The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Submodular Function Maximization over Graphs via Zero-Suppressed Binary Decision Diagrams

Shinsaku Sakaue, Masaaki Nishino, Norihito Yasuda
NTT Communication Science Laboratories, NTT Corporation
{sakaue.shinsaku, masaaki.nishino, yasuda.n}@lab.ntt.co.jp

Abstract

Submodular function maximization (SFM) has attracted much attention thanks to its applicability to various practical problems. Although most studies have considered SFM with size or budget constraints, more complex constraints often appear in practice. In this paper, we consider a very general class of SFM with such complex constraints (e.g., an s-t path constraint on a given graph). We propose a novel algorithm that takes advantage of *zero-suppressed binary decision diagrams*, which store all feasible solutions efficiently thus enabling us to circumvent the difficulty of determining feasibility. Theoretically, our algorithm is guaranteed to achieve $(1-c)$-approximations, where c is the *curvature* of a submodular function. Experiments show that our algorithm runs much faster than exact algorithms and finds better solutions than those obtained by an existing approximation algorithm in many instances. Notably, our algorithm achieves better than a 90%-approximation in all instances for which optimal values are available.

1 Introduction

Submodular function maximization (SFM) can model a large number of real-world problems (e.g., sensor placement, feature selection, and document summarization), and so continues to be a dominant subject of study. Given finite set V, set function $f : 2^V \to \mathbb{R}$ is said to be *submodular* if $f(S)+f(T) \geq f(S \cup T)+f(S \cap T)$ for all $S, T \in 2^V$, and *monotone* if $f(T) \geq f(S)$ for any $S \subseteq T$. Although SFM is NP-hard in general, greedy-based algorithms are known to offer strong approximation guarantees; some of them consider a size constraint, a budget constraint, or a matroid constraint. However, it has remained hard to design efficient approximation algorithms for SFM with more complex constraints. A typical such problem is the *submodular orienteering problem* (SOP) (Chekuri and Pal 2005):

SOP: Given budget $B > 0$ and a graph with edge costs and two specified vertices s, t, we seek to find an s-t path whose cost is at most B to maximize a submodular objective function of the set of vertices visited by the s-t path. SOP is important since it models various practical problems such as path-planning problems with robotic sensors (Singh et al. 2009), travel planning (Zeng et al. 2015), and door-to-door marketing (Zhang and Vorobeychik 2016).

SFM with complex constraints also arises in the context of wireless sensor network (WSN) design. One such example is the problem considered in (Krause et al. 2006). Given a graph that represents the connectability of sensor nodes, we seek a connected tree to maximize a submodular function that measures the informativeness of the obtained tree. Each edge in the graph has a cost, which represents the energy consumed when the two sensors placed on its ends communicate with each other, and the total cost of the obtained tree must be less than or equal to a given budget, B. In realistic WSN design problems, however, more complex constraints often appear. Below we list two such examples, which we call the *submodular Steiner tree problem* (SSTP) and the *submodular vertex-connected network problem* (SVCNP):

SSTP: As in (Ke, Liu, and Tsai 2011), many sensor placement problems have some critical areas that we are particularly interested in; a vertex that corresponds to a critical area is called a *terminal*. In SSTP, we seek a *Steiner tree* that covers all terminals under the budget constraint to maximize an objective function.

SVCNP: We aim to obtain a robust WSN that has no *cut vertex*, which is a vertex whose removal breaks the connectivity of the network; we call a network with no cut vertex *vertex-connected* (VC) network. Since the failure of a sensor placed on a cut vertex blocks data transmission, how to design a VC network has been extensively studied (Liu et al. 2006; Xiong and Li 2010). In SVCNP, we seek a VC network under the budget constraint to maximize an objective function.

The difficulty of the above problems is that the constraints are rigid. Given a subset of vertices, determining the feasibility of the subset in SOP and SSTP reduces to NP-complete problems: the *Hamilton path problem* and *Steiner tree problem*, respectively. The feasibility determination in SVCNP is known to be as hard as the *Hamilton cycle problem* if the edge costs are uniform, and it becomes more difficult if the costs are not uniform, for which no existing algorithm has been developed; SVCNP is empirically more challenging than SOP and SSTP due to the difficult feasibility determination. If we apply a naive iterative algorithm (e.g., the greedy algorithm) to such problems, we need to solve an NP-complete problem every time a solution is updated.

1.1 Our Contribution

We propose a novel approximation algorithm for SFM defined over graphs, including many hard problems such as SOP, SSTP, and SVCNP. Formally, given graph $G = (V, E)$, where V and E are the vertex set and edge set, respectively, and monotone submodular function $f : 2^V \to \mathbb{R}_+$ such that $f(\emptyset) = 0$, we consider the following problem:

$$(1) \qquad \underset{X \subseteq E}{\text{maximize}} \; f(V(X)) \quad \text{subject to } X \in \mathcal{F},$$

where $\mathcal{F} \subseteq 2^E$ is a feasible set that consists of edge subsets with certain substructures, and $V(X) := \{u \in V \mid (u, v) \text{ or } (v, u) \in X \text{ for some } v \in V\}$ is the set of ends of all $e \in X$. Problem (1) can express various constrained SFM problems; for example, an SOP can be written as problem (1) whose $\mathcal{F}$ is the set of all s-t paths whose cost is at most B.

The constraint in problem (1) can be quite complex, and so we must devise a way to handle the constraint if we are to develop a truly efficient algorithm. To do this, our algorithm takes advantage of the *zero-suppressed binary decision diagram* (ZDD) (Minato 1993). A ZDD is defined as a directed acyclic graph (DAG) and stores each feasible solution $X \in \mathcal{F}$ as a path connecting two specified nodes, called a *root node* and a *true node*. Thus, in our algorithm, problem (1) is reduced to SFM on a DAG, by which we circumvent the difficulty of determining feasibility for the rigid constraints.

Our algorithm is proved to achieve $(1-c)$-approximations, where c is the *curvature* of the submodular function (Conforti and Cornuéjols 1984); it is defined as $c := 1 - \min_{v \in V} f(v \mid V \backslash \{v\})/f(v)$, where $f(v \mid S) := f(S \cup \{v\}) - f(S)$ for any $v \in V$ and $S \subseteq V$. The curvature value is bounded from above for many practical submodular functions (Maehara et al. 2017; Sharma, Kapoor, and Deshpande 2015); in such cases our $(1 - c)$-approximation algorithm is guaranteed to find nearly optimal solutions. Unfortunately, since ZDD size can be exponential in $|V|$, our algorithm generally incurs exponential computation time. However, ZDDs that appear in practice are often of tractable size as confirmed in previous studies (e.g., (Minato 1993)) and our experiments. In the experiments, we apply our algorithm to three sensing tasks formulated as SOP, SSTP, and SVCNP, respectively; notably, to the best of our knowledge, our algorithm is the first practical approximation algorithm that is applicable to SVCNP. We show that our algorithm achieves at least a 90%-approximation in all instances for which optimal values are available, outperforming an existing approximation algorithm in most cases. Our algorithm is also shown to be much faster than exact algorithms.

1.2 Related Work

SFM has attracted much attention due to its applicability to various real-world problems (e.g., sensor placement (Krause et al. 2008; Krause, Singh, and Guestrin 2008), feature selection (Thoma et al. 2009), and document summarization (Lin and Bilmes 2010)). Most previous studies (Buchbinder et al. 2015; Conforti and Cornuéjols 1984; Nemhauser, Wolsey, and Fisher 1978; Sviridenko 2004) consider SFM with a size constraint, a knapsack constraint, or a matroid constraint, for which the greedy-based strategy is effective. On the other hand, SFM with more complex constraints (e.g., SOPs, SSTPs, and SVCNPs) often appears in practice, for which naive greedy-based methods do not work.

SOP has been extensively studied because of its applicability to many problems (Singh et al. 2009; Zeng et al. 2015; Zhang and Vorobeychik 2016). SOP was proposed in (Chekuri and Pal 2005) as an extension of the *orienteering problem*, a classical combinatorial optimization problem. For SOP, they proposed a quasi-polynomial time approximation algorithm, called the *recursive greedy algorithm*. Singh et al. scaled up the recursive greedy algorithm and applied it to path-planning problems for robot-based environment sensing. Recently, a faster bi-criterion approximation algorithm for SOP called the *generalized cost-benefit greedy algorithm* (GCB) was proposed in (Zhang and Vorobeychik 2016); it uses the cost-benefit greedy strategy with approximate cost computation. A special case of SOP is also considered in (Zeng et al. 2015). The authors studied the travel route search problem of maximizing the satisfaction of a traveler's preference, which is an SOP with a *keyword coverage* (KC) objective function. They proposed an A* algorithm with a heuristic function that suits the KC function. While the algorithm is exact (i.e., it always finds an optimal solution), its time complexity is generally exponential in $|E|$.

SFM with a budget and tree constraint is studied in (Krause et al. 2006); it is an important problem in WSN design. For this problem they proposed an efficient approximation algorithm called *pSPIEL*.

In many real-world WSN design problems, our interest is rarely spread over the whole target areas; rather we have some critical areas that are of particular interest. Such a situation is modeled as a network design problem with a Steiner tree constraint in (Ke, Liu, and Tsai 2011). However, they did not consider the submodularity of objective functions, which is known to be essential in sensor placement problems (Krause, Singh, and Guestrin 2008). Thus we consider SSTP, which is an SFM problem with a budget and Steiner tree constraints. This problem is very hard due to its rigid constraints, and thus no algorithms have been studied.

SVCNP is also a variant of the problem considered in (Krause et al. 2006). In many real-world scenarios it is important to obtain a robust WSN, and thus finding VC networks has been an attractive research subject (Liu et al. 2006; Xiong and Li 2010). Although many existing approximation algorithms consider computing the smallest VC network on graphs with uniform edge costs (Garg, Santosh, and Singla 1993; Heeger and Vygen 2016), no algorithm considers non-uniform edge costs, which frequently appear in practice. This implies no existing algorithm can determine the feasibility of a given vertex subset efficiently in SVCNPs. The SVCNP is very challenging due to the difficulty of feasibility determination, and thus no effective algorithm has been developed.

Thanks to the recent advances in algorithms for constructing *decision diagrams* (DDs), optimization methods using DDs are receiving much attention (Bergman et al. 2016; Coudert 1997; Morrison, Sewell, and Jacobson 2016). Those methods are advantageous in that their use of DDs allows them to deal with complex constraints via efficient enumeration of all feasible solutions. However, most existing optimization methods with DDs consider linear objective functions, whereas nonlinear objective functions, including submodular functions, are of critical importance in practice. To the best of our knowledge, our algorithm is the first nonlinear optimization algorithm that uses ZDDs (Minato 1993), which is a kind of DD, to deal with complex constraints. ZDDs are known to be suitable for storing specific graph substructures (e.g., s-t paths), thus enabling us to develop an efficient algorithm for SFM over graphs.

2 ZDD-based Algorithm

This section presents the approximation algorithm for problem (1). We first elucidate the properties of the ZDD that we use to store the set of all feasible solutions $\mathcal{F} \subseteq 2^E$ in problem (1). We then describe the algorithm that searches for an approximate solution on the ZDD.

2.1 Zero-suppressed Binary Decision Diagrams

For a given $\mathcal{F} \subseteq 2^E$, we define a ZDD that stores all $X \in \mathcal{F}$ as a DAG, and denote it by $Z_\mathcal{F} = (N, A)$. To avoid confusing $G = (V, E)$ with $Z_\mathcal{F} = (N, A)$, we refer to the elements in V and E as *vertex* and *edge*, respectively, whereas the elements in N and A are called *node* and *arc*, respectively. Furthermore, we refer to a path on $Z_\mathcal{F}$ as a *route*. Below we detail the properties of the ZDD $Z_\mathcal{F}$ that are needed to understand subsequent discussions.

ZDD $Z_\mathcal{F} = (N, A)$ is a DAG with root node $r \in N$, and two terminal nodes $\mathbf{0}, \mathbf{1} \in N$, called the *false node* and *true node*, respectively. Let $I := E \cup V$ be the set of all edges and vertices of G that is totally ordered (see (Kawahara et al. 2017b) for details of the ordering); the i-th element of I is denoted by I_i. Each node $n \in N \backslash \{\mathbf{0}, \mathbf{1}\}$ is labeled with $I_i \in I$; we denote the label of n by $l(n) \in I$ and let $l(r) = I_1$. Every $n \in N \backslash \{\mathbf{0}, \mathbf{1}\}$ has two outgoing arcs called 0-*arc* and 1-*arc*. If $n \in N \backslash \{\mathbf{0}, \mathbf{1}\}$ is labeled with I_i, each arc outgoing from n points to $\mathbf{0}, \mathbf{1}$, or $n \in N \backslash \{\mathbf{0}, \mathbf{1}\}$ labeled with I_j $(j > i)$; this guarantees that each label $I_i \in I$ appears at most once on any route in $Z_\mathcal{F}$. We define $\mathcal{R}_{n,n'} \subseteq 2^A$ as the set of all routes from $n \in N$ to $n' \in N$ and $\mathcal{R} := \bigcup_{n,n' \in N} \mathcal{R}_{n,n'}$ as the collection of all routes in $Z_\mathcal{F}$. Any route $R \in \mathcal{R}$ is associated with the following edge subset $E_R \subseteq E$ and vertex subset $V_R \subseteq V$:

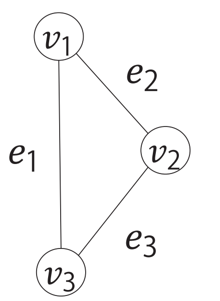
$$E_R := \{l(n) \in E \mid (n, n') \in R \text{ is 1-arc. } \},$$
$$V_R := \{l(n) \in V \mid (n, n') \in R \text{ is 1-arc. } \}.$$

Namely, when proceeding along route R, we add $l(n) \in I$ to E_R or V_R only if $(n, n') \in R$ is 1-arc. There is the following one-to-one correspondence between $X \in \mathcal{F}$ and $R \in \mathcal{R}_{r,\mathbf{1}}$:

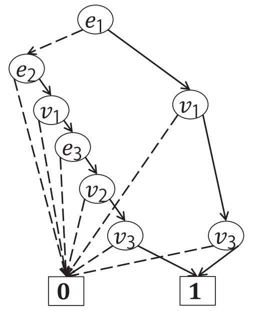
$$(2) \qquad \mathcal{F} = \{E_R \subseteq E \mid R \in \mathcal{R}_{r,\mathbf{1}}\}.$$

Thus $\mathcal{F} \subseteq 2^E$ is stored as $\mathcal{R}_{r,\mathbf{1}}$ in $Z_\mathcal{F}$. Furthermore, $V(E_R) = V_R$ holds for any $R \in \mathcal{R}_{r,\mathbf{1}}$.

Figure 1 shows an example of a ZDD that enumerates all s-t paths in the graph. In the ordinary definition of ZDDs (Minato 1993), each node is labeled using only $e \in E$. However, here we employ ZDDs whose nodes are labeled with $v \in V$ or $e \in E$ as described above since the vertex labels are necessary for our algorithm. Such ZDDs can be constructed by the



(a) Graph



(b) ZDD

Figure 1: Graph $G = (V, E)$ with $s = v_1$ and $t = v_3$, and a ZDD that stores all s-t paths in G; each node $n \in N \backslash \{\mathbf{0}, \mathbf{1}\}$ is labeled by $l(n) \in I$. The solid arcs are 1-arcs and the dashed arcs are 0-arcs. Note that we have $\mathcal{F} = \{E_R \subseteq E \mid R \in \mathcal{R}_{r,\mathbf{1}}\} = \{\{e_1\}, \{e_2, e_3\}\}$ and $\{V_R \subseteq V \mid R \in \mathcal{R}_{r,\mathbf{1}}\} = \{\{v_1, v_3\}, \{v_1, v_2, v_3\}\}$.

algorithm shown in (Kawahara et al. 2017b). Unfortunately, the size of ZDD $Z_\mathcal{F} = (N, A)$, which we define by $|N|$, generally increases exponentially with $|V|$. More precisely, $|N| \le (|E|+2|V|)d^{w(G)}$ holds where $w(G)$ is the pathwidth of G ($w(G) \le |V|$) and d is a parameter that appears when constructing a ZDD ($d = O(|w(G)|)$) as shown in (Suzuki and Minato 2016); the relationship between pathwidth and ZDD size is detailed in (Inoue and Minato 2016). Fortunately, ZDD size is often tractable in many applications; in particular, it tends to be small if pathwidth $w(G)$ is small. For example, as shown in Section 3.1, a ZDD with about 3.0×10^7 nodes can store about 6.0×10^{16} feasible solutions of an SOP, which allows our algorithm to find an approximate solution within 4.0×10^3 seconds.

2.2 GreedyDP

We here show a $(1 - c)$-approximation algorithm for problem (1). Let $Z_\mathcal{F} = (N, A)$ be a ZDD that stores $\mathcal{F}$. We define $f(R) := f(V_R)$ for all $R \in \mathcal{R}$. From the properties of ZDDs, problem (1) can be written as the following SFM on the ZDD:

$$(3) \qquad \underset{R \subseteq A}{\text{maximize}} \ f(R) \quad \text{subject to } R \in \mathcal{R}_{r,\mathbf{1}}.$$

Note that, if we obtain a $(1 - c)$-approximate solution $R \in \mathcal{R}_{r,\mathbf{1}}$ for problem (3), then $E_R \subseteq E$ is a $(1 - c)$-approximate solution for problem (1) due to the one-to-one correspondence (2).

For every $n \in N \backslash \{r\}$, we let Preds$(n) \subseteq N$ be the set of all predecessors of n, and we define $A_n := \{(n', n) \in A : n' \in \text{Preds}(n)\}$. We present here Algorithm 1, which finds a $(1 - c)$-approximate solution $R_{r,\mathbf{1}} \in \mathcal{R}_{r,\mathbf{1}}$ for problem (3). The search strategy of Algorithm 1 is based on the well-known dynamic programming (DP) approach that is used for finding the longest path on a DAG (see, e.g., (Sedgewick and Wayne 2011)). On the other hand, Algorithm 1 is greedy in that, once we obtain a route, we do not change it, and a new route $R_{r,n}$ is obtained by the greedy rule over A_n. Thus we call it *GreedyDP*. Note that, once we have constructed

Algorithm 1 GreedyDP($Z_{\mathcal{F}}, f$)

```
1: $U = N \setminus \{r\}$.
2: Create a topological order of all $n \in U$.
3: $R_{r,n} = \emptyset$ for all $n \in N$.
4: while $U \neq \emptyset$ do
5:     Let $n \in U$ be the first node in the topological order.
6:     $(n'', n) = \operatorname*{argmax}_{(n',n) \in A_n} f(R_{r,n'} \cup (n', n))$.
7:     $R_{r,n} = R_{r,n''} \cup (n'', n)$.
8:     $U = U \setminus \{n\}$.
9: end while
10: return $R_{r,\mathbf{1}}$.
```

$Z_{\mathcal{F}}$ for a feasible set $\mathcal{F}$, it can be reused when maximizing any objective functions in the same $\mathcal{F}$ using GreedyDP. This makes our algorithm efficient particularly when solving SFM problems repeatedly in the same feasible set; for example, our algorithm is really efficient when solving path planning problems for a robotic sensor that must perform sensing on a daily basis or more frequently in the same target space.

Algorithm 1 first creates a topological order for $U = N \setminus \{r\}$ so that, for all $(n', n) \in A$, n' comes before n. Algorithm 1 then finds a route $R_{r,n}$ for each $n \in N \setminus \{r\}$ in the topological order. At each iteration, $U \subseteq N \setminus \{r\}$ represents the set of nodes that have not been visited yet. Thanks to the topological sorting, we have $\mathrm{Preds}(n) \subseteq N \setminus U$ for $n \in N$ chosen in Step 5. We can easily check that Algorithm 1 requires at most $O(|N|)$ function value evaluations. The following theorem provides the approximation guarantee for GreedyDP. For proof, see the appendix.

Theorem 1. *If $R^*_{r,\mathbf{1}}$ is an optimal route, GreedyDP finds a route $R_{r,\mathbf{1}}$ satisfying $f(R_{r,\mathbf{1}}) \geq (1-c) f(R^*_{r,\mathbf{1}})$.*

3 Experiments

We conduct experiments on three sensing tasks to assess the performance of our algorithm. One is the path planning problem for a robotic sensor, which can be written as an SOP, and the others are indoor WSN design problems, which can be formulated as an SSTP and SVCNP, respectively. All experiments were performed on a computer equipped with 128 GB RAM and Xeon E5 3.1 GHz CPU. Algorithm 1 and the alternative algorithms that we used for comparison were implemented in Python. We used the C++ library (Suzuki 2016) to construct the ZDDs in our algorithm. Throughout the experiments, the limit of execution time is 10^4 seconds.

3.1 Path Planning for a Mobile Robotic Sensor

We consider the following path planning problem as considered in (Singh et al. 2009; Zhang and Vorobeychik 2016). As shown in Figures 3 (a) and (b), the 2-D target space is discretized by the grid graph $G = (V, E)$. We seek a path in G that visits a subset of vertices optimally so that a mobile robot equipped with sensors can collect as much information as possible. Using a grid graph is a natural approach when obstacles, such as buildings, are placed regularly in the target space. We discretize the target space by using 5×9 and 7×13 grids. For simplicity, we suppose that the cost of traversing one edge is always 1, and the obtained path $X \subseteq E$ must satisfy the budget constraint $|X| \leq B$ for some $B > 0$. To assess the scalability of algorithms, we consider all feasible budget values: $B = 1, 3, \dots, 43$ for the 5×9 grid and $B = 1, 3, \dots, 89$ for the 7×13 grid, where $B = 43$ and $B = 89$ are the length of the longest s-t paths in the 5×9 and 7×13 grids, respectively.

This experiment uses air quality sensor data monitored at 11 stations in Shanghai, China (Zheng, Liu, and Hsieh 2013), and we focus on PM2.5 data. Given the data obtained at the 11 stations, we consider collecting additional information using a mobile robot equipped with sensors. The robot must start from and return to specified locations.

To measure the informativeness of obtained paths, we employ the entropy function of the selected vertices, which is known to be a monotone submodular function (Krause, Singh, and Guestrin 2008). Let V_0 be the set of locations of the 11 stations. Starting from $S = V_0$, if S is the current set of vertices that have already been visited, the marginal gain of visiting new vertex $v \notin V \setminus S$ is expressed by the conditional entropy $H(v \mid S) := (1/2) \log(2\pi e \sigma^2_{v|S})$, where $\sigma^2_{v|S} := K(v,v) - K(v,S)K(S,S)^{-1}K(S,v)$ is the conditional covariance. The matrices $K(S,T) \in \mathbb{R}^{|S| \times |T|}$ are defined by a kernel function; we here use an ordinary Gaussian kernel (see, e.g., (Bishop 2006)). The parameters of $K(\cdot, \cdot)$ are fitted to the data obtained at the 11 stations.

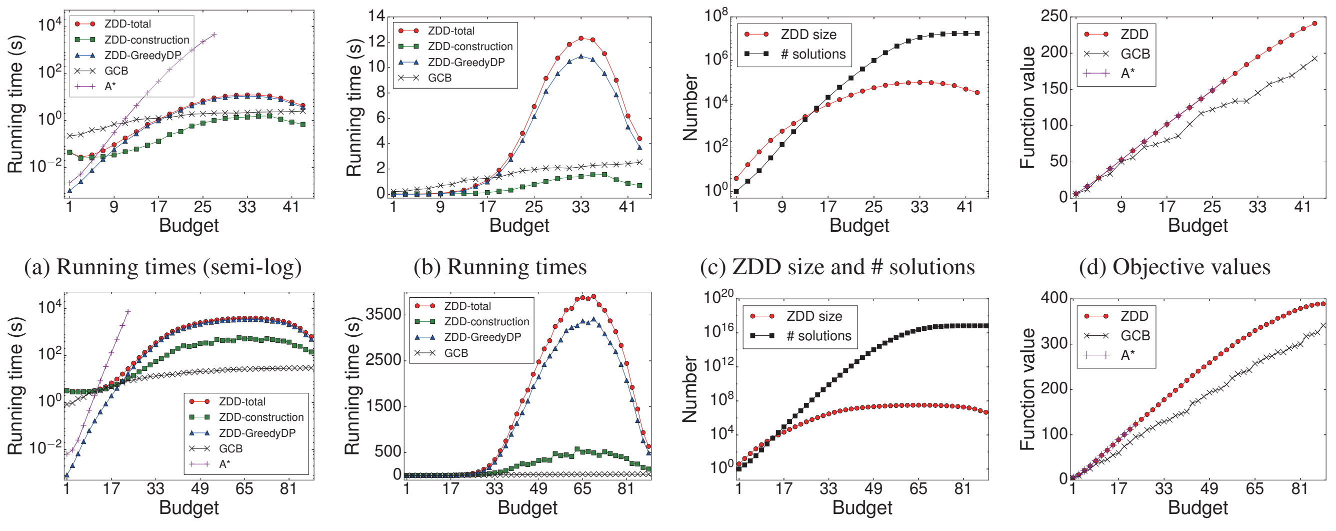
In this experiment we use the following two algorithms to benchmark our algorithm:

GCB: A generalized cost-benefit greedy algorithm (Zhang and Vorobeychik 2016). Given current solution $S \subseteq V$, GCB considers adding new vertex $v \in V \setminus S$ sequentially in non-increasing order of the cost-benefit ratios; to do this we need an approximate cost of S, which we compute here using the nearest neighbor algorithm as in the original paper. We note that this method does not always find a simple s-t path as a solution; namely, it sometimes finds a walk that visits the same vertices twice or more.

A* algorithm: An exact A* algorithm based on (Zeng et al. 2015). Since heuristic function $h(\cdot)$ used in the original A* algorithm is designed for the KC function, it is not applicable to our setting. Thus we considered two alternatives of $h(\cdot)$. Given current vertex $v \in V$ and solution $X \subseteq E$, both heuristic function values $h(X)$ bound

$$\max_{S \subseteq W : |S| \leq B - |X|} f(S \mid V(X)) \tag{4}$$

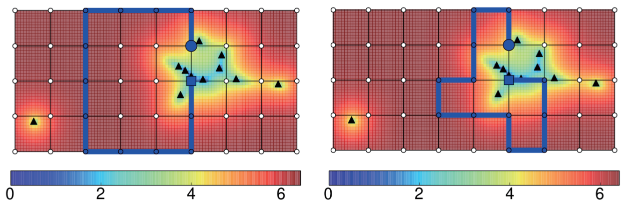
from above, where W is a set of all vertices that are reachable from v with at most $B - |X|$ cost; the A* algorithm is exact if $h(X)$ is always larger than or equal to (4). The first one is a natural generalization of the one used in (Zeng et al. 2015); letting $h'(X)$ be the objective value achieved by the greedy algorithm, which achieves a $(1 - 1/e)$-approximation for problem (4), we set $h(X) = h'(X)/(1 - 1/e)$. The second one is from (Chen, Chen, and Weinberger 2015); we let $h(X) = \max_{S \subseteq W : |S| \leq B - |X|} \sum_{v \in S} f(v \mid V(X))$, which is at least as large as (4) thanks to the submodularity. We experimentally observed that A* search with the second $h(\cdot)$ is much faster, and thus we employed it.



(a) Running times (semi-log) (b) Running times (c) ZDD size and # solutions (d) Objective values

(e) Running times (semi-log) (f) Running times (g) ZDD size and # solutions (h) Objective values

Figure 2: (a)–(d) are the results on SOPs with a 5×9 grid, and (e)–(h) are those with a 7×13 grid. (a), (e) Semi-log plot of running times for our algorithm, GCB and the A* algorithm. The running times of our algorithm are shown for constructing a ZDD (ZDD-construction), executing GreedyDP (ZDD-GreedyDP) and their summation (ZDD-total). (b), (f) Running times of our algorithm and GCB. (c), (g) Semi-log plot of the ZDD size $|N|$ and the number of feasible solutions $|\mathcal{F}|$. (d), (h) Objective values achieved by our algorithm, GCB and the A* algorithm; those of the A* algorithm are always optimal.



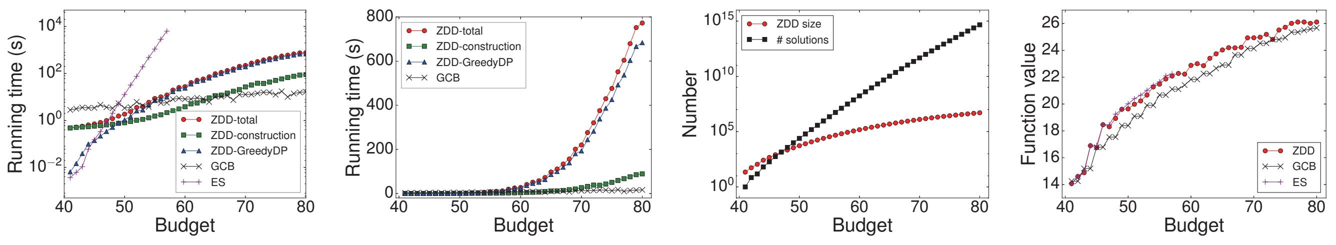
(a) Our algorithm (b) GCB

Figure 3: s-t paths with $B = 13$ obtained by our algorithm and GCB, respectively. The vertices $s, t \in V$ are indicated by the blue filled circle and square, respectively. The 11 filled black triangles indicate the locations of the 11 stations. The heat map expresses the informativeness of locations before performing sensing with the robot; locations with a warmer (darker) color are more informative.

We also tried to apply an algorithm based on the recursive greedy algorithm (Singh et al. 2009) to the problem. However, since the problem is not a Euclidean SOP, most of the techniques employed to accelerate the recursive greedy are not applicable. As a result, the approximation algorithm based on the recursive greedy took far longer than the exact A* algorithm; actually it took more than one day to find a solution for an SOP with the 5×9 grid and $B = 9$. Thus we omit comparison with the recursive greedy. We note that, even if the techniques for acceleration can be used, the recursive greedy becomes at most three orders of magnitude faster (see, (Singh et al. 2009)). Thus it is still slower than our algorithm, which took only about 10^{-1} seconds for the SOP with the 5×9 grid and $B = 9$.

Figures 2 (a)–(h) summarize the numerical results; (a)–(d) are those of the 5×9 grid, and (e)–(h) are those of the 7×13 grid. We first consider the computation cost of the algorithms. In Figures 2 (a) and (e), running times of our algorithm, GCB and the A* algorithm are shown, where the results of the A* algorithm with $B > 27$ and $B > 23$ are omitted in (a) and (e), respectively, since the time limit was exceeded. We see that our algorithm and GCB are much more scalable than the A* algorithm; our algorithm is about 5×10^2 times faster than the A* algorithm for the SOP on the 5×9 grid with $B = 27$. The detailed running time comparison of our algorithm and GCB is shown in Figures 2 (b) and (f). Although GCB is more scalable than our algorithm, both have reasonable computation costs relative to those of the other algorithms. As examined in (Minato 1993), ZDD size does not always increase with the value of B (Figures 2 (c) and (f)), and neither does the running time of our algorithm. Figures 2 (c) and (f) show that, in many instances, ZDD size is smaller than the number of feasible solutions by orders of magnitude, which makes our algorithm far more scalable than exact algorithms.

We now turn to the quality of solutions obtained by the algorithms. The objective values achieved the three algorithms are shown in Figures 2 (d) and (h), where those computed by the A* algorithm for $B \leq 27$ and $B \leq 23$, respectively, are optimal. Our algorithm significantly outperformed GCB; actually, it always achieved at least a 99%-approximation in the 26 instances to which the A* algorithm was applied, and found optimal solutions in 18 out of the 26 instances. The following two reasons explain why GCB had trouble in achieving high objective values. (1) The first reason is the difficulty of finding a feasible s-t path that covers the



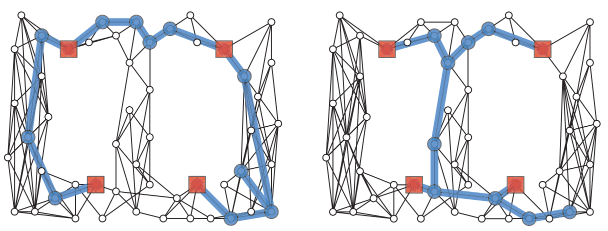
(a) Running times (semi-log) (b) Running times (c) ZDD sizes and # solutions (d) Objective values

Figure 4: (a) Semi-log plot of running times of our algorithm, GCB, and ES. For our algorithm, the running time for each procedure is shown as in Figure 2. (b) Running times of our algorithm and GCB. (c) Numbers of feasible solutions and ZDD sizes. (d) Function values obtained by the three algorithms; those of ES for $B \leq 57$ are optimal.

current solution of vertices efficiently. Every time GCB finds new solution $S \subseteq V$, we need to evaluate the cost of S by computing the minimum-cost s-t path that visits all vertices in S; this, however, is NP-hard in general. Thus GCB instead evaluates the cost of S inexactly using the nearest neighbor algorithm, which finds an s-t walk that visits all vertices in S. As a result, GCB sometimes returns an s-t walk that visits the same vertices time and again, which leads to the lower objective values. (2) The second reason is the characteristics of the cost-benefit greedy search strategy. To see this, we use Figures 3 (a) and (b) as explanatory examples. The figures show the solutions obtained by the two algorithms for the SOP on the 5×9 grid with $B = 13$. Here, our algorithm successfully found an optimal s-t path. GCB also finds an s-t path, but it fails to pass through the informative area (around the 3rd column from the left in the grid graph). Since GCB employs the cost-benefit greedy search strategy, it tends to choose vertices that are not far from s, t in the first a few iterations. Consequently, GCB fails to reach informative areas that are far from s, t. On the other hand, our algorithm seldom suffers these two problems. This is because (1) all feasible s-t paths are stored in a ZDD, and thus, (2) even if the informative areas are far from s, t, our algorithm considers s-t paths going through the areas as candidates.

3.2 Indoor WSN Design with Critical Areas

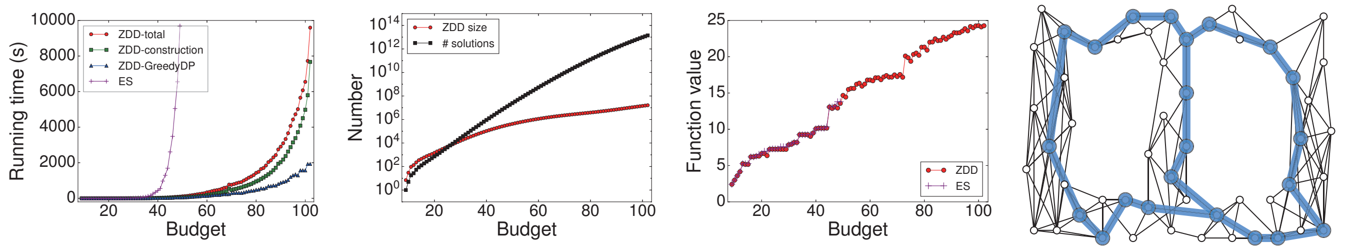
We consider an indoor WSN design problem formulated as an SSTP. To collect as much information as possible, we aim to design a network that covers as wide an area as possible under the following four constraints. (1) Sensors must be placed at some specified locations (e.g, locations close to outlets); we denote the set of locations by V. (2) The sensors must form a connected network; sensors that are too far from each other, or even nearby sensors that are obstructed by, for example, walls or radiation from appliances, cannot communicate with each other. (3) The communication cost of the obtained network must be at most B; sensors consume electric power when communicating with each other, and the total power consumption is bounded by B. (4) There are some critical locations, which are called terminals, and a sensor must be placed on every terminal. To model this situation, we construct graph $G = (V, E)$ as in Figures 5 (a) and (b). A sensor can be placed on a vertex, and two sensors can communicate if and only if they are placed on two vertices connected by an edge. The terminals are indicated by filled squares. We seek to find an optimal Steiner tree in G that includes all terminals.



(a) Our algorithm (b) GCB

Figure 5: Illustration of the problem and obtained solutions. The vertices represent the locations where sensors can be placed, and two sensors can communicate if and only if they are placed on vertices connected by an edge. The filled squares indicate terminals. Blue thick edges are the networks obtained with (a) our algorithm and (b) GCB for $B = 50$.

This experiment uses temperature field data from the Intel Berkeley Research Laboratory (Madden 2004). To construct graph G, we use the data on connectivity; two vertices are connected if and only if their average probability of successful communication is at least 40%. The resulting graph has 54 vertices and 144 edges. The edge costs are integers ranging from 1 to 5, which are computed from the success probability; the total cost of each obtained network must be at most B. As shown in (Krause, Singh, and Guestrin 2008), in the context of indoor sensor placements, mutual information $\mathrm{MI}(S) := H(V \backslash S) - H(V \backslash S \mid S)$ is suitable to measure the informativeness of given $S \subseteq V$, where $H(\cdot)$ is the entropy function used in the SOP experiments, and thus we employ it as an objective function. Unfortunately, the mutual information is non-monotone in general; this seems to be somewhat undesirable as a measure of informativeness, and it does not match the setting of problem (1). Fortunately, however, it is known to be approximately monotone if the number of deployed sensors is small enough compared to $|V|$. Thus we here make $B = 41, \ldots, 80$, so the number of deployable sensors is small enough; $B = 41$ is the cost of the minimum Steiner tree covering all terminals.



(a) Running times (b) ZDD sizes and # solutions (c) Objective values (d) The graph and a solution

Figure 6: (a) Running times of our algorithm and ES. (b) Numbers of feasible solutions and ZDD sizes. (c) Function values attained by our algorithm and ES; those of ES for $B \leq 49$ are optimal. (d) Illustration of the graph and an obtained solution; blue thick edges are the network obtained with our algorithm for $B = 101$.

We implemented and applied our algorithm to the problem; its performance is compared with that of exhaustive search (ES) and a GCB variant that employs the following approximate cost computation. Given current solution $S \subseteq V$, the cost of S is evaluated by constructing a Steiner tree that covers S using the nearest neighbor method. More precisely, starting from $v_{\mathrm{init}} \in S$, if $T \subseteq S$ is a current connected tree, we connect the pair $u \in T$ and $v \in S \backslash T$ that has the minimum shortest-path length; this procedure is continued until T covers all vertices in S. We vary v_{init} among all vertices in S and take a Steiner tree with the smallest cost.

Figure 4 (a) shows that our algorithm and GCB are far more scalable than ES; the results of ES for $B \geq 58$ are omitted since the time limit was exceeded. Compared with GCB, our algorithm performs poorly in terms of running time (Figure 4 (b)). This, however, is not such a hardship in WSN design since we rarely face situations where we must design a network quickly. Figure 4 (c) shows the numbers of feasible solutions $|\mathcal{F}|$ and ZDD sizes $|N|$. We see that ZDD size grows much more slowly than $|\mathcal{F}|$, and this is why our algorithm is more scalable than ES. Figure 4 (d) plots the objective values achieved by the two algorithms, where those obtained by ES for $B \leq 57$ are optimal. We see that our algorithm outperforms GCB in most instances and that our algorithm is almost optimal for $B \leq 57$; actually our algorithm achieved more than a 97%-approximation in all 17 instances to which ES was applied, and found optimal solutions in 5 out of the 17 instances. Figures 5 (a) and (b) illustrate the networks obtained with our algorithm and GCB for $B = 50$. The network output by our algorithm covers a wider area than that output by GCB, showing the effectiveness of our algorithm for this problem. Similar to the results of the SOP experiments, GCB again suffers from the inherent characteristics of the cost-benefit greedy strategy. Namely, in the first a few iterations, GCB chooses vertices around the center of the target space, which have large cost-benefit ratios, but choosing such vertices greedily results in a WSN that cannot cover the left and right sides of the target space because of the budget constraint.

3.3 Robust Indoor WSN Design

We consider problems raised in finding robust WSNs, which we formulate as SVCNPs. We use the same graph and objective function as those in the SSTP experiments. In the SVCNP instances, we observed that the objective value increases with B even for large B, and thus we set $B = 9, \ldots, 102$; $B = 9$ is the cost of the minimum VC network (at $B = 103$ our algorithm exceeded the time limit).

When it comes to applying our algorithm to SVCNPs, a ZDD storing all VC networks cannot be obtained by the commonly used top-down construction method (see, e.g., (Kawahara et al. 2017a)), and thus we constructed it using algebraic operations defined on a family of sets (see, (Knuth 2011)), which can be performed on ZDDs. To the best of our knowledge, no existing algorithm is applicable to SVCNP since its constraint is too complex. Thus we employ only ES to benchmark our algorithm.

As in Figure 6 (a), our algorithm is much more scalable than ES by virtue of the fact that the ZDD size grows more slowly than the number of feasible solutions (Figure 6 (b)); the running times of ES for $B \geq 50$ are omitted since the time limit was exceeded. Different from the results in SOPs and SSTPs, the ZDD-construction requires larger computation cost than ZDD-GreedyDP. This is because the ZDDs for SVCNPs are constructed using algebraic operations on ZDDs repetitively, which takes a somewhat long time. Figure 6 (c) plots the objective values achieved by the two algorithms, where those obtained with ES for $B \leq 49$ are optimal. We observed that our algorithm attained at least a 91%-approximation in all 41 instances to which ES was applied, and found optimal solutions in 24 out of the 41 instances. Figure 6 (d) illustrates a solution obtained with our algorithm for $B = 101$, showing that our algorithm successfully found a VC network that covers a wide area.

4 Conclusions and Discussions

We proposed a novel approximation algorithm for SFM over graphs, which includes many important and practical problems such as SOPs, SSTPs, and SVCNPs. Our algorithm avoids the difficulty of determining feasibility by reducing the original problems to those of finding a route in a ZDD. We proved that our algorithm achieves $(1 - c)$-approximations where c is the curvature of the objective submodular function. We observed the performance of our algorithm via numerical experiments involving three kinds of sensing tasks. The results showed that our algorithm finds better solutions than the alternative approximation algorithm in most instances, and runs much faster than the exact algorithms. Notably, our algorithm achieved more than a 90%-approximation in all instances for which optimal values were available.

In some experiments, we observed that the $(1-c)$-approximation guarantee tends to be pessimistic. Actually, our algorithm found optimal solutions for SOP instances with $c > 0.9$, implying that it might be possible to improve the $(1-c)$-approximation guarantee. We believe that submodular optimization with DDs is a promising subject of study with much room for development, and so offers a significant advance in the field of constrained submodular optimization.

Acknowledgments The authors are grateful to Hirofumi Suzuki and Hiroaki Iwashita for providing the C++ library to construct the ZDDs used in the implementation of our algorithm.

Appendix

Proof for Theorem 1

Note that the submodularity of f can be also characterized by the *diminishing return property*: $f(v \mid S) \geq f(v \mid T)$ for any $S \subseteq T$ and $v \in V \backslash T$. First, we derive an inequality for later use. From the definition of curvature c, we have the following inequality for any $T \subseteq V$ and $v \in V \backslash T$ if $f(v) > 0$ holds:

$$1 - c = \min_{S \subsetneq V,\ v' \notin S} \frac{f(v' \mid S)}{f(v')} \leq \frac{f(v \mid T)}{f(v)},$$

which means $(1-c)f(v) \leq f(v \mid T)$. If $f(v) = 0$, then we have $f(v \mid T) = 0$ by the submodularity. Thus $(1-c)f(v) \leq f(v \mid T)$ holds in both cases. Since $f(v \mid S) \leq f(v)$ holds for any $S \subseteq V$ due to the submodularity, the following inequality holds for any $v \in V$ and $S, T \subseteq V$ satisfying $v \notin T$:

(5) $$f(v \mid T) \geq (1-c) f(v \mid S).$$

Now we turn to the proof of Theorem 1. Let $R^*_{r,n} := \arg\max_{R \in \mathcal{R}_{r,n}} f(R)$ for all $n \in N$. The proof is obtained by induction on the size of U, i.e., assuming

(6) $$f(R_{r,n'}) \geq (1-c) f(R^*_{r,n'}) \quad (\forall n' \in N \backslash U),$$

we prove $f(R_{r,n}) \geq (1-c) f(R^*_{r,n})$ for node $n \in U$. Note that (6) holds if $U = N \backslash \{r\}$.

Let $n \in U$ be a node chosen in Step 5 of Algorithm 1 and $n' \in \mathrm{Preds}(n)$ satisfy $(n', n) \in R^*_{r,n}$, i.e., n' is the tail of the last arc in $R^*_{r,n}$. Furthermore, let $(n'', n) \in A_n$ be an arc chosen in Step 6. Note that (n'', n) is chosen greedily and thus satisfies the following inequality:

(7) $$\begin{aligned} f(R_{r,n}) &= f(R_{r,n''}) + f((n'', n) \mid R_{r,n''}) \\ &\geq f(R_{r,n'}) + f((n', n) \mid R_{r,n'}). \end{aligned}$$

Since (6) holds for $n' \in \mathrm{Preds}(n) \subseteq N \backslash U$ and $R^*_{r,n'} \in \mathcal{R}_{r,n'}$ is an optimal route from r to n', we get

(8) $$f(R_{r,n'}) \geq (1-c) f(R^*_{r,n'}) \geq (1-c) f(R^*_{r,n} \backslash (n', n)).$$

Now we consider two cases as follows. If (n', n) is 0-arc or $l(n') \in E$, then adding (n', n) to a route $R \in \mathcal{R}_{r,n'}$ does not increase the value of f. Thus we have

$$f((n', n) \mid R_{r,n'}) = f((n', n) \mid R^*_{r,n} \backslash (n', n)) = 0.$$

If (n', n) is 1-arc and $l(n') \in V$, then $l(n') \in V$ is added to the input of f by choosing (n', n). Since $l(n')$ appears at most once in $R_{r,n'} \cup (n', n)$ due to the property of ZDDs, we have $l(n') \notin V_{R_{r,n'}}$. Hence we obtain the following inequality by (5):

$$f((n', n) \mid R_{r,n'}) \geq (1-c) f((n', n) \mid R^*_{r,n} \backslash (n', n)).$$

Therefore, in both cases, we have

(9) $$f((n', n) \mid R_{r,n'}) \geq (1-c) f((n', n) \mid R^*_{r,n} \backslash (n', n)).$$

Substituting (8) and (9) into (7), we obtain

$$\begin{aligned} &f(R_{r,n}) \\ &\geq (1-c) \left(f(R^*_{r,n} \backslash (n', n)) + f((n', n) \mid R^*_{r,n} \backslash (n', n)) \right) \\ &= (1-c) f(R^*_{r,n}), \end{aligned}$$

where $n \in U$. By induction on the size of U, we obtain (6) for $U = \emptyset$, and thus $f(R_{r,\mathbf{1}}) \geq (1-c) f(R^*_{r,\mathbf{1}})$ follows.

References

Bergman, D.; Cire, A.; van Hoeve, W.-J.; and Hooker, J. 2016. *Decision Diagrams for Optimization*. Springer, first edition.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, New York.

Buchbinder, N.; Feldman, M.; Seffi, J.; and Schwartz, R. 2015. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM J. Comput.* 44(5):1384–1402.

Chekuri, C., and Pal, M. 2005. A recursive greedy algorithm for walks in directed graphs. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, 245–253. IEEE.

Chen, W.; Chen, Y.; and Weinberger, K. 2015. Filtered search for submodular maximization with controllable approximation bounds. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, volume 38, 156–164. PMLR.

Conforti, M., and Cornuéjols, G. 1984. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Appl. Math.* 7(3):251–274.

Coudert, O. 1997. Solving graph optimization problems with ZBDDs. In *Proceedings of European Conference on Design and Test*, 224. IEEE.

Garg, N.; Santosh, V. S.; and Singla, A. 1993. Improved approximation algorithms for biconnected subgraphs via better lower bounding techniques. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, 103–111. SIAM.

Heeger, K., and Vygen, J. 2016. Two-connected spanning subgraphs with at most $\frac{10}{7}$ OPT edges. *arXiv preprint arXiv:1609.00147*.

Inoue, Y., and Minato, S. 2016. Acceleration of ZDD construction for subgraph enumeration via path-width optimization. Technical report, TCS-TR-A-16-80, Hokkaido University.

Kawahara, J.; Inoue, T.; Iwashita, H.; and Minato, S. 2017a. Frontier-based search for enumerating all constrained subgraphs with compressed representation. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E100.A(9):1773–1784.

Kawahara, J.; Saitoh, T.; Suzuki, H.; and Yoshinaka, R. 2017b. Solving the longest oneway-ticket problem and enumerating letter graphs by augmenting the two representative approaches with ZDDs. In *Proceedings of Computational Intelligence in Information Systems Conference*, 294–305. Springer International Publishing.

Ke, W. C.; Liu, B. H.; and Tsai, M. J. 2011. Efficient algorithm for constructing minimum size wireless sensor networks to fully cover critical square grids. *IEEE Trans. Wireless Commun.* 10(4):1154–1164.

Knuth, D. E. 2011. *The Art of Computer Programming: Combinatorial Algorithms, Part 1*, volume 4A. Addison-Wesley Professional, 1st edition.

Krause, A.; Guestrin, C.; Gupta, A.; and Kleinberg, J. 2006. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, 2–10. ACM.

Krause, A.; McMahan, H. B.; Guestrin, C.; and Gupta, A. 2008. Robust submodular observation selection. *J. Mach. Learn. Res.* 9(Dec):2761–2801.

Krause, A.; Singh, A.; and Guestrin, C. 2008. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.* 9(Feb):235–284.

Lin, H., and Bilmes, J. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 912–920.

Liu, X.; Xiao, L.; Kreling, A.; and Liu, Y. 2006. Optimizing overlay topology by reducing cut vertices. In *Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 17:1–17:6. ACM.

Madden, S. 2004. Intel Lab Data. http://db.csail.mit.edu/labdata/labdata.html. Last accessed 30 November 2016.

Maehara, T.; Kawase, Y.; Sumita, H.; Tono, K.; and Kawarabayashi, K. 2017. Optimal pricing for submodular valuations with bounded curvature. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Minato, S. 1993. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proceedings of the 30th International Design Automation Conference*, 272–277. IEEE.

Morrison, D. R.; Sewell, E. C.; and Jacobson, S. H. 2016. Solving the pricing problem in a branch-and-price algorithm for graph coloring using zero-suppressed binary decision diagrams. *INFORMS J. Comput.* 28(1):67–82.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functions-I. *Math. Program.* 14(1):265–294.

Sedgewick, R., and Wayne, K. 2011. *Algorithms, 4th Edition.* Addison-Wesley.

Sharma, D.; Kapoor, A.; and Deshpande, A. 2015. On greedy maximization of entropy. In *Proceedings of the 32nd International Conference on Machine Learning*, 1330–1338.

Singh, A.; Krause, A.; Guestrin, C.; and Kaiser, W. J. 2009. Efficient informative sensing using multiple robots. *J. Artif. Int. Res.* 34(1):707–755.

Suzuki, H., and Minato, S. 2016. Adding the vertex indices for enumerating and indexing the graphs via ZDD (in Japanese). *Special Interest Group on Fundamental Problems in Artificial Intelligence* 101:41–46.

Suzuki, H. 2016. Frontier based search with vertex indices. https://github.com/hs-nazuna/FrontierBasedSearchWithVertexIndices. Last accessed 1 September 2017.

Sviridenko, M. 2004. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.* 32(1):41–43.

Thoma, M.; Cheng, H.; Gretton, A.; Han, J.; Kriegel, H.-P.; Smola, A.; Song, L.; Yu, P. S.; Yan, X.; and Borgwardt, K. 2009. Near-optimal supervised feature selection among frequent subgraphs. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, 1076–1087.

Xiong, S., and Li, J. 2010. An efficient algorithm for cut vertex detection in wireless sensor networks. In *Proceedings of the 30th International Conference on Distributed Computing Systems*, 368–377. IEEE.

Zeng, Y.; Chen, X.; Cao, X.; Qin, S.; Cavazza, M.; and Xiang, Y. 2015. Optimal route search with the coverage of users' preferences. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2118–2124.

Zhang, H., and Vorobeychik, Y. 2016. Submodular optimization with routing constraints. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Zheng, Y.; Liu, F.; and Hsieh, H.-P. 2013. U-Air: when urban air quality inference meets big data. In *Proceedings of the 19th SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1436–1444. ACM.