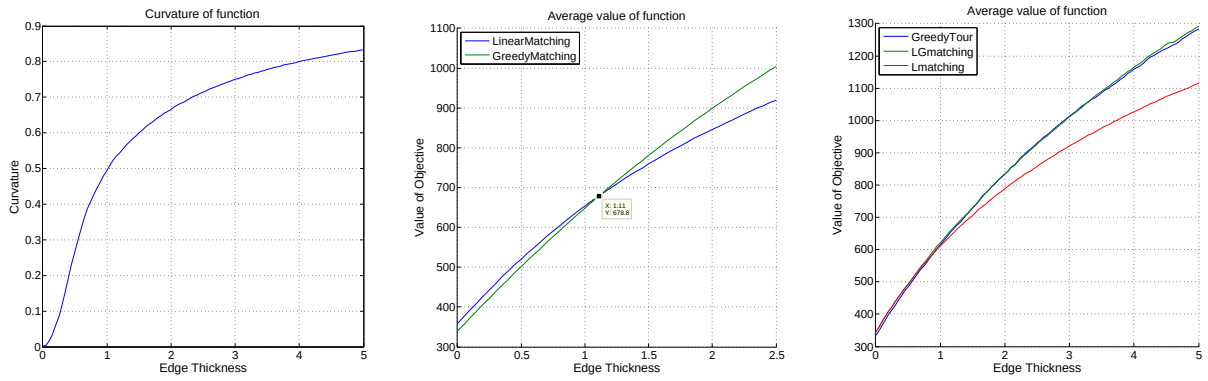
In this section we demonstrate and empirically compare the performance of the proposed algorithms on both a randomly generated submodular function, and on a submodular function arising from an application in environmental monitoring using a mobile robot.

In our implementations, we use lazy evaluations to improve the runtime of the greedy algorithm. 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 [36], the author presents an accelerated greedy algorithm that uses the decreasing marginal benefits property of submodularity to reduce the number of recalculations and thus improve the efficiency of the simple greedy approach. The idea is fairly simple. Given $A \subset B$ and two elements $e_1, e_2 \notin B$. If $\Delta(e_1|A) \leq \Delta(e_2|B)$, then $\Delta(e_1|B) \leq \Delta(e_2|B)$. Therefore, $\Delta(e_1|B)$ does not need to be calculated. The accelerated greedy algorithm has the same worst case bound as the naive version; however, empirical results have shown that it can achieve significant speed-up factors [36],[14].

6.1. Algorithm Dependence on Curvature

We begin by exploring the performance of the 2-matching based algorithm and greedy algorithm as a function of curvature. For the 2-matching algorithm we compare the greedy 2-matching presented in Section 3.3.1 with the linear relaxation in Section 3.3.2. We use a graph and submodular function that represents coverage of an environment. The particular choice of function is arbitrary, but it has the advantage that one can visualize the submodular property of the function.

The graph and submodular function are generated using the following procedure. We generate a complete graph by randomly placing vertices over a 100×100 square region. Each edge in the graph is associated with a rectangle and each rectangle is assigned a different width (or thickness)



(a) (b) (c)

Figure 4: (Left) Curvature of objective as a function of edge thickness. (Center) Comparison of the value of the 2-matching. (Right) Comparison of the value of the final tour as a function of edge thickness.

to represent different amounts of coverage. Given a set of edges S, the submodular objective function that we use is

$$w(S) = \text{coverage}(S) + \sum_{e \in S} \text{length}(e).$$

The coverage of S is simply the total area covered by the edges, and its value depends on the thickness of the edges. As the edge thickness is increased, the curvature of the function will (generally) also increase. To illustrate the function, an example of a ten vertex graph with varying edge thicknesses is given in Figure 3a. The greedy tour algorithm outputs the tour given in Figure 3b.

For the simulations in this subsection we generated graphs consisting of ten vertices and in which all edges have the same thickness (in the following subsection we show results for larger graphs). We randomly generated 20 graphs, and Figure 4a shows how the average curvature across the 20 graphs varies with edge thickness. Figure 4b compares the performance of the two proposed approaches for generating a two matching: the linear relaxation and the greedy matching. In this figure we see that when the function is linear (i.e., the edge thickness is 0), the linear approximation outperforms the greedy matching (since it is actually finding the optimal matching). As the curvature increases the gap narrows, and at a curvature of approximately 1/2, the greedy 2-matching begins to outperform the linear approximation.

In Figure 4c we compare the value of the final tour for the greedy tour and 2-matching algorithms presented in this paper. The "LGmatching" algorithm creates a greedy 2-matching as well as the linear approximation and takes the best of the two. The best edge from each subtour is removed and the tour is then constructed greedily. The "Lmatching" algorithm runs only the linear approximation to find the 2-matching. At low values of curvature the linear approximation is being used to create the tour. Eventually, greedily constructing the 2-matching becomes more rewarding and so the linear approximation is disregarded. Generally, over all the values of curvature tested, the performance of the 2-matching and the greedy algorithm are very similar.

6.2. Algorithm Comparison for High Curvature

In this section we compare the relative run-times and the performance of the algorithms for the case of graphs with high curvature. We utilize the same graph construction procedure as above, but where the objective is simply

$$w(S) = \text{coverage}(S).$$

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 edge thickness was assigned a value of 7 with probability $2/\sqrt{|E|}$, or 1 otherwise (thus, $O(|V|)$ of the edges had a high reward). The average curvature of the submodular function for each graph size was approximately 1. Also, we have obtained nearly identical results when the edge thicknesses are uniformly distributed, and so the relative performance does not seem sensitive to the exact form the submodular function.

The algorithms compared are:

- 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, then greedily connect up the tour.
  - GreedyMatching2 (GM2): Use Algorithm 2 to reduce the matching, then greedily connect up the tour.
  - GreedyMatching3 (GM3): Use Algorithm 2 to reduce the matching, then arbitrarily connect up the tour.

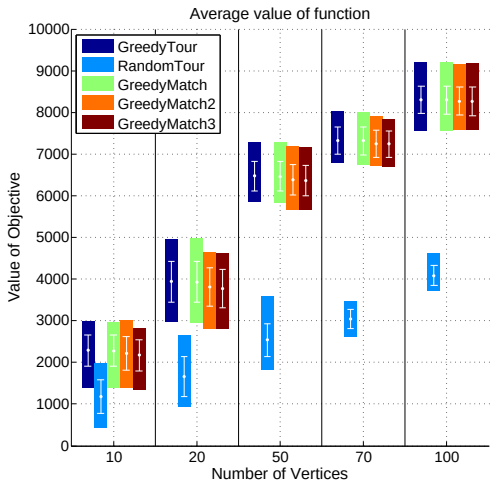
Since we do not have a complete extension of Serdyukov's to the submodular case, we have omitted it from this comparison.

The results are shown in Figure 5. The simulations were performed on a quad-core machine with a 3.10 GHz CPU and 6GB RAM. While the performance of GT and GM are similar (note that the number of ties is high especially for smaller graph sizes), GM takes significantly longer to run, due to the oracle calls needed to determine which edge to remove from each subtour. Although GM2 and GM3 had similar runtimes, 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 considerably slower than GM3 as the problem size would not be significantly reduced by first coming up with a 2-matching.

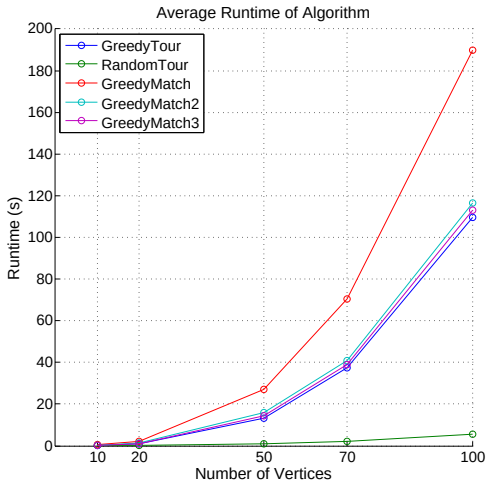
6.3. Application: Environment monitoring

The work in this paper is motivated in part by the application of monitoring an environment with a mobile robot [6, 7, 9]. In particular, consider the task of monitoring a spatial phenomenon such as the temperature profile in a region of the ocean [9]. Such phenomena are commonly approximated using Gaussian Process (GP) models, which give the correlation between the measurements of the phenomena of interest at different points in the environment [10, 11]. Given a robot traveling over the environment, the measurements made by the on-board sensors (at the visited locations) combined with the GP, aid in predicting the environmental process over the entire region of interest.

Assume that the environment is discretized into a finite set of points V. Each $v \in V$ is associated with a random variable X_v that describes the process at that point. For a multi-variate Gaussian distribution, any subset of the random variables also has a multi-variate Gaussian distribution. A



	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)



	GT	RT	GM	GM2	GM3
10	33.4	1	54.3	45.7	36.4
20	103.1	1	141.2	122.1	107.3
50	803.4	1	882.1	832.3	807
70	1903.6	1	2012.1	1942	1912.7
100	4818.9	1	4987.2	4886.8	4851.4

Figure 5: (Top left) The bars give the range of results. The white markers inside the bars show the mean and standard deviation. (Bottom left) Number of wins for each algorithm. Wins include ties (unique wins specified in parens). (Top right) Average runtime (seconds) of algorithms. (Bottom right) Average number of function calls for each algorithm.

subset A of the locations is chosen to take measurements. A method of quantifying the uncertainty in a set of measurements (given by the Gaussian random variable Y_A) is to use the entropy,

$$H(X|Y_A) = \frac{1}{2}\log(2\pi\sigma^2_{X|Y_A}).$$

The problem then becomes to choose the locations that minimize the differential entropy. An alternative measure is mutual information

$$I(X;Y_A) := H(X) - H(X|Y_A),$$

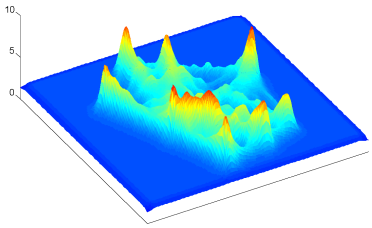
which is submodular and monotone as long as the observations Y_A are conditionally independent given X [37], [14].

For our set up, we can, in addition to discretizing the environment, discretize the edges of the graph. For any edge that is chosen as part of the path of the robot, the points on that edge are taken to be measurement locations.

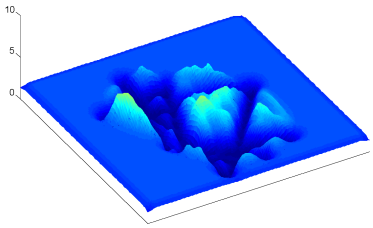
As an example, a 20×20 environment is created with grid cell size of 0.5×0.5. Any two points p_1 and p_2 have a covariance given by $e^{-k\|p_1 - p_2\|}$. The graph is generated by uniformly randomly placing 10 vertices over the region. The a priori and a posteriori variances for the greedy and random tours are shown in Figure 6. In the example shown, only the variance of the region over which the graph is defined is affected. Placing the vertices to have better coverage is a separate problem that is not addressed here.

7. Conclusions and Future Directions

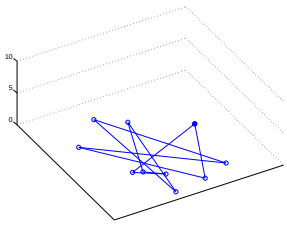
In this paper, we examine the max-TSP problem for a submodular objective function. We considered two algorithms; a greedy algorithm which achieves a $\frac{1}{2+\kappa}$ approximation and a 2-matching-



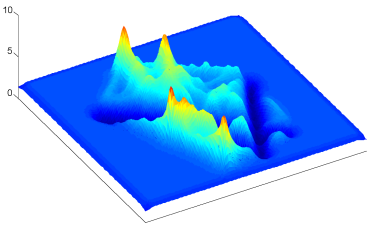
(a) A Priori Variance.

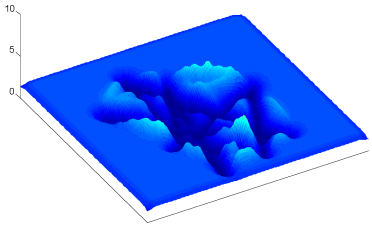


(b) Random Tour A Posteriori Variance (t=10).



(c) Greedy Tour.





(d) Greedy Tour A Posteriori Variance (t=3 and t=10).

Figure 6: Monitoring example: The graphs represent uncertainly at each point in the environment. The a priori variance is shown along with the a posteriori variance after the robot has traversed 10 edges of a random tour, as well as 3 and 10 edges of a greedy tour.

based algorithm which achieves a $\max\{\frac{2}{3(2+\kappa)}, \frac{2}{3}(1-\kappa)\}$ approximation (where κ is the curvature of the function). Both algorithms have a complexity of $O(|V|^3)$ in terms of number of oracle calls. We also discussed extending Serdyukov's algorithm. We extended these results to directed graphs and presented simulation results to empirically compare their performance as well as their dependence on curvature. We also considered an approach to integrate edge costs into the problem.

There are several directions for future work. The class of submodular functions is very broad and so adding further restrictions to the properties of the objective function may help give a better idea of how the bounds change for specific situations. For instance, utilizing some sort of locality property to define relationships between edges. Similarly, more practical definitions of curvature can be investigated, such as the notion of independence curvature we introduced in Remark 2.6. Another direction of research would be extending other algorithms. Although an attempt was made to extend Seryukov's algorithm, the question still stands as to whether or not the strategy makes sense in the case of high curvature. Additionally, there are many other simple strategies that could also be extended such as best neighbour or insertion heuristics. One other problem to investigate would be to the case where multiple tours are needed (such as with multiple patrolling robots).

Appendix A. Optimization bounds incorporating costs

We first look at how the result of maximizing (10) relates to the optimal value of (8). This result is then used to derive an approximation factor for the greedy algorithm.

Theorem A.1. *Consider a p-system and the functions f and f' as defined in* (8) *and* (10) *respectively. An α-approximation to the problem $\max_{S\in\mathcal{F}} f'(S,\beta)$, corresponds to an approximation*

of $\alpha OPT - \left(1 + \frac{p-2}{p}\alpha\right)\beta Mn$*, for the problem* $\max_{S \in \mathcal{F}} f(S, \beta)$*, where* n *is the size of the maximum cardinality basis.*

Proof. Let S be the solution obtained by a α-approximation algorithm to f'. Let T be the optimal solution using f'. Let Z be the optimal solution using f. Note the following inequality:

$$|A| \leq |B|p \implies \alpha|B| - |A| \geq |B|(\alpha - p) \geq |A|(\frac{\alpha}{p} - 1).$$

By using the property of p-systems that for any two bases A and B, $|A| \leq p|B|$, and by applying Lemma 5.2 to the fact that $f'(S, \beta) \geq \alpha f'(T, \beta)$,

$$f(S, \beta) \geq \alpha f(T, \beta) + \beta M \left(\alpha|T| - |S|\right) \geq \alpha f(T, \beta) - \beta M|S| \left(1 - \frac{\alpha}{p}\right) \geq \alpha f(T, \beta) - \beta Mn \left(1 - \frac{\alpha}{p}\right).$$

Deriving a similar expression using the fact that $f'(T, \beta) \geq f'(Z, \beta)$ and substituting, we obtain

$$f(S, \beta) \geq \alpha f(Z, \beta) - \alpha\beta M|T| \left(1 - \frac{1}{p}\right) - \beta Mn \left(1 - \frac{\alpha}{p}\right) \geq \alpha f(Z, \beta) - \beta Mn \left(1 + \frac{p-2}{p}\alpha\right).$$

□

Remark A.2. In the special case of a 1-system (this includes matroids), or more generally any problem where the output to the algorithm will always be the same size, we have $|S| = |T| = |Z|$ and also $T = Z$ following from Remark 5.3. This means that if an algorithm outputs a solution with relative error of α using f', that same solution will give an approximation of $f(S, \beta) \geq \alpha \text{OPT} - (1 - \alpha)\beta Mn$, for the function f. •

Appendix A.1. Greedy with Costs

Corollary A.3. *The problem* $\max_{S \in \mathcal{F}} f(S, \beta)$ *can be approximated to* $(p+1)^{-1}OPT - \beta(1 + \epsilon)Mn$ *(where* $\epsilon \in [-\frac{1}{2}, 1)$ *is a function of* p*) using a greedy algorithm.*

Proof. Run the greedy algorithm using $f'(S, \beta)$ to obtain the set T_G. Let T and Z be defined as in Theorem A.1. Since f' is a non-negative monotone function, $f'(T_G, \beta) \geq \frac{1}{p+1} f'(T, \beta)$. So applying Theorem A.1,

$$f(T_G, \beta) \geq \frac{1}{p+1} f(Z, \beta) - \beta Mn \left(1 + \frac{p-2}{p(p+1)}\right) \tag{A.1}$$

□

From Remark 5.3, the following can be deduced about the greedy algorithm.

Theorem A.4. *Let* G_1 *be the greedy solution obtained maximizing* (8) *and* G_2 *be the greedy solution maximizing* (10)*. Then* $G_1 = G_2$*.*

Proof. At each iteration i of the greedy algorithm, we are finding the element that will give the maximum value for a set of size $i + 1$. Since comparison is being done between sets of the same size, the same element will be chosen at each iteration. □

Summarizing, for any p-system, applying the greedy algorithm using the non-monotone objective (8) leads to the same set as using (10) and the resulting bound is given by (A.1).

References

[1] M. L. Fisher, G. L. Nemhauser, L. A. Wolsey, An analysis of approximations for finding a maximum weight hamiltonian circuit, Operations Research 27 (4) (1979) pp. 799–809.

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

[3] K. Paluch, M. Mucha, A. Madry, A 7/9 - approximation algorithm for the maximum traveling salesman problem, in: I. Dinur, K. Jansen, J. Naor, J. Rolim (Eds.), Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 298–311.

[4] R. Hassin, S. Rubinstein, Better approximations for max TSP, Information Processing Letters 75 (1998) 181–186.

[5] A. Van Zuylen, Deterministic approximation algorithms for the maximum traveling salesman and maximum triangle packing problems, Discrete Applied Mathematics 161 (13-14) (2013) 2142–2157.

[6] S. L. Smith, M. Schwager, D. Rus, Persistent robotic tasks: Monitoring and sweeping in changing environments, IEEE Transactions on Robotics 28 (2) (2012) 410–426.

[7] J. Clark, R. Fierro, Mobile robotic sensors for perimeter detection and tracking, ISA Transactions 46 (1) (2007) 3–13.

[8] D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. Mclain, S.-M. Li, R. Mehra, Cooperative forest fire surveillance using a team of small unmanned air vehicles, International Journal of Systems Sciences 37 (6) (2006) 351–360.

[9] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, G. S. Sukhatme, Persistent ocean monitoring with underwater gliders: Adapting sampling resolution, Journal of Field Robotics 28 (5) (2011) 714–741.

[10] X. Lan, M. Schwager, Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 13), 2013, pp. 2407–2412.

[11] C. Guestrin, A. Krause, A. Singh, Near-optimal sensor placements in Gaussian processes, in: Int. Conf. on Machine Learning, Bonn, Germany, 2005.

[12] A. Singh, A. Krause, C. Guestrin, W. Kaiser, Efficient informative sensing using multiple robots, Journal of Artificial Intelligence Research 34 (2009) 707–755.

[13] J. Binney, A. Krause, G. Sukhatme, Informative path planning for an autonomous underwater vehicle, in: Robotics and Automation (ICRA), 2010 IEEE International Conference on, 2010, pp. 4791–4796.

[14] D. Golovin, A. Krause, Adaptive submodularity: Theory and applications in active learning and stochastic optimization, Journal of Artificial Intelligence Research 42 (2011) 427–486.

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

[16] A. Schrijver, A combinatorial algorithm minimizing submodular functions in strongly polynomial time, Journal of Combinatorial Theory, Series B 80 (2) (2000) 346 – 355.

[17] S. Iwata, L. Fleischer, S. Fujishige, A combinatorial strongly polynomial algorithm for minimizing submodular functions, J. ACM 48 (4) (2001) 761–777.

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

[19] M. Conforti, 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 7 (3) (1984) 251 – 274.

[20] 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.

[21] Y. Filmus, J. Ward, A tight combinatorial algorithm for submodular maximization subject to a matroid constraint, in: Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on, 2012, pp. 659–668.

[22] U. Feige, V. S. Mirrokni, J. Vondrak, Maximizing non-monotone submodular functions, SIAM Journal on Computing 40 (4) (2011) 1133–1153.

[23] J. Lee, V. S. Mirrokni, V. Nagarajan, M. Sviridenko, Non-monotone submodular maximization under matroid and knapsack constraints, in: ACM Symposium on Theory of Computing, Bethesda, MD, 2009, pp. 323–332.

[24] J. Lee, M. Sviridenko, J. Vondrák, Submodular maximization over multiple matroids via generalized exchange properties, Math. Oper. Res. 35 (4) (2010) 795–806.

[25] A. Gupta, A. Roth, G. Schoenebeck, K. Talwar, Constrained non-monotone submodular maximization: Offline and secretary algorithms, in: Internet and Network Economics, Springer, 2010, pp. 246–257.

[26] G. Calinescu, C. Chekuri, M. Pal, J. Vondrak, Maximizing a monotone submodular function subject to a matroid constraint, SIAM Journal on Computing 40 (6) (2011) 1740–1766.

[27] J. Vondrák, C. Chekuri, R. Zenklusen, Submodular function maximization via the multilinear relaxation and contention resolution schemes, in: Proceedings of the 43rd annual ACM symposium on Theory of computing, 2011, pp. 783–792.

[28] S. T. Jawaid, S. L. Smith, The maximum traveling salesman problem with submodular rewards, in: American Control Conference, Washington, DC, 2013, pp. 4003–4008.

[29] B. Korte, J. Vygen, Combinatorial Optimization: Theory and Algorithms, 4th Edition, Vol. 21 of Algorithmics and Combinatorics, Springer, 2007.

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

[31] G. L. Nemhauser, L. A. Wolsey, M. L. Fisher, An analysis of approximations for maximizing submodular set functions - I, Mathematical Programming 14 (1978) 265–294.

[32] U. Feige, A threshold of ln n for approximating set cover, J. ACM 45 (4) (1998) 634–652.

[33] J. Mestre, Greedy in approximation algorithms, in: Y. Azar, T. Erlebach (Eds.), Algorithms ESA 2006, Vol. 4168, Springer Berlin / Heidelberg, 2006, pp. 528–539.

[34] T. A. Jenkyns, The greedy travelling salesman's problem, Networks 9 (4) (1979) 363–373.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 2nd Edition, MIT Press, 2001.

[36] M. Minoux, Accelerated greedy algorithms for maximizing submodular set functions, in: J. Stoer (Ed.), Optimization Techniques, Vol. 7 of Lecture Notes in Control and Information Sciences, Springer Berlin / Heidelberg, 1978, pp. 234–243.

[37] A. Krause, C. Guestrin, Near-optimal nonmyopic value of information in graphical models, in: Twenty-first conference on uncertainty in artificial intelligence (UAI), 2005, p. 5.