Graph-based maximization algorithms for submodular functions and influence maximization on social networks

PhD Thesis

Eszter Csókás
Supervisor: Tamás Vinkó, PhD

Doctoral School of Computer Science
Department of Computational Optimization
Faculty of Science and Informatics
University of Szeged

Szeged
2025

1 Submodular function maximization based on graph properties

Submodular function maximization Let $N = \{1,\dots,n\}$ be a finite set. The function $f : 2^N \to \mathbb{R}$ is called *submodular* if it fulfills $f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$ for all $S, T \subseteq N$. Perhaps more intuitive is the following equivalent definition: $f(\{i\} \mid S) \geq f(\{i\} \mid T)$ holds for every $S \subseteq T \subseteq N$ and $i \in N \setminus S$ where $f(\{i\} \mid S) := f(S \cup \{i\}) - f(S)$. A submodular function f is called *non-decreasing* if $f(S) \leq f(T)$ holds for all $S \subseteq T \subseteq N$. In the rest of the thesis it is assumed that f is a non-decreasing submodular function.

The submodular maximization problem with a cardinality constant k (where $0 < k \leq n$) is defined in the following way:

$$\begin{array}{rl} \max & f(S) \\ \text{subject to} & |S| \leq k,\ S \subseteq N. \end{array} \tag{1}$$

Solvability The greedy method for solving the non-decreasing, monotone submodular maximization problem with cardinality constraint was proposed in [11]. They showed that the greedy strategy achieves an $(1-1/e)$-approximation of the optimal solution.

Although this is a simple and efficient technique for solving many optimization problems and is therefore very commonly used, the global optimum is often more needed in real-world applications, which was the motivation for a new solution method proposed in [10]. This is based on a mixed integer programming (MIP) model:

$$\begin{array}{ll} \max & z \\ \text{s.t.} & z \leq f(S) + \sum\limits_{i \in N \setminus S} f(\{i\} \mid S) \cdot y_i, \quad S \in F, \\ & \sum\limits_{i \in N} y_i \leq k, \\ & y_i \in \{0,1\},\ \ i \in N, \end{array} \tag{2}$$

where $f(T \mid S) := f(S \cup T) - f(S)$ for all $S, T \subseteq N$ and F denotes the set of all feasible solutions satisfying the cardinality constraint $\mid S \mid \leq k$.

Since the number of constraints increases exponentially in (2), this motivated a new procedure, the so-called constraint generation (CG) algorithm, proposed in [10]. CG is an iterative algorithm that starts with solving a reduced problem. The reduced problem consists of a set of constraints generated from the initial set, which is extended on demand at each iteration by the addition of a feasible solution. So in essence it solves many reduced MIP problems, which are not always sufficiently efficient in applications. For this reason, the branch-and-bound algorithmic approach is often used, which exploits the relaxation of MIP.

1.1 Constraint generation approaches

I/1, I/2 and I/3 contribute to the research effort invested into submodular function maximization with cardinality constraints by introducing efficient variations of a recently proposed constraint generation (CG) algorithm [12]. More precisely, we introduce variants of the CG algorithm which take into account certain properties of the input graph aiming at informed selection of the constraints.

Algorithm 1 CG($S^{(0)}$)

```
Input The initial feasible solution $S^{(0)}$.
Output The optimal solution $S^*$ of problem (2).
Step 1: Set $Q \leftarrow S^{(0)}$, $S^* \leftarrow S^{(0)}$ and $t \leftarrow 1$.
Step 2: Solve MIP($Q$). Let $z^{(t)}$ be the optimal value of MIP($Q$) and $S^{(t)}$ is the set corresponding to the
        optimal solution $\mathbf{y}^{(t)}$ of MIP($Q$).
Step 3: If $f(S^{(t)}) > f(S^*)$, then let $S^* \leftarrow S^{(t)}$.
Step 4: If $z^{(t)} = f(S^*)$ holds, then output the solution $S^*$ and exit.
Step 5: Set $Q \leftarrow Q \cup \{S^{(t)}\}$, $t \leftarrow t+1$ and return to Step 2.
```

Constraint generation algorithm (CG) A constraint generation algorithm was proposed in [10], which starts from a reduced MIP problem to start with a few constraints to solve. It is an iterative algorithm, and in every iteration solving the reduced MIP problem while adding a new constraint. Let define the reduced problem MIP(Q), where $Q \subseteq F$ is a set of feasible solutions:

$$\begin{array}{ll} \max & z \\ \text{s.t.} & z \leq f(S) + \displaystyle\sum_{i \in N \setminus S} f(\{i\} \mid S) \cdot y_i, \quad S \in Q, \\ & \displaystyle\sum_{i \in N} y_i \leq k, \\ & y_i \in \{0,1\}, \quad i \in N. \end{array} \tag{3}$$

The pseudo code of CG is shown in Algorithm 1. The starting point of the algorithm is a set $Q = \{S^{(0)}_{[0]}, \dots, S^{(0)}_{[k]}\}$, where $S_{[i]}$ is the first i elements of a feasible solution $S^{(0)}$ which comes from the order of the greedy algorithm's solution. In the t-th iteration we solve the problem MIP(Q), $Q = \{S^{(0)}_{[0]}, \dots, S^{(0)}_{[k]}, \dots, S^{(t-1)}\}$ to obtain the optimal solution $\mathbf{y}^{(t)} = (y^{(t)}_1, \dots, y^{(t)}_n)$ and $z^{(t)}$ the optimal value which gives an upper bound of the problem (2). Let S^* be the best feasible solution of problem (2) up to this point and $S^{(t)} \in F$ be the set which is generated the optimal solution $\mathbf{y}^{(t)}$ of MIP(Q), i.e., $\mathbf{y}^{(t)}$ is the characteristic vector of $S^{(t)}$. When $f(S^{(t)}) > f(S^*)$ holds, then update S^* with $S^{(t)}$. If $z^{(t)} > f(S^*) \geq f(S^{(t)})$ holds, then we have that $S^{(t)} \notin Q$, so (in Step 5) add $S^{(t)}$ to Q. This effectively adds the following constraint to MIP(Q):

$$z \leq f(S^{(t)}) + \sum_{i \in N \setminus S^{(t)}} f(\{i\} \mid S^{(t)}) \cdot y_i. \tag{4}$$

The algorithm stops when $z^{(t)} = f(S^*)$ is satisfied which means that it is proven that the optimal solution is found.

ICG In [12], the authors proposed an improved generation method based on CG, where not one but several constraints are added per iteration. It complements Algorithm 1 by creating a new set Q^+ containing the elements of the set Q and the result of the internal sub-algorithm (SUB-ICG).

Step 5 of the CG algorithm is completed by calling SUB-ICG and adding its return value to the set Q^+. If a solution is found in this part of the algorithm whose function value is greater than the current $f(S^*)$, it is also updated.

The SUB-ICG is an iterative algorithm that generates $\lambda = 10 \cdot k$ new feasible solutions (i.e., k vertices are selected). To do this, it uses a heuristic method to assign a value p_i to the vertices $i \in N$. This is based on the number of times the vertices $i \in N$ appear in the sets $S \in Q$. These are recalculated after updating the set Q. Finally, the final selection order is given by r_i, which is generated randomly such that $0 \leq r_i \leq p_i$.

We propose three modifications of ICG in which either certain characteristics of the graph describing the problem is used or the submodularity property of the function to be maximized is exploited.

ICG with reduced k (ICG($k-1$)) We created ICG with reduced k. What has been changed from the ICG is that in SUB-ICG we choose $k-1$ nodes for the constraints. Thus, the function value calculated in (3) computes the value of the k-th vertex when adding it to the set. We have kept this change for the remaining algorithm variants, i.e., we choose $k-1$ nodes for the constraints in both GCG and ECG, which are presented below.

ICG using graph structure (GCG) In this variant of ICG($k-1$), we changed the heuristic that calculated the value p_i to select the nodes in the SUB-ICG. The new heuristic is based on the structure of the input graph. If the graph is fully connected, then we do not consider all of the edges. Specifically, we compute the median of the outgoing edges' weights for every vertex $j \in M$. Edges with weights less than the median are ignored. The p_i value is the sum of the weights of the incoming edges at node $i \in N$ normalized with the degree of the node in M corresponding to the edge:

$$p_i = \begin{cases} \displaystyle\sum_{\substack{j:(j,i)\in E(G),\\ w_{ji}\geq m_j}} \frac{w_{ji}}{d_j} & \text{if } G \text{ is fully connected bipartite graph,} \\ \displaystyle\sum_{j:(j,i)\in E(G)} \frac{w_{ji}}{d_j} & \text{otherwise,} \end{cases} \tag{5}$$

where G is the input graph of the optimization problem, $E(G)$ is the set of edges of G, the edges have w_{ji} weights and d_j is the degree of the node $j \in M$. This defines the value of p_i and based on this we set the value of r_i by uniformly at random such that the relation $0 \leq r_i \leq p_i$ holds.

We also use the concept coming from ICG($k-1$), so we select $k-1$ nodes.

As an illustrative example, see the graph on Figure 1, where the labels of the nodes are indicated as black numbers. The results of equation (5) are shown in Figure 1: the value of p_i for the node is to the left.

ICG using enumeration (ECG) In contrast to ICG and GCG, in this variation no sub-algorithm is repeated within the main algorithm, which can result in less computation time. Instead of the sub-algorithm, we add a few additional steps to the Algorithm 1 so that we still generate λ constraints per iteration. We choose some nodes from the union of the set $S^{(t)}$ and a randomly chosen set Q of feasible solutions satisfying certain conditions. The choice is based on a graph structure according to the largest p_i values. From these we generate subsets with cardinality at most $k-1$

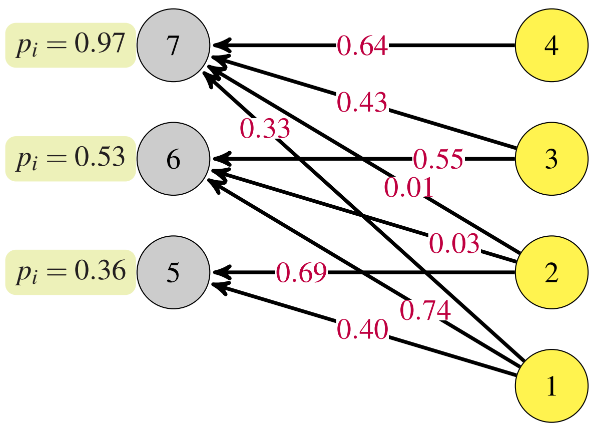

Figure 1: *Example graph to calculate the p_i values*

and compute their function values. Of these we keep λ subsets with the smaller function values. These correspond to the return sets of SUB-ICG.

Conclusion We use 3 different benchmark instances for testing and comparison. According to our benchmarking results, we cannot declare a clear winner among the algorithms and it is not surprising as the investigated problem is NP-hard. However, for every instances there exists at least one of our algorithms which is computationally more efficient than the ICG algorithm.

1.2 On the initial set of constraints

It can be observed that the choice of the starting point plays a role in the efficiency (i.e., its runtime) of the CG algorithm. More precisely, starting from a high function-valued initial point *might not* provide the fastest runtime. This effect is illustrated in Figure 2.

We created 250 test cases, part of which started from a random starting point, while in the other part we chose 3 of the best 5 vertices belonging to the global optimum, fixed them and randomly added 2 other nodes. The reason for this is that we did not get an initial set with a larger function value in the random choices, so we have also biased the sensitivity analysis a bit towards the more interesting scenarios.

Figure 2 shows the scatter plot of the 250 test cases. It is important to note that in this figure, the x-axis shows the runtime of the CG algorithm and the y-axis shows the function value of the initial set. In the figure, two sets of points can be roughly separated, due to the semi-random chosen test cases. The green dot indicates the original CG algorithm starting from the initial point proposed by greedy. Note that for the other results, we used all subsets of randomly generated points as starting point since we did not have any order. This setup is appropriate, because for the *NS* method, which will be presented later, the starting point is all the subset of the selected vertices. This figure perfectly illustrates that the running time of a CG algorithm can be very different even if the function value of the initial points are similar.

The phenomenon introduced informally above is the main motivation for Thesis I/4 and I/5.

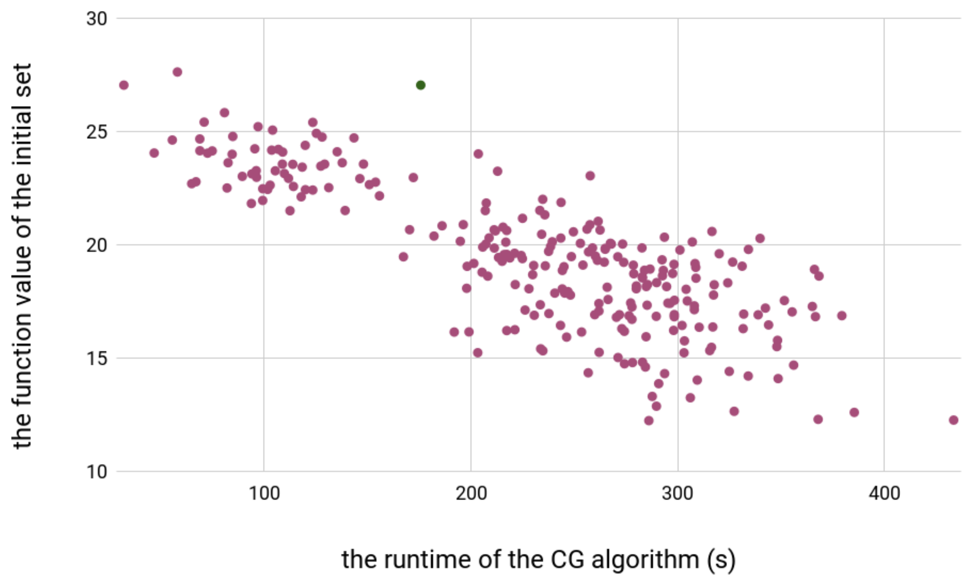

Figure 2: *Visualization of the sensibility of the starting point: starting points with similar function values can have rather different running times*

New centrality values To find a new starting point for a CG-type algorithm we use the input graph's structure.

To choose k node as a new starting point, first of all, we calculate a new centrality value ns_i to every node $i \in N$. This centrality is adding up the weights of the incoming edges at node $i \in N$ normalized with the degree of the targets node, then multiplying the sum with the degree of the source node:

$$ns_i = d_i \cdot \sum_{j:(j,i)\in E(G)} \frac{w_{ji}}{d_j}, \tag{6}$$

where G is the input graph of the optimization problem, $E(G)$ is the set of edges of G, the edges have w_{ji} weights, d_j is the degree of the node $j \in M$ and d_i is the degree of node $i \in N$.

Choose node i with highest ns_i value and delete node i with their edges and recalculate all the ns_i for every node $i \in N$. The next vertex is chosen for the starting point based on the recomputed centrality value. Repeat this method until k nodes are selected.

See the graph on Figure 3 as a small illustrative example, where the labels of the nodes are marked with black numbers. The vertices signed by their labels and their corresponding ns_i values calculated by (6) are shown next to them highlighted by tanning color. Taking the ns_i values into account, we first choose node 7 and then delete this vertex with its edges. Then, the result graph is shown on Fig. 4 with its recalculated ns_i values. Accordingly, the next selected vertex is 5 and not node 6, but note that in the first step it seemed that node 6 is the better choice.

With this procedure we choose k of vertices.

Finally, we generate all subsets from the k vertices proposed by the new centrality metric. We start a CG-type algorithm from the constraints defined by these subsets.

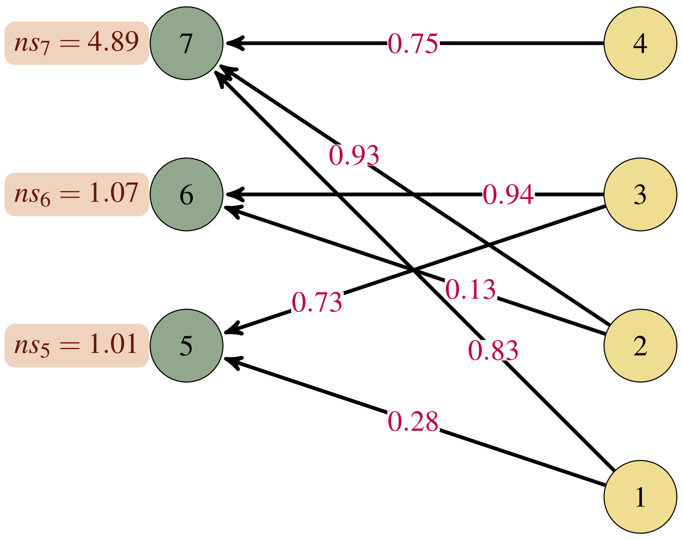

Figure 3: *Example graph to calculate the ns_i values; initial step*

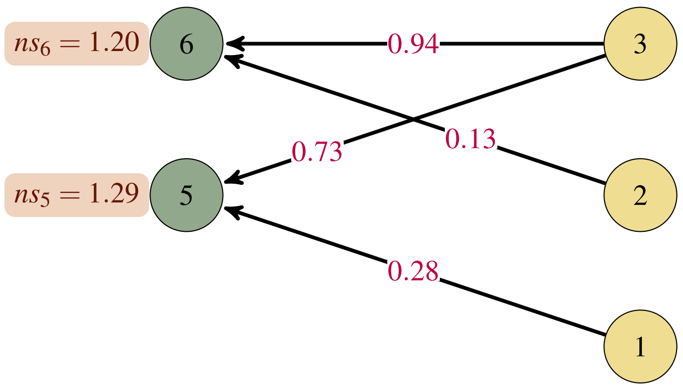

Figure 4: *Example graph to calculate the ns_i values; result of the first iteration*

Conclusion We used five different algorithm variants for the non-decreasing submodular function maximization problem based on a MIP formulation using constraint generation approach which we started from the greedy's solution and also from the new starting point proposed by the centrality metric. Furthermore, we used two modern implementations of Nemhauser and Wolsey's MIP model for submodular function maximization problem based on lazy constraint generation, which we started from the GRASP heuristic and also from the new starting point proposed by the centrality metric. According to our benchmarking results, algorithms starting from the new initial set reduced the runtime by a factor of 5.37 for all test cases. Overall, we can conclude the initial set suggested by the new centrality metric is worth using, as shown by our run-time tests and, in their absence, the relative gap tests.

2 Influence maximization under deterministic linear threshold model

Influence maximization (IM) is a combinatorial optimization problem in which, given a weighted directed graph G, a diffusion (or spreading) model, and an integer $k \geq 1$, it is required to identify the so-called seed nodes $v_1, \ldots, v_k \in V$ which can make the largest influence in the network [8]. Formally, the optimization problem can be described as

$$\max_{S \subset V, |S| = k} \sigma(S).$$

[8] investigated the influence maximization problem using spreading models with stochastic parameters.

Several diffusion models have been proposed in the literature. We use the linear threshold model [6], where let $b_{i,j} \in (0,1)$ be the edge weight between node i and j, $\theta_i \in (0,1]$ be the threshold of node i, and set $\hat{N}(i)$ be the already influenced in-neighbors of node i. In the deterministic LT model (DLTM) all the θ_i threshold values are fixed.

The second group of thesis was inspired by the model proposed in [9], where two integer linear programming formulations of influence maximization based on the DLTM were recently proposed and studied.

2.1 An exact method

Thesis points II/1, II/2 and II/3 are obtained from the analysis and correction of the algorithm proposed in [9].

A 0-1 linear programming model The formulation of the basic model is a special 0-1 LP, in which $\mathbf{x} \in \{0,1\}^{n \times \mathcal{T}}$ is the decision variable, $n = |V|$, and the index $\mathcal{T} > 1$ is also part of the optimization problem. Hence, $\mathbf{x}$ is a binary matrix in which choosing the rows in the first column to be equal to 1 represents the selection of the seed nodes. This should be done in such a way that, given certain constraints dictated by IM, the sum of the last column is to be maximized.

Assuming that $\mathcal{T} > 1$ is a given integer constant, let $T = \{2, \ldots, \mathcal{T}\}$ be the set of time periods describing the diffusion process. Let integer $k > 0$ be the number of seed nodes to be selected. The set of in-neighbors of node i is denoted by $N_{in}(i)$.

In the following the binary LP formulation is given, inspired by the basic model of [9], where the cost of selecting a seed node is equal to 1.

$$\max \sum_{i=1}^{n} x_{i,\mathcal{T}} \tag{7}$$

$$\sum_{i=1}^{n} x_{i,1} \leq k \tag{8}$$

$$\sum_{j \in N_{in}(i)} b_{j,i} x_{j,t-1} \geq \theta_i x_{i,t} \quad \forall (i \in V, t \in T) \tag{9}$$

$$\sum_{j \in N_{in}(i)} b_{j,i} x_{j,t-1} \leq \theta_i + x_{i,t} \quad \forall (i \in V, t \in T) \tag{10}$$

$$x_{i,t-1} \leq x_{i,t} \quad \forall (i \in V, t \in T) \tag{11}$$

$$\mathbf{x} \in \{0,1\}^{n \times \mathcal{T}}. \tag{12}$$

The objective function in fact has the form

$$\min_{\mathcal{T}} \max \sum_{i=1}^{n} x_{i,\mathcal{T}}$$

and together with constraints (8) - (12) we have a bilevel optimization problem. It is shown that linear bilevel problems are strongly NP-hard [7].

The AMPL modeling language [5], which we used for implementation and numerical experiments is not suitable for directly describing bilevel optimization models. Hence, we need to consider and treat $\mathcal{T}$ as a constant.

An iterative algorithm The solution method for the bilevel optimization problem proposed in [9] is shown in Algorithm 2.

Algorithm 2

Step 1 Start the iteration from $\mathcal{T} := 2$.

Step 2 Solve the optimization problem (7) - (12) with fixed $\mathcal{T}$.

Step 3 If $\mathbf{x}_{i,\mathcal{T}} = \mathbf{x}_{i,\mathcal{T}-1} \quad \forall (i \in V)$, i.e., the last two columns of $\mathbf{x}$ are the same then STOP, the optimum is found. Otherwise, let $\mathcal{T} := \mathcal{T}+1$ and go back to Step 2.

Analysis In the following, I give the steps needed to obtain theoretical results, without providing proofs of them.

For a start, it turns out that the optimization problem (7)-(12) needs to be modified.

Proposition 1. *For the correctness of Algorithm 2, the constraint* (9) *has to be replaced by*

$$\sum_{j \in N_{in}(i)} b_{j,i} x_{j,t-1} \geq \theta_i (x_{i,t} - x_{i,t-1}) \ \ \forall (i \in V, t \in T). \tag{13}$$

Remark 1. *Note that constraint* (13) *is equivalent to constraint* (9) *together with adding loop edges to all the nodes. However, it turns out that from the computational efficiency point of view using* (13) *directly is more beneficial.*

In the following we show that Algorithm 2 can get stuck in locally optimal solution even if the newly added constraint (13) is taken into account.

Proposition 2. *For the optimization problem* (7)*,* (8)*,* (10) - (13)*, there is a graph for which*

$$(\sigma, \mathcal{T}) = (\sigma, \mathcal{T}+1) \quad \textit{and} \quad (\sigma, \mathcal{T}+1) < (\sigma, \mathcal{T}+2).$$

We conclude that an extension of the optimization model (7), (8), (10) - (13) is needed in order to have a strategy about when to stop the iterative algorithm to be sure that it indeed reached the globally optimal solution. At that end, the following constraint is added:

$$\sum_{i=1}^{n} x_{i,\mathcal{T}-1} + 1 \leq \sum_{i=1}^{n} x_{i,\mathcal{T}}. \tag{14}$$

The purpose of constraint (14) is to force that for a given $\mathcal{T}$, the last step of the diffusion must have at least one more influenced node than in the previous step. We can thus guarantee no repetition in the last two columns of matrix $\mathbf{x}$.

The following proposition claims that although constraint (14) guarantees no repetition in the last two columns of $\mathbf{x}$, we can obtain such result in which column duplication appears inside the solution matrix.

Proposition 3. *For increasing $\mathcal{T}$ values the solutions of* (7), (8), (10) - (14) *do not necessarily form a monotonically increasing sequence. Moreover, it can also happen that repetition occurs for consecutive columns in matrix* $\mathbf{x}$.

This can be avoided by changing constraint (10) into

$$\sum_{j \in N_{in}(i)} b_{j,i} x_{j,t-1} \leq \theta_i + x_{i,t} - \varepsilon \qquad \forall (i \in V, t \in T), \tag{15}$$

where $\varepsilon > 0$ is a small constant to make sure that the node is activated when the sum of the edge weight of the already influenced in-neighbors of node is equal to the threshold. The following proposition claims that adding constraint (14) to the ILP model does not prune the globally optimal solution.

Proposition 4. *The globally optimal solution of* (7), (8), (11) - (13), (15) *satisfies constraint* (14) *as well.*

In addition to the previous proposition, it can also be shown that adding constraint (14) to the ILP model does not change the globally optimal solution.

Proposition 5. *The diffusion value $\mathcal{T}^*$ and influence value σ^* corresponding to the globally optimal solution of* (7), (8), (11) - (13), (15) *are respectively the same as the values $\mathcal{T}^{**}$ and σ^{**} corresponding to the global optimum of* (7), (8), (11) - (15).

Now we need to find stopping conditions to the iterative procedure. Clearly, one of them is when all nodes are influenced. The other one is when the model becomes infeasible.

Proposition 6. *If the problem* (7), (8), (11) - (15) *becomes infeasible for a given $\mathcal{T}$ value, then it remains to be infeasible for the further iteration steps as well.*

Finally, implicated by Proposition 5 and 6 we have the following consequence.

Corollary 7. *The problem* (7), (8), (11) - (15) *is feasible in the iteration steps* $2, \ldots, \mathcal{T}^*$, *i.e., before finding the global optimum.*

Conclusion Summarizing the above analysis, the correct model is:

$$\max \sum_{i=1}^{n} x_{i,\mathcal{T}} \tag{16}$$

$$\sum_{i=1}^{n} x_{i,1} \leq k \tag{17}$$

$$\sum_{j \in N(i)} b_{j,i} x_{j,t-1} \geq \theta_i (x_{i,t} - x_{i,t-1}) \ \ \forall (i \in V, t \in T) \tag{18}$$

$$\sum_{j \in N(i)} b_{j,i} x_{j,t-1} \leq \theta_i + x_{i,t} - \varepsilon \ \ \forall (i \in V, t \in T) \tag{19}$$

$$x_{i,t-1} \leq x_{i,t} \ \ \forall (i \in V, t \in T) \tag{20}$$

$$\sum_{i=1}^{n} x_{i,\mathcal{T}-1} + 1 \leq \sum_{i=1}^{n} x_{i,\mathcal{T}} \tag{21}$$

$$\mathbf{x} \in \{0,1\}^{n \times \mathcal{T}}. \tag{22}$$

The correct iterative algorithm to find the globally optimal solution of the influence maximization problem under deterministic linear threshold diffusion model is given in Algorithm 3.

Algorithm 3

Step 1 Start the iteration with $\mathcal{T} := 2$.

Step 2 Solve the problem defined by the set of equations $\{(16)$ - $(22)\}$ for the diffusion time value $\mathcal{T}$.

Step 3 If the solution becomes infeasible or all the nodes are influenced then STOP, the global optimum is found. Otherwise, let $\mathcal{T} = \mathcal{T} + 1$ and go back to Step 2.

2.2 A heuristic for seeds selection

The aim is to find a correlation between the input graph and the solution method. More precisely, whether the seed vertices selected at initial time have some distinguishable property; or whether we can exclude vertices from the initial set based on a feature. A good approach to this can be to use graph centrality, which establishes an ordering between nodes based on the values assigned to the vertices.

Combining these, I created two new centrality metrics that can be computed from the structure of the input graph. These can be used to minimize the number of possible seed nodes. The solver then chooses from the reduced set of possible seed nodes provided by the centrality metrics. By solving the problem in this way, the solution time is reduced. What makes them distinguished from other centralities is that they take into account not only the direction and the weight of the graph edges, but also the weight, i.e. threshold, of the nodes.

Influenceability ($\mathcal{I}_{in}$) It measures how easy is to activate a node. To calculate this, we examine the incoming edges from the neighbours, namely which edges and combinations of edges are able to reach or exceed the threshold value of the node. We define the weighted incidence, denoted by w_{to} as follows. Take the number of edge combinations that are able to activate the node by dividing by the number of edges in the edge combination and all occurrences of a given number of edge combinations. Finally, sum the w_{to} values and obtain $\mathcal{I}_{in}^{(p)} = \sum w_{to}(i)$, where $i \in V(G)$.

The final centrality metrics are obtained by combining with the measure of node and its neighbors. The influenceability value of a node is obtained by adding to the value of $\mathcal{I}_{in}^{(p)}$ the approximation of the influenceability of its in-neighbours:

$$\mathcal{I}_{in}(i) = \mathcal{I}_{in}^{(p)}(i) + \sum_{j \in N_{in}(i)} \frac{\mathcal{I}_{in}^{(p)}(j)}{|N_{out}(j)| - 1} \quad \forall (i \in V). \tag{23}$$

Note that if $|N_{out}(j)| \leq 1$, then let $|N_{out}(j)| = 2$ for the divisor to be 1.

Ability-to-influence ($\mathcal{I}_{out}$) This indicates the influencing role of the node on its neighbors. Specifically, we look at all the combinations of incoming edges to the neighbourhood which include the edge from the investigated node. Of these, we count the ones whose sum of weights reaches the threshold value of the node and calculate the weighted incidence value for this case, denote w_{from}. As calculated for the influenceability, we divide the number of infecting edges by the number of edges in the edge combination and all occurrences of a given number of edge combinations. Finally, summarized w_{from} for each investigated combinations of edges.

The ability-to-influence value of a vertex is obtained by adding to the value of $\mathcal{I}_{out}^{(p)}$ the approximation of the ability-to-influence of its out-neighbors:

$$\mathcal{I}_{out}(i) = \mathcal{I}_{out}^{(p)}(i) + \sum_{j \in N_{out}(i)} \frac{\mathcal{I}_{out}^{(p)}(j)}{|N_{in}(j)| - 1} \quad \forall (i \in V). \tag{24}$$

Note that if $|N_{in}(j)| \leq 1$, then let $|N_{out}(j)| = 2$ for the divisor to be 1.

Potential seed selections Using the two centrality values, we want to determine which vertices can be seeds. Therefore, first, the centrality values are normalized between 0 and 1 in a way that all the elements are divided with the maximum. Such normalization is denoted in each case by $||.||$. Then, we sort the nodes according to their centrality value. We put them in descending order according to their ability-to-influence value, since seed vertices should have good ability-to-influence's value. Conversely, we rank the vertices in ascending order according to their influenceability value, since seed vertices are unlikely to be easily infected. We take the weighted sum of the two order values for each node to get $\mathcal{I}$. This is shown in equation (25):

$$\mathcal{I}(i) = \alpha \cdot \text{ord}(||\mathcal{I}_{out}(i)||) + (1 - \alpha) \cdot \text{ord}(||\mathcal{I}_{in}(i)||) \quad \forall (i \in V). \tag{25}$$

Finally, to form the set of potential seed nodes, choose the subset of $V(G)$ according to $\mathcal{I}$. This is controlled by a parameter $0 < r < 1$, thus the cardinality of the candidate seeds set is $r \cdot |V(G)|$.

Proposed heuristic: IAtI Here we describe our proposal for a heuristic which selects a candidate seeder set of graph nodes based on the new centrality metrics. Since it is using the Influenceability and the Ability-to-Influence measures we refer to it as IAtI-heuristic. The method is described in Algorithm 4.

Algorithm 4 IAtI-heuristic(r, α)

```
Input  A directed graph $G$ with edge weights and node threshold values.
Step 1  Calculate $\mathcal{I}_{in}^{(p)}$ and $\mathcal{I}_{out}^{(p)}$ for all $i \in V$ and then $\mathcal{I}_{in}$ and $\mathcal{I}_{out}$ using Eq. (23) and (24), respectively.
Step 2  Form $\mathcal{I}$ for each vertex according to the Equation (25) using the input parameter $\alpha$.
Step 3  Define $S \subseteq V(G)$ to be the set of possible seeds: choose the top $r \cdot |V(G)|$ number of nodes from $\mathcal{I}$.
Step 4  Let $\mathcal{T} := 2$ and start the iteration.
Step 5  Solve the ILP defined by $\{(16)$ - $(22)\}$ for the diffusion time value $\mathcal{T}$, so that the seed vertices can be chosen exclusively from the set $S$.
Step 6  If all the nodes are influenced or the solution becomes infeasible then stop the iteration. Otherwise, let $\mathcal{T} = \mathcal{T} + 1$ and go back to Step 5.
```

Conclusion Using the two centrality metrics, we selected vertices that have a high probability of being seed nodes. The solver now selects seed vertices only among these. This reduces the computational complexity of the task and therefore, compared to running the ILP solver on the unrestricted model, it speeds up the procedure. IAtI algorithm was compared with Greedy and with the global optimum, also. The IAtI algorithm is slower than Greedy, but in many cases it gives a better solution and in most cases it finds the global optimum.

3 Contributions of the thesis

In the **first thesis group**, the contributions are related to Part I, the maximization of submodular functions. A detailed presentation is given in Chapters 2 and 3.

I/1. I developed a version of the constraint generating algorithm that works with subsets of $k-1$ elements.

I/2. I created another version of the constraint generation algorithm, GCG, which exploits the structure of the graph in a heuristic step.

I/3. I described another version of the constraint generating algorithm, ECG, which generates subsets directly.

I/4. I introduced a new centrality metric, which is an initial point selection strategy for solving submodular function maximization.

I/5. I show that the new starting point selection strategy is better than the commonly used greedy method or the recently published GRASP heuristic.

In the **second thesis group**, the contributions are related to Part II, the maximization of influence spread. A detailed presentation is given in Chapters 4 and 5.

II/1. I discovered that the integer model proposed in [9] is not correct in all cases.

II/2. I showed that the model needs to be completed to get the correct fit.

II/3. I demonstrated step-by-step that the new model we have constructed is correct.

II/4. By examining the step-by-step solution of the correct model, I presented two new centrality metrics for Influenceability and Ability-to-influence.

II/5. I demonstrated that by using a combination of the two centrality metrics, we can reduce the number of possible seed nodes at the initial moment.

Table 1 summarizes the relation between the thesis points and the corresponding publications.

Table 1: *Correspondence between the thesis points and my publications.*

Publication	**Thesis point**									
	I/1	I/2	I/3	I/4	I/5	II/1	II/2	II/3	II/4	II/5
[P1]						●	●	●		
[P2]	●	●	●							
[P3]									●	●
[P4]				●	●					

The author's publications on the subjects of the thesis

[P1] **Eszter Csókás** and Tamás Vinkó. An exact method for influence maximization based on deterministic linear threshold model. *Central European Journal of Operations Research*, 31, 269-286, 2023.

[P2] **Eszter Csókás** and Tamás Vinkó. Constraint generation approaches for submodular function maximization leveraging graph properties. *Journal of Global Optimization*, 88, 377-394, 2024.

[P3] **Eszter Csókás** and Tamás Vinkó. A heuristic for influence maximization under deterministic linear threshold model. *Informatica*, 48, 4, 2025.

[P4] **Eszter Csókás** and Tamás Vinkó. On the initial set of constraints for graph-based submodular function maximization. *Acta Cybernetica*, 2024.

Other References

[5] Robert Fourer, David Gay, and Brian Kernighan. *AMPL. A modeling language for mathematical programming*. Thomson, 1993.

[6] Mark Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.

[7] Pierre Hansen, Brigitte Jaumard, and Gilles Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217, 1992.

[8] David Kempe, Jon Kleinberg, and Éva 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*, pages 137–146, 2003.

[9] Muhammed Emre Keskin and Mehmet Güray Güler. Influence maximization in social networks: an integer programming approach. *Turkish Journal of Electrical Engineering & Computer Sciences*, 26(6):3383–3396, 2018.

[10] George Nemhauser and Laurence Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. *Studies on Graphs and Discrete Programming*, pages 279–301, 1981.

[11] George Nemhauser, Laurence Wolsey, and Marshall Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[12] Naoya Uematsu, Shunji Umetani, and Yoshinobu Kawahara. An efficient branch-and-cut algorithm for submodular function maximization. *Journal of the Operations Research Society of Japan*, 63(2):41–59, 2020.

4 Összefoglalás

Ez a disszertáció az optimalizálás és a hálózattudomány területén végzett kutatásaim eredményeit mutatja be. Célkitűzéseim közé tartozott, hogy az irodalmi áttekintést követően továbbfejlesszem az adott problémák megoldására szolgáló algoritmusokat az input gráfok különböző tulajdonságainak figyelembevételével. Az ilyen típusú megközelítés egyre nagyobb népszerűségnek örvend, és számomra is jelentős inspirációt nyújtott a kutatás során.

A dolgozatom két fő részből áll. Az első részben a szubmoduláris függvények maximalizálásával foglalkoztam, ezen belül olyan feladatokkal, melyek rendelkeznek gráfos reprezentációval. A második részben a befolyásterjedés maximalizálásával foglalkoztam, amit szintén gráfokon értelmezünk.

Az első fejezetben a dolgozat értéséhez szükséges optimalizálási és a gráfelméleti bevezető található. A 2. fejezetben korlátozó feltétel generáló megközelítésekről van szó. A szakirodalomban található megoldó eljárást tanulmányoztam és készítettem el annak három további változatát. Kifejlesztettem a korlátozó generáló algoritmus azon változatát, amely $k-1$ elemszámú részhalmazokkal dolgozik. Valamint egy másik változatát hoztam létre, a GCG-t, mely egy heurisztikus lépésben kihasználja a gráf szerkezetét. Végül megalkottam a korlátozó generáló algoritmus azon változatát, az ECG-t, ami közvetlen módon generálja a részhalmazokat. A 3. fejezetben a szubmoduláris függvények maximalizálására szolgáló eljárások kezdőpont választását vizsgáltuk meg. Kidolgoztam egy új stratégiát, amely egy új centralitási metrika alapján választ kiindulási pontot. A kezdőpont választási stratégiát a saját korlátozó feltétel generáló algoritmusainkkal és egy modern megoldó algoritmussal, mely a szakirodalomban fellelhető. Bemutattam, hogy az új kezdőpont választási stratégia jobb, mint az általában használt mohó eljárás vagy a nemrég publikált GRASP heurisztika.

A dolgozat második felében a befolyásterjedés maximalizálásával foglalkoztam, azon belül is azokkal a modellekkel, amelyek a determinisztikus lineáris küszöbmodellt használják terjedési modellként. Ebben a részben is gráfos reprezentációval bíró feladatokat vizsgáltam, egész pontosan azok megoldó algoritmusait. A 4. fejezetben egy 0-1 típusú programozási modellt analizáltam, amelyről beláttam, hogy nem helyes minden gráfpéldány esetén. Lépésenként bizonyítottam be a hibákat és azok javításait, végül megalkottam a helyes modellt, melynek helyességét igazoltam. Az 5. fejezetben a helyes modell segítségével kapott eredményeket vizsgáltam és kerestem a kapcsolatot a megoldás és a kiindulási (seed) csúcsok között. Végül két centralitási metrikát javasoltam a befolyásolhatóságra és a befolyásoló képességre. Kimutattam, hogy a két centralitási metrika kombinációját használva, redukálni lehet a kezdeti pillanatban lehetséges seed csúcsok számát. Ezzel pedig csökkenthető az egzakt megoldó futási ideje.

Nyilatkozat

Csókás Eszter "Graph-based maximization algorithms for submodular functions and influence maximization on social networks" című PhD diszszertációjában a következő eredményekben Csókás Eszter hozzájárulása volt meghatározó:

- A **2. fejezetben** felhasznált [P2] publikációban megjelent kutatás esetén: a három algoritmus variáns kifejlesztése a témavezetővel közös, míg az implementálás, az eljárások hatékonyságára végzett numerikus vizsgálatok egyéni munka volt.
- A **3. fejezetben** felhasznált [P4] publikációban megjelent kutatás esetén: a centralitási metrika ötlete a témavezetővel együttműködve született, majd az ebből készített algoritmus variánsok tervezése, megvalósítása és tesztelése egyéni munka.
- A **4. fejezetben** felhasznált [P1] publikációban megjelent kutatás esetén: a megoldó algoritmus javítására szolgáló állítások és azok bizonyításai a témavezetővel végzett közös munka eredményei, a helyes algoritmus implementálása, tesztelése saját munka.
- Az **5. fejezetben** felhasznált [P3] publikációban megjelent kutatás esetén: A központisági metrika alapötletét és megfogalmazását a témavezetővel közösen dolgoztuk ki. A technikai részeket önállóan végeztem, azaz a megvalósítást és a numerikus tesztelést.

Ezek az eredmények Csókás Eszter PhD disszertációján kívül más tudományos fokozat megszerzésére nem használhatók fel.

Szeged, 2025. május 19.

[illegible]
Csókás Eszter
jelölt

[illegible]
Dr. Vinkó Tamás
témavezető

Az Informatika Doktori Iskola vezetője kijelenti, hogy jelen nyilatkozatot minden társszerzőhöz eljuttatta, és azzal szemben egyetlen társszerző sem emelt kifogást.

Szeged, 2025/05/19

[illegible]
Dr. Jelasity Márk
Informatika Doktori Iskola vezetője