Constrained Monotone k-Submodular Function Maximization Using Multi-objective Evolutionary Algorithms with Theoretical Guarantee

Chao Qian, Jing-Cheng Shi, Ke Tang, *Senior Member, IEEE,* Zhi-Hua Zhou, *Fellow, IEEE*

***Abstract*—The problem of maximizing monotone k-submodular functions under a size constraint arises in many applications, and it is NP-hard. In this paper, we propose a new approach which employs a multi-objective evolutionary algorithm to maximize the given objective and minimize the size simultaneously. For general cases, we prove that the proposed method can obtain the asymptotically tight approximation guarantee, which was also achieved by the greedy algorithm. Moreover, we further give instances where the proposed approach performs better than the greedy algorithm on applications of influence maximization, information coverage maximization and sensor placement. Experimental results on real-world data sets exhibit the superior performance of the proposed approach.**

***Index Terms*—Submodular optimization, constrained optimization, multi-objective evolutionary algorithms, theoretical analysis, experimental studies.**

I. Introduction

In many areas such as combinatorial optimization and machine learning, we often encounter the problem of selecting a subset with a size constraint from a total set of items. The involved objective functions are often monotone and submodular, e.g., in maximum coverage [7] and feature selection [15], [17]. This kind of problem, maximizing monotone submodular functions under a size constraint, is NP-hard, and the greedy algorithm achieves the $(1-\frac{1}{e})$-approximation guarantee [22], which is optimal in general [21].

In this paper, we study a generalization of the above problem, i.e., maximizing monotone k-submodular functions under a size constraint, which appears in many practical applications, e.g., influence maximization with k kinds of topics and sensor placement with k kinds of sensors [25]. That is, k pairwise disjoint subsets instead of a single subset need to be selected, and the objective functions are k-submodular. Note that although k-submodular is a generalization of submodular (i.e., $k = 1$) [12], the methods on submodular function maximization cannot be directly applied to the general k.

C. Qian is with the Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: chaoqian@ustc.edu.cn)

J.-C. Shi and Z.-H. Zhou are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mails: {shijc, zhouzh}@lamda.nju.edu.cn)

K. Tang is with the Shenzhen Key Laboratory of Computational Intelligence, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: tangk3@sustc.edu.cn)

Studies on k-submodular function optimization start with $k = 2$, i.e., bisubmodular, including both minimization [9], [18] and maximization [31], [35]. Since 2012, more attention has been drawn to the case of general k. The minimization of k-submodular functions is solvable in polynomial time [32]. However, maximizing k-submodular functions is NP-hard. Ward and Živnỳ [35] first gave a $\frac{1}{3}$-approximation guarantee by a deterministic greedy algorithm. This bound was later improved by Iwata et al. [13] to $\frac{1}{2}$ with a randomized greedy algorithm. In addition, Iwata et al. [13] also proved a $\frac{k}{2k-1}$-approximation guarantee for monotone k-submodular function maximization.

Most of the above-mentioned studies considered the unconstrained cases. Meanwhile, real-world applications of k-submodular function maximization are often subject to a size constraint [31]. To the best of our knowledge, there has existed only one work for maximizing monotone k-submodular functions under a size constraint. Ohsaka and Yoshida [25] proved that the greedy algorithm can obtain the $\frac{1}{2}$-approximation guarantee under the total size constraint, which is asymptotically tight since exponential running time is required for achieving an approximation ratio of $\frac{k+1}{2k}+\epsilon$ for any $\epsilon > 0$ [13]. Note that they also studied the problem under the individual size constraint, while we focus on the total size constraint in this paper.

Motivated by the success of multi-objective evolutionary algorithms (MOEAs) for solving constrained optimization problems over the continuous solution space [2], [4], [33] as well as the Boolean solution space (i.e., $\{0,1\}^n$) [26], [27], [28], [29], we propose a Multi-Objective evolutionary optimization method for maximizing Monotone k-Submodular functions with a size constraint (where the solution space is $\{0,1,\ldots,k\}^n$), called MOMS. The MOMS method first reformulates the original constrained problem as a bi-objective optimization problem that maximizes the given objective and minimizes the size simultaneously, then employs a simple MOEA [16] combined with randomized local search to solve it, and finally selects the best feasible solution from the produced non-dominated solution set.

We first theoretically analyze the performance of MOMS. The main results are that,

- MOMS can achieve the $\frac{1}{2}$-approximation guarantee in polynomial time for general cases (i.e., Theorem 1), which reaches the asymptotically tight bound also obtained by the greedy algorithm [25].

- Three representative monotone k-submodular problems, i.e., influence maximization, information coverage maximization and sensor placement, are investigated. For each problem, we give an instance and prove that MOMS can find an optimal solution while the greedy algorithm cannot (i.e., Theorems 2, 3 and 4).

These results show that compared with the best-so-far method for the studied problem, the greedy algorithm [25], the proposed method MOMS has the same general approximation guarantee, but can achieve better solutions in cases.

We have also conducted experiments on the applications of influence maximization, information coverage maximization and sensor placement. The results on 14 real-world data sets exhibit the superior performance of MOMS over the greedy algorithm. Note that for sensor placement with $k = 1$, the greedy algorithm has been proved to be able to achieve the nearly optimal solution [30]. Thus, the better performance of MOMS on sensor placement also discloses that MOMS can bring a performance improvement even when the greedy algorithm has been nearly optimal.

The rest of the paper first introduces the studied problem, and then presents the proposed MOMS method, its theoretical analysis and empirical study. Finally we conclude this paper.

II. Monotone k-Submodular Function Maximization under a Size Constraint

In this section, we first introduce the general k-submodular problem, and then give three special instances, influence maximization, information coverage maximization and sensor placement, which will be studied in this paper.

A. The General k-Submodular Problem

Given a finite nonempty set $V = \{e_1, \ldots, e_n\}$ and a positive integer k, we study the functions $f : (k+1)^V \to \mathbb{R}$ defined on k disjoint subsets of V, where $(k+1)^V = \{(X_1, \ldots, X_k) \mid X_i \subseteq V, \forall i \neq j : X_i \cap X_j = \emptyset\}$. Note that $(X_1, \ldots, X_k)$ can be naturally represented by a vector $\boldsymbol{x} \in \{0, 1, \ldots, k\}^n$, where the j-th value $x_j = i$ means that $e_j \in X_i$, and $x_j = 0$ means that e_j does not appear in any subset. That is, $X_i = \{e_j \mid x_j = i\}$. In this paper, we will not distinguish $\boldsymbol{x} \in \{0, 1, \ldots, k\}^n$ and its corresponding k disjoint subsets $(X_1, \ldots, X_k)$ for notational convenience.

For $\boldsymbol{x} = (X_1, \ldots, X_k)$ and $\boldsymbol{y} = (Y_1, \ldots, Y_k)$, we define $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$ if $X_i \subseteq Y_i$ for any i. The monotonicity and k-submodularity are then defined as follows. When $k = 1$, it is easy to see that k-submodular is just submodular.

Definition 1 (Monotone)**.** *A function $f : (k+1)^V \to \mathbb{R}$ is monotone if for any $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$, $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$.*

Definition 2 (k-Submodular)**.** *[25] A function $f : (k+1)^V \to \mathbb{R}$ is k-submodular if for any $\boldsymbol{x}$ and $\boldsymbol{y}$,*

$$f(\boldsymbol{x}) + f(\boldsymbol{y}) \geq f(\boldsymbol{x} \sqcap \boldsymbol{y}) + f(\boldsymbol{x} \sqcup \boldsymbol{y}),$$

where $\boldsymbol{x} \sqcap \boldsymbol{y} = (X_1 \cap Y_1, \ldots, X_k \cap Y_k)$, and

$$\boldsymbol{x} \sqcup \boldsymbol{y} = \Big((X_1 \cup Y_1) \setminus \bigcup_{i \neq 1} (X_i \cup Y_i), \ldots, (X_k \cup Y_k) \setminus \bigcup_{i \neq k} (X_i \cup Y_i)\Big).$$

Let $[k]$ denote the set $\{1, 2, \ldots, k\}$. We then give some concepts in Definitions 3-6, that will be used in the paper. The support of a solution $\boldsymbol{x} = (X_1, \ldots, X_k)$ is the union of its k disjoint subsets, i.e., $supp(\boldsymbol{x}) = \bigcup_{i \in [k]} X_i$. For the vector representation $\boldsymbol{x} \in \{0, 1, \ldots, k\}^n$, $X_i = \{e_j \mid x_j = i\}$, and thus $supp(\boldsymbol{x})$ can also be represented as $\{e_j \mid x_j > 0\}$. The orthant submodularity intuitively means that the marginal gain by adding a single element into a solution decreases as the k subsets contained by the solution extend. The pairwise monotonicity is weaker than the monotonicity, because if a function is monotone, it must be pairwise monotone. In [35], the k-submodularity is proved to be equivalent to both the orthant submodularity and the pairwise monotonicity.

Definition 3 (Support)**.** *For a solution $\boldsymbol{x} = (X_1, \ldots, X_k)$, the support $supp(\boldsymbol{x})$ is the union of its k disjoint subsets, i.e.,*

$$supp(\boldsymbol{x}) = \bigcup_{i \in [k]} X_i.$$

Definition 4 (Marginal Gain/Loss)**.** *For a function $f : (k+1)^V \to \mathbb{R}$ and a solution $\boldsymbol{x} = (X_1, \ldots, X_k)$, the marginal gain $\Delta^+_{e,i} f(\boldsymbol{x})$ with respect to $e \notin supp(\boldsymbol{x})$ and $i \in [k]$ is the increment on f by adding e into X_i, i.e.,*

$$\begin{aligned}\Delta^+_{e,i} f(\boldsymbol{x}) =& f(X_1, \ldots, X_{i-1}, X_i \cup \{e\}, X_{i+1}, \ldots, X_k) \\ &- f(X_1, \ldots, X_k);\end{aligned}$$

and the marginal loss $\Delta^-_{e,i}$ with respect to $e \in X_i$ is decrement on f by deleting e from X_i, i.e.,

$$\begin{aligned}\Delta^-_{e,i} f(\boldsymbol{x}) =& f(X_1, \ldots, X_k) \\ &- f(X_1, \ldots, X_{i-1}, X_i \setminus \{e\}, X_{i+1}, \ldots, X_k).\end{aligned}$$

Definition 5 (Orthant Submodular)**.** *A function $f : (k+1)^V \to \mathbb{R}$ is orthant submodular if for any $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$, $e \notin supp(\boldsymbol{y})$ and $i \in [k]$,*

$$\Delta^+_{e,i} f(\boldsymbol{x}) \geq \Delta^+_{e,i} f(\boldsymbol{y}). \tag{1}$$

Definition 6 (Pairwise Monotone)**.** *A function $f : (k+1)^V \to \mathbb{R}$ is pairwise monotone if for any $\boldsymbol{x}$, $e \notin supp(\boldsymbol{x})$ and $i, j \in [k]$ with $i \neq j$,*

$$\Delta^+_{e,i} f(\boldsymbol{x}) + \Delta^+_{e,j} f(\boldsymbol{x}) \geq 0.$$

Our studied problem as presented in Definition 7 is to maximize a monotone k-submodular function f with an upper limit on $|supp(\boldsymbol{x})|$, where $|\cdot|$ denotes the size of a set. Let $\mathbf{0}$ denote the all-zeros vector. Without loss of generality, we assume that f is normalized, i.e., $f(\mathbf{0}) = 0$. Ohsaka and Yoshida [25] have recently proved that the greedy algorithm can obtain the asymptotically tight $\frac{1}{2}$-approximation guarantee. As shown in Algorithm 1, the greedy algorithm iteratively selects a combination (e, i) with the largest improvement on f. Note that $\boldsymbol{x}(e)$ denotes the value of $\boldsymbol{x}$ on the element e, i.e., $e \in X_{\boldsymbol{x}(e)}$. Ohsaka and Yoshida [25] also studied the problem under the individual size constraint, i.e., $|X_i| \leq b_i$ for each $i \in [k]$. In this paper, we focus on the total size constraint, i.e., $|supp(\boldsymbol{x})| = |\bigcup_{i \in [k]} X_i| \leq b$.

Algorithm 1 Greedy Algorithm

Input: a monotone k-submodular function $f : \{0, \dots, k\}^n \to \mathbb{R}^+$ and a budget $b \in [n]$
Output: a solution $\boldsymbol{x}$ with $|supp(\boldsymbol{x})| = b$
Process:

```
Let t = 0 and x = 0.
repeat
    (e, i) = arg max_{e∈V\supp(x), i∈[k]} Δ^+_{e,i} f(x).
    Let x(e) = i, and t = t + 1.
until t = b
return x
```

Definition 7 (The General k-Submodular Problem)**.** *Given a monotone k-submodular function $f : \{0, 1, \dots, k\}^n \to \mathbb{R}^+$ and a budget $b \in [n]$, the task is to find*

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\{0,1,\dots,k\}^n}{\arg\max} f(\boldsymbol{x}) \quad s.t. \quad |supp(\boldsymbol{x})| \le b. \tag{2}$$

B. Influence Maximization

Influence maximization is to identify a set of influential users in social networks. Let a directed graph $G = (V, E)$ represent a social network, where each node is a user and each edge $(u, v) \in E$ has a probability $p_{u,v}$ representing the strength of influence from user u to v. Given a budget b, influence maximization is to find a subset S of V with $|S| = b$ such that the expected number of nodes activated by propagating from S is maximized [14]. A fundamental propagation model is Independence Cascade (IC), as shown in Definition 8. Starting from a seed set $A_0 = S$, it uses a set A_t to record the nodes activated at time t, and at time $t+1$, each inactive neighbor v of $u \in A_t$ becomes active with probability $p_{u,v}$; this process is repeated until no nodes get activated at some time.

Definition 8 (Independence Cascade (IC))**.** *[14] Given a directed graph $G = (V, E)$ with edge probabilities $p_{u,v}$ for any $(u, v) \in E$ and a seed set $S \subset V$, the IC model propagates as follows:*
1. let $A_0 = S$ and $t = 0$.
2. repeat until $A_t = \emptyset$
3. for each edge (u, v) with $u \in A_t$ and $v \in V \setminus \bigcup_{i\le t} A_i$
4. v is added into A_{t+1} with probability $p_{u,v}$.
5. let $t = t + 1$.

As the user-to-user influence usually depends on some topic, Barbieri et al. [1] introduced the topic-aware model, where each edge (u, v) has a probability vector $(p^1_{u,v}, \dots, p^k_{u,v})$ representing the influence strength from u to v on each topic. Let $\boldsymbol{x} \in \{0, 1, \dots, k\}^n$ represent an assignment of topics to n users, where $x_j = i$ means that user j is with topic i. Thus, X_i contains all the nodes with topic i. The k-topic IC model propagates from X_i using probabilities $p^i_{u,v}$ independently for each topic. The set of nodes activated by propagating from X_i is denoted as $A(X_i)$, which is a random variable. Then, influence maximization with k kinds of topics as shown in Definition 9 is to maximize the expected total number of nodes that get activated in at least one propagation process. It has been proved to be monotone and k-submodular [25]. Note that $\mathbb{E}[\cdot]$ denotes the expectation of a random variable.

Definition 9 (Influence Maximization)**.** *Given a directed graph $G = (V, E)$ with $|V| = n$, edge probabilities $p^i_{u,v}$ ($(u, v) \in E, i \in [k]$) and a budget b, the task is to find*

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\{0,1,\dots,k\}^n}{\arg\max} \mathbb{E}\big[|\bigcup\nolimits_{i\in[k]} A(X_i)|\big] \quad s.t. \quad |supp(\boldsymbol{x})| \le b.$$

C. Information Coverage Maximization

Considering that an inactive node may be informed of information by any of its active neighbor nodes, Wang et al. [34] proposed a new problem, i.e., information coverage maximization, which is to maximize the expected number of both active nodes and informed nodes. An inactive node is informed if there exists at least one active neighbor node. For each $v \in A(X_i)$, let $N(v)$ denote the set of inactive neighbor nodes of v. Then, the set of active nodes and informed nodes by propagating from X_i can be represented as

$$AI(X_i) = A(X_i) \cup (\cup_{v\in A(X_i)} N(v)).$$

Information coverage maximization with k kinds of topics as shown in Definition 10 is to maximize the expected total number of nodes that get activated or informed in at least one propagation process. Because $\mathbb{E}[|AI(X_i)|]$ is monotone and submodular [34], it is easy to show that $\mathbb{E}\big[|\bigcup_{i\in[k]} AI(X_i)|\big]$ is monotone and k-submodular, the proof of which is as same as that for influence maximization in [25].

Definition 10 (Information Coverage Maximization)**.** *Given a directed graph $G = (V, E)$ with $|V| = n$, edge probabilities $p^i_{u,v}$ ($(u, v) \in E, i \in [k]$) and a budget b, the task is to find*

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\{0,1,\dots,k\}^n}{\arg\max} \mathbb{E}\big[|\bigcup\nolimits_{i\in[k]} AI(X_i)|\big] \quad s.t. \quad |supp(\boldsymbol{x})| \le b.$$

D. Sensor Placement

Given a limited number of sensors, the sensor placement problem is to decide where to place them such that the uncertainty is mostly reduced. Assume that there are k kinds of sensors, and let $\boldsymbol{x} \in \{0, 1, \dots, k\}^n$ represent a placement of sensors on n locations, where $x_j = i$ means that location j is installed with the i-th kind of sensor. We use O^i_j to denote a random variable representing the observations collected from location j with sensor i. Note that the conditional entropy (i.e., remaining uncertainty) of a total set U of random variables having observed a subset S is $H(U \mid S) = H(U) - H(S)$, where $H(\cdot)$ denotes the entropy. Thus, minimizing the uncertainty of U is equivalent to maximizing the entropy of S. Let $U = \{O^i_j \mid j \in [n], i \in [k]\}$. Then, sensor placement is to maximize the entropy of $\{O^{x_j}_j \mid x_j > 0\}$ using at most b number of sensors, as shown in Definition 11. It has been proved to be monotone and k-submodular [25]. Note that for the special case $k = 1$ (i.e, submodular), Sharma et al. [30] have shown that the greedy algorithm can achieve the nearly optimal solution by proving that the submodular objective function of sensor placement is close to modular.

Definition 11 (Sensor Placement)**.** *Given n locations, k kinds of sensors and a budget b, the task is to find*

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\{0,1,\ldots,k\}^n} H(\{O_j^{x_j} \mid x_j > 0\}) \ \ \textit{s.t.}\ \ |supp(\boldsymbol{x})| \leq b.$$

III. The MOMS Method

In this section, we propose a Multi-Objective evolutionary optimization method for the problem of maximizing Monotone k-Submodular functions under a size constraint, briefly called MOMS. The MOMS method first reformulates the original problem Eq. (2) as a bi-objective minimization problem

$$\arg\min_{\boldsymbol{x}\in\{0,1,\ldots,k\}^n} \quad (f_1(\boldsymbol{x}), f_2(\boldsymbol{x})),$$

where

$$f_1(\boldsymbol{x}) = \begin{cases} +\infty, & |supp(\boldsymbol{x})| \geq 2b \\ -f(\boldsymbol{x}), & \text{otherwise} \end{cases}, \quad f_2(\boldsymbol{x}) = |supp(\boldsymbol{x})|.$$

That is, MOMS maximizes the original objective f and minimizes the size of $supp(\boldsymbol{x})$ simultaneously.

In the bi-objective setting, both the two objective values have to be considered for comparing two solutions $\boldsymbol{x}$ and $\boldsymbol{x}'$. $\boldsymbol{x}$ *weakly dominates* $\boldsymbol{x}'$ (i.e., $\boldsymbol{x}$ is *better* than $\boldsymbol{x}'$, denoted as $\boldsymbol{x} \preceq \boldsymbol{x}'$) if $f_1(\boldsymbol{x}) \leq f_1(\boldsymbol{x}') \wedge f_2(\boldsymbol{x}) \leq f_2(\boldsymbol{x}')$ (i.e., $\boldsymbol{x}$ has a smaller or equal value on both the objectives); $\boldsymbol{x}$ *dominates* $\boldsymbol{x}'$ (i.e., $\boldsymbol{x}$ is *strictly better*, denoted as $\boldsymbol{x} \prec \boldsymbol{x}'$) if $\boldsymbol{x} \preceq \boldsymbol{x}'$ and either $f_1(\boldsymbol{x}) < f_1(\boldsymbol{x}')$ or $f_2(\boldsymbol{x}) < f_2(\boldsymbol{x}')$ (i.e., $\boldsymbol{x}$ has a smaller value on one objective, and meanwhile has a smaller or equal value on the other objective). The domination relationship can be summarized as follows:

(1) $\boldsymbol{x} \preceq \boldsymbol{x}'$ if $f_1(\boldsymbol{x}) \leq f_1(\boldsymbol{x}') \wedge f_2(\boldsymbol{x}) \leq f_2(\boldsymbol{x}')$,

(2) $\boldsymbol{x} \prec \boldsymbol{x}'$ if $\boldsymbol{x} \preceq \boldsymbol{x}' \wedge (f_1(\boldsymbol{x}) < f_1(\boldsymbol{x}') \vee f_2(\boldsymbol{x}) < f_2(\boldsymbol{x}'))$.

But if neither $\boldsymbol{x}$ is better than $\boldsymbol{x}'$ nor $\boldsymbol{x}'$ is better than $\boldsymbol{x}$, they are *incomparable*.

After the transformation, a simple multi-objective evolutionary algorithm (MOEA) with mutation only [16] is employed to solve the reformulated bi-objective minimization problem. It has been shown to be among the best-so-far algorithms for optimizing some P problems [23], [24] as well as approximating some NP-hard problems [8], [36]. As described in Algorithm 2, it starts from the all-zeros solution (line 1) and then iteratively tries to improve the solutions in the population P (lines 3-16). In each iteration, a new solution $\boldsymbol{y}$ is generated by mutating an archived solution $\boldsymbol{x}$ selected from the current P (lines 4-5); if $\boldsymbol{y}$ is not dominated by any previously archived solution (line 6), it will be added into P, and meanwhile those previously archived solutions weakly dominated by $\boldsymbol{y}$ will be removed from P (line 7). Note that in order to utilize the local information well to improve the efficiency, a randomized local search (RLS) procedure is incorporated to improve the newly included solution $\boldsymbol{y}$ (line 8). RLS as shown in Algorithm 3 adopts the random sampling technique [20], and performs a sequence of greedy local moves. In each step, it first selects a random subset R of elements and then adds one element from R with the largest gain or deletes one element from R with the smallest loss. The addition or deletion depends on the relationship between $|supp(\boldsymbol{y})|$ and b. The generated solutions

Algorithm 2 MOMS

Input: a monotone k-submodular function $f : \{0,\ldots,k\}^n \to \mathbb{R}^+$ and a budget $b \in [n]$
Parameter: the number T of iterations
Output: a solution $\boldsymbol{x}$ with $|supp(\boldsymbol{x})| \leq b$
Process:

1: Let $\boldsymbol{x} = \boldsymbol{0}$ and $P = \{\boldsymbol{x}\}$.
2: Let $t = 0$.
3: **while** $t < T$ **do**
4: Select $\boldsymbol{x}$ from P uniformly at random.
5: $\boldsymbol{y} = Mutation(\boldsymbol{x})$.
6: **if** $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \prec \boldsymbol{y}$ **then**
7: $P = (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{y} \preceq \boldsymbol{z}\}) \cup \{\boldsymbol{y}\}$.
8: $Q = RLS(f, b, \boldsymbol{y})$.
9: **for** each $\boldsymbol{q} \in Q$
10: **if** $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \prec \boldsymbol{q}$ **then**
11: $P = (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{q} \preceq \boldsymbol{z}\}) \cup \{\boldsymbol{q}\}$.
12: **end if**
13: **end for**
14: **end if**
15: $t = t + 1$.
16: **end while**
17: **return** $\arg\min_{\boldsymbol{x}\in P, |supp(\boldsymbol{x})|\leq b} f_1(\boldsymbol{x})$

by RLS are then used to update P (lines 9-13). Note that the choice of the sample size $|R|$ in each iteration of RLS will be made clear in theoretical analysis.

Definition 12 (Mutation)**.** *Given a solution $\boldsymbol{x} \in \{0,\ldots,k\}^n$, the mutation operator generates a new solution $\boldsymbol{y}$ by independently flipping each position of $\boldsymbol{x}$ with probability $\frac{1}{n}$, where the flipping on one position changes the current value to a different value selected uniformly at random. That is, for all $j \in [n]$,*

$$y_j = \begin{cases} x_j, & \textit{with probability } 1 - 1/n, \\ i, & \textit{otherwise}, \end{cases}$$

where i is uniformly randomly chosen from $\{0,\ldots,k\} \setminus \{x_j\}$.

MOMS repeats for T iterations. The value of T is a parameter, which could affect the quality of the produced solution. Their relationship will be analyzed in the next section, and we will use the theoretically derived T value in the experiments.

After running T iterations, the best solution (i.e., having the smallest f_1 value) satisfying the size constraint in P is selected as the final solution (line 17). Note that the smallest f_1 value corresponds to the largest value on the original objective f.

In the bi-objective transformation, the goal of setting f_1 to $+\infty$ is to exclude overly infeasible solutions, that is, the size of the support is at least $2b$. These infeasible solutions having $f_1 = +\infty \wedge f_2 \geq 2b$ are dominated by any feasible solution (e.g., the empty solution having $f_1 = 0$ and $f_2 = 0$), and therefore never introduced into the population P.

To the best of our knowledge, MOEAs have not been specially designed for the constrained k-submodular maximization problem before. Furthermore, previous applications of MOEAs often lack theoretical justification. We thus first

Algorithm 3 Randomized Local Search (RLS)

Input: a monotone k-submodular function $f : \{0,\dots,k\}^n \to \mathbb{R}^+$, a budget $b \in [n]$ and a solution $\boldsymbol{x} \in \{0,\dots,k\}^n$
Output: a set Q of solutions
Process:

1: Let $Q = \emptyset$, and $j = |supp(\boldsymbol{x})|$.
2: **if** $j < b$ **then**
3: **repeat**
4: $R \leftarrow \frac{n-|supp(\boldsymbol{x})|}{b-|supp(\boldsymbol{x})|}\ln(2(b-j))$ elements uniformly sampled from $V \setminus supp(\boldsymbol{x})$ with replacement.
5: $(e,i) = \arg\max_{e\in R, i\in[k]} \Delta^+_{e,i} f(\boldsymbol{x})$.
6: Let $\boldsymbol{x}(e) = i$, and $Q = Q \cup \{\boldsymbol{x}\}$.
7: **until** $|supp(\boldsymbol{x})| = b$
8: **else if** $j > b$ **then**
9: **repeat**
10: $R \leftarrow \frac{b+1}{|supp(\boldsymbol{x})|-b}$ elements uniformly sampled from $supp(\boldsymbol{x})$ with replacement.
11: $e = \arg\min_{e\in R} \Delta^-_{e,\boldsymbol{x}(e)} f(\boldsymbol{x})$.
12: Let $\boldsymbol{x}(e) = 0$, and $Q = Q \cup \{\boldsymbol{x}\}$.
13: **until** $|supp(\boldsymbol{x})| = b$
14: **end if**
15: **return** Q

propose a simple multi-objective evolutionary method for the ease of theoretical analysis. The following analysis will show that MOMS has already performed well both theoretically and empirically. Complex MOEAs for this problem will be studied in our future work.

IV. Theoretical Analysis: General Cases

This section theoretically investigates the general performance of MOMS. We prove its general approximation bound in Theorem 1, where OPT denotes the optimal function value of Eq. (2). Note that the $\frac{1}{2}$ polynomial-time approximation guarantee achieved by MOMS is asymptotically tight, since exponential running time is required for achieving an approximation ratio of $\frac{k+1}{2k} + \epsilon$ for any $\epsilon > 0$ [13]. The proof is inspired from that of Theorem 3.1 in [25]. If the running time is allowed to be infinite, MOMS can eventually find an optimal solution, since the employed mutation operator in Definition 12 is a global search operator which leads to a positive probability of producing any solution in each iteration.

Theorem 1. *For maximizing a monotone k-submodular function under a size constraint, the expected number T of iterations until MOMS first finds a solution $\boldsymbol{x}$ with $|supp(\boldsymbol{x})| \le b$ and $f(\boldsymbol{x}) \ge OPT/2$ is at most $8eb$.*

Proof. Assume that the population P always contains at least one solution $\boldsymbol{x}$ such that $|supp(\boldsymbol{x})| < b$ and

$$f(\boldsymbol{x}) \ge OPT - f(\boldsymbol{z}) \tag{3}$$

for some solution $\boldsymbol{z}$ with $\boldsymbol{x} \sqsubseteq \boldsymbol{z}$ and $|supp(\boldsymbol{z})| = b$. By selecting $\boldsymbol{x}$ in line 4 of Algorithm 2 and flipping no position in line 5, $\boldsymbol{x}$ is regenerated. Because $\boldsymbol{x}$ is previously archived in P, there must exist no solution in P which dominates $\boldsymbol{x}$. Thus, the condition of line 6 is satisfied, and $\boldsymbol{x}$ will go into the randomized local search procedure in line 8.

We then claim that with probability at least $\frac{1}{2}$, RLS on $\boldsymbol{x}$ can generate a solution with the $\frac{1}{2}$-approximation guarantee. Since $|supp(\boldsymbol{x})| < b$, Algorithm 3 will perform lines 3-7, which iteratively adds one element into $\boldsymbol{x}$ until $|supp(\boldsymbol{x})| = b$. In the first iteration, the sampled R in line 4 is denoted as $R^{(1)}$, the selected element from R and its value in line 5 are denoted as $e^{(1)}$ and $i^{(1)}$, respectively, and the generated solution in line 6 is denoted as $\boldsymbol{x}^{(1)}$. We define $\boldsymbol{z}^{(1/2)}$ as a vector obtained from $\boldsymbol{z}$ by assigning 0 to one element e' in $S^{(1)} = supp(\boldsymbol{z}) \setminus supp(\boldsymbol{x})$. Note that $\boldsymbol{x} \sqsubseteq \boldsymbol{z}$, thus $supp(\boldsymbol{x}) \subseteq supp(\boldsymbol{z})$. Assume that $R^{(1)} \cap S^{(1)} \neq \emptyset$. If $e^{(1)} \in R^{(1)} \cap S^{(1)}$, $e' = e^{(1)}$; otherwise, e' is an arbitrary element in $R^{(1)} \cap S^{(1)}$. Because $(e^{(1)}, i^{(1)})$ is the combination which increases f the most (see line 5), we have

$$f(\boldsymbol{x}^{(1)}) - f(\boldsymbol{x}) \ge \Delta^+_{e',\boldsymbol{z}(e')} f(\boldsymbol{x}).$$

Since $\boldsymbol{x} \sqsubseteq \boldsymbol{z}^{(1/2)}$ and $e' \notin supp(\boldsymbol{z}^{(1/2)})$, we can apply the orthant submodularity (i.e., Eq. (1)) to derive that

$$\Delta^+_{e',\boldsymbol{z}(e')} f(\boldsymbol{x}) \ge \Delta^+_{e',\boldsymbol{z}(e')} f(\boldsymbol{z}^{(1/2)}) = f(\boldsymbol{z}) - f(\boldsymbol{z}^{(1/2)}).$$

Combining the above two inequalities, we get

$$f(\boldsymbol{x}^{(1)}) - f(\boldsymbol{x}) \ge f(\boldsymbol{z}) - f(\boldsymbol{z}^{(1/2)}).$$

We define $\boldsymbol{z}^{(1)}$ as a vector obtained from $\boldsymbol{z}^{(1/2)}$ by letting $\boldsymbol{z}^{(1/2)}(e^{(1)}) = i^{(1)}$. Note that $|supp(\boldsymbol{z}^{(1)})| = b$. Due to the monotonicity of f, $f(\boldsymbol{z}^{(1)}) \ge f(\boldsymbol{z}^{(1/2)})$. Thus, we get

$$f(\boldsymbol{x}^{(1)}) - f(\boldsymbol{x}) \ge f(\boldsymbol{z}) - f(\boldsymbol{z}^{(1)}).$$

Using the same analysis procedure, we easily get $f(\boldsymbol{x}^{(i)}) - f(\boldsymbol{x}^{(i-1)}) \ge f(\boldsymbol{z}^{(i-1)}) - f(\boldsymbol{z}^{(i)})$ for the i-th iteration. Note that lines 4-6 will repeat $L = b - |supp(\boldsymbol{x})|$ iterations, and $\boldsymbol{z}^{(L)} = \boldsymbol{x}^{(L)}$. Let $\boldsymbol{x}^{(0)} = \boldsymbol{x}$ and $\boldsymbol{z}^{(0)} = \boldsymbol{z}$. Then, we have

$$\begin{aligned} f(\boldsymbol{x}^{(L)}) - f(\boldsymbol{x}) &= \sum\nolimits_{i=1}^{L} \left(f(\boldsymbol{x}^{(i)}) - f(\boldsymbol{x}^{(i-1)})\right) \\ &\ge \sum\nolimits_{i=1}^{L} \left(f(\boldsymbol{z}^{(i-1)}) - f(\boldsymbol{z}^{(i)})\right) = f(\boldsymbol{z}) - f(\boldsymbol{z}^{(L)}). \end{aligned}$$

By combining $\boldsymbol{z}^{(L)} = \boldsymbol{x}^{(L)}$ with the above inequality, we get

$$f(\boldsymbol{x}^{(L)}) \ge (f(\boldsymbol{x}) + f(\boldsymbol{z}))/2 \ge OPT/2,$$

where the second inequality is by Eq. (3).

The above analysis relies on the assumption that

$$\forall 1 \le i \le L : R^{(i)} \cap S^{(i)} \neq \emptyset.$$

In the following, we show that our choice of the sample size $|R^{(i)}|$ makes this assumption hold with probability at least $\frac{1}{2}$. Note that $|S^{(i)}| = |supp(\boldsymbol{z}^{(i-1)}) \setminus supp(\boldsymbol{x}^{(i-1)})| = b - |supp(\boldsymbol{x}^{(i-1)})|$, because $|supp(\boldsymbol{z}^{(i-1)})| = b$ and $supp(\boldsymbol{x}^{(i-1)}) \subseteq supp(\boldsymbol{z}^{(i-1)})$. From the procedure of generating $R^{(i)}$ in line 4 of Algorithm 3, we can derive that

$$\begin{aligned} &\Pr(R^{(i)} \cap S^{(i)} = \emptyset) \\ &= \left(1 - \frac{b - |supp(\boldsymbol{x}^{(i-1)})|}{n - |supp(\boldsymbol{x}^{(i-1)})|}\right)^{\frac{n-|supp(\boldsymbol{x}^{(i-1)})|}{b-|supp(\boldsymbol{x}^{(i-1)})|}\ln(2L)} \le \frac{1}{2L}, \end{aligned}$$

where the inequality is by $(1-1/m)^m \leq 1/e$. The assumption then holds with probability at least $1-L\cdot\frac{1}{2L}=\frac{1}{2}$ by the union bound. Thus, our claim holds.

Once such a solution $\boldsymbol{x}^{(L)}$ is generated by RLS, it will be used to update the population P (lines 9-13 of Algorithm 2); this makes P always contain a solution $\boldsymbol{y}\preceq\boldsymbol{x}^{(L)}$, i.e.,

$$f(\boldsymbol{y})\geq f(\boldsymbol{x}^{(L)})\geq OPT/2;$$
$$|supp(\boldsymbol{y})|\leq|supp(\boldsymbol{x}^{(L)})|=b.$$

Thus, in order to find a solution with the $\frac{1}{2}$-approximation guarantee, the required number of times to regenerate $\boldsymbol{x}$ in line 5 of Algorithm 2 is at most 2 in expectation. Let $P_{\max}$ denote the largest size of the population P during the run of MOMS. The probability of selecting $\boldsymbol{x}$ in line 4 is at least $\frac{1}{P_{\max}}$ due to uniform selection. From Definition 12, we know that the probability of flipping no position of $\boldsymbol{x}$ in line 5 is $(1-\frac{1}{n})^n\geq\frac{1}{2e}$. Thus, the probability of regenerating $\boldsymbol{x}$ in each iteration is at least $\frac{1}{2eP_{\max}}$, and then the expected number of iterations is at most $2eP_{\max}$. Note that the solutions in P are incomparable, thus each value of one objective can correspond to at most one solution in P. Because any solution with $|supp(\cdot)|\geq 2b$ is excluded from P (since $f_1=+\infty$), the second objective $f_2=|supp(\cdot)|$ can only belong to $\{0,1,\ldots,2b-1\}$. Thus, $P_{\max}\leq 2b$, which implies that the expected number of iterations until achieving the $\frac{1}{2}$-approximation guarantee is at most $2\cdot(2e\cdot 2b)=8eb$.

Finally, we only need to verify our assumption. The initial solution $\mathbf{0}$ satisfies Eq. (3) by letting $\boldsymbol{z}$ be an optimal solution. Furthermore, it will always be in P, because it has the smallest f_2 value 0 and no other solution can weakly dominate it. Thus, our assumption holds and then the theorem holds. □

V. Theoretical Analysis: Special Cases

We have proved that MOMS can generally achieve the $\frac{1}{2}$-approximation bound. Previous studies have shown that the greedy algorithm can also achieve this asymptotically tight bound [25]. We then further compare MOMS with the greedy algorithm on the applications of influence maximization, information coverage maximization and sensor placement. For each application, we give an instance where MOMS can perform better than the greedy algorithm.

A. Influence Maximization

For influence maximization, we analyze Example 1 where each user has an equal influence probability on each topic. We prove in Theorem 2 that MOMS can find a global optimal solution while the greedy algorithm cannot. The proof idea is that the greedy algorithm easily gets trapped in a local optimal solution, while MOMS can efficiently find an infeasible solution with size $b+1$, from which backward randomized local search (i.e., lines 9-13 of Algorithm 3) can produce a global optimal solution. Let s_i be a random variable such that $s_i=1$ if the node v_i is activated in the propagation process and $s_i=0$ otherwise. In the proof, the objective function in Definition 9 is equivalently computed by $\mathbb{E}[\sum_{i=1}^n s_i]=\sum_{i=1}^n\mathbb{E}[s_i]$, where the equality is by the linearity of expectation.

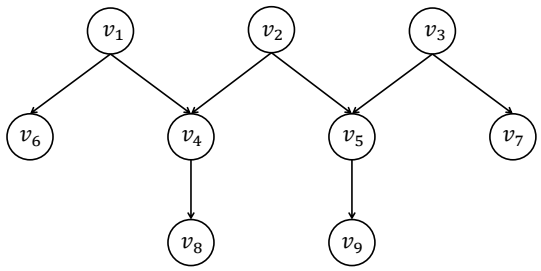


Figure 1. A social network graph, where each edge has a probability vector $(\frac{1}{k},\ldots,\frac{1}{k})$.

Example 1. *The parameters of influence maximization in Definition 9 are set as: the graph $G=(V,E)$ is shown in Figure 1 where each edge has a probability vector $(\frac{1}{k},\ldots,\frac{1}{k})$, and the budget $b=2$.*

Theorem 2. *For Example 1, the expected number T of iterations until MOMS first finds a global optimal solution is $O(bn)$, while the greedy algorithm cannot find a global optimal solution.*

Proof. We first analyze $f(\boldsymbol{x})$ with $|supp(\boldsymbol{x})|\leq b=2$. Note that x_j is the value on the node v_j. For $|supp(\boldsymbol{x})|=1$, it is easy to see that the solution $(0,i,0,\ldots,0)$ with $i\in[k]$ has the largest f value $1+\frac{2}{k}+\frac{2}{k^2}$, which is calculated by $\mathbb{E}[s_2]=1$, $\mathbb{E}[s_4]=\mathbb{E}[s_5]=\frac{1}{k}$, $\mathbb{E}[s_8]=\mathbb{E}[s_9]=\frac{1}{k^2}$ and $\mathbb{E}[s_1]=\mathbb{E}[s_3]=\mathbb{E}[s_6]=\mathbb{E}[s_7]=0$. For $\boldsymbol{x}$ with $x_2>0$ and $|supp(\boldsymbol{x})|=2$, we can derive that, for any $i,j\in[k]$,

$$f((j,i,0,\ldots,0))=f((0,i,j,0,\ldots,0))$$
$$=\begin{cases}2+\frac{4}{k}+\frac{2}{k^2}-\frac{1}{k^3} & \text{if}\quad i=j\\ 2+\frac{4}{k}+\frac{2}{k^2}-\frac{1}{k^4} & \text{if}\quad i\neq j\end{cases},$$
$$f((0,i,0,j,0,\ldots,0))=f((0,i,0,0,j,0,\ldots,0))$$
$$=\begin{cases}2+\frac{2}{k}+\frac{1}{k^2} & \text{if}\quad i=j\\ 2+\frac{2}{k}+\frac{2}{k^2}-\frac{1}{k^3} & \text{if}\quad i\neq j\end{cases},$$
otherwise, $f(\boldsymbol{x})\leq 2+2/k+2/k^2$.

It is also easy to verify that $\boldsymbol{x}_{global}=(i,0,j,0,\ldots,0)$ is a global optimal solution with the objective value $2+\frac{4}{k}+\frac{2}{k^2}$.

According to the above calculation results, the greedy algorithm will first find $(0,i,0,\ldots,0)$, and then go to the solution $\boldsymbol{x}_{local}=(j,i,0,\ldots,0)$ or $(0,i,j,0,\ldots,0)$, where $i\neq j$. Thus, it cannot find a global optimal solution.

For MOMS, we first show that it can efficiently find $\boldsymbol{x}_{local}$. The required number of iterations is denoted by T_1. Note that the initial solution $\mathbf{0}$ will always be in P. By selecting $\mathbf{0}$ in line 4 of Algorithm 2 and flipping no position, which happens with probability at least $\frac{1}{2b}\cdot(1-\frac{1}{n})^n\geq\frac{1}{4eb}$, $\mathbf{0}$ is regenerated in line 5. Because $\mathbf{0}$ cannot be dominated by any solution, it will go into the RLS procedure, which produces $\boldsymbol{x}_{local}$ with probability $\Omega(1)$, since the random sample R in line 4 of RLS can cover any specific element with probability $\Omega(1)$. We pessimistically assume that $\boldsymbol{x}_{global}$ is not found, since we are to derive the upper bound on the number of iterations for finding $\boldsymbol{x}_{global}$. Then, $\boldsymbol{x}_{local}$ will be included into P and

always exist in P, because it has the second largest f value among the solutions with $|supp(\boldsymbol{x})| = 2$. Thus, $\mathbb{E}[T_1] = O(b)$.

After that, by selecting $\boldsymbol{x}_{local}$ in line 4 and flipping the only 0 value in its first three positions to a value $l \neq i, j$ in line 5, which happens with probability at least $\frac{1}{2b} \cdot \frac{1}{n}(1-\frac{1}{n})^{n-1}\frac{k-2}{k}$, a solution $\boldsymbol{x}^* = (j, i, l, 0, \ldots, 0)$ or $(l, i, j, 0, \ldots, 0)$ is generated. Such a solution has the largest f value among the solutions with $|supp(\boldsymbol{x})| = 3$, and thus no solution can dominate it and $\boldsymbol{x}^*$ will go into the RLS procedure. Since $|supp(\boldsymbol{x}^*)| = b + 1$, lines 10-12 of Algorithm 3 will be performed once. If the random sample R in line 10 covers v_2, which happens with probability $1-(1-\frac{1}{b+1})^{b+1} \geq 1-\frac{1}{e}$, x_2^* will be set to 0, which leads to the smallest loss. Thus, $\boldsymbol{x}_{global}$ has been found. Denote the number of iterations in this phase as T_2. We then have $\mathbb{E}[T_2] \leq 2ebn\frac{k}{k-2} \cdot \frac{e}{e-1} = O(bn)$.

By combining the above two phases, we get that the expected number of iterations for MOMS finding $\boldsymbol{x}_{global}$ is at most $\mathbb{E}[T_1] + \mathbb{E}[T_2] = O(bn)$. □

B. Information Coverage Maximization

For information coverage maximization, we consider Example 2, which also uses the social network graph in Figure 1. The objective function in Definition 10 can be computed by $\sum_{i=1}^{n}\mathbb{E}[s_i]$, where $s_i = 1$ if the node v_i is activated or informed in the propagation process and $s_i = 0$ otherwise. We prove in Theorem 3 that MOMS performs better than the greedy algorithm.

Example 2. *The parameters of information coverage maximization in Definition 10 are set as: the graph $G = (V, E)$ is shown in Figure 1 where each edge has a probability vector $(\frac{1}{k}, \ldots, \frac{1}{k})$, and the budget $b = 2$.*

Theorem 3. *For Example 2, the expected number T of iterations until MOMS first finds a global optimal solution is $O(bn)$, while the greedy algorithm cannot find a global optimal solution.*

Proof. We first analyze $f(\boldsymbol{x})$. When $|supp(\boldsymbol{x})| = 1$, the solution $(0, i, 0, \ldots, 0)$ with $i \in [k]$ has the largest f value $3 + \frac{2}{k}$, which is calculated by $\mathbb{E}[s_2] = \mathbb{E}[s_4] = \mathbb{E}[s_5] = 1$, $\mathbb{E}[s_8] = \mathbb{E}[s_9] = \frac{1}{k}$ and $\mathbb{E}[s_1] = \mathbb{E}[s_3] = \mathbb{E}[s_6] = \mathbb{E}[s_7] = 0$. When $x_2 > 0$ and $|supp(\boldsymbol{x})| = 2$, we have, for any $i, j \in [k]$,

$$f((j, i, 0, \ldots, 0)) = f((0, i, j, 0, \ldots, 0)) = 5 + \frac{3}{k} - \frac{1}{k^2},$$
$$f((0, i, 0, j, 0, \ldots, 0)) = f((0, i, 0, 0, j, \ldots, 0)) = 4 + \frac{1}{k},$$

otherwise, $f(\boldsymbol{x}) \leq 4 + 2/k$.

In the case of $b = 2$, it is easy to verify that $(i, 0, j, 0, \ldots, 0)$ is a global optimal solution with the objective value $6+\frac{2}{k}$. When $|supp(\boldsymbol{x})| = 3$, the solution $(i, j, l, 0, \ldots, 0)$ has the largest f value $7 + \frac{4}{k} - \frac{2}{k^2}$.

Thus, the structure of the f function here is similar to that for influence maximization. We then can use the same proof as Theorem 2 to prove that MOMS can find a global optimal solution in $O(bn)$ expected number of iterations, while the greedy algorithm will get trapped in a local optimal solution. □

C. Sensor Placement

For sensor placement, we analyze Example 3 where only 4 locations with a specific kind of sensor can have different observations. Note that O_j^i denotes the observations from location j by installing the i-th kind of sensor, as introduced in Section II-D. Theorem 4 shows that MOMS is better than the greedy algorithm. The proof idea is similar to that of Theorem 2. The objective function (i.e., entropy) in Definition 11 is calculated using the observed frequency.

Example 3. *The parameters of sensor placement in Definition 11 are set as: the budget $b = 3$, and the observations collected from n locations by installing k kinds of sensors are*

$$O_1^1 = \{1, 1, 1, 2, 1, 2, 3, 3\},\ O_2^2 = \{1, 1, 1, 1, 2, 2, 2, 2\},$$
$$O_3^3 = \{1, 1, 2, 2, 1, 1, 2, 2\},\ O_4^4 = \{1, 2, 1, 2, 1, 2, 1, 2\},$$

and for any other $i \in [k], j \in [n]$, $O_j^i = \{1, 1, 1, 1, 1, 1, 1, 1\}$.

Theorem 4. *For Example 3, the expected number T of iterations until MOMS first finds a global optimal solution is $O(kbn)$, while the greedy algorithm cannot find a global optimal solution.*

Proof. By analyzing $f(\boldsymbol{x})$ with $|supp(\boldsymbol{x})| \leq 3$, we can easily follow the solution path of the greedy algorithm. When $|supp(\boldsymbol{x})| = 1$, $f((1, 0, \ldots, 0)) = 1.5$, $f((0, 2, 0, \ldots, 0)) = f((0, 0, 3, 0, \ldots, 0)) = f((0, 0, 0, 4, 0 \ldots, 0)) = 1$, and $f(\boldsymbol{x}) = 0$ otherwise. Thus, the greedy algorithm first finds $(1, 0, \ldots, 0)$. When $|supp(\boldsymbol{x})| = 2$ and $x_1 = 1$, we have $f((1, 2, 0, \ldots, 0)) = f((1, 0, 3, 0, \ldots, 0)) = f((1, 0, 0, 4, 0, \ldots, 0)) = 2.16$, and $f(\boldsymbol{x}) = 1.5$ otherwise. Thus, the next solution can be any of these three solutions. Since $f((1, 2, 3, 0, \ldots, 0)) = 2.5$ and $f((1, 2, 0, 4, 0 \ldots, 0)) = f((1, 0, 3, 4, 0, \ldots, 0)) = 2.75$, the greedy algorithm will output the solution $\boldsymbol{x}_{local} = (1, 2, 0, 4, 0, \ldots, 0)$ or $(1, 0, 3, 4, 0, \ldots, 0)$. It is also easy to verify that the global optimal solution $\boldsymbol{x}_{global} = (0, 2, 3, 4, 0, \ldots, 0)$ with $f(\boldsymbol{x}_{global}) = 3$. Thus, the greedy algorithm cannot find $\boldsymbol{x}_{global}$.

For MOMS, we can first use the same proof as Theorem 2 to derive that $\boldsymbol{x}_{local}$ will be found in the expected number of iterations $\mathbb{E}[T_1] = O(b)$. Then, by selecting $\boldsymbol{x}_{local}$ in line 4 and using mutation in line 5, the solution $\boldsymbol{x}^* = (1, 2, 3, 4, 0, \ldots, 0)$ will be generated with probability at least $\frac{1}{2b} \cdot \frac{1}{n}(1-\frac{1}{n})^{n-1}\frac{1}{k} \geq \frac{1}{2ekbn}$. Because $f(\boldsymbol{x}^*)$ reaches the largest f value 3 and we pessimistically assume that $\boldsymbol{x}_{global}$ has not been found, no solution in P can dominate $\boldsymbol{x}^*$ and it will go into the randomized local search procedure, which quickly finds $\boldsymbol{x}_{global}$ by letting $x_1^* = 0$ with probability at least $1 - \frac{1}{e}$. Thus, $\mathbb{E}[T_2] \leq 2ekbn \cdot \frac{e}{e-1}$. By combining these two phases, we get that the expected number of iterations for MOMS finding $\boldsymbol{x}_{global}$ is at most $\mathbb{E}[T_1] + \mathbb{E}[T_2] = O(kbn)$. □

VI. Empirical Study

In the above theoretical analysis, we have proved that MOMS can generally achieve the asymptotically tight approximation bound (which was also achieved by the greedy algorithm), and can perform better than the greedy algorithm on some artificial instances. Here, we also conducted experiments on influence maximization, information coverage

Table I
STATISTICS OF DATA SETS FOR INFLUENCE MAXIMIZATION AND INFORMATION COVERAGE MAXIMIZATION.

Data set	Type	#Nodes	#Edges
p2p-Gnutella04	Directed	10,879	39,994
p2p-Gnutella05	Directed	8,846	31,839
p2p-Gnutella06	Directed	8,717	31,525
p2p-Gnutella08	Directed	6,301	20,777
p2p-Gnutella09	Directed	8,114	26,013
p2p-Gnutella24	Directed	26,518	65,369
p2p-Gnutella25	Directed	22,687	54,705
p2p-Gnutella30	Directed	36,682	88,328
ca-HepPh	Undirected	12,008	118,521
ca-HepTh	Undirected	9,877	25,998
ego-Facebook	Undirected	4,039	88,234
weibo	Directed	10,000	162,371

maximization and sensor placement to investigate the actual performance of MOMS on real-world data sets.

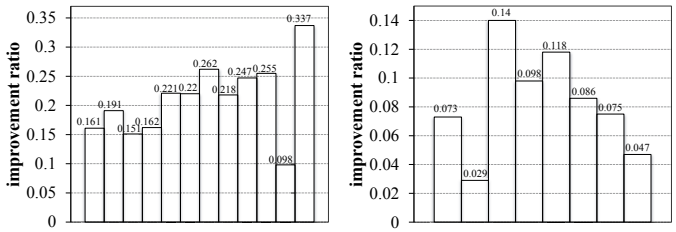
Since the greedy algorithm is the previous best algorithm both theoretically and empirically [25], we compare MOMS with it. Note that although there has existed various MOEAs, they have not been specially designed for these applications before. We thus do not compare MOMS with them. The number T of iterations of MOMS is set to $\lfloor 8eb \rfloor$ as suggested by Theorem 1. We test the budget b from 5 to 10. As MOMS is a randomized algorithm, we repeat the run 10 times independently and report the average f values.

A. Influence Maximization

We first test the special case of $k = 1$, i.e., each edge on the network has one propagation probability instead of one probability vector. We use 12 real-world data sets[1], the details of which are shown in Table I. The data set *p2p-Gnutella04* is collected from the Gnutella peer-to-peer file sharing network from August 4 2002, and other *p2p-Gnutella* data sets are collected similarly. The data sets *ca-HepPh* and *ca-HepTh* are collected from the e-print arXiv, which cover collaborations between authors who submitted papers to High Energy Physics - Phenomenology category and Theory category in the period from January 1993 to April 2003, respectively. The *ego-Facebook* data set is collected from survey participants using Facebook.app. The last data set *weibo* is crawled from Weibo.com, which is a Chinese microblogging site like Twitter. On each network, the propagation probability of one edge from node i to j is estimated by $\frac{weight(i,j)}{indegree(j)}$, as widely used in [3], [10].

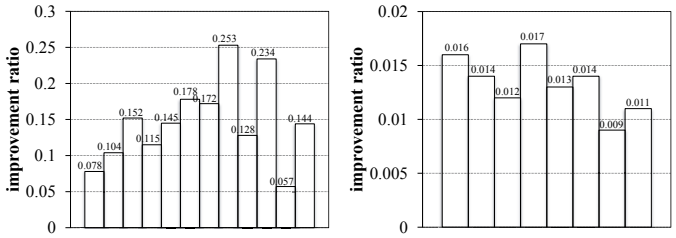
For the general case $k \geq 2$, i.e., each edge on the network has one propagation probability vector of length k, we use a real-world data set[2] collected from the social news website Digg over one month in 2009. It contains two tables, the friendship links between users and the user votes on news stories [11]. After preprocessing, we get a directed graph with 3523 nodes and 90244 edges. We set the number of topics $k = 2, 3, \ldots, 9$, respectively, and use the method in [1] to estimate the edge probabilities on each topic from the user votes.

[1]http://snap.stanford.edu/data/index.html
[2]http://www.isi.edu/~lerman/downloads/digg2009.html



(a) $k = 1$ (b) $k \geq 2$

Figure 2. Average improvement ratio from the greedy algorithm (i.e., $(f_{MOMS} - f_{Greedy})/f_{Greedy}$) at $b = 10$ for influence maximization: (a) on the 12 data sets with $k = 1$; (b) on the data set *Digg* with $k = 2, 3, \ldots, 9$.



(a) $k = 1$ (b) $k \geq 2$

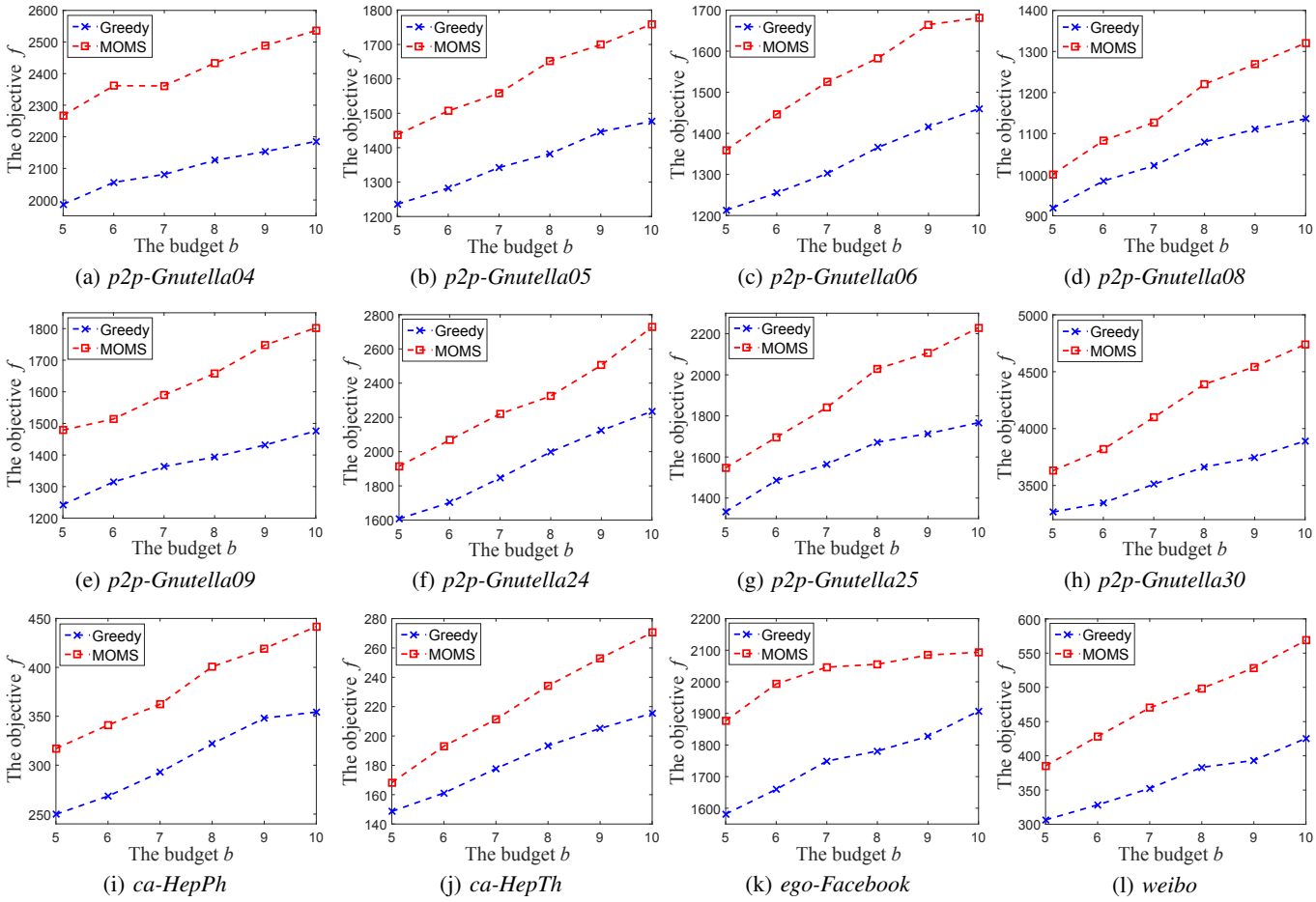
Figure 3. Average improvement ratio from the greedy algorithm (i.e., $(f_{MOMS} - f_{Greedy})/f_{Greedy}$) at $b = 10$ for information coverage maximization: (a) on the 12 data sets with $k = 1$; (b) on the data set *Digg* with $k = 2, 3, \ldots, 9$.

Note that for estimating the objective function of influence maximization in Definition 9, i.e., the expected number of nodes that get activated, we simulate the diffusion process 30 times independently and use the average as an estimation (that is, the objective function evaluation is noisy). But for the final output solutions of the algorithms, we average over 10,000 times for more accurate estimation. Since the behavior of the greedy algorithm is randomized under noise, we also repeat its run 10 times independently and report the average f values.

The results are plotted in Figures 4 and 5. We can observe that MOMS performs consistently better than the greedy algorithm on all data sets, which supports our theoretical analysis. Figure 2 also depicts the improvement ratio from the greedy algorithm (i.e., $(f_{MOMS} - f_{Greedy})/f_{Greedy}$) at $b = 10$, which can exceed 33% and 14% for $k = 1$ and $k \geq 2$, respectively, at best. We can also observe from Figures 4 and 5 that the gap between MOMS and the greedy algorithm has the trend of increasing with b at most cases. This may be because the more complex problem landscape led by larger b makes the greedy algorithm easier get trapped in local optima.

B. Information Coverage Maximization

To compare MOMS with the greedy algorithm on the task of information coverage maximization, we also use the 12 real-world data sets in Table I and the *Digg* data set. For estimating the objective function of information coverage maximization in Definition 10, i.e., the expected number of nodes that get activated or informed, we use the same process as that used in the experiment of influence maximization.

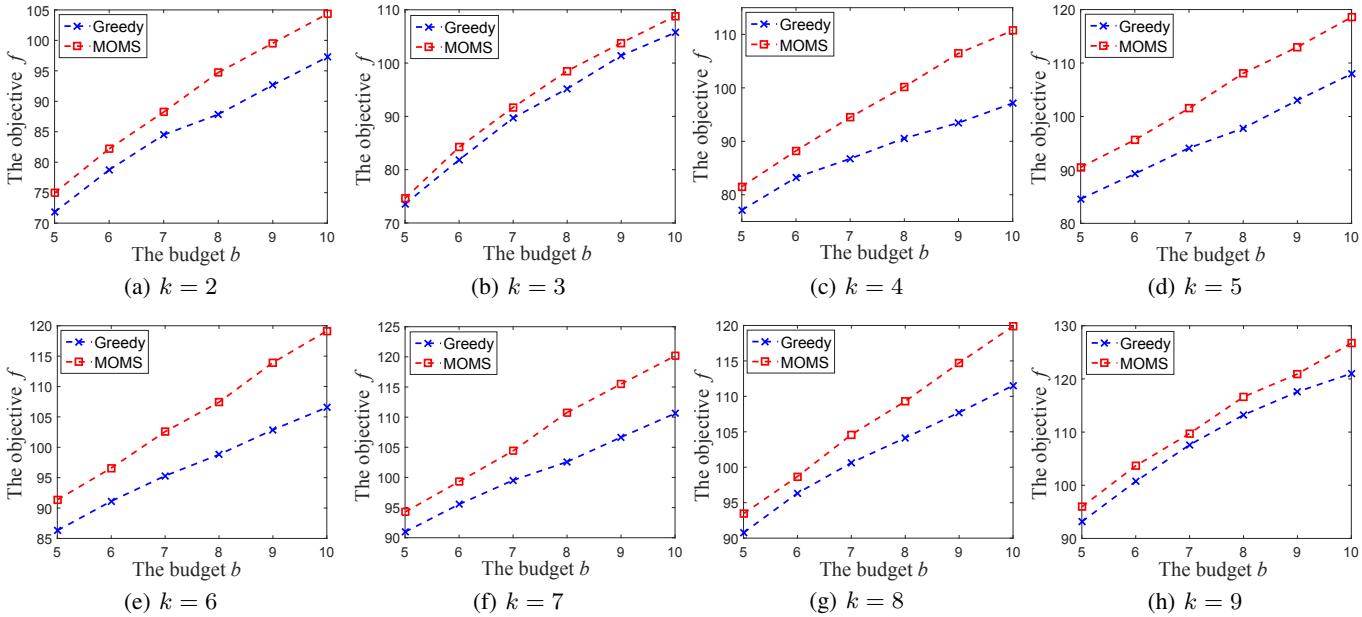


(a) *p2p-Gnutella04* (b) *p2p-Gnutella05* (c) *p2p-Gnutella06* (d) *p2p-Gnutella08*

(e) *p2p-Gnutella09* (f) *p2p-Gnutella24* (g) *p2p-Gnutella25* (h) *p2p-Gnutella30*

(i) *ca-HepPh* (j) *ca-HepTh* (k) *ego-Facebook* (l) *weibo*

Figure 4. Influence maximization with $k = 1$ on the 12 data sets in Table I. The objective f: the average number of active nodes (the larger the better).

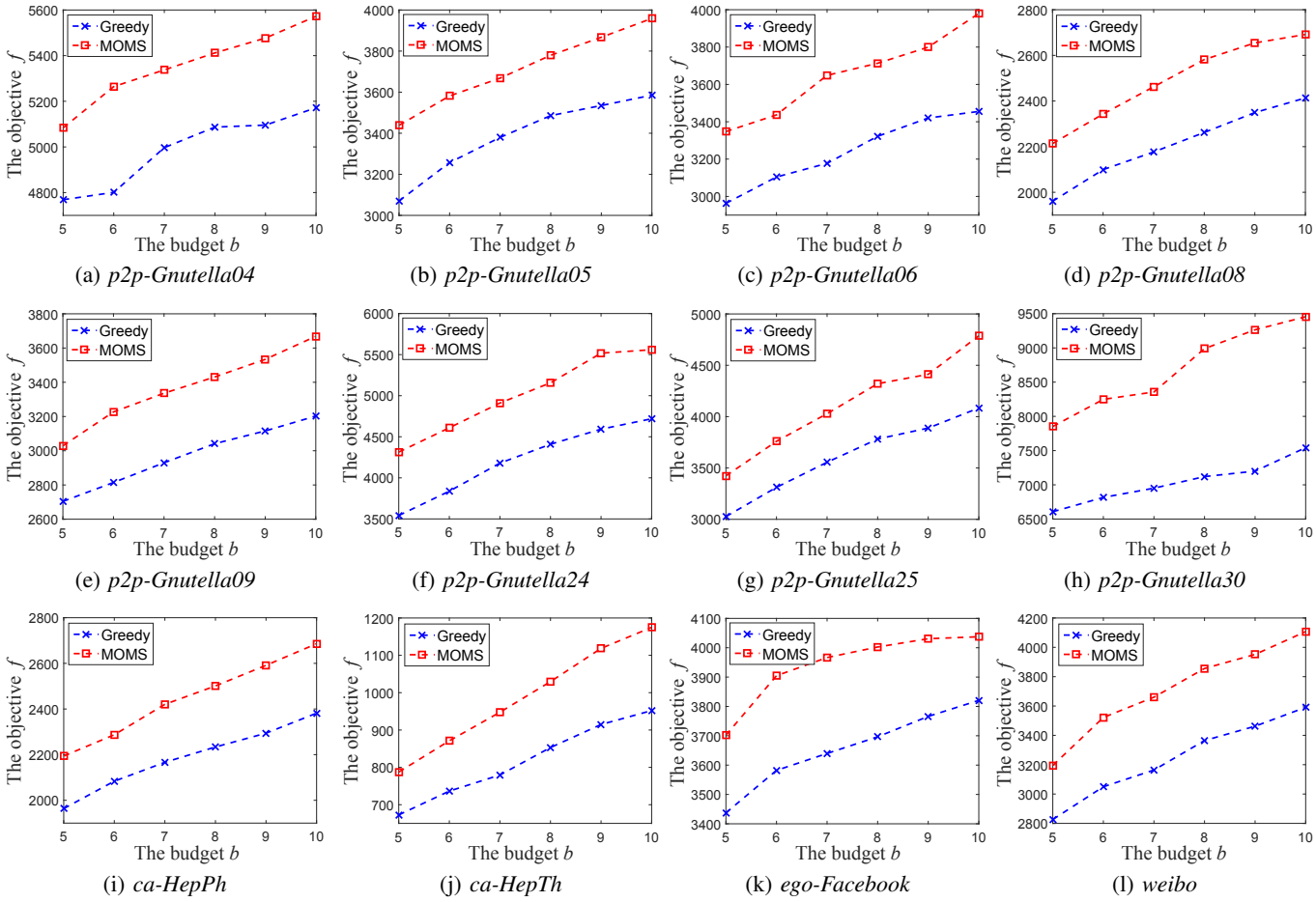


(a) $k = 2$ (b) $k = 3$ (c) $k = 4$ (d) $k = 5$

(e) $k = 6$ (f) $k = 7$ (g) $k = 8$ (h) $k = 9$

Figure 5. Influence maximization with $k \geq 2$ on the data set *Digg*. The objective f: the average number of active nodes (the larger the better).

The results plotted in Figures 6 and 7 show that MOMS always performs better than the greedy algorithm. Note that on the data set *ego-Facebook* (i.e., Figure 6(k)), MOMS almost finds an optimal solution at $b = 10$, since the objective function value has reached the largest possible value, i.e., the total number 4039 of nodes of this network. The improvement

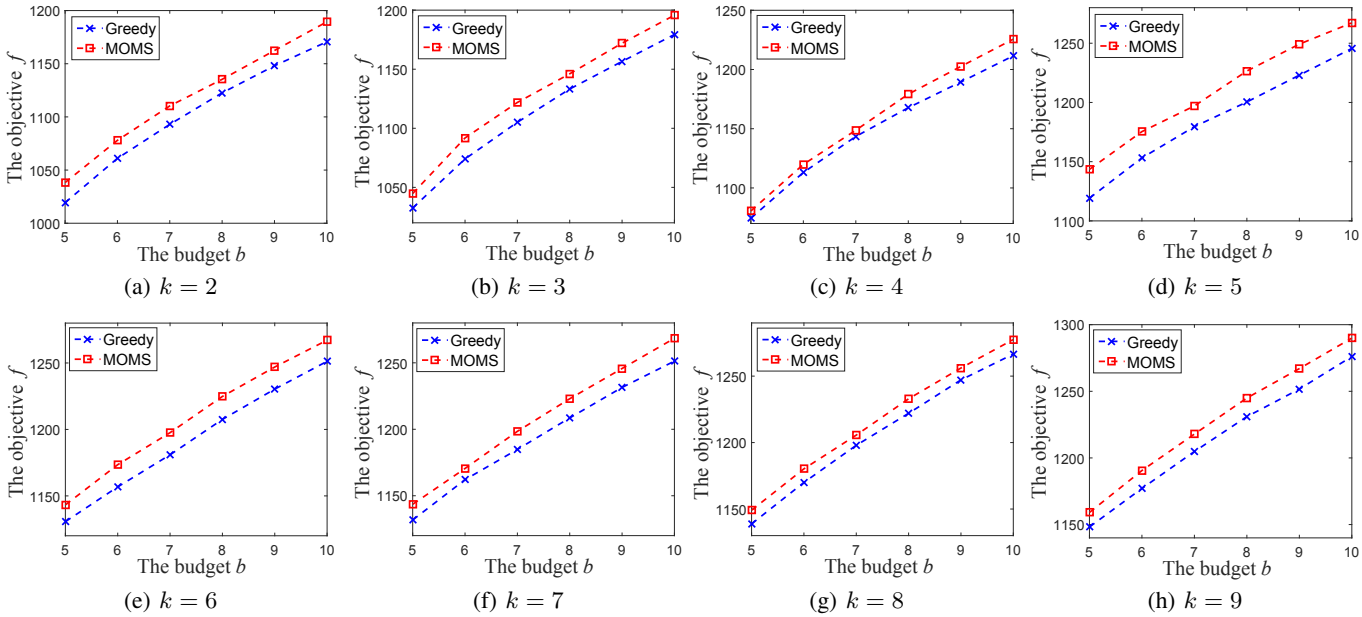


(a) *p2p-Gnutella04* (b) *p2p-Gnutella05* (c) *p2p-Gnutella06* (d) *p2p-Gnutella08*

(e) *p2p-Gnutella09* (f) *p2p-Gnutella24* (g) *p2p-Gnutella25* (h) *p2p-Gnutella30*

(i) *ca-HepPh* (j) *ca-HepTh* (k) *ego-Facebook* (l) *weibo*

Figure 6. Information coverage maximization with $k = 1$ on the 12 data sets in Table I. The objective f: the average number of active nodes and informed nodes (the larger the better).

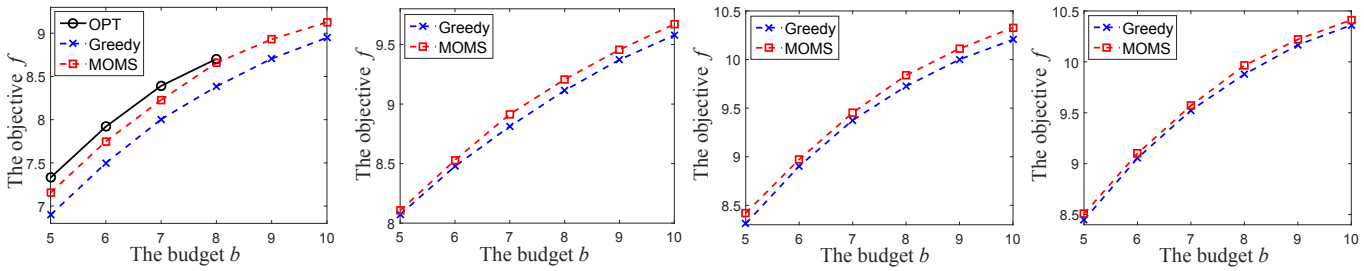


(a) $k = 2$ (b) $k = 3$ (c) $k = 4$ (d) $k = 5$

(e) $k = 6$ (f) $k = 7$ (g) $k = 8$ (h) $k = 9$

Figure 7. Information coverage maximization with $k \geq 2$ on the data set *Digg*. The objective f: the average number of active nodes and informed nodes (the larger the better).

ratio from the greedy algorithm at $b = 10$ is shown in Figure 3. We can see that the improvement ratio can exceed 25% for $k = 1$ at best. For $k \geq 2$, the improvement ratio is relatively low, i.e., between 1% and 2%.



(a) $k = 1$ (b) $k = 2$ (c) $k = 3$ (d) $k = 4$

Figure 8. Sensor placement on the data set collected from the Intel Berkeley Research lab. The objective f: the entropy (the larger the better).

C. Sensor Placement

For the comparison on the application of sensor placement, we use a real-world data set[3] collected from sensors installed at 55 locations of the Intel Berkeley Research lab between February 28 and April 5, 2004. The light, temperature, voltage and humidity measures are extracted and their values are discretized into 5, 12, 6 and 13 bins with equal size, respectively. Thus, we have $k = 4$ kinds of sensors and $n = 55$ locations. The problem is to select b locations and the corresponding sensors such that the entropy is maximized. We compare the performance of MOMS and the greedy algorithm at $k = 1, 2, 3, 4$, respectively. For $k = 1$, we only use the light sensor at each location; for $k = 2$, the light and temperature sensors can be selected at each location; for $k = 3$, we use the light, temperature and voltage sensors; for $k = 4$, we use all the four kinds of sensors. Note that the objective function (i.e., the entropy) of sensor placement in Definition 11 is calculated using the observed frequency.

The results are plotted in Figure 8. We can see that MOMS is better than the greedy algorithm for any k and b. We also calculate the improvement ratios of MOMS from the greedy algorithm at $b = 10$, which are 2%, 1%, 1.2% and 0.5% for $k = 1, 2, 3, 4$, respectively.

D. Running Time

Considering the running time (in the number of objective function evaluations), the greedy algorithm needs $O(kbn)$ time, while MOMS needs $8eb$ iterations $\times$ $O(kn \ln^2 b)$ time per iteration (i.e., the worst case of RLS). Note that we do not consider the lazy evaluation technique [19] for the fairness of comparison. Thus, MOMS is slower by a factor of $8e \ln^2 b$. Since this is a theoretical upper bound for MOMS being good, we record the actual time until MOMS achieves a better performance than the greedy algorithm. Table II shows the results at $b = 10$ for influence maximization and information coverage maximization. We can see that MOMS takes only at most 5% (i.e., $5.2/(8e \ln^2 10)$) of the theoretical time to achieve a better performance. For sensor placement, the running time of the greedy algorithm at $b = 10$ for $k = 1, 2, 3, 4$ is 505, 1010, 1515 and 2020, respectively; and the running time of MOMS until finding a solution better than that found by the greedy algorithm is 499, 3625, 3718 and 15401, respectively. Thus, the ratios between the running time of MOMS and the greedy algorithm are 1.0, 3.6, 2.5 and 7.6, respectively. Comparing with the theoretical ratio $8e \ln^2 b$, MOMS takes only at most 7% (i.e., $7.6/(8e \ln^2 10)$) of the theoretical time to be better than the greedy algorithm. These observations are expected, because the theoretical upper bound is the worst case running time, which is derived based on some pessimistic assumptions.

By selecting the greedy algorithm as the baseline, we also plot the curve of the f value over the running time for MOMS to investigate how the solution quality changes with the number of objective function evaluations. The value of b is set to 5. We compute the optimal solution by exhaustive enumeration, denoted as OPT. Note that OPT is calculated only for sensor placement (where $n = 55$) due to the computation time limit. The results on sensor placement are shown in Figure 9. Note that one unit on the x-axis corresponds to kbn number of objective function evaluations, i.e., the running time of the greedy algorithm. We can observe that MOMS takes more time to perform as well as the greedy algorithm in most cases, but can find better solutions by performing even more objective function evaluations. If the running time is further increased, the solution quality found by MOMS can be quite close to the optimum. For example, the ratio of the f value between MOMS using $50kbn$ number of objective function evaluations and OPT for $k = 1, 2, 3, 4$ is 97.8%, 100%, 99.0% and 99.7%, respectively. The results on influence maximization and information coverage maximization are similar, which are provided in the supplementary material due to space limitation.

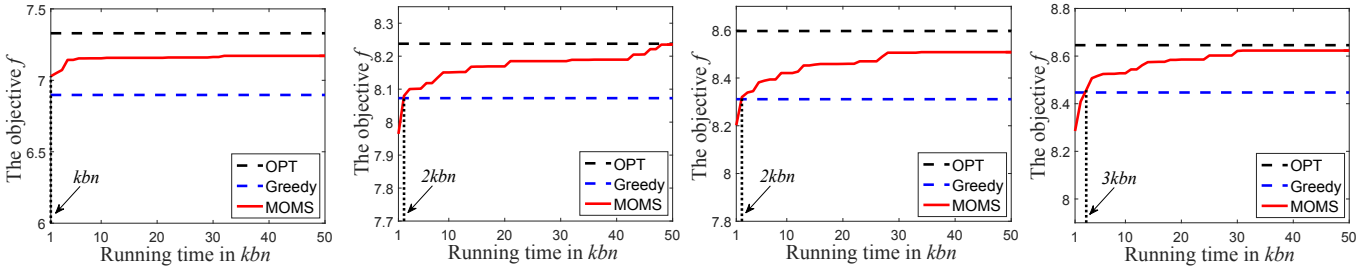
E. Discussion

The above experimental results have shown that MOMS is consistently better than the greedy algorithm. However, we also note that the improvement ratio can be quite different, which depends on concrete k-submodular applications and data sets. For example, the improvement ratio at $b = 10$ for influence maximization can exceed 33%, while that for sensor placement is at most 2%. The relatively small performance improvement of MOMS in some situations may be because the greedy algorithm has already performed very well. For example, for the sensor placement task with $k = 1$ where the improvement ratio of MOMS is empirically low, Sharma et al. [30] proved that the greedy algorithm can achieve the nearly optimal solution, which is also validated by experiments here. We can see from Figure 8(a) that the curve of the greedy algorithm is close to that of OPT. Note that OPT is calculated

[3]http://db.csail.mit.edu/labdata/labdata.html

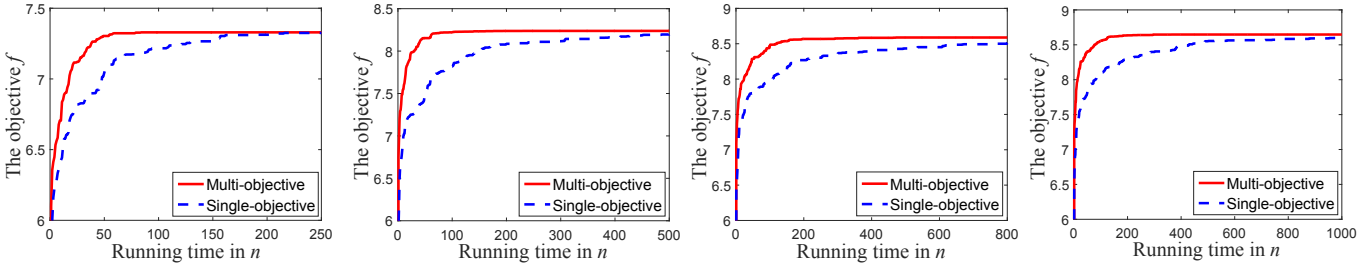
Table II
RUNNING TIME COMPARISON (IN THE NUMBER OF OBJECTIVE FUNCTION EVALUATIONS) FOR INFLUENCE MAXIMIZATION (IM) AND INFORMATION COVERAGE MAXIMIZATION (ICM) AT $b = 10$, WHERE THE COLUMNS OF MOMS-IM AND MOMS-ICM RECORD THE TIME OF MOMS UNTIL FINDING A BETTER SOLUTION THAN THE GREEDY ALGORITHM FOR IM AND ICM, RESPECTIVELY. NOTE THAT THE TIME OF THE GREEDY ALGORITHM IS FIXED FOR EACH DATA SET. THE NUMBER IN () DENOTES THE RATIO BETWEEN THE TIME OF MOMS AND THE GREEDY ALGORITHM.

Data set	Greedy	MOMS-IM	MOMS-ICM	Data set	Greedy	MOMS-IM	MOMS-ICM
p2p-Gnutella04	108745	126976 (1.2)	192145 (1.8)	*p2p-Gnutella25*	226825	225421 (1.0)	472909 (2.1)
p2p-Gnutella05	88415	137236 (1.6)	171623 (1.9)	*p2p-Gnutella30*	366755	754672 (2.1)	383159 (1.0)
p2p-Gnutella06	87125	205176 (2.4)	79976 (0.9)	*ca-HepPh*	120035	147153 (1.8)	180425 (1.5)
p2p-Gnutella08	62965	89184 (1.4)	63717 (1.0)	*ca-HepTh*	98725	151776 (1.5)	114486 (1.2)
p2p-Gnutella09	81095	53645 (0.7)	87705 (1.1)	*ego-Facebook*	40345	47462 (1.2)	55697 (1.4)
p2p-Gnutella24	265135	900287 (3.4)	402142 (1.5)	*weibo*	99955	176296 (1.8)	130161 (1.3)
Digg	Greedy	MOMS-IM	MOMS-ICM	*Digg*	Greedy	MOMS-IM	MOMS-ICM
$k = 2$	70370	148621 (2.1)	144585 (2.1)	$k = 6$	211110	228134 (1.1)	692729 (3.3)
$k = 3$	105555	544573 (5.2)	400764 (3.8)	$k = 7$	246295	532098 (2.2)	941425 (3.8)
$k = 4$	140740	161975 (1.2)	499061 (3.5)	$k = 8$	281480	541230 (2.0)	1397520 (5.0)
$k = 5$	175925	345494 (2.0)	325537 (1.9)	$k = 9$	316665	1083952 (3.4)	1290150 (4.1)



(a) $k = 1$ (b) $k = 2$ (c) $k = 3$ (d) $k = 4$

Figure 9. The objective f v.s. the running time for MOMS on sensor placement with $b = 5$. The objective f: the entropy (the larger the better).



(a) $k = 1$ (b) $k = 2$ (c) $k = 3$ (d) $k = 4$

Figure 10. The objective f v.s. the running time on sensor placement with $b = 5$, where "multi-objective" and "single-objective" denote the algorithm in the multi-objective and single-objective optimization setting, respectively. The objective f: the entropy (the larger the better).

only for $b = 5, 6, 7, 8$ due to the computation time limit. The approximation ratio is 94.1%, 94.7%, 95.4% and 96.3%, respectively, which implies that the greedy algorithm achieves the nearly optimal solution. Thus, these observations also disclose that MOMS can bring a performance improvement even when the greedy algorithm has been nearly optimal.

Compared with the greedy algorithm, MOMS mainly has two specific characteristics: the mutation operator and the domination-based comparison. The mutation operator as described in Definition 12 is a global search operator, which allows MOMS to produce new solutions by flipping any positions. Note that the greedy algorithm can flip only one position with the value 0 in each iteration. The domination relation can be used for comparing two solutions, due to the transformation of the original problem into a bi-objective one. By the domination-based comparison, MOMS naturally maintains several solutions in the population, while the greedy algorithm maintains only one solution. These two characteristics may make MOMS have a better ability of avoiding local optima than the greedy algorithm.

To validate the effectiveness of the bi-objective transformation, we have conducted experiments on sensor placement with the budget $b = 5$. We compare MOMS with its counterpart in the original single-objective setting. By single-objective optimization, the domination-based comparison is replaced by a common comparison rule for constrained problems [5]: two feasible solutions are compared based on their objective f values, and a feasible solution is always better than an infeasible one. For the fairness of comparison, we also delete the RLS procedure in implementing these two algorithms, since

RLS may bring a different effect in bi-objective and single-objective optimization. We plot the curves of the objective f value over the running time. The results are shown in Figure 10. Note that one unit on the x-axis corresponds to n number of objective function evaluations. We can observe that by the bi-objective transformation, the algorithm can find a much better solution using the same number of objective function evaluations. This shows the advantage of the bi-objective transformation. As the running time continues to increase, the two algorithms will gradually find the same good solution. This is expected, since both of them use the global search operator, which leads to a positive probability of producing any solution; and will eventually find a global optimal solution if the running time goes to infinity. Note that the benefit of the bi-objective transformation is limited to the studied problem, and the single-objective optimization setting can be better in other cases.

VII. Conclusion

In this paper, we propose a multi-objective evolutionary optimization approach for the problem of maximizing monotone k-submodular functions under a size constraint, called MOMS. We first prove that MOMS can achieve the asymptotically tight approximation bound for general cases, which was also obtained by the greedy algorithm; and then show that MOMS can be better than the greedy algorithm on special instances. The empirical results on the applications of influence maximization, information coverage maximization and sensor placement verify the superior performance of MOMS. Particularly, for sensor placement with $k = 1$ where the greedy algorithm achieves the nearly optimal solution, the comparison results show that MOMS can bring a performance improvement even when the greedy algorithm has been almost optimal.

Note that for solving the bi-objective reformulation of the original studied problem, MOMS employs a simple MOEA [16], which uses mutation only. In the future, we will try to use some state-of-the-art MOEAs such as NSGA-II [6] and MOEA/D [37] in MOMS, which may bring a larger performance improvement; and study their theoretical performance.

Acknowledgments

This research was supported by the National Science Foundation of China (61603367, 61333014, 61672478), the Young Elite Scientists Sponsorship Program by CAST (2016QNRC001) and the Collaborative Innovation Center of Novel Software Technology and Industrialization. K. Tang was also supported in part by the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284). This research was started when C. Qian was at the LAMDA Group, Nanjing University. Z.-H. Zhou is the corresponding author.

References

[1] N. Barbieri, F. Bonchi, and G. Manco, "Topic-aware social influence propagation models," in *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM'12)*, Brussels, Belgium, 2012, pp. 81–90.

[2] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 658–675, 2006.

[3] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, Paris, France, 2009, pp. 199–208.

[4] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11, pp. 1245–1287, 2002.

[5] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[7] U. Feige, "A threshold of $\ln n$ for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[8] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt, "Approximating covering problems by randomized search heuristics using multi-objective models," *Evolutionary Computation*, vol. 18, no. 4, pp. 617–633, 2010.

[9] S. Fujishige and S. Iwata, "Bisubmodular function minimization," *SIAM Journal on Discrete Mathematics*, vol. 19, no. 4, pp. 1065–1073, 2005.

[10] A. Goyal, W. Lu, and L. V. Lakshmanan, "Simpath: An efficient algorithm for influence maximization under the linear threshold model," in *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM'11)*, Vancouver, Canada, 2011, pp. 211–220.

[11] T. Hogg and K. Lerman, "Social dynamics of digg," *EPJ Data Science*, vol. 1, no. 5, pp. 1–26, 2012.

[12] A. Huber and V. Kolmogorov, "Towards minimizing k-submodular functions," in *Combinatorial Optimization*, 2012, pp. 451–462.

[13] S. Iwata, S. Tanigawa, and Y. Yoshida, "Improved approximation algorithms for k-submodular function maximization," in *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*, Arlington, VA, 2016, pp. 404–413.

[14] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, Washington, DC, 2003, pp. 137–146.

[15] C.-W. Ko, J. Lee, and M. Queyranne, "An exact algorithm for maximum entropy sampling," *Operations Research*, vol. 43, no. 4, pp. 684–691, 1995.

[16] M. Laumanns, L. Thiele, and E. Zitzler, "Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 170–182, 2004.

[17] K. Li, F. He, and X. Chen, "Real-time object tracking via compressive feature selection," *Frontiers of Computer Science*, vol. 10, no. 4, pp. 689–701, 2016.

[18] S. T. McCormick and S. Fujishige, "Strongly polynomial and fully combinatorial algorithms for bisubmodular function minimization," *Mathematical Programming*, vol. 122, no. 1, pp. 87–120, 2010.

[19] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," *Optimization Techniques*, vol. 7, pp. 234–243, 1978.

[20] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause, "Lazier than lazy greedy," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, Austin, TX, 2015, pp. 1812–1818.

[21] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of Operations Research*, vol. 3, no. 3, pp. 177–188, 1978.

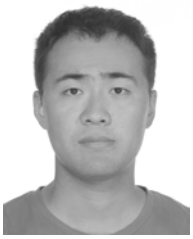
[22] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions – I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[23] F. Neumann and I. Wegener, "Minimum spanning trees made easier via multi-objective optimization," *Natural Computing*, vol. 5, no. 3, pp. 305–319, 2006.

[24] F. Neumann, J. Reichel, and M. Skutella, "Computing minimum cuts by randomized search heuristics," *Algorithmica*, vol. 59, no. 3, pp. 323–342, 2011.

[25] N. Ohsaka and Y. Yoshida, "Monotone k-submodular function maximization with size constraints," in *Advances in Neural Information Processing Systems 28 (NIPS'15)*, Montreal, Canada, 2015, pp. 694–702.

[26] C. Qian, Y. Yu, and Z.-H. Zhou, "On constrained Boolean Pareto optimization," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, Buenos Aires, Argentina, 2015, pp. 389–395.
[27] ——, "Subset selection by Pareto optimization," in *Advances in Neural Information Processing Systems 28 (NIPS'15)*, Montreal, Canada, 2015, pp. 1765–1773.
[28] C. Qian, J.-C. Shi, Y. Yu, and K. Tang, "On subset selection with general cost constraints," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, Melbourne, Australia, 2017, pp. 2613–2619.
[29] C. Qian, J.-C. Shi, Y. Yu, K. Tang, and Z.-H. Zhou, "Parallel Pareto optimization for subset selection," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, New York, NY, 2016, pp. 1939–1945.
[30] D. Sharma, A. Deshpande, and A. Kapoor, "On greedy maximization of entropy," in *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, Lille, France, 2015, pp. 1330–1338.
[31] A. P. Singh, A. Guillory, and J. Bilmes, "On bisubmodular maximization," in *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS'12)*, La Palma, Canary Islands, 2012, pp. 1055–1063.
[32] J. Thapper and S. Živnỳ, "The power of linear programming for valued CSPs," in *Proceedings of the 53rd IEEE Annual Symposium on Foundations of Computer Science (FOCS'12)*, New Brunswick, NJ, 2012, pp. 669–678.
[33] S. Venkatraman and G. G. Yen, "A generic framework for constrained optimization using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, pp. 424–435, 2005.
[34] Z. Wang, E. Chen, Q. Liu, Y. Yang, Y. Ge, and B. Chang, "Maximizing the coverage of information propagation in social networks," in *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*, Buenos Aires, Argentina, 2015, pp. 2104–2110.
[35] J. Ward and S. Živnỳ, "Maximizing bisubmodular and k-submodular functions," in *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, Portland, OR, 2014, pp. 1468–1481.
[36] Y. Yu, X. Yao, and Z.-H. Zhou, "On the approximation ability of evolutionary optimization with application to minimum set cover," *Artificial Intelligence*, vol. 180, pp. 20–33, 2012.
[37] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.



Chao Qian received the BSc and PhD degrees in computer science from Nanjing University, China, in 2009 and 2015, respectively. Since then, he joined the School of Computer Science & Technology at University of Science and Technology of China as an associate researcher. His research interests are mainly in artificial intelligence, evolutionary computation and machine learning, particularly, the foundation of evolutionary algorithms and its application in machine learning. He has several papers in leading international journals and conference proceedings, including *Artificial Intelligence*, *Evolutionary Computation*, *IEEE Transactions on Evolutionary Computation*, NIPS, IJCAI, AAAI, etc. He has won the ACM GECCO 2011 Best Paper Award (Theory Track), the IDEAL 2016 Best Paper Award, the IBM PhD Fellowship Award, and has been in the team of the PAKDD 2012 Data Mining Competition (Open Category) Grand Prize Winner. He has also been selected to the Young Elite Scientists Sponsorship Program by CAST.

Jing-Cheng Shi received the BSc degree in the Department of Mathematics from Nanjing University, China, in 2015. In the same year, he was admitted to study for a MSc degree in the Department of Computer Science & Technology at Nanjing University without entrance examination. His research interests are mainly in artificial intelligence, evolutionary computation and machine learning.

Ke Tang (M'07-SM'13) received the B.Eng. degree from Huazhong University of Science and Technology, Wuhan, China, in 2002, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2007, respectively. Currently, he is a Professor in the Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech), Shenzhen, China. Prior to that, he was an Associate Professor (2007-2011) and a Professor (2011-2017) in the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China. He has authored/co-authored more than 120 refereed publications. His major research interests include evolutionary computation, machine learning, and their real-world applications. Dr. Tang is an Associate Editor of the IEEE Transactions on Evolutionary Computation and served as a member of Editorial Boards for a few other journals. He received the Royal Society Newton Advanced Fellowship (2015) and the 2018 IEEE Computational Intelligence Society Outstanding Early Career Award.

Zhi-Hua Zhou (S'00-M'01-SM'06-F'13) received the BSc, MSc and PhD degrees in computer science from Nanjing University, China, in 1996, 1998 and 2000, respectively, all with the highest honors. He joined the Department of Computer Science & Technology at Nanjing University as an Assistant Professor in 2001, and is currently Professor and Standing Deputy Director of the National Key Laboratory for Novel Software Technology; he is also the Founding Director of the LAMDA group. His research interests are mainly in artificial intelligence, machine learning and data mining. He has authored the books *Ensemble Methods: Foundations and Algorithms* and *Machine Learning* (in Chinese), and published more than 150 papers in top-tier international journals or conference proceedings. He has received various awards/honors including the National Natural Science Award of China, the PAKDD Distinguished Contribution Award, the IEEE ICDM Outstanding Service Award, the IEEE CIS Outstanding Early Career Award, the Microsoft Professorship Award, etc. He also holds 18 patents. He is an Executive Editor-in-Chief of the *Frontiers of Computer Science*, Associate Editor-in-Chief of the *Science China Information Sciences*, Action Editor or Associate Editor of the *Machine Learning*, *ACM Transactions on Intelligent Systems and Technology*, *IEEE Transactions on Neural Networks and Learning Systems*, etc. He served as Associate Editor-in-Chief for *Chinese Science Bulletin* (2008-2014), Associate Editor for *IEEE Transactions on Knowledge and Data Engineering* (2008-2012), Neural Networks (2014-2016), *Knowledge and Information Systems* (2003-2008), etc. He founded ACML (Asian Conference on Machine Learning), served as Advisory Committee member for IJCAI (2015-2016), Steering Committee member for ICDM, PAKDD and PRICAI, and Chair of various conferences such as General co-chair of PAKDD 2014 and ICDM 2016, Program co-chair of SDM 2013 and IJCAI 2015 Machine Learning Track, and Area chair of NIPS, ICML, AAAI, IJCAI, KDD, ICDM, etc. He is/was the Chair of the IEEE CIS Data Mining Technical Committee (2015-2016), the Chair of the CCF-AI(2012-), and the Chair of the Machine Learning Technical Committee of CAAI (2006-2015). He is a Fellow of the ACM, AAAI, AAAS, IEEE, IAPR, IET/IEE and CCF.