# LEADER-FOLLOWER GAMES FOR INFLUENCE SPREAD IN SOCIAL NETWORKS

by

Kübra Tanınmış Ersüs

M.S., Industrial Engineering, Boğaziçi University, 2014

B.S., Industrial Engineering, Boğaziçi University, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering
Boğaziçi University
2020

# ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude and thanks to my advisor Prof. Necati Aras and co-advisor Prof. İ. Kuban Altınel. I am truly grateful for having their guidance, support, and patience throughout this work. This thesis would not be completed without their invaluable contributions. I feel privileged to know them and to have the opportunity to work with them.

I would like to thank Assist. Prof. Evren Güney, Assist. Prof. Hande Küçükaydın, and Prof. Taner Bilgiç for their illuminating comments and suggestions throughout my study, and for being part of my thesis committee. Their constructive feedback had a great impact on the progress of my work. I also would like to thank Assoc. Prof. Tınaz Ekim Aşıcı for being part of my thesis committee and for her valuable comments.

I also would like to thank my friends Gökçe, Vuslat, Zeynep, Betül, Mert, and Gökalp for their friendship and assistance whenever I needed during my graduate study. We had a lot of experience and great times together in the beautiful campus of our university, and it was a pleasure to work with them.

Finally, I am deeply grateful to my family for their love and support. I thank my mother and my father for their encouragement, and my dear sisters Büşra and Şeyma for always being by my side. I am especially thankful to my beloved husband Ahmet for being extremely supportive and thoughtful throughout my study, and for cheering me up whenever I feel stressful. They all made it possible for me to complete my thesis successfully.

This study was partially supported by Boğaziçi University Scientific Research Project under the Grant number: BAP 12745D. I also thank TUBITAK for their financial support during my doctoral study under the BIDEB-2211 program.

# ABSTRACT

## LEADER-FOLLOWER GAMES FOR INFLUENCE SPREAD IN SOCIAL NETWORKS

Influence Maximization Problem involves finding a set of individuals in a social network to trigger an influence/information spread, i.e., a seed set, such that the maximum possible number of individuals are influenced. In this thesis, two competitive variants of the Influence Maximization Problem where two players make decisions sequentially in the form of a Stackelberg game are introduced. In the Influence Maximization with Deactivation the objectives of the leader and the follower are maximization and minimization of the spread, respectively. While the Misinformation Spread Minimization Problem has a reverse situation, in both problems the leader takes the follower's optimal response into account. Both problems are formulated as mixed-integer bilevel linear programs with a two-stage stochastic program in the lower level, by enumerating the diffusion scenarios due to the well known Linear Threshold model. In the solution phase, firstly several matheuristic methods are proposed where the follower's optimal decisions are approximated via Sample Average Approximation method. For the MSMP, additionally a state-of-the art influence maximization method is integrated into the developed algorithms. The proposed methods are evaluated on small instances where an optimal solution can be found via complete enumeration as well as larger instances where the algorithms are compared to each other. Then, a branch-and-cut algorithm for mixed-integer bilevel problems is enriched with a variable bound update procedure for the Influence Maximization with Deactivation which is shown to increase the time efficiency. An exact algorithm for bilevel interdiction problems is improved substantially, tested on various problem types including the Misinformation Spread Minimization Problem and yields significantly better results.

# ÖZET

## SOSYAL AĞLARDA ETKİ YAYILIMINA YÖNELİK ÖNCÜ-İZLEYİCİ OYUNLARI

Etki Enbüyükleme Problemi, bir sosyal ağ üzerinde etki/bilgi yayılımını başlattıklarında mümkün olan en yüksek sayıda kişinin etkilenmesine yol açacak bir dizi kişinin, yani bir çekirdek kümesinin bulunmasını içerir. Bu tezde, iki oyuncunun bir Stackelberg oyunu biçiminde sırayla karar verdiği, Etki Enbüyükleme probleminin iki rekabetçi türü tanıtılmıştır. Etkisizleştirme Durumunda Etki Enbüyükleme probleminde öncünün ve izleyicinin amaçları, sırasıyla yayılımın enbüyüklenmesi ve enküçüklenmesidir. Yanlış Bilgi Yayılımı Enküçükleme problemi tam tersi bir durumu ele alırken, her iki problemde de öncü kendi kararında izleyicinin eniyi karşılığını hesaba katar. Her iki problem de, yaygın olarak kullanılan Doğrusal Eşik modelinden kaynaklanan yayılım senaryoları numaralandırılarak, alt düzeyi iki aşamalı stokastik program olan bir iki düzeyli karışık tamsayılı doğrusal program olarak gösterimlenmiştir. Çözüm aşamasında ilk olarak, izleyicinin eniyi kararlarının Örneklem Ortalaması Yaklaştırma yöntemi ile bulunduğu matsezgisel yöntemler önerilmiştir. Yanlış Bilgi Yayılımı Enküçüklemesi için, geliştirilen algoritmalarla gelişmiş bir etki enbüyükleme yöntemi bütünleştirilmiştir. Önerilen yöntemler, tam birerleme yoluyla eniyi çözümün bulunabileceği küçük örneklerin yanı sıra algoritmaların birbirleriyle karşılaştırıldığı daha büyük örnekler yardımıyla da değerlendirilmiştir. Daha sonra, iki düzeyli karışık tamsayılı doğrusal programlama problemleri için bir dal-kesme algoritması, Etkisizleştirme Durumunda Etki Enbüyüklemesi için bilgisayısal etkinliği arttırdığı gösterilen değişken sınır güncelleme yaklaşımıyla zenginleştirilmiştir. İki düzeyli saldırı problemleri için kesin (eniyi) çözüm veren bir algoritma önemli ölçüde geliştirilmiş ve Yanlış Bilgi Yayılımı Enküçükleme problemini de içeren çeşitli problem türleri üzerinde denenerek daha iyi sonuçlar verdiği gösterilmiştir.

# TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . x
LIST OF TABLES . . . xiii
LIST OF SYMBOLS . . . xvi
LIST OF ACRONYMS/ABBREVIATIONS . . . xx
1. INTRODUCTION . . . 1
2. BACKGROUND INFORMATION . . . 5
2.1. Bilevel Programming . . . 5
2.2. Sample Average Approximation Method . . . 6
2.3. The Branch-and-Cut Algorithm with Intersection Cuts . . . 8
2.4. The $x$-space Algorithm . . . 9
3. LITERATURE REVIEW . . . 12
3.1. Influence Maximization Problems in Social Networks . . . 12
3.1.1. Influence Maximization in the Absence of a Competitor . . . 13
3.1.2. Competitive Influence Spread Problems . . . 18
3.2. Bilevel Programming . . . 25
3.2.1. Mixed Integer Bilevel Linear Programming . . . 26
3.2.2. Interdiction Problems . . . 29
4. INFLUENCE MAXIMIZATION PROBLEM WITH DEACTIVATION . . . 32
4.1. Problem Definition and Mathematical Models . . . 32
4.1.1. Formulation with Threshold Scenarios . . . 36
4.1.2. Formulation with Live-arc Scenarios . . . 39
4.2. Solution Methods . . . 42
4.2.1. Sample Average Approximation Method for the Follower's Problem 43
4.2.1.1. The Threshold Model . . . 43
4.2.1.2. The Live-arc Model . . . 45

4.2.1.3. Sampling Methods . . . . . 47
4.2.2. Complete Enumeration of the Leader's Solutions . . . . . 49
4.2.3. Matheuristic Methods for the Leader's Problem . . . . . 50
4.2.3.1. Simulated Annealing Based Matheuristic . . . . . 51
4.2.3.2. Tabu Search Based Matheuristic . . . . . 54
4.2.3.3. Simple Heuristics . . . . . 55
5. A BRANCH-AND-CUT ALGORITHM . . . . . 57
5.1. The High Point Relaxation and Bilevel Feasibility Constraints . . . . . 57
5.2. Bilevel-free Sets . . . . . 59
5.3. Alternating Bound Update Procedure . . . . . 62
5.4. Implementation Details . . . . . 65
6. MISINFORMATION SPREAD MINIMIZATION PROBLEM . . . . . 67
6.1. Problem Definition and Mathematical Model . . . . . 67
6.2. Solution Methods . . . . . 73
6.2.1. Sample Average Approximation Method for the Follower's Problem 74
6.2.2. Tabu Search Based Matheuristic . . . . . 75
6.2.3. Greedy Heuristic for the Protection Decision . . . . . 79
6.2.4. Binary Search Algorithm . . . . . 81
7. IMPROVED X-SPACE ALGORITHM FOR MIN-MAX BILEVEL PROBLEMS 86
7.1. Improved $x$-space Algorithm . . . . . 86
7.1.1. A New Lower Bound Problem Formulation . . . . . 88
7.1.2. Greedy Maximum Covering . . . . . 93
7.2. Implementation on the Misinformation Spread Minimization Problem . 95
7.2.1. Reformulation of the Problem . . . . . 95
7.2.2. Improved $x$-space Algorithm Implementation . . . . . 96
8. COMPUTATIONAL RESULTS . . . . . 99
8.1. Experimental Settings and Instance Generation . . . . . 99
8.2. Results for the Influence Maximization Problem with Deactivation . . . 100
8.2.1. Sample Average Approximation Method Results . . . . . 100
8.2.1.1. Optimality Gaps . . . . . 103
8.2.1.2. Effect of the Formulation . . . . . 105

8.2.2. Matheuristic Method Results . . . . . 110
8.2.2.1. Assessment of the Performances of the Matheuristics . 110
8.2.2.2. Comparison of the Outputs of the Matheuristics . . . . 113
8.2.2.3. Results Obtained by Simple Heuristics . . . . . 114
8.2.2.4. The Effect of the Formulation on Tabu Search Based Matheuristic Results . . . . . 115
8.3. Results of the Branch-and-Cut Algorithm with Intersection Cuts . . . . 120
8.3.1. Solution Time Reduction Due to Alternating Bound Update . . 121
8.3.2. The Comparison of Bilevel-free Sets . . . . . 122
8.3.3. Assessment of the Branch-and-Cut Algorithm with BFS6 . . . . 123
8.4. Results for the Misinformation Spread Minimization Problem . . . . . 124
8.4.1. IMM Algorithm Integration Results . . . . . 126
8.4.2. Assessment of Tabu Search Based Matheuristic and Greedy Protection Heuristic . . . . . 128
8.4.3. Assessment of the Binary Search algorithm . . . . . 129
8.4.4. Comparison of Tabu Search Based Matheuristic and Greedy Protection Heuristic . . . . . 130
8.4.5. A Solution Approach Based on Centrality Measures . . . . . 135
8.5. Results of the Improved $x$-space Algorithm . . . . . 136
8.5.1. Comparison of the Algorithms on Bilevel Knapsack Problem and Bilevel Clique Problem Instances . . . . . 137
8.5.2. Instance Generation and Results Obtained on the Misinformation Spread Minimization Problem Instances . . . . . 145
9. CONCLUSION . . . . . 150
9.1. Summary of the Contributions . . . . . 150
9.2. Future Research Directions . . . . . 152
REFERENCES . . . . . 154
APPENDIX A: k-means++ Algorithm . . . . . 170
APPENDIX B: Branch-and-Cut Method with Intersection Cuts . . . . . 171
B.1. Illustration of the Method . . . . . 171
B.2. Obtaining Intersection Cuts . . . . . 171

B.3. The Separation Mixed-integer Linear Program . . . . . 174
B.4. Additional Results . . . . . 175
APPENDIX C: $x$-space Algorithm . . . . . 177
APPENDIX D: Instance Generation . . . . . 180

# LIST OF FIGURES

Figure 4.1. Comparison of two IMPD ULP solutions. . . . . 34

Figure 4.2. Pseudo-code of the Sample Average Approximation (SAA) algorithm. 47

Figure 4.3. Pseudo-code of Simulated Annealing based matheuristic (SAM). . 53

Figure 4.4. Pseudo-code of Tabu Search based matheuristic (TSM) for IMPD. 55

Figure 5.1. Pseudo-code of Alternating Bound Update (ABU) procedure. . . . 65

Figure 6.1. Comparison of two MSMP ULP solutions. . . . . 70

Figure 6.2. A sample directed graph with unit arc weights. . . . . 73

Figure 6.3. Pseudo-code of Tabu Search based matheuristic (TSM) for the MSMP. . . . . 78

Figure 6.4. Pseudo-code of 1-Swap operator of TSM. . . . . 79

Figure 6.5. Pseudo-code of the Greedy Protection heuristic (GPH). . . . . 80

Figure 6.6. A small network with unit arc weights. . . . . 81

Figure 6.7. Pseudo-code of the Binary Search algorithm (BSA). . . . . 85

Figure 7.1. Pseudo-code of the Improved $x$-space (IXS) algorithm. . . . . 94

Figure 8.1. Comparison of LHS and SRS methods. . . . . 103

Figure 8.2. SAA gap estimates for cardinality-based instances. . . . . 105

Figure 8.3. SAA gap estimates for cost-based instances. . . . . 105

Figure 8.4. Average SAA times for cardinality-based instances. . . . . 108

Figure 8.5. Average SAA times for cost-based instances. . . . . 109

Figure 8.6. Budget class vs. average improvement in the spread. . . . . 119

Figure 8.7. Effect of BSA parameter $\beta$ on optimality gap. . . . . 130

Figure 8.8. Effect of BSA parameter $\beta$ on solution times. . . . . 131

Figure 8.9. Assessment of the results obtained by means of centrality measures. 136

Figure 8.10. Comparison of the methods for BKP instances. . . . . 142

Figure 8.11. Comparison of IXS and XS for BKP instances. . . . . 143

Figure 8.12. Comparison of the methods for BCP instances. . . . . 146

Figure 8.13. Comparison of the methods for MSMP instances. . . . . 149

Figure A.1. Separation of an Intersection Cut. . . . . 170

Figure B.1. IC example using $S^+(\hat{y})$. . . . . 172

Figure B.2. Separation of an Intersection Cut. . . . . 173

Figure C.1. The $x$-space (XS) algorithm. . . . . 177

Figure D.1. Selection of a subgraph of the co-authorship network. . . . . 180

# LIST OF TABLES

Table 8.1. Gap(%) Estimates of SAA algorithm for arbitrary leader strategy. 102

Table 8.2. Effect of the IMPD formulation on SAA results for cardinality-based instances. . . . . 106

Table 8.3. Effect of the IMPD formulation on SAA results for cost-based instances. . . . . 107

Table 8.4. Results obtained by SAM and TSM for cardinality-based IMPD instances. . . . . 111

Table 8.5. Results obtained by SAM and TSM for cost-based IMPD instances. 112

Table 8.6. Results of SAM and TSM for $n = 30$. . . . . 113

Table 8.7. Relative difference of the objective values found in SAM and TSM. 114

Table 8.8. Comparison of the simple heuristics with SAM and TSM for cardinality-based instances. . . . . 115

Table 8.9. Comparison of the simple heuristics with SAM and TSM for cost-based instances. . . . . 115

Table 8.10. Results of the TSM for cardinality-based instances. . . . . 117

Table 8.11. Results of the TSM for cost-based instances. . . . . 118

Table 8.12. Average percent improvement in the spread based on the budget classes. . . . . 119

Table 8.13. Results on the co-authorship network from `arXiv`. . . . . 120

Table 8.14. Percent reduction in solution times due to ABU. . . . . 122

Table 8.15. Number of optimally solved instances. . . . . 123

Table 8.16. Number of optimally solved instances. . . . . 123

Table 8.17. Average optimality gaps. . . . . 123

Table 8.18. Average optimality gaps. . . . . 124

Table 8.19. Assessment of the B&C algorithm. . . . . 125

Table 8.20. Average percent differences in the objective values of B&C and CE. 125

Table 8.21. Change in the number of SAA calls. . . . . 127

Table 8.22. Assessment of TSM and GPH. . . . . 128

Table 8.23. Number of SAA calls and tree nodes in BSA for different $\beta$. . . . . 129

Table 8.24. Comparison of the performances of TSM and GPH. . . . . 132

Table 8.25. Comparison of the performances of TSM and GPH. . . . . 133

Table 8.26. Results on the co-authorship network from `arXiv`. . . . . 135

Table 8.27. Results obtained on BKP instances. . . . . 140

Table 8.28. Results obtained on BCP instances using BCP′ formulation with penalty term in the objective function. . . . . 144

Table 8.29. Results obtained on BCP instances. . . . . 145

Table 8.30. Results obtained on MSMP instances. . . . . 148

Table B.1. Solution times with Alternating Bound Update. . . . . 175

Table B.2. Solution times without Alternating Bound Update. . . . . 176

Table B.3. Number of proven optimalities with (without) Alternating Bound Update. . . . . 176

Table B.4. Number of proven optimalities with (without) Alternating Bound Update. . . . . 176

# LIST OF SYMBOLS

| | |
|---|---|
| $A$ | Set of all arcs in the network |
| $A_t$ | Set of active nodes at step $t$ of diffusion process |
| $a_i^r$ | Nodes that can reach node $i$ in scenario $r$ |
| $a_{ji}(\mathbf{x}, r)$ | Parameter indicating if $j$ is reachable from $i$ in scenario $r$ |
| $B(\mathbf{x}, S_q)$ | Index set of solutions in $S_q$ that are blocked by $\mathbf{x}$ |
| $\bar{B}(\mathbf{x}, S_q)$ | Index set of solutions in $S_q$ that are not blocked by $\mathbf{x}$ |
| $b_i$ | In-degree of node $i$ |
| $b_{ji}^r$ | Nodes on the path from $j$ to $i$ in scenario $r$ |
| $C$ | Leader's budget to activate nodes |
| $C_\tau$ | Blocker set of $y^\tau$ |
| $c_i$ | Leader's cost of activating node $i$ |
| $c_q$ | Objective value of maximum covering problem at iteration $q$ |
| $\hat{c}_q$ | Objective value of greedy covering algorithm at iteration $q$ |
| $\bar{c}$ | Average activation cost in IMPD |
| $\text{conv}(\mathcal{X})$ | Convex hull of set $\mathcal{X}$ |
| $d$ | Network density |
| $E$ | Follower's budget to deactivate nodes |
| $e_i$ | Follower's cost of deactivating node $i$ |
| $f(S)$ | Pseudo objective value function used in SAM and TSM |
| $\widetilde{\mathcal{F}^\circ}$ | Relaxed feasible region of bilevel interdiction problem |
| $G$ | Graph |
| $G_Y$ | Subgraph of $G$ obtained by removing nodes in $Y$ |
| $g(\mathbf{x}, \mathbf{y}^\star, \boldsymbol{\theta})$ | Number of influenced nodes for threshold $\boldsymbol{\theta}$, leader solution $\mathbf{x}$, and optimal follower solution $\mathbf{y}^\star$ for $\mathbf{x}$ |
| $HC^+(\mathbf{x})$ | Set used for obtaining Hypercube IC's |
| $h$ | Number of protected nodes in MSMP |
| $J$ | Index set of follower solutions in $S_q$ |
| $J^B$ | Index set of follower solutions that must be blocked |

| | |
|---|---|
| $k$ | Number of activated nodes |
| $L$ | Cycle length in SAM |
| $\mathrm{LB}_q(\mathbf{x})$ | Inner problem of $\mathrm{LB}_q$ |
| $M$ | Number of ALLP's solved in SAA |
| $m$ | Number of arcs |
| $m_{ji}^r$ | Binary variable indicating if node $j$ influences $i$ in scenario $r$ |
| $N$ | Number of scenarios used to formulate an ALLP |
| $N'$ | Number of scenarios used in the second stage of SAA |
| $N''$ | Number of scenarios used in the third stage of SAA |
| $n$ | Number of nodes |
| $P^*(S)$ | Optimal follower seed for leader solution $S$ in MSMP |
| $p$ | Rewire probability used to generate WS networks |
| $p_0$ | Initial probability of acceptance in SAM |
| $p_{ij}$ | Probability of node $i$ to activate node $j$ in the IC model |
| $p_r$ | Probability of scenario $r$ |
| $R$ | Set of threshold or live-arc scenarios |
| $r$ | Cooling ratio in SAM |
| $r_{pm}$ | Performance ratio of method $m$ for problem $p$ |
| $S$ | Leader solution with set representation |
| $S_q$ | Set of follower solution at iteration $q$ of IXS |
| $S^+(\mathbf{y})$ | BFS for given follower solution $\mathbf{y}$ |
| $T$ | Temperature in SAM |
| $T_i$ | Triggering node set of node $i$ |
| $t_{\max}$ | Time limit of an algorithm |
| $t_{pm}$ | Time needed to solve problem $p$ with method $m$ |
| $t_q$ | Objective of the LB problem at iteration $q$ of IXS |
| $u_{ir}$ | Binary variable indicating if node $i$ is influenced in scenario $r$ |
| $V$ | Set of all nodes in the network |
| $V_r$ | Set of nodes with a predecessor in live-arc scenario $r$ |
| $w_{ij}$ | Weight of arc $(i, j)$ |

| | |
|---|---|
| $x_i$ | Binary variable indicating if node $i$ is activated (protected) by the leader in IMPD (MSMP) |
| $\mathbf{x}^q$ | Optimal solution of the problem $\text{LB}_q$ |
| $\mathcal{X}^\circ$ | Intersection of $\mathcal{X}$ and $\{0,1\}^n$ |
| $\mathbb{X}$ | Feasible region of ULP in IMPD and MSMP |
| $y_i$ | Binary variable indicating if node $i$ is deactivated (activated) by the follower in IMPD (MSMP) |
| $\bar{\mathbf{y}}$ | Vector of all follower variables |
| $\mathbf{y}^q$ | Optimal solution of the problem $\text{UB}_q$ |
| $\tilde{\mathbf{y}}^m$ | Optimal follower first-stage solution of the $m^{th}$ ALLP in SAA |
| $\tilde{\mathbf{y}}^*$ | Best follower decision found in SAA |
| $\bar{Z}$ | Upper bound on $Z^*(\mathcal{X},\mathcal{F})$ |
| $Z^*(\mathcal{X},\mathcal{F})$ | Optimal objective value of bilevel interdiction problem |
| $\hat{z}$ | Approximate optimal objective values of bilevel problem |
| $z_q$ | Objective of the UB problem at iteration $q$ of IXS |
| $z(\mathbf{x})$ | Objective value of LLP for leader solution $\mathbf{x}$ |
| $\hat{z}_N(\mathbf{x})$ | Objective value of an ALLP with $N$ scenarios for leader solution $\mathbf{x}$ |
| $\hat{z}_N^m(\mathbf{x})$ | Objective value of the $m^{th}$ ALLP solved in SAA |
| $\bar{z}_N^M(\mathbf{x})$ | Average objective value of $M$ ALLP's with $N$ scenarios |
| $\hat{z}_N(\mathbf{x},\mathbf{y})$ | Estimated objective value of $(\mathbf{x},\mathbf{y})$ with a sample of $N$ scenarios |
| $\hat{z}^*_{SAM}$ | Approximate objective value of the best solution in SAM |
| $\hat{z}^*_{TSM}$ | Approximate objective value of the best solution in TSM |
| $\hat{z}_{IMM}(S)$ | Estimated spread found by IMM when $S$ is protected |
| $\hat{z}_{SAA}(\mathbf{x})$ | Approximated LLP objective value via SAA for $\mathbf{x}$ |
| $\hat{z}_{SAA}(S)$ | Approximated LLP objective value via SAA for $S$ |
| $z^*(\mathbf{x},S_q)$ | Optimal objective value of $\text{LB}_q(\mathbf{x})$ in MSX algorithm |
| $\alpha_\tau$ | Binary variable indicating if follower solution $\mathbf{y}^\tau$ is blocked |
| $\beta$ | Threshold parameter of Binary Search algorithm |
| $\delta_r(i)$ | Predecessor of node $i$ in live-arc scenario $r$ |
| $\epsilon$ | Small positive number |

| | |
|---|---|
| $\eta$ | Cycle length update parameter in SAM |
| $\theta_i$ | Influence threshold of node $i$ |
| $\theta_{ir}$ | Influence threshold of node $i$ in threshold realization $r$ |
| $\lambda$ | Dual variable of $\mathrm{LB}_q(\mathbf{x})$ formulation |
| $\lambda_{ij}$ | Length of the shortest path from $i$ to $j$ |
| $\mu$ | Frequency penalty coefficient in TSM |
| $\pi_i$ | Proportion of visited solutions in TSM such that $i$ is in the seed |
| $\nu$ | Proportion of evaluated neighbors in TSM |
| $\xi$ | Set of all variables in HPR |
| $\overline{\xi_j}^{new}$ | Implied upper bound of variable $\xi_j$ in a subproblem |
| $\underline{\xi_j}^{new}$ | Implied lower bound of variable $\xi_j$ in a subproblem |
| $\overline{\xi_j}$ | Upper bound of variable $\xi_j$ |
| $\underline{\xi_j}$ | Lower bound of variable $\xi_j$ |
| $\sigma_{\mathcal{M}}(G, X)$ | Spread of seed set $X$ on graph $G$ under diffusion model $\mathcal{M}$ |
| $\phi$ | Temperature update parameter in SAM |

# LIST OF ACRONYMS/ABBREVIATIONS

| | |
|---|---|
| ABU | Alternating Bound Update |
| ALLP | Approximate Lower Level Problem |
| BSA | Binary Search Algorithm |
| BLP | Bilevel Linear Programming Problem |
| BFS | Bilevel-free Set |
| CE | Complete Enumeration |
| DEF | Deterministic Equivalent Formulation |
| DLT | Deterministic Linear Threshold |
| GPH | Greedy Protection Heuristic |
| HPR | High Point Relaxation |
| IC | Independent Cascade |
| IC | Intersection Cut |
| IMM | Influence Maximization with Martingales |
| IMP | Influence Maximization Problem |
| IMPD | Influence Maximization Problem with Deactivation |
| IXS | Improved $x$-space Algorithm |
| LB | Lower bound |
| LHS | Latin Hypercube Sampling |
| LLP | Lower-level problem |
| LP | Linear Programming Problem |
| LT | Linear Threshold |
| MIBLP | Mixed-integer Bilevel Linear Programming Problem |
| MILP | Mixed-integer Linear Programming Problem |
| MINLP | Mixed-integer Nonlinear Programming Problem |
| MSMP | Misinformation Spread Minimization Problem |
| OSN | Online Social Networking Site |
| RHS | Right-hand Side |
| RIMF | $r$-interdiction Median Problem with Fortification |

| | |
|---|---|
| SAA | Sample Average Approximation |
| SAM | Simulated Annealing based matheuristic |
| SBP | Stochastic Bilevel Programming Problem |
| SH | Simple Heuristic |
| SH-AC | Activation Cost Cased Simple Heuristic |
| SH-DC | Deactivation Cost Based Simple Heuristic |
| SH-OD | Out-degree Based Simple Heuristic |
| SH-SP | Individual Spread Based Simple Heuristic |
| TSM | Tabu Search Based Matheuristic |
| UB | Upper Bound |
| ULP | Upper-level Problem |
| XS | $x$-space Algorithm |

# 1. INTRODUCTION

A social network consists of a group of individuals or communities and the relationships between them. The relationships can be defined in various contexts such as economic, social, political, and analyzing the structure of these relationships helps to answer questions in different disciplines (Wasserman and Faust, 1994). Although this research area has been around for decades, it has attracted much more interest along with the growth in the number and prevalence of online social networking sites (OSN) such as Facebook and Twitter (Scott, 2012). One of the issues of primary interest in social network analysis is *social influence*. It is based on the fact that people's ideas can affect those of their friends, i.e., actors they are linked to. The existence of social influence helps transferring an idea or information through a network. It may even result in the spread of ideas like an epidemic (Chen *et al.*, 2013). Today, it is quite common to see that an interesting video published by an average OSN user with a few hundreds of friends is viewed by thousands of people through sharing with friends. Fake news fabricated by a few users sometimes reaches millions of people.

Viral marketing is one of the widespread applications of influence spread through a social network. Ideas or new technologies can quickly reach the masses when influential individuals are targeted initially. Many companies work with people called "influencers" to promote their products or services. On the other hand, determining the most influential people in a network is not always an easy question to answer. Influence Maximization Problem (IMP) involves finding the $k$ most influential people in a network. More precisely, it is the problem of selecting $k$ individuals to start an influence spread so that the expected number of people who are eventually influenced is maximized. It is a stochastic optimization problem due to the uncertainties in the underlying diffusion model. The initial spreaders are called the *seed set*, and finding the optimal seed set is $\mathcal{NP}$-hard under various well-known diffusion models (Kempe *et al.*, 2003). For the environments without competition, developing efficient approximation algorithms for the IMP is very valuable and there is a substantial amount of research

in this area. On the other hand, competition is a prevalent factor in the spread of influence in real social networks such as the spread of new technologies of two competing companies, the campaigns of political parties, the spread of rumors etc.

Even though competition may appear within a wide variety of problem settings, it is possible to group them in three main classes. In the first one, competing agents choose seed sets in order to maximize their own influence spread or to minimize the opponents' spread while multiple ideas spread over the network simultaneously. The problems in the second class involve preventing the spread of an undesired information by blocking individuals or relations after the seed set or the current set of adopters is realized. The last class deals with the case where two competitors who are aware of each other act sequentially. One of them tries to maximize the spread of influence as much as possible while the other one seeks to prevent it. The problems that we study in this dissertation belong to this last class, which can be seen as a Stackelberg game (Stackelberg *et al.*, 1952). In Stackelberg games, the first player (leader) acts first and makes a decision anticipating the reaction of the second player (follower) whose problem is assumed to be known by the leader. Such games are also known as leader-follower games and they can be modelled as bilevel programs, where the optimization problem of the follower is embedded into the leader's problem as a constraint (Bracken and McGill, 1973).

In the first problem we propose, the leader selects a seed set to maximize the spread whereas the follower deactivates some of the seed nodes to minimize the same measure. An example of such a diffusion process is the misinformation spread by an adversary organization like a terrorist group with the purpose of causing chaos in the society. In this context, solving the follower's problem answers the question of which individuals among a group of initial spreaders, i.e., the seed, must be prevented from influencing other people. The follower may represent the authority, who can restrain people for isolation purpose. Then, solving the problem of the leader corresponds to finding the strategy that maximizes the spread in the existence of a rational follower. In other words, the seed selected by the leader identifies the individuals that are most

likely to be targeted by the adversary. We refer to this problem as the Influence Maximization Problem with Deactivation (IMPD).

In the second problem, the leader seeks to minimize the spread knowing that the follower seeks to maximize it. To this end, the leader determines a given number of nodes to protect from being influenced, and then the follower activates a given number of the unprotected nodes to start the spread. The leader's problem can be seen as the minimization of the misinformation spread by preventing influential people from transmitting the information. As an example, we can consider the spread of false news on online social networks. If people are informed about the subject before they meet the false news, they will most probably not share but only ignore it. In this context, a protected node can be interpreted as an individual who has the information about the truth and does not share the false news or misinformation with friends. The problem of the leader is to decide the individuals to inform about the truth before a malicious and rational opponent starts a misinformation spread. This problem is called the Misinformation Spread Minimization Problem (MSMP).

We formulate the proposed problems as stochastic mixed-integer bilevel programs. Then, deterministic equivalents of these formulations are obtained. Several heuristic methods, which involve stochastic programming solution methods as subroutines, are developed for their solutions. Under a simplifying assumption about the level of uncertainty, two exact solution methods in the literature are adapted to the problems and improved using problem specific features.

The outline of the dissertation is as follows. A brief background information on the subjects that are frequently mentioned throughout the thesis is provided in Chapter 2. In Chapter 3, a detailed literature review about the optimization of influence spread in social networks and bilevel programming solution approaches is presented. In Chapter 4, the IMPD is introduced formally and two different bilevel programming formulations that can be used for its solution are developed. Several solution methods for the leader's and the follower's problem are proposed. Then, a state-of-the-art mixed-

integer bilevel programming solution technique is adapted to the problem and improved by developing an iterative problem reduction procedure in Chapter 5. In Chapter 6, the MSMP is introduced and formulated as a bilevel program. The proposed solution approaches are described in detail as well as the adaptation of some of the methods proposed to solve the IMPD to the MSMP. It is explained how to integrate a heuristic solution method for the solution of the follower's problem, without compromising the leader's solution quality. A generic algorithm for the solution of bilevel interdiction problems is proposed in Chapter 7 using the results of a related study. The implementation of this algorithm on the MSMP is possible after some modifications on the problem formulation, which is presented in the same chapter. Next, the results of an extensive computational study including all of the solution methods proposed for the leader's and follower's problems are presented in Chapter 8. Lastly, the contributions are summarized and possible future research topics are mentioned in Chapter 9.

# 2. BACKGROUND INFORMATION

## 2.1. Bilevel Programming

Bilevel programs (BP) are defined as "mathematical programs with optimization problems in the constraints" in Bracken and McGill (1973). Bilevel programming problems involve two decision makers who decide sequentially and these decision makers usually have conflicting objectives. The general structure of a basic BP is given as

$$\begin{aligned}
\min_{\mathbf{x}\in\mathcal{X}} \quad & F(\mathbf{x},\mathbf{y}) \\
\text{s.t.} \quad & G(\mathbf{x},\mathbf{y}) \leq 0 \\
& \text{where, for each } \mathbf{x} \text{ given by the first level, } \mathbf{y} \text{ solves} \\
& \min_{\mathbf{y}\in\mathcal{Y}} \quad f(\mathbf{x},\mathbf{y}) \\
& \text{s.t.} \quad g(\mathbf{x},\mathbf{y}) \leq 0,
\end{aligned} \tag{2.1}$$

where $\mathbf{x}$ and $\mathbf{y}$ represent the decision variables of the upper level problem (ULP) and lower level problem (LLP), respectively. Feasible solution sets $\mathcal{X}$ and $\mathcal{Y}$ may include additional restrictions such as integrality of some variables. For a given decision $\mathbf{x}$ of the ULP, the LLP is solved with the knowledge of $\mathbf{x}$ in order to optimize its objective function $f(\mathbf{x},\mathbf{y})$. Therefore, the ULP is solved taking the optimal response of the lower level into consideration. The set of optimal responses to an upper level decision forms the *rational reaction set* of the lower level. A solution $(\mathbf{x},\mathbf{y})$ is *bilevel feasible* if $\mathbf{y}$ is in the rational reaction set of the LLP for $\mathbf{x}$ and *bilevel infeasible* otherwise. If the rational reaction set is not a singleton, two approaches can be used to handle the follower's decision: optimistic and pessimistic (Dempe, 2018). In the former, it is assumed that among the optimal solutions the follower chooses the one that is best for the leader's problem whereas the worst one is chosen in the latter. Note that if the objective functions of the players are the same (with opposite optimization directions), then an assumption on the behavior of the follower is not needed since any follower

decision which optimizes the LLP yields the same objective value for the ULP along with the associated leader decision.

In a bilevel linear program (BLP) both the ULP and LLP are linear programs. In case of a linear LLP with continuous decision variables, a bilevel problem can be reformulated as a single-level problem using the Karush-Kuhn-Tucker optimality conditions (Colson *et al.*, 2007) even if the ULP has integrality restrictions. If the objective functions of the LLP and ULP are the same, it is also possible to write the dual formulation of the LLP to obtain a single-level formulation. If some of the decision variables are restricted to be integers, then it is called a mixed-integer bilevel linear program (MIBLP) for which it is not possible to obtain a single-level formulation in most cases.

The problems we propose in this dissertation are in the form of a Stackelberg game where the leader and the follower of the game decide sequentially to optimize their respective objectives and the leader anticipates the response of the follower. Due to the structural similarity between a Stackelberg game and a BP, our problems can be formulated as bilevel programs. The combinatorial nature of the problems (due to activation, deactivation and protection decisions which will be explained in relevant chapters) leads to MIBLPs with integrality restrictions in both levels of the formulations. We propose exact and heuristic solution methods for their solution.

## 2.2. Sample Average Approximation Method

Stochastic programs are optimization models that include uncertainty in their parameters. In a two-stage stochastic program, there exist two types of decision variables. The value of the *first-stage* variables must be determined while the uncertainty is still present. Then, the initially unknown data become available, and the values of the *second-stage* (*recourse*) variables are determined.

Sample Average Approximation (SAA) is a widely used method to solve two-stage stochastic programs with an objective function that is not easy to estimate for

a given first-stage solution (Mak *et al.*, 1999; Norkin *et al.*, 1998). It is based on repeatedly taking samples from the scenario space of the random events, i.e., possible realizations of the unknown parameters of the problem. The outputs of the method are stochastic lower and upper bounds on the optimal objective value, one of which is an approximation of the optimal objective value depending on the optimization direction. The main steps can be summarized as follows. First, the problem is solved several times with a commercial solver for different random samples of scenarios. For a minimization problem, the average of the optimal objective values obtained provides a stochastic lower bound, i.e., a lower bound estimate on the true optimal objective value. Each one of these problems yields a first-stage solution (some of them may be identical). The best candidate among them, which is expected to be a near-optimal solution, is determined with respect to some criterion. Then, an unbiased estimate of the objective value should be obtained for that first stage solution, which is usually possible by means of Monte Carlo simulations. This value provides an upper bound estimate on the optimal objective value, as it is associated with a feasible solution of a minimization problem. At the same time, it is an approximation of the optimal objective value since it is an estimate of the objective value for a near-optimal solution. The quality of the approximation depends on the choice of the sample size. As they become larger, better bounds are obtained at the expense of increased solution times. The optimality gap can be estimated as the difference between the lower and upper bound estimates computed in the first and the last stages, respectively.

The uncertainties in the diffusion models assumed in our problems lead to stochastic mathematical programs. The lower levels of the developed bilevel formulations turn out to be two-stage stochastic programs with excessive number of possible scenarios. Therefore, SAA is considered as an appropriate solution approach for the LLP's of our bilevel programs. The detailed procedure will be explained in Section 4.2.1 along with some properties of the bounds that the method yields and the implementation details such as the sampling methods used.

### 2.3. The Branch-and-Cut Algorithm with Intersection Cuts

A state-of-the-art approach for solving MIBLPs is due to Fischetti *et al.* (2016). They basically modify a standard mixed-integer linear program (MILP) solver for the purpose of solving MIBLPs using Intersection Cuts (Balas, 1971) to cut off bilevel infeasible solutions from the solution space. The B&C algorithm they propose solves the *high point relaxation* of the original problem, which is a single-level MILP obtained by relaxing the bilevel feasibility condition from the *value function formulation* of the original MIBLP. Consider the bilevel programming model

$$\begin{aligned}
\min_{\mathbf{x}} \quad & F(\mathbf{x}, \mathbf{y}) \\
\text{s.t.} \quad & G(\mathbf{x}, \mathbf{y}) \leq 0 \\
& x_i \text{ integer}, \ i \in J_x \\
& \text{where, for each } \mathbf{x} \text{ given by the first level, } \mathbf{y} \text{ solves} \\
& \min_{\mathbf{y}} \quad f(\mathbf{x}, \mathbf{y}) \\
& \text{s.t.} \quad g(\mathbf{x}, \mathbf{y}) \leq 0 \\
& \qquad y_i \text{ integer}, \ i \in J_y,
\end{aligned} \tag{2.2}$$

where $J_x$ and $J_y$ denote the set of upper-level (leader) and lower-level (follower) variables with integrality restrictions, respectively. For a given leader solution $\bar{\mathbf{x}}$, the value function is defined as

$$\Theta(\bar{\mathbf{x}}) = \min_{\mathbf{y}} \{ f(\bar{\mathbf{x}}, \mathbf{y}) : g(\bar{\mathbf{x}}, \mathbf{y}) \leq 0, \ y_i \text{ integer}, \ i \in J_y \}, \tag{2.3}$$

which leads to the value function formulation of (2.2) as follows:

$$\begin{aligned}
\min_{\mathbf{x}} \quad & F(\mathbf{x}, \mathbf{y}) \\
\text{s.t.} \quad & G(\mathbf{x}, \mathbf{y}) \leq 0 \\
& g(\mathbf{x}, \mathbf{y}) \leq 0
\end{aligned} \tag{2.4}$$

$$x_i \text{ integer}, \ i \in J_x$$
$$y_i \text{ integer}, \ i \in J_y$$
$$f(\mathbf{x}, \mathbf{y}) \leq \Theta(\mathbf{x}).$$

Notice that the value function constraint forces a feasible solution $(\mathbf{x}, \mathbf{y})$ of (2.4) to be bilevel feasible since any solution satisfying $f(\mathbf{x}, \mathbf{y}) \leq \Theta(\mathbf{x})$ implies that $\mathbf{y}$ is an optimal response of the follower to $\mathbf{x}$, i.e., $\mathbf{y}$ is in the rational reaction set of the follower. Therefore, solving (2.2) and (2.4) are equivalent. Removing the value function constraint implies relaxing the bilevel feasibility condition and leads to the high point relaxation (HPR) problem. The HPR is solved with an MILP solver, and every time an integer solution is found the bilevel feasibility of the solution is checked by computing $\Theta(\mathbf{x})$, i.e., solving the follower's problem. If the current solution is bilevel feasible, then it is treated as a feasible solution and the incumbent value is updated if necessary. Otherwise, an Intersection Cut (IC), which eliminates the current solution but none of the bilevel feasible solutions, is added to the current subproblem of the B&C tree. Note that it is necessary to define a *bilevel-free* set which contains the current bilevel infeasible solution in order to generate the ICs .

We implement the algorithm on the Influence Maximization Problem with Deactivation, assuming a finite number of diffusion scenarios. We improve it by adding a variable bound update scheme which reduces the size of subproblems and possibly prunes them. The details of the adaptation of the method to our problem and the improvement mechanism are explained in Chapter 5. Besides, we evaluate the effect of different bilevel-free sets to the performance of the algorithm and generate valid inequalities for the original formulation in the same chapter.

## 2.4. The $x$-space Algorithm

Tang *et al.* (2016) propose an exact algorithm for a special class of bilevel problems called *interdiction problems*. A bilevel interdiction problem is usually a zero sum game, where the aim of the leader is to damage the objective value of the follower by

interdicting some of its decisions. The $x$-space algorithm is an iterative algorithm for min-max bilevel interdiction problems with only binary variables in both levels. It is based upon their theorem which states that if the feasible region of the LLP (follower's problem), which is a non-convex set, is replaced by its convex hull, the optimal objective value of the leader remains the same. When the integrality restrictions disappear, the LLP becomes a linear program. Since the objective functions of the leader and the follower are the same (except the directions of optimization), the dual of the LLP can be obtained and added to the ULP as a constraint set by discarding the objective function. The resulting formulation is a single-level MILP and can be solved by using an MILP solver.

However, finding the convex hull can be very difficult. Therefore, it is inner-approximated iteratively using a set of feasible LLP solutions and expanding this set at each iteration of the $x$-space algorithm. Convex combinations of a set of feasible LLP solutions constitute a subset of the true convex hull and therefore a restricted feasible region of the LLP. The objective value of the restricted problem yields a lower bound on the true optimal objective value. Consequently, reducing the bilevel problem to an MILP as described above would result in underestimating the optimal leader objective value. This procedure constitutes the lower bound generation step of the algorithm. Solving the generated MILP yields a feasible leader solution and then solving the LLP for this leader solution provides an upper bound since it is associated with a feasible leader solution. The set of LLP solution is expanded by adding the newly found follower solution to it. In this way, a better approximation of the convex hull is obtained.

This algorithm solves upper bound and lower bound problems until convergence and requires the dualization of the follower's problem to develop the lower bound problem formulation. We enhance the algorithm in Chapter 7 by reformulating the lower bound problem using the properties of an optimal solution to the original formulation so that dualization is not needed any more. Following the reformulation, a greedy covering heuristic is integrated into the solution scheme to reduce the solution time. We test the modified algorithm on two well known problem families, i.e. Bilevel

Knapsack Problem and Bilevel Clique Problem , as well as the Misinformation Spread Minimization Problem, assuming a limited number of diffusion scenarios.

# 3. LITERATURE REVIEW

The problems addressed in this dissertation are competitive variants of the well-known IMP. They can be formulated as bilevel programming models since they are in the form of a Stackelberg game. Therefore, we first present a comprehensive review of non-competitive and competitive influence maximization (or minimization) problems in Section 3.1. Then, we present a brief survey on bilevel programming problems including exact and heuristic solution approaches.

## 3.1. Influence Maximization Problems in Social Networks

Social influence arises from the ability of individuals to affect the people they communicate in a real or online social network. There are various applications making use of social influence and the fact that it can spread through a social network, which include viral marketing and political campaign management. Before an attempt to use the phenomenon for some application, it is important to verify the existence of influence in the network since it can be confused with other factors such as homophily. There are studies that aim to distinguish influence from other factors that may cause correlation in the behaviors of connected entities (Anagnostopoulos *et al.*, 2008; Aral *et al.*, 2009; Sun and Tang, 2013). In addition to verification of the influence, the quantification of the influence (Tang *et al.*, 2009), the construction of the influence spread model (Goyal *et al.*, 2010), and spread maximization are the problems that attract attention.

The first study on the maximization of influence belongs to Domingos and Richardson (2001). They claim that considering customers as independent entities causes sub-optimal marketing decisions. They model the influences of customers on each other as a Markov random field where the probability of buying a product depends both on the customer's own preferences and the influence of the other customers. The authors extend their model in Richardson and Domingos (2002) for variable marketing actions and apply their methods on a real knowledge-sharing network. These studies

have triggered a cascade of influence maximization related works some of which involve competition over the spread of influence.

### 3.1.1. Influence Maximization in the Absence of a Competitor

Kempe *et al.* (2003) represent a social network with a graph and define the IMP first time as a discrete optimization problem as finding a set of $k$ nodes (called *seed set*) to start a diffusion so that the expected number of influenced nodes at the end of the diffusion is maximized. They offer two basic diffusion models: Linear Threshold (LT) and Independent Cascade (IC). These models and their special cases are used in most of the work in the IMP literature. In the LT model each node has a random influence threshold uniformly distributed in the interval $[0, 1]$, and node $i$ is influenced by its neighbor $j$ with some weight $w_{ji}$ if $j$ is an influenced node. Once the initial seed is selected (nodes in this set are active now), any node whose active neighbors' total weight exceeds its influence threshold, becomes active. At each step, nodes which satisfy the threshold condition are added to the active nodes set until it is not possible to activate any other node.

The diffusion process in the IC model again starts with the selection of seed nodes. Then, in the next step each active node $i$ tries to activate its each inactive neighbor $j$ and succeeds with some probability $p_{ij}$. An important feature of the IC model is that once node $i$ tries activating $j$ and fails, it cannot activate $j$ in the following steps. At each step, all active nodes try to activate their inactive neighbors which they did not try before to activate. The spread continues until no more activation can be done. For both diffusion models, Kempe *et al.* (2003) prove that the problem is $\mathcal{NP}$-hard and its objective function is a monotone submodular set function. Greedy algorithms provide approximation guarantees for problems with a monotone submodular objective function (Cornuejols *et al.*, 1977; Nemhauser *et al.*, 1978). Using this property, Kempe *et al.* (2003) propose a greedy algorithm with a $(1-1/e-\epsilon)$ approximation ratio, where $e$ is Euler's number and $\epsilon$ is an arbitrarily small positive number.

Leskovec *et al.* (2007) consider a sensor location problem in which a subset of nodes are selected for monitoring and an outbreak that spreads over the network can be detected when it reaches one of the monitored nodes. The objective is expressed as a reward function related to the quick detection of the spread, and it is shown to be submodular. The proposed algorithm, Cost Effective Lazy Forward (CELF), is a greedy method which makes use of submodularity to eliminate unnecessary computations. The problem is closely related to the selection of the most influential nodes in a network. They explain this relation by showing that one of the diffusion models in Kempe *et al.* (2003) is a special case of the outbreak detection problem. As a result, CELF can be considered as an improved version of the algorithm Greedy in Kempe *et al.* (2003) in terms of time complexity while maintaining the same approximation ratio. Kimura *et al.* (2007) propose a method for the efficient computation of marginal gains due to adding a node to the seed set, using bond percolation and graph theory. Since a bond percolation process declares the edges of a graph *occupied* or *unoccupied* like the *triggering model* in Kempe *et al.* (2003) labels them as *live* or *blocked*, they yield similar results. Namely, there exists a bond percolation (or triggering) model that is equivalent to the IC model in terms of the distribution of active (influenced) nodes. It is true for the LT model as well. They show that individual spreads (influence degrees) can be computed more efficiently by using these equivalent models instead of IC or LT within the original greedy algorithm. CELF and Greedy are further improved in Chen *et al.* (2009) for the IC model by using a similar approach to the one in Kimura *et al.* (2007) to compute the marginal gains. They generate subgraphs of the original one by determining the edges that will propagate the influence (analogous with live edges in the triggering model and occupied edges in the bond percolation model) and apply graph search algorithms such as depth-first search to compute the spread exactly for a specific diffusion scenario. They integrate their approach to the CELF algorithm. The authors also propose a degree discount heuristic method with a better time efficiency with some compromise from spread.

Chen *et al.* (2010a) show that the computation of the influence spread is $\#P$ hard for the IC model and develop a scalable heuristic method called *Maximum Influence*

*Arborescence* (MIA) and its improved version *Prefix Excluding MIA* (PMIA) with a tunable parameter to determine the balance between solution quality (value of spread) and running time. Their method is based on using local arborescence structures to estimate spread and it performs well for networks with millions of nodes. The IRIE algorithm in Jung *et al.* (2012) brings an efficient influence ranking (IR) method and an influence estimation (IE) method together, and improves PMIA in terms of memory usage and running time for the IC model. It is claimed that it performs better than the other state-of-the-art heuristics especially for dense networks.

As is the case with the IC model, the computation of the influence spread for a given seed set is $\#P$ hard for the LT model, as shown by Chen *et al.* (2010b). Nevertheless, if the diffusion model is LT and the underlying network is a directed acyclic graph, then exact influence spread computation can be done in a time proportional to the size of the network. Chen *et al.* (2010b) propose a scalable algorithm which intends to solve spread problem with LT diffusion model effectively and compare its performance to the Greedy algorithm and its improved versions in terms of speed and solution quality. Another study focusing on the scalability issue belongs to Wang *et al.* (2012). They propose a heuristic method which makes use of local influence regions of nodes to ease the computations for the IC model, and is scalable to networks with millions of nodes. It performs close to the existing greedy algorithms in terms of spread while it is faster by orders of magnitude. Thus, it has a better solution quality performance compared to the existing scalable heuristics. Again for the IC model, Borgs *et al.* (2014) come up with an algorithm that has the approximation ratio of the greedy methods with probability 3/5 and is runtime optimal. They offer a novel sampling procedure for building a stochastic hypergraph of the original graph which is the first stage of the two-stage algorithm proposed. In the second stage, the seed set is obtained in a greedy fashion on the hypergraph.

The *Two-phase Influence Maximization* (TIM) algorithm and its improved version TIM$^+$ in Tang *et al.* (2014) are based on the sampling idea in Borgs *et al.* (2014) which is referred to as "Reverse Influence Sampling". In its parameter estimation

phase, the number of samples required to achieve an approximation guarantee is estimated. Then, in the node selection phase samples are generated and the seed set is determined. The resulting algorithms provide a $(1 - 1/e - \epsilon)$ approximation with probability $1 - n^{-l}$, where $l$ is a parameter which affects the time complexity of the algorithm, and run in near-linear time for the Triggering Model in Kempe *et al.* (2003) which is a generalization of the IC model. Later, the empirical efficiency of TIM (and TIM$^+$) is improved in Tang *et al.* (2015) while maintaining the same theoretical time efficiency and approximation ratio using a statistical tool called *martingales* in the parameter estimation phase. The new algorithm *Influence Maximization with Martingales* (IMM) is applicable to several other diffusion models and claimed to perform better than the other state-of-the-art methods. Nguyen *et al.* (2016) propose the *Stop-and-Stare Algorithm* (SSA) algorithm and its dynamic version D-SSA which are based on a new sampling approach, and claim that they have the same theoretical efficiency with IMM and better practical efficiency. However, their analysis turns out to be incomplete according to Huang *et al.* (2017). Galhotra *et al.* (2016) introduce an opinion-aware diffusion model which takes the personal opinions of the nodes into account and propose the problem of Effective Opinion Maximization. They propose linear time and linear space algorithms EaSyIM and OSIM for the opinion-oblivious and the opinion-aware case, respectively. It is shown that it has better memory usage compared to algorithms TIM$^+$ and IMM as well as some other recent techniques.

An important aspect of the IMP is the uncertainty in the propagation process. Nevertheless, there are studies assuming deterministic diffusion models with the perspective that the uncertain parameters can be estimated efficiently with relevant techniques such as surveys, data mining tools etc. As the IC model's core is the existence of influence probabilities, the deterministic version of the LT model (DLT), where the node thresholds are known, is more meaningful in this context. In one such work, Chen (2009) considers a fixed spread ratio that should be reached (instead of a fixed number of nodes to activate) and addresses the problem of minimizing the size of initial seed (target set) that causes all or a fraction of the nodes in the network to be influenced at the end. It is shown that the optimal spread value is hard to approximate within

near-polynomial time for this diffusion model. A polynomial-time algorithm for optimal solution is provided for networks with a tree structure. Lu *et al.* (2011) show that there is no polynomial time $n^{1-\epsilon}$ approximation of the problem for the case where at most two active neighbors is needed to activate a node and that the computation of spread for a given seed set can be done in linear time under the DLT model. The inapproximability result is generalized to any case of the DLT model in Lu *et al.* (2012). Gursoy and Gunnec (2018) address the Targeted and Budgeted Influence Maximization Problem where nodes may have heterogeneous activation costs and profits, and the objective is to maximize the profit due to influencing nodes. They propose a greedy algorithm which is based on computing potential gains due to selecting a node as seed.

In another line of research, mathematical programming based methods are used for the solution of the IMP. Güney (2017) formulates the IMP as a stochastic program for the IC model and approximates the optimal spread by the SAA method. The formulation is further improved in Güney (2019) and a linear programming relaxation based method is proposed which achieves the $(1 - 1/e - \epsilon)$ ratio asymptotically with linear time complexity. Wu and Küçükyavuz (2018) propose a decomposition-based method called Delayed Constraint Generation and generate strong optimality cuts using the submodularity of the spread function. They investigate the performance of the algorithm under different diffusion models assuming a finite number of scenarios. Similarly, Güney *et al.* (2019) model the problem as a stochastic maximal covering location problem and solve it by a branch-and-cut method exploiting submodularity while approximating the problem with a finite number of diffusion scenarios. Li *et al.* (2019) address a generalization of the IMP called Cost-aware Target Viral Marketing and seek to find optimal solutions. Other than formulating and solving the problem via the SAA method, they offer an integer programming based method (TIPTOP) producing $(1 - \epsilon)$ optimal solutions for arbitrary $\epsilon$. TIPTOP is based on reducing the sample size efficiently so that the resulting integer programming formulation of the problem is small enough to be solved for networks with millions of nodes. Fischetti *et al.* (2018) address another generalization, the Generalized Least Cost Influence Problem, where the objective is to minimize the paid incentives to achieve a given influence

ratio. The diffusion model is very similar to the DLT except the thresholds can be reduced via partial incentives in addition to activating the seed nodes by paying a full incentive (such as an activation cost). The authors develop a set covering formulation, strengthen it with valid inequalities and propose exact and heuristic methods for its solution. Meanwhile, Günneç *et al.* (2019) address the same problem with another deterministic threshold model and show that it is $\mathcal{NP}$-hard. They develop a dynamic programming algorithm and a totally unimodular integer programming formulation of the problem for trees. Keskin and Güler (2018) formulate the IMP as an integer program for the DLT model by explicitly keeping track of the steps of the diffusion process, and propose a iterative heuristic.

For a review and detailed classification of the existing techniques in Influence Maximization as well as widely used diffusion models, we refer the reader to survey papers such as Li *et al.* (2018) and Sumith *et al.* (2018).

### 3.1.2. Competitive Influence Spread Problems

The aforementioned studies consider a single decision maker that chooses a seed set to start a diffusion process and assume that there is no competition. However, most of the applications of Influence Maximization including viral marketing, which is the seminal motivation, are likely to involve competition.

Several studies address problems involving competing information types that propagate on the same social network simultaneously. The decision makers select a seed set to either maximize their own influence spread or to minimize the competitor's spread. The former can be considered as influence maximization in the existence of competing or adversary influence spread(s). For example, In (Carnes *et al.*, 2007) the objective is the maximization of the spread given the seed set of the competitor. In other words, the decision is a best response to an existing opponent. They propose two diffusion models that are competitive variants of the IC model and show the submodularity of the spread function which results in an approximation guarantee for greedy

algorithms. Borodin *et al.* (2010) address another best response problem under the competitive LT models proposed in the same study. They show that approximating the optimal spread is $\mathcal{NP}$-hard for those models. The problem In (Shirazipourazad *et al.*, 2012) is finding a seed set with the minimum size such that the eventual spread is larger than the opponent's spread whose initially active nodes are known. In other words, the objective is to defeat the adversary. The problem is shown to be $\mathcal{NP}$-hard under the diffusion models proposed by Carnes *et al.* (2007). In the problem proposed by Venkatramanan and Kumar (2014), the influences of two content creators spread over two different social networks simultaneously. Each player seeks to optimize its budget allocation between the networks. The best response functions are obtained and used to compute a Nash equilibria. Keskin and Güler (2018) and Kahr *et al.* (2020) focus on developing integer programming formulations for the problem of finding the best response to an opponent's known seed set, under the competitive variants of the DLT and IC models, respectively. Due to the assumed diffusion model, the model in the latter turns out to be a stochastic programming formulation, which they solve via a Benders Decomposition based algorithm.

Chen *et al.* (2011) introduce a different problem structure where negative opinions may emerge during the spread of influence for a product. They propose the Independent Cascade model with Negative Opinions (IC-N). In this model, once a node is activated, it adopts the positive or the negative opinion with a probability associated with product quality. Therefore, negative and positive opinions propagate simultaneously. The spread function of the positive opinion, which is tried to be maximized, is shown to be submodular under this model. Since the proposed greedy algorithm is not time efficient, they propose a heuristic method that uses the local tree structures around nodes to approximate the spread efficiently. Jung *et al.* (2012) adapt the IRIE method which they propose for the IMP under the IC model to the competitive version with the IC-N model, and compare it to the methods in (Chen *et al.*, 2011).

Another subclass of the problems with multiple propagation processes includes models where the aim is to minimize the competitor's spread with a counter campaign

or idea propagation. Budak *et al.* (2011) introduce the Influence Limitation Problem where the spread of an undesired campaign is detected with some delay and the aim is to find a seed set with a given size for a new *limiting campaign* in order to minimize the spread of the first campaign. They propose two competitive IC diffusion models. In one of them, the influence probabilities depend of the campaign type while they do not in the other. For both models, the objective function is submodular. In addition to investigating the performance of a greedy algorithm, the authors propose another greedy approach for the case with incomplete data about the initial states of the nodes. Later, Wu and Pan (2017) develop two efficient heuristics for the same problem and same diffusion models, using the maximum influence arborescence structures to compute the spread faster.

Similarly, He *et al.* (2012) propose the Influence Blocking Maximization Problem. In this problem, the objective is to minimize the negative spread by starting a counter (positive) spread or equivalently to maximize the reduction in the number of nodes that adopts the negative idea. Here, the negative seed nodes are known and the decision maker needs to select a fixed number of positive seed nodes. Under the competitive LT model which is similar to the ones in Borodin *et al.* (2010), the problem is $\mathcal{NP}$-hard and its objective function is submodular. Due to the time inefficiency of a classical greedy algorithm, they make use of the spread computation approach of Chen *et al.* (2010b) to develop a heuristic. The Node Protectors problem considered by Nguyen *et al.* (2012) is different in terms of the objective function although the framework is similar. There, the objective is to minimize the size of the positive seed set while limiting the ratio of negatively influenced people to a given fraction. The authors show that the optimal objective value cannot be approximated in polynomial time. They propose greedy methods and a community based heuristic for the cases where the initial negative seed is known and unknown to the decision maker. Likewise, Fan *et al.* (2013) propose the Least Cost Rumor Blocking Problem where the objective is to minimize the number of positive spread initiators (protectors) such that the ratio of negatively influenced *bridge end* nodes is limited to a given fraction when the community that is the source of the rumor is known. Here, bridge ends represent the nodes that are considered critical in

spreading the rumor as they are connected to the source community. The objective is a submodular function under two IC based diffusion models considered. Kaur and He (2017) introduce the Blocking Negative Influential Node Set Problem and consider it from a host's perspective. Initially, there are negative seed sets in the network for each company involved and they are known to the host of the network. The problem of the host is to allocate positive seed nodes to each company such that the negative spreads are minimized for all of them. A greedy heuristic is proposed for its solution.

The studies mentioned in this section so far involve the best response type decisions in their problems. It is also possible that competing decision makers act simultaneously (Alon *et al.*, 2010; Bharathi *et al.*, 2007; Goyal *et al.*, 2019; Tzoumas *et al.*, 2012). In this class of problems, there exist multiple players who usually seek to maximize their own influence spread by selecting an initial set of active nodes in a non-cooperative fashion. Tzoumas *et al.* (2012) study a problem with two players and assume a diffusion model called "interaction scheme" which can be reduced to the LT model as a special case. They investigate the existence and computation of a pure strategy Nash equilibrium, show that an equilibrium may not exist for this game and compute bounds on the price of anarchy and the price of stability. Goyal *et al.* (2019) consider another two-player game where the players have budgets for activating nodes. They analyze the conditions for which there is a bound on the price of anarchy and the effect of budget differences (imbalances) on the equilibrium. In (Alon *et al.*, 2010) $n$ players compete and each one of them selects one seed node. Then, the diffusion starts and unfolds according to a deterministic model. They study the relation between the network diameter and existence of Nash equilibria where diameter is defined as the maximum distance between any pair of nodes in the underlying graph. One of the problems proposed by Clark and Poovendran (2011) involves simultaneous seed selection of two players under a Markovian influence spread model called *Dynamic Influence in Competitive Environments* (DICE). Different from the models in aforementioned studies, DICE allows switching between ideas. Under this model, the payoff functions of both players are submodular. Tsai *et al.* (2012) address a security game on a network of local leaders, and formulate a variant of the Influence Blocking Maximization

Problem. This is a zero-sum game where one player seeks to maximize its spread while the other one wants to minimize it. Instead of finding the best response, the players select their seed sets simultaneously and two competing ideas start spreading at the same time. They propose heuristic methods that perform especially well in sparsely connected graphs.

Competition in influence spread does not necessarily mean multiple diffusion processes, campaigns, opinions etc. Sometimes, it is tried to prevent or limit an undesired spread such as a rumor, fake news, a computer virus or a disease in a passive way. In such situations, some edges or nodes of the network may be blocked. Blocking a network entity usually implies the removal of that entity from the network since it cannot transmit information and contribute to the spread any more. As an example, in disease spread literature blocking corresponds to either the immunization of individuals by means of vaccination or prevention strategies such as putting in quarantine. The problem Cohen *et al.* (2003) consider is preventing an epidemic (due to a disease or a computer virus) via immunization of a minimum fraction of nodes. While random immunization requires immunizing almost all of the nodes, the heuristic approach they propose reduces this fraction significantly. They also analyze the strategy under the well-known Susceptible-Infected-Removed (SIR) disease spread model (Cooke, 1979). Holme (2004) also studies efficient vaccination strategies under the SIR model using only local information of nodes in the network. They propose an iterative vaccination model where at each step the neighbor of the previous vaccinated person with most arcs out of the neighborhood is chosen.

Kuhlman *et al.* (2010) address the problem of minimization of the spread of a contagion such as rumors by removing a small number of nodes from the network. They assume that the set of initially infected nodes is known and that the diffusion model is a complex contagion model which consists of the deterministic threshold functions for each node. They show that the problem is $\mathcal{NP}$-hard and propose heuristics. The problem in (Yu *et al.*, 2008) is another node blocking problem. It involves a dynamic network structure in the sense that the set of nodes and edges can change in time. A

variant of the IC model is assumed where an active node can try to activate its neighbors at all steps of a given finite horizon. The objective is to minimize the average influence degree of all nodes in the network obtained after blocking. The influence degree of a node is defined as the expected number of influenced nodes when it is the only seed node. In other words, the information about a seed set is not of concern. The authors analyze several network structural measures to determine the best spread blockers. In (Wang *et al.*, 2013), the set of infected nodes is given and the diffusion model is the IC. They propose a greedy method for the minimization of spread by blocking nodes and show that it performs better than centrality based heuristics. Yao *et al.* (2015) propose a topic-aware IC model where the influence probabilities depend on the topic and develop two heuristics based on node centrality to minimize the spread for a given seed set. Wang *et al.* (2017) introduce a blocking time (duration that the user account is blocked) constraint for the rumor influence minimization problem in online social networks, taking the user experiences into account. They consider threshold times after which user utility decreases, and make use of survival theory to investigate the likelihood that a user will be infected before the threshold time. They devise greedy algorithms for the solution. Tan *et al.* (2019) propose a node blocking strategy based on *activation increments* of nodes which is basically the difference in total activation probabilities of a node's neighbors when that node is infected and when it is recovered. The main idea of their method is blocking nodes with large activation increments. They assume a variant of the SIR model for the diffusion.

Instead of isolating or immunizing nodes to limit the influence spread, several studies consider blocking edges for the same purpose. Kimura *et al.* (2008a,b, 2009) and Kuhlman *et al.* (2013) aim to find a set of edges whose removal from the network minimizes a spread-based measure. This measure is the average influence degree of all nodes in the network for IC and LT diffusion models given by Kimura *et al.* (2008a) and Kimura *et al.* (2008b), respectively. They refer to this measure as the *contamination degree* of a graph. To minimize the contamination degree, they propose greedy algorithms which make use of the bond percolation method proposed by Kimura *et al.* (2007) to estimate influence degrees. The authors define a new measure called *worst*

*contamination degree* in (Kimura *et al.*, 2009) which is the maximum influence degree of all nodes. Clearly, its value does not depend on a fixed seed set as well. They compare the performance of a greedy method with degree-based methods. Kuhlman *et al.* (2013) consider deterministic threshold models (simple contagion where contacting with one infected neighbor is enough to be infected and complex contagion where more than one infected neighbor are needed). They provide two problem formulations. In the first one, the objective is to minimize the number of affected nodes within a blocking budget. In the second one, the objective is to minimize the blocking budget to protect all non-seed nodes from the spread. They provide complexity results on both problems and propose heuristic methods.

A different competitive problem setting involves sequential decisions of multiple players. The mechanism in best response type of problems may look like sequential decision making as a decision maker seeks to find an optimal response to a realized action/decision of some other agent. However, here we refer to the problems where we are interested in finding an optimal decision for each player involved. Bharathi *et al.* (2007) address a problem with multiple players trying to maximize their spread and give a competitive generalization of the IC model. Under this model, they first show that if the seed sets of all other players' are known, the objective function of the last player is monotone and submodular, and thus it can be approximated within a $(1 - 1/e - \epsilon)$ factor. Then, they briefly discuss a strategy of the first-mover (leader) in a two-player game which is in the form of a Stackelberg game (Stackelberg *et al.*, 1952). Kostka *et al.* (2008) analyze a two-player sequential game where both players aim to maximize their spread under a particular deterministic diffusion model by making use of concepts from location theory. They show that finding even an approximate solution for the leader is $\mathcal{NP}$-complete and that the leader does not always win. Clark and Poovendran (2011) address the same problem with a Markovian diffusion model and show that both players' problems have submodular objective functions under that model. Therefore, greedy approximation algorithms with provable guarantees can be developed. Unlike the previous studies, Hemmati *et al.* (2014) consider a Stackelberg game in which both players have the same objective function but with opposite objectives, i.e., a zero-sum

game. The leader seeks to minimize the spread by protecting some nodes whereas the second player (follower) wants to maximize it by targeting unprotected nodes for activation. The authors adopt a deterministic linear threshold diffusion process and assume finite number of time periods (diffusion steps) to model the problem as a discrete bilevel program. They propose several exact methods to solve the follower's problem and a cutting-plane algorithm for the solution of the leader's problem.

As can be understood from the review presented above, despite the abundance of the studies that focus on scalable algorithms for the IMP or on the best response influence maximization/minimization problems in competitive environments, sequential games have not been studied well yet. In particular, there is no study which considers the problem from the perspective of both players under a widely used stochastic diffusion model. Therefore, we propose two such problems in this thesis: Influence Maximization with Deactivation, and Misinformation Spread Minimization. As they can be modeled as mixed-integer bilevel programs, we present a brief summary of bilevel programming concentrating on this class and its particular subclass of interdiction problems in the next section.

## 3.2. Bilevel Programming

Several studies such as Jeroslow (1985), Ben-Ayed and Blair (1990) and Bard (1991) prove that solving a BLP which is the simplest version of bilevel programs, is $\mathcal{NP}$-hard. Besides, Vicente *et al.* (1994) prove that checking local optimality of a BLP solution is $\mathcal{NP}$-hard. Consequently, solving an MIBLP is $\mathcal{NP}$-hard as well. There exists a number of traditional approaches to solve BLPs. A prominent one of them is replacing the optimization problem of the follower with its Kuhn-Tucker conditions to obtain a single-level formulation and linearize the appearing bilinear terms (Fortuny-Amat and McCarl, 1981), with the drawback that it may lead to very large MILPs. This approach is also applicable for MIBLPs with integrality restrictions in the ULP only. Bard and Moore (1990) address a general bilevel program (either linear or quadratic), and adopts a similar approach to solve it. After replacing the LLP with its

optimality conditions, they propose a branch and bound algorithm which branches on Lagrangean multipliers to solve the resulting single-level formulation. Other approaches for BLPs include vertex enumeration, complementary pivot algorithms and penalty function methods (Lu *et al.*, 2016). Colson *et al.* (2007), Lu *et al.* (2016), and Dempe (2018) provide comprehensive reviews on existing approaches and algorithms as well as application examples.

### 3.2.1. Mixed Integer Bilevel Linear Programming

In this section, we present a brief summary of widely used general purpose MIBLP solution methods and their evolution in time. We do not include the techniques which are proposed for a specific problem or problem family. In one of the earliest studies, Moore and Bard (1990) consider a general MIBLP and show that relaxing the integrality constraints does not yield a valid bound on the optimal objective value of the problem. They develop an enumeration scheme which is proved to generate optimal solutions if all the leader variables are integers and the follower's problem is a linear program (LP). They also propose heuristic methods and test their performances on randomly generated instances with up to 40 variables. The problem addressed by Bard and Moore (1992) is pure binary and the leader has no constraints other than binary restrictions. The leader's objective function is integrated into the follower's problem by converting it to a parameterized constraint. Then, the resulting problem is solved via an implicit enumeration mechanism where the leader's objective value is improved at each iteration by changing the value of the parameter incrementally. Besides, the solutions generated at each iteration are bilevel feasible, i.e., the follower's part of the solution is in the rational reaction set of the follower. The test results on random problem instances with up to 50 variables show that the proposed algorithm performs better than the enumeration algorithm of Moore and Bard (1990) in terms of solution time.

DeNegre and Ralphs (2009) discuss the computational challenges of solving an MIBLP, i.e., why the existing methods for MILPs cannot be easily generalized to MI-

BLPs. Then, they propose a branch-and-cut method to deal with these challenges, which is actually an improvement of the branch-and-bound algorithm in Moore and Bard (1990) for the pure integer case. As opposed to Bard and Moore (1992), they allow constraints in the ULP albeit in only upper level variables. They generate valid inequalities to obtain improved bounds by making use of the bilevel feasibility conditions in addition to the standard feasibility conditions for MILPs. In addition, the resulting method does not require specialized branching strategies.

Another study which addresses general MIBLPs is due to Xu and Wang (2014). They assume a pure integer ULP and known bounds on the leader variables. The proposed algorithm uses a branch-and-bound scheme where the subproblems are *high point relaxations* of a parametric version of the initial MIBLP. A high point relaxation problem is obtained via removing the objective function of the follower and adding its constraints to the leader's problem. This idea was first used by Bialas and Karwan (1984). Bilevel infeasible subproblem solutions are discarded by a special branching rule. The algorithm is proved to converge correctly in a finite number of iterations. Infeasible and unbounded problem cases are also addressed, thus it is not needed to assume a bounded feasible region as is the case in most of the previous works. The authors provide computational results on more than 100 test instances with different sizes up to 920 variables. Caramia and Mari (2015) propose two methods, a cutting plane method, and a branch-and-cut algorithm for the pure integer case. While the main idea is similar to the one in DeNegre and Ralphs (2009), different types of valid inequalities are generated to cut off bilevel infeasible solutions. The proposed valid inequalities are shown to be more effective in terms of solution time on test instances with up to 25 variables.

Fischetti *et al.* (2016) propose another branch-and-cut algorithm for general MIBLPs with the assumption that the leader variables appearing in the follower's problem are all integer and bounded. They describe a procedure to transform a standard MILP solver into an MIBLP solver by making use of Intersection Cuts which are originally introduced by Balas (1971) for eliminating integer infeasible points in MILPs. The algo-

rithm is described briefly in Section 2.3. Fischetti *et al.* (2017b) enhance their method with a bilevel specific preprocessing procedure and additional separation algorithms to cut off bilevel infeasible solutions. The Watermelon Algorithm of Wang and Xu (2017) uses multiway disjunction cuts to remove bilevel infeasible solutions, motivated by the removal of watermelon seeds from a watermelon by a spoon. Similar to the method in their former study (Xu and Wang, 2014), multiple new nodes are generated at each iteration and the algorithm is proved to be finitely convergent. Differently, the method does not require known variable bounds. The algorithm is examined on the same test instances with up to 920 variables, which results in its superiority. Lozano and Smith (2017) introduce a sampling-based approach for general MIBLPs with a pure integer ULP and a mixed-integer LLP. The main idea is to sample solutions from the follower's solution space and solve a single-level relaxation problem at each iteration to obtain better bounds on the optimal objective value. While the relaxation problem yields upper bounds, each bilevel feasible solution obtained provides a lower bound. The algorithm terminates in a finite number of steps.

Exact approaches can be very time consuming for even medium-sized bilevel problems. Therefore, much research focuses on heuristic approaches to solve BLPs. Some examples for the linear case are the hybrid tabu search method of Gendreau *et al.* (1996), the genetic algorithm of Hejazi *et al.* (2002) and the particle swarm optimization method of Kuo and Huang (2009). Wen and Huang (1996) propose a tabu-search based method for an MIBLP where the leader controls the binary variables and the lower level variables are continuous. Nishizaki and Sakawa (2005) introduce a genetic algorithm where a penalty is incurred on the fitness of individuals if the bilevel feasibility is violated. Angelo and Barbosa (2015) design an integrated algorithm that involves two metaheuristic methods, ant colony optimization for the ULP and an evolutionary algorithm for the LLP, for a pure integer MIBLP. It is an important aspect of the heuristic approaches in bilevel programming that if the LLP is not solved optimally, the algorithm may yield bilevel infeasible solutions and they must be treated accordingly, which makes the efficient solution of LLP an important research issue.

The studies mentioned so far deal with deterministic bilevel problems. In case of uncertainty in any level of a BP, the resulting problem becomes a stochastic bilevel program (SBP). There exist studies that focus on reformulating a linear SBP as a single-level MILP. For example Fampa *et al.* (2008) consider a pricing problem in an electricity market, Dempe *et al.* (2017) introduce a general method for the reformulation under certain assumptions including discrete probability distribution, and Alizadeh *et al.* (2013) address a pricing problem on a transportation network. On the other hand, there are few works addressing a discrete stochastic bilevel problem. In the bilevel knapsack problem Özaltın *et al.* (2010) study, the LLP is a stochastic 0-1 knapsack problem where the knapsack capacity (the right-hand side) is a function of the leader's decision and a random variable. They reformulate the problem as a two-stage stochastic program and propose a branch-and-cut algorithm, assuming integer leader variables. Kosuch *et al.* (2012) formulate a linear SBP with a probabilistic knapsack constraint in the ULP and reformulate it first as an MIBLP by enumerating all scenarios, and then as a BLP. They develop an iterative algorithm based on Lagrangean relaxation. To the best of our knowledge, stochastic programming and mixed-integer bilevel programming have not been involved in the same optimization problem in any other study.

### 3.2.2. Interdiction Problems

A special class of bilevel optimization problems is known as Interdiction Problems where the conflict of interest between the decision makers is at the highest level possible. While players usually have conflicting objectives in a general BP, in an interdiction problem the objective of the leader is to cause the maximum deterioration in the objective value of the follower. In other words, the players have the same objective function with opposite optimization directions which makes the problem a *zero-sum game*.

A basic bilevel interdiction problem can be formulated as mathematical optimization model (3.1) where $\mathcal{X}$ and $\mathcal{Y}$ denote the ULP and LLP feasible regions, respectively (Wood, 2010). In most models considered, the leader restricts the decisions of the fol-

lower by interdicting some of the lower-level variables, i.e., modifying their bounds. As a general example, we may take the knapsack interdiction problem where the leader interdicts a subset of the items to minimize the profit of the follower due to solving a knapsack problem on non-interdicted items (Caprara *et al.*, 2016). Likewise, Furini *et al.* (2019) address the maximum-clique interdiction problem where a set of vertices is removed from the network to minimize the size of the resulting maximum clique. These problems are examples to full interdiction, i.e., an interdicted item cannot be used at all while it is also possible to disable items partially as is the case in Aksen *et al.* (2014). This is not the case in our problems. Due to the nature of the interdiction decisions, the ULP is usually an MILP in bilevel interdiction problems whereas the LLP can be an LP, an MILP, or any other problem type depending on the context.

$$\begin{aligned} &\min_{\mathbf{x}\in\mathcal{X}} z(\mathbf{x}) \\ &\text{s.t. } z(\mathbf{x}) = \max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f(\mathbf{x},\mathbf{y}). \end{aligned} \tag{3.1}$$

Most of the interdiction models studied are related to network interdiction problems in which there is an attacker who interdicts (damages) some network components so that the defender cannot use them to optimize its objective. In one of the earliest studies on the topic, Wood (1993) deals with a network interdiction problem where the arcs of the network consisting of roads and rivers are interdicted to minimize the flow of drugs. Since the LLP is a max-flow problem, its dual is used to obtain a single-level formulation. Israeli and Wood (2002) address the Shortest Path Interdiction Problem where the leader interdicts arcs to maximize the length of the shortest $s$-$t$ path. They use the same duality based approach for modeling the problem as an MILP, and develop decomposition algorithms for its efficient solution. Shen *et al.* (2012) introduce node deletions to reduce the connectivity of a network. They consider measures such as number of connected components and size of the largest connected component. Although the LLPs are mixed-integer, they show that an optimal solution remains the same when the integrality constraints are relaxed, which allows a single-level reformulation. We refer the reader to (Wood, 2010) and (Smith and Song, 2020) for a detailed

survey on network interdiction models.

Although few in number, some research focuses on developing solution approaches for general bilevel interdiction models with a discrete LLP. Tang *et al.* (2016) address min-max bilevel problems with mixed-integer LLP and propose three iterative algorithms that converge in finite number of steps when the leader's variables are binary. The algorithms are based on solving lower bound and upper bound problem formulations they develop iteratively until convergence. The detailed description is provided in Section 2.4. The problem considered in Lozano and Smith (2016) is a three-level problem with fortification, interdiction, and recourse decisions, respectively. Fortification means protection of some items in advance so that an attacker cannot interdict them, and the resulting problem has a min-max-min objective. They propose a sampling based iterative method and allow the recourse problem to be in any form while the first two levels include binary variables only. The most recent technique belongs to Fischetti *et al.* (2019), to the best of our knowledge. They develop a branch-and-cut algorithm for the interdiction problems whose LLP satisfies the so-called *down-monotonicity* assumption which implies the following. Consider two interdiction decisions such that the interdicted elements in the second one is a subset of the interdicted elements in the first one. If a lower-level solution is feasible for the first decision, then it is also feasible for the second one. Although it is satisfied in several well known problem families such as the knapsack interdiction problems, it is violated by the problems examined in this dissertation.

The class of competitive spread problems where the nodes or the arcs of a network are blocked/removed to limit the influence spread has clear similarities with interdiction problems. Some network items are made unusable (cannot spread ideas) with the aim of damaging the opponent's objective, i.e., the spread. Nevertheless, most of such problems do not have the hierarchical structure mentioned in this section. Therefore, our problem can be considered as a stochastic bilevel interdiction problem, without satisfying down-monotonicity.

# 4. INFLUENCE MAXIMIZATION PROBLEM WITH DEACTIVATION

In this chapter, we study a competitive extension of the IMP. After its definition, some application examples and information about the general assumptions, we present two alternative bilevel mathematical models in order to handle the inherent uncertainty in two different ways. Then, we propose explicit solution methods for the problems of both players involved.[1]

## 4.1. Problem Definition and Mathematical Models

IMP is defined as finding an initial set of $k$ nodes to start a diffusion process in a social network so that the expected total number of affected nodes at the end of the process is maximized. The problem introduced in this section is a competitive version of the IMP, which can be regarded as a Stackelberg game between two players. Given a social network, the first player (leader) determines a subset of nodes (individuals) to *activate* in order to propagate a belief, idea, or campaign. The activated nodes, also called the seed set, are not only influenced directly as a result of the campaign, but they also have the capability of influencing other individuals. Given that the leader decides on the seed set, the second player (follower) having perfect information on the seed set *deactivates* some of these nodes so that they cannot influence the remaining ones. This has the consequence of keeping the number of influenced nodes at a low level. Deactivation can be achieved in different ways depending on the problem context. If we regard the follower as a competing company, it may convince a seed node not to spread the idea by giving promotions or offering financial incentives. Alternatively, if the follower represents the security forces, it can remove the links of a seed node by detaining that node. After the follower's decision, the nodes that are activated by the leader and not deactivated by the follower, influence other nodes in the network according to a certain diffusion model. The goal of the leader is to maximize the

[1] (Tanınmış *et al.*, 2019) and (Tanınmış *et al.*, 2020a) are based on the content of this chapter.

expected number of influenced nodes whereas the follower tries to minimize it. We refer to the problem as the Influence Maximization Problem with Deactivation (IMPD).

Let the social network be represented by a directed graph $G = (V, A)$, where $V$ and $A$ denote the set of nodes and arcs, respectively. We define $G_Y$ as the subgraph of $G$ that is obtained by removing the nodes in $Y$ along with the arcs incident to them, and $\sigma_{\mathcal{M}}(G, X)$ as the expected spread for an initially active node set, i.e., seed set, $X$, on graph $G$ under some diffusion model $\mathcal{M}$. Besides, let $c(\cdot)$ and $e(\cdot)$ be the activation and deactivation cost functions, while $C$ and $E$ are the associated budgets. Then, the IMPD can be defined for any diffusion model as follows:

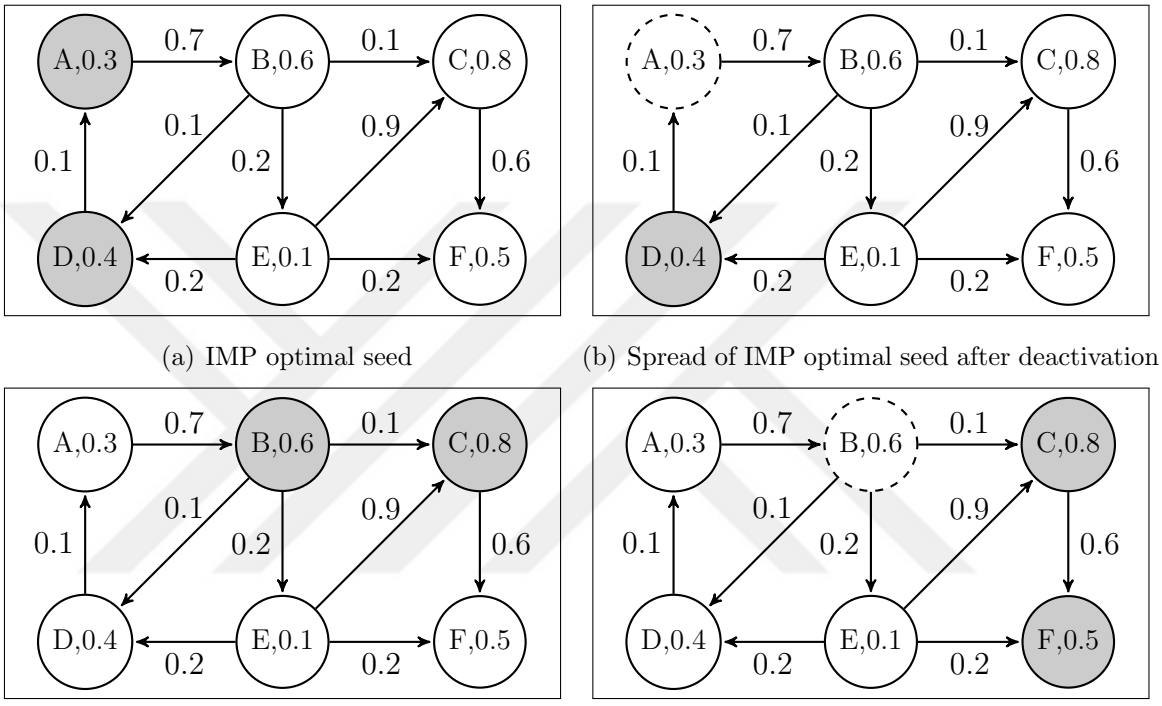
$$\max_{\substack{X \subset V, \\ c(X) \leq C}} \sigma_{\mathcal{M}}(G_{Y^*(X)}, X \setminus Y^*(X)), \tag{4.1}$$

where

$$Y^*(X) = \arg\min_{\substack{Y \subset X, \\ e(Y) \leq E}} \sigma_{\mathcal{M}}(G_Y, X \setminus Y). \tag{4.2}$$

In order to model this problem explicitly, we assume that the influence spreads according to the well known Linear Threshold (LT) diffusion model, where a node becomes influenced only if the total weight on the incoming arcs from its influenced neighbors exceeds a random threshold value. An illustration of the problem is presented in Figure 4.1 for two different leader decisions under the assumption that node thresholds are deterministic. Threshold values are shown next to the node labels, and arc weights are indicated on the arcs. Activation and deactivation costs are assumed to be unity, and the budgets of the leader and the follower are taken as two and one, respectively. The optimal seed nodes that are selected in IMP and IMPD are displayed as shaded circles in Figure 4.1(a) and Figure 4.1(c), respectively. Recall that in the IMP the optimal seed is determined by not anticipating the reaction of an opponent. Also note that the resulting number of influenced nodes would be six and four in Figure

4.1(a) and Figure 4.1(c), respectively if there were no deactivation. Subsequently, the follower deactivates one of the seed nodes, which are displayed as dashed circles. The influenced nodes at the end of diffusion process are shaded in Figure 4.1(b) and Figure 4.1(d). As can be observed, the IMP optimal seed cannot preserve its superiority under the existence of a rational opponent.




(a) IMP optimal seed

(b) Spread of IMP optimal seed after deactivation

(c) IMPD optimal seed

(d) Spread of IMPD optimal seed after deactivation

Figure 4.1. Comparison of two IMPD ULP solutions.

Stackelberg games can be formulated as bilevel programs where the problem of the follower is a constraint set in the leader's problem. On the other hand, the uncertainty of node thresholds in the LT diffusion model turns our problem into a stochastic optimization problem. Therefore, a discrete stochastic bilevel programming model for the IMPD is proposed in (4.3)–(4.6). The sets, parameters, and decision variables used in the formulation are provided below. Note that the model takes into account the possibility that not all individuals (nodes) are alike in terms of activation and deactivation. Therefore, it is assumed that there is a budget for both the leader and the follower, and there is a cost for activation and deactivation associated with each individual. This cost may be the same for each individual giving rise to a cardinality-based IMPD, or it may change with respect to individuals resulting in cost-based

IMPD.

*Sets and Parameters:*

$V$ : set of all nodes in the network where $|V| = n$

$c_i$ : leader's cost of activating node $i \in V$

$e_i$ : follower's cost of deactivating node $i \in V$

$\theta_i$ : influence threshold of node $i \in V$

$C$ : leader's budget to activate nodes

$E$ : follower's budget to deactivate nodes

*Decision variables:*

$x_i$ : 1 if node $i$ is activated by the leader; 0 otherwise

$y_i$ : 1 if node $i$ is deactivated by the follower; 0 otherwise

Using this notation, the stochastic bilevel programming model for the IMPD can be defined as follows:

IMPD-S:

$$\max_{\mathbf{x}} \ \mathbb{E}_{\boldsymbol{\theta}}[g(\mathbf{x}, \mathbf{y}^\star, \boldsymbol{\theta})] \tag{4.3}$$

s.t.

$$\mathbf{c}^T\mathbf{x} \leq C \tag{4.4}$$

$$\mathbf{x} \in \{0,1\}^n \tag{4.5}$$

$$\mathbf{y}^\star \in \arg\min_{\mathbf{y}} \left\{ \mathbb{E}_{\boldsymbol{\theta}}[g(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})] : \mathbf{e}^T\mathbf{y} \leq E, \mathbf{y} \leq \mathbf{x}, \mathbf{y} \in \{0,1\}^n \right\} \tag{4.6}$$

where $g(\mathbf{x}, \mathbf{y}^\star, \boldsymbol{\theta})$ represents the number of influenced nodes for a given threshold realization (scenario) vector $\boldsymbol{\theta}$ corresponding to a given seed selection strategy $\mathbf{x}$ of the leader and optimal seed deactivation strategy $\mathbf{y}^\star$ of the follower. For a given realization of $\boldsymbol{\theta}$, $g(\mathbf{x}, \mathbf{y}^\star, \boldsymbol{\theta})$ can be computed in an algorithmic way in polynomial time using the LT diffusion model. $\mathbb{E}_{\boldsymbol{\theta}}[g(\mathbf{x}, \mathbf{y}^\star, \boldsymbol{\theta})]$ denotes the spread, i.e., the expected number of influenced nodes, where the expectation is taken with respect to the probabilistic threshold vector $\boldsymbol{\theta}$. Note that the optimization problem of the follower given by (4.6)

becomes a constraint in the leader's problem. In other words, the leader tries to give the best decision $\mathbf{x}$ with the anticipation that the follower gives the best response $\mathbf{y}^\star$.

The objective function (4.3) of the leader is to maximize the spread. Constraint (4.4) is the budget constraint of the leader. Constraints (4.5) put binary restriction on variables $\mathbf{x}$. Constraints (4.6) state that $\mathbf{y}^\star$ must be in the rational reaction set (optimal response) of the follower. The first constraint of this set ($\mathbf{e}^T\mathbf{y} \leq E$) limits the deactivation cost by the available budget. The second one, $\mathbf{y} \leq \mathbf{x}$, ensures that a node cannot be deactivated unless it is activated by the leader. The last one is the binary restriction on variables $\mathbf{y}$. The rational reaction set includes $\mathbf{y}$'s that satisfy the aforementioned three constraints and minimize the spread for a given $\mathbf{x}$.

There is no closed-form expression to calculate $\mathbb{E}_{\boldsymbol{\theta}}[g(\mathbf{x}, \mathbf{y}^\star, \boldsymbol{\theta})]$. However, it is possible to develop a deterministic formulation equivalent to (4.3)–(4.6) that is obtained by enumerating each possible realization of the random events. In Section 4.1.1, we present a deterministic equivalent formulation (DEF) based on enumerating all possible threshold realizations under the assumption that the number of threshold realizations is finite. Then, another DEF is developed in Section 4.1.2 by making use of the equivalence of the LT and the *Trigerring* models (Kempe *et al.*, 2015).

### 4.1.1. Formulation with Threshold Scenarios

In the LT model, the state of a node at any step of the diffusion process is determined based on two parameters as described in Kempe *et al.* (2003). Let $G = (V, A)$ denote a directed graph where $V$ and $A$ denote the set of nodes and arcs, respectively. The first parameter is an arc weight which is denoted by $w_{ij}$, $(i, j) \in A$, and it represents a deterministic measure of influence that node $i$ has on node $j$. The arc weights should satisfy the condition $w_{ij} \in [0, 1], (i, j) \in A$ and $\sum_i w_{ij} \leq 1, j \in V$. The second parameter is a node threshold denoted by $\theta_i, i \in V$. This parameter is random and distributed uniformly in the interval $[0, 1]$ independent of the other nodes. At any time of the diffusion process node $i$ becomes influenced (activated) if the total

weight on the arcs incoming from its active neighbors exceeds its threshold $\theta_i$. Note that Kempe *et al.* (2003) consider (undirected) graphs, therefore the relationships are bidirectional.

In addition to the parameters $c_i$, $e_i$, $C$, and $E$, and decision variables $x_i$ and $y_i$ used in the definition of IMPD-S, the additional notation needed for the so-called deterministic equivalent formulation of IMPD-S with threshold scenarios is given below.

*Sets and Parameters:*

| | | |
|---|---|---|
| $R$ | : | set of all possible threshold scenarios |
| $\theta_{ir}$ | : | threshold of node $i \in V$ in threshold scenario $r \in R$ |
| $w_{ij}$ | : | weight of arc $(i, j)$, 0 if the arc does not exist |
| $p_r$ | : | probability of threshold scenario $r \in R$ |

*Decision variable:*

| | | |
|---|---|---|
| $u_{ir}$ | : | 1 if node $i \in V$ is influenced in threshold scenario $r \in R$; 0 otherwise |

As a result, IMPD-t can be written as follows:

IMPD-t:

$$z^* = \max_{\mathbf{x}} z(\mathbf{x}) \tag{4.7}$$

s.t.

$$\sum_{i \in V} c_i x_i \leq C \tag{4.8}$$

$$x_i \in \{0, 1\} \qquad i \in V \tag{4.9}$$

where

$$z(\mathbf{x}) = \min_{\mathbf{y},\mathbf{u}} \sum_{r \in R} \sum_{i \in V} p_r u_{ir} \tag{4.10}$$

s.t.

$$\sum_{i \in V} e_i y_i \leq E \tag{4.11}$$

$$y_i \leq x_i \qquad i \in V \tag{4.12}$$

$$u_{ir} \geq x_i - y_i \qquad i \in V, r \in R \tag{4.13}$$

$$u_{ir} + y_i \geq \sum_{j \in V} w_{ji} u_{jr} - \theta_{ir} + \epsilon \qquad i \in V, r \in R \tag{4.14}$$

$$u_{ir}, y_i \in \{0, 1\} \qquad i \in V, r \in R. \tag{4.15}$$

Here, (4.7)–(4.9) represent the leader's ULP, and (4.10)–(4.15) is the follower's LLP. The objective function $z(\mathbf{x})$ denotes the minimum average number of influenced nodes for a given leader decision $\mathbf{x}$ and optimal follower decision $\mathbf{y}$ corresponding to $\mathbf{x}$. Note that it is an approximation to the value of the spread. Inequality (4.8) is the budget constraint of the leader. Decision variables $x_i$ take binary values by Constraint (4.9). Constraint (4.11) in the LLP is the budget constraint of the follower, and Constraint (4.12) implies that a node can only be deactivated by the follower if it is activated by the leader. Constraint (4.13) forces the seed nodes that are not deactivated ($x_i = 1, y_i = 0$) to be influenced by default, i.e., $u_{ir} = 1$. In other cases, i.e., ($x_i = 0, y_i = 0$) and ($x_i = 1, y_i = 1$), the value of $u_{ir}$ is determined by Constraint (4.14) and the objective function. This is achieved as follows: the summation on the right-hand side (RHS) of Constraint (4.14) is the total weight on the incoming links to node $i$ from its influenced predecessors. This total weight cannot exceed one by the definition of link weights in the LT diffusion model. If the total weight is equal to or exceeds the threshold $\theta_{ir}$ of node $i$ in scenario $r$, the RHS becomes positive between zero and one, which forces the left-hand side to be at least one. In that case, if $i$ is a deactivated node ($x_i = 1, y_i = 1$), $u_{ir}$ is set to zero due to the minimization objective (satisfying the assumption that a deactivated node cannot be influenced). Otherwise, $u_{ir}$ is set to one, i.e., node $i$ which is not a seed node, is influenced in scenario $r$. If the RHS is non-positive, then a node is not influenced due to the minimization of the objective function. $\epsilon$ is a small positive number and should guarantee that the RHS is positive when the total weight is equal to or greater than the threshold. Notice that for a non-deactivated node $i$ (i.e., $y_i = 0$), $u_{ir} = 1$ if $\sum_{j \in V} w_{ji} u_{jr} - \theta_{ir} > 0$. However, $u_{ir}$ must also be set to one if $\sum_{j \in V} w_{ji} u_{jr} - \theta_{ir} = 0$. This can be ensured by choosing a small positive value for $\epsilon$ which does not give rise to $u_{ir} = 1$ if $\sum_{j \in V} w_{ji} u_{jr} - \theta_{ir} < 0$ with a small absolute value. More specifically, $\epsilon$ has to be set to such a small positive

value so as to guarantee that $\epsilon < \min_{i,r}\{|\sum_{j\in V} w_{ji}u_{jr} - \theta_{ir}| : \sum_{j\in V} w_{ji}u_{jr} - \theta_{ir} < 0\}$. Since we use a precision level of four significant digits after the decimal point in our computations, the value of $\epsilon$ is set to $10^{-5}$. Lastly, $u_{ir}$ and $y_i$ are restricted to take binary values by Constraint (4.15). Note that if the threshold parameter values $\theta_i$ were deterministic, then the scenario index $r$ would disappear from the model (4.10)–(4.15), and IMPD would be a much simpler problem to solve.

### 4.1.2. Formulation with Live-arc Scenarios

Kempe *et al.* (2015) define the Triggering model with the purpose of analyzing some properties of the IMP. In this model, a triggering set $T_i$ denotes a random subset of the neighbors of node $i$ which is obtained using some probability distribution. At any step of the diffusion process, an inactive node $i$ becomes activated if at least one of the nodes in $T_i$ is active. An arc $(j, i) \in A$ is called *live* if $j \in T_i$, and *blocked* otherwise. It follows that, given a seed set, i.e., a set of initially active nodes, the expected number of active nodes at the end of a diffusion process according to the Triggering model is equal to the expected number of nodes that are reachable from the seed nodes using the paths consisting of live arcs only. It is also shown that there exist a probability distribution for $T_i, i \in V$ such that the distribution of active nodes in the Triggering model is equivalent to the one in the LT model for a given seed set. In this probability distribution, at most one incoming neighbor of node $i$ can appear in $T_i$ and the probability of having $j$ in $T_i$ is $w_{ji}$. As a result, the probability that node $i$ has an empty triggering set $T_i$ is $1 - \sum_j w_{ji}$. In other words, the arc $(j, i)$ is selected as live with probability $w_{ji}$. Here, the property of the LT model stating that $\sum_i w_{ij} \leq 1, j \in V$ allows the interpretation of the arc weights as the probability of being live in the Triggering model.

The equivalence between the diffusion models under the $T_i$ distribution described above is reached by showing the equality of the following two probabilities in Kempe *et al.* (2015). The first one is the probability that node $i$ becomes active in step $t + 1$ of the diffusion process under the LT model, given the set of active nodes in steps $t - 1$

and $t$, and that $i$ is not activated yet. The second is the probability that node $i$ is determined as reachable from a given seed set via the live arcs at step $t+1$ if it is not determined as reachable in the previous steps. Both probabilities are computed as

$$\frac{\sum_{j \in A_t \setminus A_{t-1}} w_{ji}}{1 - \sum_{j \in A_t} w_{ji}} \tag{4.16}$$

where $A_t$ denotes the set of active nodes at step $t$ of the diffusion for the LT model, and the set of nodes that are determined as reachable from the seed via live arc paths in step $t$ for the Triggering model. Using the equivalence of the LT and the Triggering models under the conditions described above, random events of the diffusion process can be represented as live-arc scenarios which are basically subgraphs of the original directed graph $G$, instead of the threshold scenarios in the LT model.

In the DEF of the IMPD-S with live-arc scenarios, it is possible to represent a live-arc scenario with a vector of size $n = |V|$, which holds the indices of the predecessor nodes for all $i \in V$ in the relevant scenario (subgraph). Recall that there can be at most one incoming arc to a node in a scenario. In other words, each node can have at most one predecessor. Therefore, such a one dimensional vector is sufficient to represent a scenario. In addition to defining new sets and parameters, it is necessary to slightly change the definition of some of the previously defined parameters and variables for developing the new formulation. The list of additional (and modified) sets, parameters and decision variables are presented below. It is followed by the new deterministic equivalent formulation, IMPD-La.

*Sets and Parameters:*

$R$ : set of all possible live-arc scenarios

$V_r$ : set of nodes with a predecessor in scenario $r \in R$

$\delta_r(i)$ : predecessor of node $i \in V$ in scenario $r \in R$.

$p_r$ : probability of live-arc scenario $r \in R$

*Decision variables:*

$u_{ir}$ : 1 if node $i \in V$ is influenced in live-arc scenario $r \in R$; 0 otherwise

IMPD-La:

$$z^* = \max_{\mathbf{x}} z(\mathbf{x}) \tag{4.17}$$

s.t.

$$\sum_{i \in V} c_i x_i \leq C \tag{4.18}$$

$$x_i \in \{0, 1\} \qquad i \in V \tag{4.19}$$

where

$$z(\mathbf{x}) = \min_{\mathbf{y},\mathbf{u}} \sum_{r \in R} \sum_{i \in V} p_r u_{ir} \tag{4.20}$$

s.t.

$$\sum_{i \in V} e_i y_i \leq E \tag{4.21}$$

$$y_i \leq x_i \qquad i \in V \tag{4.22}$$

$$u_{ir} \geq x_i - y_i \qquad i \in V, r \in R \tag{4.23}$$

$$y_i + u_{ir} \geq u_{\delta_r(i),r} \qquad i \in V_r, r \in R \tag{4.24}$$

$$u_{ir} \geq 0, y_i \in \{0, 1\} \qquad i \in V, r \in R. \tag{4.25}$$

Formulation IMPD-La is different from IMPD-t in two constraints. Firstly, Constraint (4.14) where the influence state of a node is checked using threshold values is replaced with (4.24) which ensures that node $i$ gets influenced in scenario $r$ if it is not a deactivated node and its predecessor in scenario $r$ is influenced. If $i$ is a deactivated node ($y_i = 1$), then $u_{ir}$ is set to zero in an optimal follower solution due to the objective function regardless of the state of $i$'s predecessor in scenario $r$. Since the seed nodes which are not deactivated are forced to be influenced by Constraint (4.23), these nodes also enable their successors to be influenced in Constraint (4.24). This process eventually leads to the outcome that all the nodes that can be influenced ultimately achieve the state of being influenced. Secondly, the integrality restrictions on some of the variables are changed in Constraint (4.25). Although all variables should take

binary values as before, a binary restriction on variable $u_{ir}$ is not needed any more. Since $x_i$ and $y_i$ take binary values by (4.19) and Constraint (4.25), $u_{ir}$ is set to one if $i$ is influenced in scenario $r$ and zero otherwise, due to the optimization direction and Constraint (4.23) and Constraint (4.24).

## 4.2. Solution Methods

IMPD is a Stackelberg game between two decision makers and can be formulated by means of a bilevel program, as can be seen above. On the other hand, it can be reduced to a problem involving one decision maker, namely IMP, by setting the deactivation budget to zero and all activation costs to unity. Once the deactivation decisions do not exist, the problem becomes one of finding a seed set of fixed size so that the resulting spread is maximized. Since this reduced problem was proven to be $\mathcal{NP}$-hard by Kempe *et al.* (2003), IMPD is clearly $\mathcal{NP}$-hard as well.

A brief survey on the techniques for solving bilevel programming problems is presented in Section 3.2. As mentioned there, there exist several basic approaches to deal with BLP, MIBLP and SBPs. For example, a BLP can be reduced to a single-level MILP or MINLP using the Karush-Kuhn-Tucker optimality conditions or making use of the duality theory (Colson *et al.*, 2005). Such approaches are also applicable to the MIBLPs with a convex LLP. Likewise, an SBP with a continuous LLP can be reformulated as a two-stage stochastic program as suggested in several studies. In both of the proposed formulations the ULP and LLP include binary decision variables, which makes impossible the use of the aforementioned approaches. Another approach to solve discrete bilevel optimization problems is to develop a heuristic method for the leader's problem, which requires to solve the follower's problem (4.10)–(4.15) or (4.20)–(4.25) to optimality for each candidate solution of the leader. We propose two matheuristics that can be used commonly for the formulations IMPD-t and IMPD-La, based on an approximation of the follower's problem described in the sequel.

### 4.2.1. Sample Average Approximation Method for the Follower's Problem

4.2.1.1. The Threshold Model. In the LT diffusion model, each of the threshold parameter values, $\theta_i$, follows a continuous uniform distribution. Therefore, it is not possible to enumerate all possible realizations of the threshold vector in IMPD-t. Even if $\theta_i$'s had a discrete distribution, it would require the solution of a large-sized integer problem in the lower level, leading to an inefficient solution approach. The SAA method can be utilized in such a case by using random samples that are generated from the set of all realizations. Our application of the SAA method is based on the framework proposed by Kleywegt *et al.* (2002), and it consists of three stages. The first stage employs the solution of the follower's approximate LLP, referred to as ALLP-t:

$$\hat{z}_N(\mathbf{x}) = \min_{\mathbf{y},\mathbf{u}} \frac{1}{N} \sum_{r=1}^{N} \sum_{i \in V} u_{ir} \tag{4.26}$$

s.t.

$$(4.11), (4.12), \text{and}$$

$$u_{ir} \geq x_i - y_i \qquad i \in V, r = 1, ..., N \tag{4.27}$$

$$u_{ir} + y_i \geq \sum_{j \in V} w_{ji} u_{jr} - \theta_{ir} + \epsilon \qquad i \in V, r = 1, ..., N \tag{4.28}$$

$$u_{ir}, y_i \in \{0, 1\} \qquad i \in V, r = 1, ..., N \tag{4.29}$$

Here, $N$ represents the sample size, namely the number of threshold scenarios. Notice that each scenario $r$ consists of $|V|$ threshold values, $\theta_i$, associated with each node $i \in V$. The optimal objective value $\hat{z}_N(\mathbf{x})$ of the ALLP-t for a given leader decision $\mathbf{x}$ is a negatively biased estimator of the optimal objective value $z(\mathbf{x})$ of the original LLP. That is, $\mathbb{E}[\hat{z}_N(\mathbf{x})] \leq z(\mathbf{x})$, which can be explained by means of the relationship

$$\mathbb{E}\left[\hat{z}_N(\mathbf{x})\right] = \mathbb{E}\left[\min_{\mathbf{y},\mathbf{u}} \frac{1}{N} \sum_{r=1}^{N} \sum_{i \in V} u_{ir}\right] \leq \min_{\mathbf{y},\mathbf{u}} \mathbb{E}\left[\frac{1}{N} \sum_{r=1}^{N} \sum_{i \in V} u_{ir}\right] = \min_{\mathbf{y},\mathbf{u}} \sum_{r \in R} \sum_{i \in V} p_r u_{ir} = z(\mathbf{x}). \tag{4.30}$$

Inequality (4.30) is due to the interchange of the minimum and expectation operators, as can be seen in Norkin *et al.* (1998). In order to approximate the objective value $z(\mathbf{x})$,

we initially generate $M$ independently and identically distributed threshold samples of size $N$, and solve the ALLP-t for each sample $m$ (also called batch $m$) to obtain an optimal follower decision $\tilde{\mathbf{y}}^m$ with objective value $\hat{z}_N^m(\mathbf{x})$. The average of these objective values

$$\bar{z}_N^M(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} \hat{z}_N^m(\mathbf{x}), \tag{4.31}$$

constitutes a statistical lower bound on the optimal spread $z(\mathbf{x})$ because of the negative bias of each $\hat{z}_N^m(\mathbf{x})$ as discussed before. This completes the first stage of the SAA method. Notice that the vector $\mathbf{y}$ represent the first-stage variables in the context of two-stage stochastic programming models for which the SAA method is developed, as they are not associated with a scenario.

The second stage of the SAA method involves determining the best follower response $\tilde{\mathbf{y}}^*$ among $\tilde{\mathbf{y}}^m, m \in M$ that were found in the first stage. To this end, the ALLP-t can be solved for each $m$ by fixing the optimal follower decision $\tilde{\mathbf{y}}^m$ and using a larger number of threshold scenarios $N' \gg N$. Notice that the only decision variables remaining in the ALLP-t model are the second-stage variables $u_{ir}$ since leader's decision $\mathbf{x}$ is also given in the LLP. However, we apply a more efficient procedure and calculate the resulting objective value denoted as $\hat{z}_{N'}(\mathbf{x}, \tilde{\mathbf{y}}^m)$, in an iterative way by fixing the activated nodes $\mathbf{x}$ and deactivated nodes $\tilde{\mathbf{y}}^m$ in the network, and unfolding the influence propagation until no more nodes are influenced. Our experiments revealed that this algorithmic approach of computing the spread is faster than solving the mathematical model. The best follower decision $\tilde{\mathbf{y}}^*$ for a given leader decision $\mathbf{x}$ is found as the one $\tilde{\mathbf{y}}^m$ that provides the smallest spread value $\hat{z}_{N'}(\mathbf{x}, \tilde{\mathbf{y}}^m)$, namely $\hat{z}_{N'}(\mathbf{x}, \tilde{\mathbf{y}}^*) = \min_m \hat{z}_{N'}(\mathbf{x}, \tilde{\mathbf{y}}^m)$. Kleywegt *et al.* (2002) show that $\hat{z}_{N'}(\mathbf{x}, \tilde{\mathbf{y}}^\star)$ converges to $z(\mathbf{x})$ with probability one as $N' \to \infty$.

The third and last stage of the SAA method consists of computing an unbiased estimate of the optimal spread $z(\mathbf{x})$. This is achieved by taking an independent sample size $N'' \gg N'$ (Güney, 2017; Verweij *et al.*, 2003). Note that this again requires either

the solution of the ALLP-t in terms of second-stage variables $u_{ir}$ by fixing leader's decision variable $\mathbf{x}$ and follower's first-stage decision variable $\tilde{\mathbf{y}}^*$ or adopting the algorithmic way. As before, the latter approach turns out to be more efficient. It is clear that $E[\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*)] \geq z(\mathbf{x})$ as the objective value on the left-hand side is associated with a feasible solution while the one on the right-hand side is the optimal objective value of the LLP. In other words, we obtain an upper bound estimate on the optimal objective value $z(\mathbf{x})$ of the LLP in IMPD-t. Thus, we approximate $z(\mathbf{x})$ from above.

In the SAA method, it is possible to calculate an estimator of the gap for the optimal objective value of the stochastic integer programming model in question. In our case, the optimality gap for $z(\mathbf{x})$ can be estimated by the quantity

$$\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*) - \bar{z}_N^M(\mathbf{x}), \tag{4.32}$$

which actually overestimates the gap due to the bias of $\bar{z}_N^M(\mathbf{x})$. Besides the estimation of this gap, we can also compute the variance of the estimated gap as follows:

$$\begin{aligned} \mathrm{Var}\big(\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*) - \bar{z}_N^M(\mathbf{x})\big) &= \mathrm{Var}\big(\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*)\big) + \mathrm{Var}\big(\bar{z}_N^M(\mathbf{x})\big) \\ &= \frac{1}{N''(N''-1)} \sum_{r=1}^{N''} \big[\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*) - g(\mathbf{x}, \mathbf{y}^*, \theta_r)\big]^2 + \\ &\quad \frac{1}{M(M-1)} \sum_{m=1}^{M} \big[\hat{z}_N^m(\mathbf{x}) - \bar{z}_N^M(\mathbf{x})\big]^2. \end{aligned} \tag{4.33}$$

<u>4.2.1.2. The Live-arc Model.</u> As opposed to IMPD-t, formulation IMPD-La is based on a scenario definition with a discrete probability space. Nevertheless, the SAA method is still applicable and can be utilized due to the following reasons. A live-arc scenario corresponds to a subset of the arcs in the network as described in Section 4.1.2. Even though the number of scenarios is finite, it increases exponentially with the network size. Let $b_i$ be the in-degree, i.e., the number of incoming arcs of node $i$ in the original network. Then, $\prod_{i \in V} b_i$ is a lower bound on the number of scenarios. In the worst case, if $\sum_{j \in V} w_{ji} < 1$ for all $i$, then the exact number of scenarios is $\prod_{i \in V}(b_i + 1)$.

It is not practical to solve the deterministic equivalent formulation as a single MILP, even for very small networks. The decomposition based two-stage stochastic programming methods for problems with recourse, such as the $L$-Shaped method, assume a small number of scenarios. The Integer $L$-Shaped method proposed by Laporte and Louveaux (1993), which allows binary variables in both stages, depends on the assumption that the objective value can easily be computed for a given first-stage solution (a first stage solution is represented by the variable $\mathbf{y}$ in our problem), which is not the case for the LLP of the IMPD-La. For these reasons, we proceed with the SAA method to approximate the lower level objective value. The approximation problem ALLP-La is formulated as:

$$\hat{z}_N(\mathbf{x}) = \min_{\mathbf{y},\mathbf{u}} \frac{1}{N} \sum_{r=1}^{N} \sum_{i \in V} u_{ir} \tag{4.34}$$

s.t.

$$(4.21), (4.22), \text{and}$$

$$u_{ir} \geq x_i - y_i \qquad i \in V, r = 1, ..., N \tag{4.35}$$

$$y_i + u_{ir} \geq u_{\delta_r(i),r} \qquad i \in V_r, r = 1, ..., N \tag{4.36}$$

$$u_{ir} \geq 0, y_i \in \{0, 1\} \qquad i \in V, r = 1, ..., N. \tag{4.37}$$

The properties of the statistical bounds that the SAA method yields for the threshold model which are explained in Section 4.2.1.1 are valid for the live-arc model as well. Likewise, the optimality gaps can be estimated by using the estimator in (4.32). The basic change is the approximation problem solved in the first stage of the algorithm which will be referred to as ALLP in the rest unless a particular one of the two formulations is meant. Since we use an MILP solver to solve these problems, it is possible to present a common SAA algorithm for the approximate solution of the LLP in formulations IMPD-t and IMPD-La. We present the steps of the SAA method implemented for a given leader strategy $\mathbf{x}$ in Algorithm 4.2. Lines 1–6 correspond to stage 1 of the algorithm, where lower bound $\bar{z}_N^M$ and candidate solutions $\tilde{\mathbf{y}}^m$ are generated. Lines 7–11 comprise stage 2 where the best response $\tilde{\mathbf{y}}^*$ is determined.

The remaining lines form the third stage, which involves the computation of the upper bound estimate $\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*)$ as well as an estimate of the optimality gap, and its variance.

**Algorithm 4.2** SAA($\mathbf{x}$)

```
Choose sample sizes N, N', N'' such that N < N' < N'', and the number of batches M
for m = 1, ..., M do
    Generate N threshold realizations
    Solve the ALLP to obtain the optimal objective value ẑ_N^m(x) and an optimal solution ỹ^m
end for
Compute z̄_N^M(x) = (1/M) Σ_{m=1}^M ẑ_N^m(x)
Generate N' threshold realizations
for m = 1, ..., M do
    Compute ẑ_{N'}(x, ỹ^m) algorithmically
end for
Choose the best solution ỹ* = arg min_m ẑ_{N'}(x, ỹ^m)
Generate N'' threshold realizations
Compute ẑ_{N''}(x, ỹ*) algorithmically
Compute the optimality gap ẑ_{N''}(x, ỹ*) − z̄_N^M(x) and its variance given in (4.33)
```

Figure 4.2. Pseudo-code of the Sample Average Approximation (SAA) algorithm.

4.2.1.3. Sampling Methods. The optimality gaps produced by the SAA method are usually overestimated due to the bias in the lower bound estimate, which is the output of stage 1. Increasing the sample size decreases the bias, however using very large sample sizes reduces efficiency due to exponentially increasing MILP solution times. If an improved sampling technique is used, both the variance and accuracy of the bound estimates can be improved. We consider three sampling procedures for the sampling of scenarios.

(i) Simple Random Sampling (SRS): In this method, the scenarios are selected from the scenario space independently with their respective probabilities.

(ii) k-Means Clustering: Clustering techniques may help to sample scenarios that will represent the scenario space better than random sampling. Let $N$ denote the size

of the samples used in stage 1. In the method proposed by Emelogu *et al.* (2016), first a sample of size $N_C$ is generated by simple random sampling. Then these points are split into $N$ clusters via one of the well known clustering methods. A representative point from each cluster is selected to be used in the ALLP formulation. We use $k$-means++ method to cluster the sample points (Arthur and Vassilvitskii, 2006). The basic difference of $k$-means++ from $k$-means is that the initial cluster centers are not selected randomly, but after the first randomly selected center, the next is selected with a probability proportional to its distance to the currently fixed cluster centers. The procedure is repeated until there is a desired number of cluster centers. The algorithm is given in Appendix A.

(iii) Latin Hypercube Sampling (LHS): This method generates more evenly distributed sample points and is known to reduce the variance (McKay *et al.*, 1979). Linderoth *et al.* (2006) use the LHS method while generating the samples used in the SAA scheme and observe better optimality gap and variance estimates. Suppose that we need to generate a LHS sample $\mathbf{X}_1, ...\mathbf{X}_N$ of parameter $\mathbf{X}$ which has $K$ components and the distribution of $X_k$ is denoted by $F_k$. $P$ is an $N \times K$ matrix whose columns are random permutations of $\{0, ..., N-1\}$ and $\epsilon_{jk}$ are iid random numbers in $[0, 1]$. The general formula is:

$$X_{jk} = F_k^{-1}(\frac{p_{jk} + \epsilon_{jk}}{N}), j \in \{1, ..., N\}, k \in \{1, ..., K\} \tag{4.38}$$

In other words, the sample space is divided into $N$ equal probability intervals for each component of the parameter and exactly one sample point is generated from each interval. The distribution of node thresholds is uniform in $[0, 1]$ in our problem. Therefore, samples of size $N$ can be generated using $\theta_{ri} = (p_{ri} + \epsilon_{ri})/N$, $i \in V, r \in \{1, ..., N\}$. For the live-arc scenarios, the predecessor of node $i$ in scenario $r$ is determined using the value of $\theta_{ri}$ computed as above and cumulative weights of the arcs incoming to $i$.

### 4.2.2. Complete Enumeration of the Leader's Solutions

Note that we solve the LLP of IMPD-t given in (4.10)–(4.15) and IMPD-La in (4.20)–(4.25) in an approximate way using the SAA method. Since the estimated objective value of the follower for a given leader strategy $\mathbf{x}$ is given as $\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^\star)$, where $\tilde{\mathbf{y}}^\star$ is the best follower response to $\mathbf{x}$, we rewrite the ULP as

$$\hat{z} = \max_{\mathbf{x} \in \mathbb{X}} \hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*), \tag{4.39}$$

where $\mathbb{X} = \{\mathbf{x} : \mathbf{c}^T\mathbf{x} \leq C, \mathbf{x} \in \{0, 1\}\}$. Although the bilevel feasibility of a solution found in this way is not guaranteed since the optimality of the follower solution is not certain, it is observed in the computational experiments, whose results are given in Section 8.2.1.1, that the (over)estimated optimality gaps are quite small. Therefore, it is very unlikely that the leader solution found by this approach is suboptimal.

One method to solve ULP (4.39) to optimality is to enumerate all feasible solutions of the ULP. Then, the estimated optimal objective value of the follower, $\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*)$, is computed using the SAA algorithm given in Figure 4.2 for each feasible solution; we will denote this value as $\hat{z}_{SAA}(\mathbf{x})$ in the sequel. Note that it is non-decreasing in the set of activated nodes, i.e., the seed set, which allows the elimination of the solutions that has no chance of optimality. Let $\mathbb{X}'$ denote the set of maximal subsets of $\mathbb{X}$ ($A \subseteq \mathbb{X}$ is maximal if there is no $B \subseteq \mathbb{X}$ such that $A \subset B$). Hence, we can consider solving the problem $\hat{z} = \max_{\mathbf{x} \in \mathbb{X}'} \hat{z}_{SAA}(\mathbf{x})$. Obviously, the activation budget $C$ and the network size $n$ determine the size of $\mathbb{X}'$. The number of candidate feasible solutions to be evaluated increases exponentially with increasing values of these parameters. Therefore, this method can only be helpful for very small networks. In this thesis, the results of the complete enumeration method are used as references while evaluating the performance of the proposed matheuristics for small instances.

### 4.2.3. Matheuristic Methods for the Leader's Problem

Matheuristics are algorithms that consist of a metaheuristic solution procedure combined with a mathematical programming based method to solve an optimization problem (Boschetti *et al.*, 2009). We propose two matheuristics based on Simulated Annealing and Tabu Search. Both of them perform a search in the solution space of the leader's decision variables $\mathbf{x}$ and for each solution visited, the optimal spread is estimated using the SAA algorithm. Therefore, each method can be regarded as a metaheuristic coupled with the solution of a stochastic mathematical programming model solved by means of SAA. There are two important aspects with regard to these methods. First, only feasible solutions are considered due to the costly $\hat{z}_{SAA}(\mathbf{x})$ computation of the LLPs. Moreover, it is not allowed to visit the same solution more than once for the sake of computational efficiency. This is achieved by using an explicit memory structure that stores each solution generated by means of a hash function.

Two approaches are considered for generating an initial solution. The first one is based on the influence degrees of nodes as well as the activation and deactivation costs. The influence degree of node $i$, is the spread resulting from selecting it as the only seed node in case without deactivation. The nodes are sorted in non-increasing order of their influence degrees. If it is a cost-based IMPD instance, then the second half of the nodes are removed from the list, while the first half is sorted again in terms of increasing value of the $c_i/e_i$ ratio. The nodes at the top of the final list are selected as seed nodes until the budget constraint is violated. This selection favors relatively more influential nodes that can be activated at a low cost but deactivated at a high cost. The second approach for initial solution generation is based only on the activation and deactivation costs of the nodes. In this approach, nodes are sorted in non-decreasing order with respect to their activation costs $c_i$. Ties are broken in favor of the largest deactivation cost $e_i$.

Let $S = \{i \in V : x_i = 1\}$ represent the set of nodes in the network that are selected as the seed nodes in a leader solution $\mathbf{x}$. In the sequel, we use $\hat{z}_{SAA}(\mathbf{x})$ and

$\hat{z}_{SAA}(S)$ interchangeably to simplify the notation. Before going into the details of the matheuristics, we would like to point out that the objective value of a candidate leader solution $S$ is computed as

$$f(S) = \hat{z}_{SAA}(S) + \frac{C - \sum_{i \in S} c_i}{\bar{c}}, \tag{4.40}$$

where $\bar{c}$ is the average activation cost of the nodes in the network. The second term on the right-hand side of Equation (4.40) represents the average number of nodes that can be activated using the remaining budget for a solution $S$. The rationale of using this pseudo objective value is to reward a seed set that achieves a lower spread in comparison with another seed set albeit at a lower activation cost. This implies that we also take into account the potential of a seed set to improve the spread. Note that this approach does not favor an unpromising solution under the assumption that for any feasible solution there exist nodes in the network not belonging to the seed set and having an activation cost less than $\bar{c}$. This is a reasonable assumption since the size of the seed set is much smaller than the number of nodes in the network.

4.2.3.1. Simulated Annealing Based Matheuristic. The first method we propose is a matheuristic based on Simulated Annealing referred to as SAM. At each iteration of SAM, an eligible solution is selected randomly from the neighborhood of the current solution. The latter is updated if either the neighbor has a better objective value (higher spread) or the neighbor has a worse objective value but still accepted as the current solution with a probability that is a function of the amount of the deterioration in the objective value and the current temperature. Let $\mathbf{x}^*_{SAM}$ denote the best leader strategy that SAM finds, and $\hat{z}^*_{SAM}$ denote the corresponding objective value. Then, $\hat{z}^*_{SAM}$ is a lower bound on $\hat{z}$ given in Equation (4.39).

Before moving on to the details of SAM, we give the definition of an eligible solution. A solution or seed set $S$ is said to be *eligible* if it is feasible (i.e., $\sum_{i \in S} c_i \leq C$), not generated before, and has a positive spread. The last property implies that all the nodes of an eligible solution cannot be deactivated by the follower, namely $\sum_{i \in S} e_i > E$.

*Neighborhood Structure:* Three types of move operators are implemented: 1-Add, 1-Drop, and 1-Swap. A 1-Add move randomly chooses a node $i \in V \setminus S$ such that $S' = S \cup \{i\}$ is an eligible solution. A 1-Drop move randomly chooses a node $i \in S$ such that $S' = S \setminus \{i\}$ is eligible. Finally, a 1-Swap move randomly exchanges two nodes, $i \in S$ and $j \in V \setminus S$ to generate a new eligible solution $S' = (S \setminus \{i\}) \cup \{j\}$. All three move operators have the same chance of being selected, i.e., are chosen with equal probability of 1/3. If there does not exist an eligible solution in the selected move operator, then the remaining ones are tried.

*Initial Temperature and Temperature Update:* The initial temperature is calculated based on the approach proposed in Ohlmann and Thomas (2007). In this approach, a number of random solutions are generated, and from the neighborhood of each solution, a random neighbor is created. After the absolute value of the difference between each solution and its neighbor is computed, the average absolute difference is computed, which essentially reflects the objective value changes in the landscape of the objective function. The initial temperature to be used is determined by assuming that a random neighbor of the current solution whose objective value is worse than that of the current solution and equal to the average absolute difference is accepted with an initial probability of acceptance, $p_0$. In our implementation we generate 20 solutions. At the end of each cycle, we check whether the proportion of accepted solutions is greater than a threshold value $\phi$. If this is the case, then the current temperature is reduced to half, as the number of accepted solutions turns out to be large. Otherwise, the temperature is updated in a geometric fashion using a cooling ratio of $r$, as is done frequently in the literature. In other words, $T \leftarrow rT$.

*Cycle Length:* The number of iterations in each cycle (i.e., the cycle length $L$) varies during the search. The initial cycle length $L_0$ is the average neighborhood size of the initial solution obtained by the two approaches explained earlier, over the three move types (e.g., $L_0 = (k + (n - k) + (n - k)k)/3$ for an initial seed of size $k$). As the temperature decreases throughout the iterations, the acceptance probability of worsening solutions decreases, which makes finding and accepting better solutions

more difficult. Therefore, the number of iterations is increased gradually by setting $L \leftarrow (1+\eta)L$, where $\eta \in (0,1)$.

*Termination Criterion:* SAM is executed for a time limit equal to $t_{\max}$. However, if there is no eligible neighbor in any type of moves at an iteration, then the algorithm is stopped. This is more likely at low temperatures for small instances with a small neighborhood size since the probability of acceptance is small at these temperatures.

The Pseudo-code of SAM is displayed in Figure 4.3.

**Algorithm 4.3** SAM

```
Compute initial temperature T
Generate the initial solution (seed set) S and set S*_SAM ← S, ẑ*_SAM ← ẑ_SAA(S), j ← 0
Choose L, r, φ, and η
while CPU time ≤ t_max do
  for l = 1 to L do
    Set S' ← 1-Add(S) or S' ← 1-Drop(S) or S' ← 1-Swap(S) with equal probability
    Compute Δ = f(S') − f(S)
    if Δ ≥ 0 or (Δ ≤ 0 and e^(Δ/T) > U(0,1)) then
      Update the current solution S ← S', j ← j + 1
    end if
    if ẑ_SAA(S') > ẑ*_SAM then
      Update the incumbent solution S*_SAM ← S', ẑ*_SAM ← ẑ_SAA(S')
    end if
  end for
  if j/L > φ then
    T ← T/2
  else
    T ← rT
  end if
  Set L ← (1 + η)L, j ← 0
end while
Return S*_SAM and ẑ*_SAM
```

Figure 4.3. Pseudo-code of Simulated Annealing based matheuristic (SAM).

4.2.3.2. Tabu Search Based Matheuristic. In this method, the solution space of the leader is explored using Tabu Search that is implemented with a candidate list strategy, which helps to reduce the computational effort. At each iteration of Tabu Search based matheuristic (TSM), only a promising subset of the neighboring solutions, i.e., those belonging to the candidate list, is considered. The proportion of the size of this subset within the size of the complete neighborhood is controlled by a parameter, denoted as $\nu$. A preprocessing step is carried out before executing the TSM in order to determine the promising candidates at each iteration. This preprocessing step makes use of the length of the shortest path $\lambda_{ij}$ from node $i$ to node $j$ on $G' = (V, A)$ where the length of the arc $(i, j)$ is computed as $-\log(w_{ij})$. It is computed between each pair of nodes in the network using Dijkstra's Algorithm (Dijkstra, 1959). Larger weights on a path between two nodes indicate higher chance of influencing each other under the LT diffusion model. Hence, a small $\lambda_{ij}$ is an indicator of high direct/indirect influence of node $i$ on node $j$.

*Neighborhood Structure:* The move operators 1-Add, 1-Drop and 1-Swap used in SAM are also employed in TSM so that none of the matheuristics is favored over the other. At each iteration, one of the moves is selected with equal probability. As mentioned before, instead of generating all the neighboring solutions using a move operator, we determine a candidate list and take into this list only promising solutions existing in the neighborhood. This approach is based on the idea that if a node has a smaller chance to be influenced by the seed nodes, then adding this node to the seed set has more potential to improve the spread. Let $S$ be the current seed set. The neighborhood size for the 1-Add move is $|V \setminus S|$. To determine a promising subset of neighboring solutions to be included in the candidate list, a score given as $\sum_{i \in S} \lambda_{ij}$ is assigned first to each eligible neighbor $S' = S \cup \{j\}$. Then, the candidates are sorted by their score in non-increasing order. The first $\nu|V \setminus S|$ elements in the candidate list are evaluated and the best one is chosen. We evaluate all eligible candidates in the 1-Drop move since the size of this neighborhood is relatively small. Its value is equal to $|S|$ at most. The size of the 1-Swap move operator is $|S||V \setminus S|$. The score function for this operator is $\sum_{i \in S} \lambda_{ij}$ for any neighbor $S' = (S \setminus \{k\}) \cup \{j\}$ (i.e., the candidates are evaluated only based on the node that will be added to the seed). The eligible candidates are

first sorted in non-increasing order in terms of their score. Then, top $\nu|V \setminus S||S|$ in the list are taken into consideration, and the best promising neighbor is determined.

*Tabu Structure:* Since the solutions are mapped to integers and stored in a list using a hash function, all solutions visited before are declared tabu in our matheuristic.

*Diversification Strategy:* A frequency-based long-term memory is utilized in order to penalize the frequently observed nodes in the seed set. To this end, we keep track of the proportion $\pi_i$ of the visited solutions that have node $i$ in the seed set. Each candidate solution $S$ is penalized by adding the term $-\mu \sum_{i \in S} \pi_i$ to its objective value.

*Termination Criterion:* We allocate a time limit equal to $t_{\max}$ as is the case with SAM. TSM terminates before $t_{\max}$ only if no eligible solutions can be obtained by any move operator.

The Pseudo-code of TSM is displayed in Figure 4.4.

**Algorithm 4.4** TSM

Generate the initial solution (seed set) $S$ and set $S^*_{TSM} \leftarrow S$, $\hat{z}^*_{TSM} \leftarrow \hat{z}_{SAA}(S)$
Choose $\nu$ and $\mu$
**while** CPU time $\leq t_{\max}$ **do**
  Set $S' \leftarrow$ 1-Add$(S, \nu)$ or $S' \leftarrow$ 1-Drop$(S)$ or $S' \leftarrow$ 1-Swap$(S, \nu)$ with equal probability
  **if** $\hat{z}_{SAA}(S') > \hat{z}^*_{TSM}$ **then**
    Update the incumbent solution $S^*_{TSM} \leftarrow S', \hat{z}^*_{TSM} \leftarrow \hat{z}_{SAA}(S')$
  **end if**
  Update the parameter $\pi_i, i = 1, \ldots n$
**end while**
Return $S^*_{TSM}$ and $\hat{z}^*_{TSM}$

Figure 4.4. Pseudo-code of Tabu Search based matheuristic (TSM) for IMPD.

4.2.3.3. Simple Heuristics. An effective heuristic method developed for solving a bilevel problem should anticipate the follower's response to generate a good decision for the

leader. Simpler heuristic methods, however, can be devised by ignoring the follower's objective function in the LLP, and solve the leader's ULP provided that the constraints of the ULP does not include the follower's decision variables. Since this is the case in IMPD, we devise such simple heuristics for comparison purposes. Each simple heuristic (SH) adopts a measure for selecting the nodes as a seed node as long as the activation budget is not exceeded. The first one is referred to as SH-AC since the activation cost is used as the measure for seed node selection. All nodes are sorted in non-decreasing order of the activation cost. Then, starting with the first node in the list and breaking ties in favor of the largest deactivation cost, nodes are added to the seed set until the activation budget is exceeded. The second SH is called SH-DC since nodes are sorted in non-increasing order of the deactivation cost. Nodes are added to the seed set starting from the first node in the list where ties are broken in favor of the smallest activation cost. SH-OD employs the out-degree of the nodes as the measure and seed nodes are determined by starting from the ones having the largest out-degree, and ties are broken in favor of the smallest activation cost. SH-SP is based on the first approach that is utilized in generating an initial solution for SAM and TSM, which is described in Section 4.2.3. We should remark that SH-AC and SH-DC cannot be implemented for cardinality-based instances because they have unit activation and deactivation costs. The results of these methods are compared to those of the matheuristic methods in Section 8.2.2.3.

# 5. A BRANCH-AND-CUT ALGORITHM

One of the most recent methods for MIBLPs is the branch-and-cut (B&C) algorithm of Fischetti *et al.* (2016). In their method, intersection cuts that have been introduced by Balas (1971) to solve MILPs are used to solve an MIBLP by cutting off bilevel infeasible solutions as described in Section 2.3. The results show that this method significantly outperforms the existing techniques for general MIBLPs. Fischetti *et al.* (2017b) add new features to their algorithm such as preprocessing procedures and a new type of intersection cut. The most important feature of the method is that basically a standard MILP solver is used to solve an MIBLP with some modifications in the rules of a standard B&C algorithm. Although the method is designed for deterministic problems, it can be utilized to solve the deterministic equivalent formulation of a stochastic bilevel program with a finite number of scenarios. In this chapter, we implement the method on the deterministic equivalent of IMPD, with live-arc scenarios whose formulation IMPD-La is given in (4.17)–(4.25).

In addition to the adaptation of the algorithm to our problem, we propose a variable bound update scheme for reducing the feasible regions of the subproblems and prune them if it is possible. Besides, we tighten the initial single-level MILP formulation with bilevel feasibility constraints. We also consider several bilevel-free set definitions with the aim of obtaining a deeper cut.

## 5.1. The High Point Relaxation and Bilevel Feasibility Constraints

The B&C algorithm starts with developing the *value function formulation* of the bilevel formulation at hand. It is a single-level formulation and given below for our problem.

$$\max_{\mathbf{x},\mathbf{y},\mathbf{u}} \frac{1}{|R|} \sum_{r \in R} \sum_{i \in V} u_{ir} \tag{5.1}$$

s.t.

$$\sum_{i \in V} x_i = k \tag{5.2}$$

$$\sum_{i \in V} y_i = l \tag{5.3}$$

$$y_i \leq x_i \qquad i \in V \tag{5.4}$$

$$u_{ir} \geq x_i - y_i \qquad i \in V, r \in R \tag{5.5}$$

$$y_i + u_{ir} \geq u_{\delta_r(i),r} \qquad i \in V_r, r \in R \tag{5.6}$$

$$\frac{1}{|R|} \sum_{r \in R} \sum_{i \in V} u_{ir} \leq z(\mathbf{x}) \tag{5.7}$$

$$x_i \in \{0,1\}, y_i \in \{0,1\}, u_{ir} \geq 0 \qquad i \in V, r \in R \tag{5.8}$$

Here, the *follower value function* $z(\mathbf{x})$ is defined as before as in (4.20)–(4.25). Dropping the bilevel feasibility condition (5.7) from this formulation leads to the so-called *high point relaxation* (HPR), on which the B&C algorithm is applied. An HPR solution $(\mathbf{x}, \mathbf{y}, \mathbf{u})$ is called bilevel infeasible if it does not satisfy (5.7). Notice that a bilevel feasible solution satisfies Constraint (5.7) as equality. In the B&C algorithm, the HPR is solved with a standard commercial MILP solver with some modification in node processing, branching, and fathoming rules. Whenever an integer solution is found, the bilevel feasibility is checked by solving the follower's problem optimally. If it is not bilevel feasible, an appropriate Intersection Cut (IC), which cuts off that point from the LP relaxation of the HPR, is added to the current subproblem. A sample implementation given in Fischetti *et al.* (2016) is provided in Appendix B.1.

Now, we suggest inequalities that may lead to a stronger HPR formulation. The follower constraints (5.5) and (5.6), which state the conditions that a node can be influenced, normally belong to a formulation with an objective of minimizing $u_{ir}$'s. When the follower objective is removed, these constraints do not restrict the values that $u_{ir}$ take. Assuming that explicit upper bounds (of one) on those variables are introduced, they are all set to one in favor of the leader. Although the objective value decreases and bilevel feasible solutions are reached as more ICs are added, adding

problem specific constraints to the HPR formulation in advance to eliminate bilevel infeasible solutions may increase the efficiency of the algorithm. To this end, we come up with the constraints

$$u_{ir} \leq 1 - y_i \qquad i \in V, r \in R \tag{5.9}$$

$$u_{ir} \leq x_i + u_{\delta_r(i),r} \qquad i \in V_r, r \in R, . \tag{5.10}$$

which are satisfied for any bilevel feasible HPR solution. The above inequalities are not directly implied by the constraints of HPR. However, it is possible to observe in an optimal follower solution that a deactivated node ($y_i = 1$) cannot be influenced in any scenario independent of the state of its predecessor. Besides, it cannot be influenced unless it is a seed node ($x_i = 1$) or its predecessor is influenced ($u_{\delta_r(i),r} = 1$). Based on preliminary experiments whose results are provided in Section 8.3 we decide to include inequalities(5.9) and (5.10) in the HPR formulation.

## 5.2. Bilevel-free Sets

Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a feasible solution of the HPR formulation of a general MIBLP, i.e., satisfying integrality requirements, but not a bilevel feasible solution. Fischetti *et al.* (2016) obtain an IC which is violated by $(\mathbf{x}^*, \mathbf{y}^*)$ by using the corner polyhedron of the optimal basis for the relevant LP (Conforti *et al.*, 2011) and a convex set containing $(\mathbf{x}^*, \mathbf{y}^*)$ but not any bilevel feasible point in its interior, i.e., a *bilevel-free* set. Note that a bilevel-free set can have bilevel feasible points in its frontier. They introduce the set

$$S^+(\hat{\mathbf{y}}) = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n : d^T\mathbf{y} \geq d^T\hat{\mathbf{y}}, A\mathbf{x} + B\hat{\mathbf{y}} \leq b + \mathbf{1}\}, \tag{5.11}$$

where $\hat{\mathbf{y}}$ is an arbitrary follower solution, $d^T\mathbf{y}$ is the follower's objective function and $A\mathbf{x} + B\mathbf{y} \leq b$ is the set of follower constraints with a non-zero $\mathbf{x}$ coefficient with $\mathbf{1}$ denoting the vector of ones. They prove that this set does not contain any bilevel feasible solution in its interior with the assumption that $A\mathbf{x} + B\mathbf{y} - b$ is integer for all

HPR solutions. They suggest two options for the selection of $\hat{\mathbf{y}}$. In the first one, $\hat{\mathbf{y}}$ is an optimal follower response to $\mathbf{x}^*$, therefore it can be obtained by solving the follower's problem. For the second one, they make use of the following argument. Consider a polyhedron $S = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n : \alpha_i^T \mathbf{x} + \beta_i^T \mathbf{y} \leq \gamma_i,\ i = 1, ..., k\}$ which does contain any bilevel feasible points in its interior. The $i^{\text{th}}$ facet can be removed from $S$ if

$$\sum_{j=1}^{n_1} \max\{\alpha_{ij} x_j^+, \alpha_{ij} x_j^-\} + \sum_{j=1}^{n_2} \max\{\beta_{ij} y_j^+, \beta_{ij} y_j^-\} < \gamma_i. \tag{5.12}$$

In other words, if the complementary half-space of a facet cannot contain a bilevel feasible solution then it is removable. They formulate an MILP whose optimal solution $\hat{\mathbf{y}}$ results in the removal of maximum number of facets of $S^+(\hat{\mathbf{y}})$ based on the result presented above.

We adopt the bilevel-free set in Fischetti *et al.* (2017b) for our implementation and consider several variations to examine their effect on the solution procedure. This investigation may be fruitful since the definition of the bilevel feasible set determines the depth of the IC. Let $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{u}^*)$ denote the current HPR solution and $(\hat{\mathbf{y}}, \hat{\mathbf{u}})$ denote some follower solution for the IMPD. We present seven bilevel-free set (BFS) definitions i.e., separation alternatives for our problem below. Note that the additional constraints (5.9) and (5.10) are not included in the definition of any BFS since shrinking the bilevel-free set reduces the depth of the cut and this is not preferable.

- BFS1: This set directly follows from (5.11), and is stated as

$$\begin{aligned} S^+(\hat{\mathbf{y}}, \hat{\mathbf{u}}) = \{(\mathbf{x}, \mathbf{y}, \mathbf{u}) \in \mathbb{R}^{(n_1+n_2)} : & \sum_{r \in R} \sum_{i \in V} u_{ir} \geq \sum_{r \in R} \sum_{i \in V} \hat{u}_{ir}, \\ & \hat{y}_i - x_i \leq 1, x_i - \hat{y}_i - \hat{u}_{ir} \leq 1, i \in V, r \in R\} \end{aligned} \tag{5.13}$$

where $(\hat{\mathbf{y}}, \hat{\mathbf{u}})$ is an optimal follower solution for $\mathbf{x}^*$, $n_1$ and $n_2$ are the numbers of

leader and follower variables, respectively. Note that the interior of this set,

$$S(\hat{\mathbf{y}}, \hat{\mathbf{u}}) = \{(\mathbf{x}, \mathbf{y}, \mathbf{u}) \in \mathbb{R}^{(n_1+n_2)} : \sum_{r \in R} \sum_{i \in V} u_{ir} > \sum_{r \in R} \sum_{i \in V} \hat{u}_{ir}, \\ \hat{y}_i - x_i \leq 0, x_i - \hat{y}_i - \hat{u}_{ir} \leq 0, i \in V, r \in R\}, \tag{5.14}$$

does not contain any bilevel feasible solution because any $(\mathbf{x}, \mathbf{y}, \mathbf{u}) \in S(\hat{\mathbf{y}}, \hat{\mathbf{u}})$ has a worse follower objective value than $(\mathbf{x}, \hat{\mathbf{y}}, \hat{\mathbf{u}})$ for the same leader solution. Moreover, the interior contains the initial point $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{u}^*)$. Therefore, it can be used to obtain an IC. During the implementation, the last inequality in the definition of $S^+(\hat{\mathbf{y}}, \hat{\mathbf{u}})$ is replaced by $x_i \leq \min_r\{1+\hat{y}_i+\hat{u}_{i,r}\}$ to eliminate redundant inequalities.

- BFS2: $S^+(\hat{\mathbf{y}}, \hat{\mathbf{u}})$ in (5.13) is modified by determining the removable facets. A facet $\alpha_i^T x+\beta_i^T y \leq \gamma_i$ is removable according to Fischetti *et al.* (2017b) if $\alpha_i^T x+\beta_i^T y \geq \gamma_i$ does not contain bilevel feasible solutions. Consequently, the facet $\hat{y}_i - x_i \leq 1$ in (5.13) is removable for $\hat{y}_i = 0$ since $x_i \leq -1$ does not contain a bilevel feasible solution. Since there is a cardinality restriction on the deactivation decisions, this rule results in the removal of most of these facets. Likewise, $x_i - \hat{y}_i - \hat{u}_{ir} \leq 1$ is removable when $\hat{y}_i + \hat{u}_{ir} \geq 1$ since $x_i \geq 2$ does not contain any bilevel feasible solution.
- BFS3: In this separation alternative, we expand the bilevel-free set further using the results for zero-sum games like ours. Let $\hat{z} = \sum_{r \in R} \sum_{i \in V} \hat{u}_{ir}$ denote the objective value of the follower for the selected follower solution and $z_{inc}$ is the current best leader objective value. $z_{inc}$ is a lower bound on the objective value of the follower in an optimal leader solution since they have the same objective functions and the leader seeks to maximize it. If $\hat{z} < z_{inc}$, then the facet which is associated with the objective value in (5.13), i.e., $\sum_{r \in R} \sum_{i \in V} u_{ir} \geq \hat{z}$, can be removed since the halfspace $\sum_{r \in R} \sum_{i \in V} u_{ir} \leq \hat{z}$ cannot contain an optimal solution, even though it may contain bilevel feasible solutions.
- BFS4: The set in (5.13) used with a different follower solution instead of the optimal one. A separation MILP is solved to determine the follower solution with

maximum number of removable facets. The formulation is provided in Appendix B. Note that this results in solving two MILPs for each integer feasible but bilevel infeasible HPR solution, one is the follower's problem to decide bilevel feasibility and the other is the separation MILP to determine the follower solution which will be used to define $S^+(\hat{\mathbf{y}}, \hat{\mathbf{u}})$.

- BFS5: In BFS4, the information about which facets are removable is obtained as a result of the separation MILP. In this separation alternative, we remove the objective value facet if possible, as is the case in BFS3.
- BFS6: We define the set

$$S^+(\hat{\mathbf{y}}, \hat{\mathbf{u}}) = \{(\mathbf{x}, \mathbf{y}, \mathbf{u}) \in \mathbb{R}^{(n_1+n_2)} : \sum_{r \in R} \sum_{i \in V} u_{ir} \geq \sum_{r \in R} \sum_{i \in V} \hat{u}_{ir}, \mathbf{x}^T \mathbf{x}^* \geq k - 1\}, \tag{5.15}$$

  which contains all the points with $\mathbf{x} = \mathbf{x}^*$ and suboptimal $(\mathbf{y}, \mathbf{u})$ pairs in its interior, where $(\hat{\mathbf{y}}, \hat{\mathbf{u}})$ is the optimal follower solution for $\mathbf{x}^*$. Since the current HPR solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{u}^*)$ is bilevel infeasible (otherwise we do not add an IC), it is contained in the interior as well, which implies that the new set is a proper BFS. The objective value facet is removed if possible as in BFS3 and BFS5.
- *Hypercube Intersection Cuts*: This set depends only on $\mathbf{x}^*$ (Fischetti *et al.*, 2017b), and is defined for our problem as

$$\mathrm{HC}^+(\mathbf{x}^*) = \{(\mathbf{x}, \mathbf{y}, \mathbf{u}) \in \mathbb{R}^{(n_1+n_2)} : x_i^* - 1 \leq x_i \leq x_i^* + 1, i \in V\}. \tag{5.16}$$

  which contains a bilevel feasible solution $(\mathbf{x}^*, \hat{\mathbf{y}}, \hat{\mathbf{u}})$. Although it is not bilevel-free, it can be used to obtain an IC while treating the bilevel feasible solution in it, which we cut off, as a candidate solution, and updating the incumbent solution if necessary.

## 5.3. Alternating Bound Update Procedure

The B&C algorithm can be improved by adding a preprocessing procedure which possibly reduce the size of the subproblems prior to solving them. In this section, we

propose a variable bound update scheme that can be implemented via solver callbacks before the solution of each subproblem. This scheme takes the current variable bounds as input and tightens them based on the information implied by the model constraints, if possible. A similar approach, which is referred to as "logical tests", is used in Sherali and Tuncbilek (1992) within a branch-and-bound algorithm to solve a location-allocation problem.

Let $\xi$ denote the vector of all variables in the HPR formulation. Suppose that we arrange the constraints of the HPR in such a way that they are expressed as $\sum_j A_{ij}\xi_j \leq b_i$, $i = 1, ..., m$, where $A_i$ is a row vector of appropriate size and $m$ is the total number of constraints including the bilevel feasibility constraints in Section 5.1. Note that $A_{ij}$ takes a value in $\{-1, 0, 1\}$ in the HPR formulation. It is clear that

(i) if the coefficient of $\xi_j$ is positive in the $i^{th}$ constraint ($A_{ij} = 1$), then an upper bound on the value of $b_i - \sum_{k \neq j} A_{ik}\xi_k$ yields an upper bound on $\xi_j$,

(ii) if the coefficient of $\xi_j$ is negative in the $i^{th}$ constraint ($A_{ij} = -1$), then a lower bound on the value of $\sum_{k \neq j} A_{ik}\xi_k - b_i$ yields a lower bound on $\xi_j$.

Considering all constraints in which $\xi_j$ appears results in a bound update if they yield a tighter bound than the current one. The bound update rules can be summarized as follows:

$$\overline{\xi_j}^{new} = \min\Big\{\overline{\xi_j}, \min_{i:A_{ij}>0}\big\{b_i - \sum_{k \neq j} \min\{A_{ik}\overline{\xi_k}, A_{ik}\underline{\xi_k}\}\ \big\}\Big\}, \tag{5.17}$$

$$\underline{\xi_j}^{new} = \max\Big\{\underline{\xi_j}, \max_{i:A_{ij}<0}\big\{\sum_{k \neq j} \min\{A_{ik}\overline{\xi_k}, A_{ik}\underline{\xi_k}\}\ - b_i\big\}\Big\}. \tag{5.18}$$

In order to define the bound update rules explicitly for each variable, we consider the HPR formulation involving (5.1)–(5.6), (5.8)–(5.10). The resulting rules are stated below, where a bold-written expression indicates that it is obtained through one of the

bilevel feasibility constraints (5.9) and (5.10).

$$\overline{x_i}^{new} = \min\left\{\overline{x_i},\, k - \sum_{j\neq i} \underline{x_j},\, \min_r\{\overline{u_{ir}} + \overline{y_i}\}\right\}$$

$$\underline{x_i}^{new} = \max\left\{\underline{x_i},\, k - \sum_{j\neq i} \overline{x_j},\, \underline{y_i},\, \boldsymbol{\max_r\{\underline{u_{ir}} - \overline{u_{\delta_r(i),r}}\}}\right\}$$

$$\overline{y_i}^{new} = \min\left\{\overline{y_i},\, l - \sum_{j\neq i} \underline{y_j},\, \overline{x_i},\, \boldsymbol{\min_r\{1 - \underline{u_{ir}}\}}\right\}$$

$$\underline{y_i}^{new} = \max\left\{\underline{y_i},\, l - \sum_{j\neq i} \overline{y_j},\, \max_r\{\underline{x_i} - \overline{u_{ir}}\},\, \max_r\{\underline{u_{\delta_r(i),r}} - \overline{u_{ir}}\}\right\}$$

$$\overline{u_{ir}}^{new} = \min\left\{\overline{u_{ir}},\, \boldsymbol{1 - \underline{y_i},\, \overline{x_i} + \overline{u_{\delta_r(i),r}}},\, \min_{j:\delta_r(j)=i}\{\overline{u_{jr}} + \overline{y_j}\}\right\}$$

$$\underline{u_{ir}}^{new} = \max\left\{\underline{u_{ir}},\, \underline{x_i} - \overline{y_i},\, \underline{u_{\delta_r(i),r}} - \overline{y_i},\, \boldsymbol{\max_{j:\delta_r(j)=i}\{\underline{u_{jr}} - \overline{x_j}\}}\right\}.$$

The preliminary study has shown that some of these expressions never lead to a better bound than the current one. When these redundant terms are removed, the final version of the above expressions becomes

$$\overline{x_i}^{new} = \min\left\{\overline{x_i},\, k - \sum_{j\neq i} \underline{x_j}\right\} \tag{5.19}$$

$$\underline{x_i}^{new} = \max\left\{\underline{x_i},\, k - \sum_{j\neq i} \overline{x_j},\, \underline{y_i}\right\} \tag{5.20}$$

$$\overline{y_i}^{new} = \min\left\{\overline{y_i},\, l - \sum_{j\neq i} \underline{y_j},\, \overline{x_i}\right\} \tag{5.21}$$

$$\underline{y_i}^{new} = \max\left\{\underline{y_i},\, l - \sum_{j\neq i} \overline{y_j}\right\} \tag{5.22}$$

$$\overline{u_{ir}}^{new} = \min\left\{\overline{u_{ir}},\, \boldsymbol{1 - \underline{y_i},\, \overline{x_i} + \overline{u_{\delta_r(i),r}}}\right\} \tag{5.23}$$

$$\underline{u_{ir}}^{new} = \max\left\{\underline{u_{ir}},\, \underline{x_i} - \overline{y_i},\, \underline{u_{\delta_r(i),r}} - \overline{y_i}\right\}. \tag{5.24}$$

It is possible that an update affects the terms in another variable's rules. Therefore, we alternate between computing the new (implied) bounds and updating the current ones until the newly computed bounds and the current ones are identical for all variables. The algorithm is referred to as Alternating Bound Update (ABU) and its Pseudo-code is presented as Algorithm 5.1. Note that all of the variables in the HPR must have

binary values in a bilevel feasible solution. Therefore, we skip computing the new lower bound of a variable if the new upper bound is less than the current one, which implies that the value of that variable is fixed.

**Algorithm 5.1** ABU
**Input**: Current upper and lower bound vectors $\overline{\boldsymbol{\xi}}$ and $\underline{\boldsymbol{\xi}}$;
Define $\overline{\boldsymbol{\xi}}^{new} = \overline{\boldsymbol{\xi}}$ and $\underline{\boldsymbol{\xi}}^{new} = \underline{\boldsymbol{\xi}}$, set $j \leftarrow 0$
**while** $j = 0$ or $\overline{\boldsymbol{\xi}}^{new} < \overline{\boldsymbol{\xi}}$ or $\underline{\boldsymbol{\xi}}^{new} > \underline{\boldsymbol{\xi}}$ **do**
  Set $\overline{\boldsymbol{\xi}} \leftarrow \overline{\boldsymbol{\xi}}^{new}$, $\underline{\boldsymbol{\xi}} \leftarrow \underline{\boldsymbol{\xi}}^{new}$ and $j \leftarrow j + 1$
  **for** each variable $\xi_j$ **do**
    Compute the new upper bound $\overline{\xi_j}^{new}$
    **if** $\overline{\xi_j}^{new} = \overline{\xi_j}$ **then**
      Compute the new lower bound $\underline{\xi_j}^{new}$
    **end if**
  **end for**
**end while**
Return $\overline{\boldsymbol{\xi}}^{new}$ and $\underline{\boldsymbol{\xi}}^{new}$

Figure 5.1. Pseudo-code of Alternating Bound Update (ABU) procedure.

## 5.4. Implementation Details

The modification necessary on the B&C rules can be implemented using CPLEX callbacks. The facts that most of the operations such as accessing the subproblem LP at B&C nodes of an MILP, getting the relevant basis and the basis inverse matrix are not available on object oriented interfaces, restrict us to use the Callable Library (C application programming interface) of CPLEX. A lazy constraint callback function is defined via `CPXXsetlazyconstraintcallbackfunc()` to check the bilevel feasibility of every integer solution found. Let $z^*$ denote the optimal objective value of the node LP and $\mathbf{x}^*$ the value of the leader solution. If $z^* > z(\mathbf{x}^*)$, then the current solution is not bilevel feasible and an IC is generated using the procedure described in Appendix B.2. This cut is locally valid since it uses local information such as variable bounds and the subproblem's optimal basis, therefore it is added using the `CPXXcutcallbackaddlocal()` function. It is possible that all the facets are removed in

BFS3 and BFS5 separation options when a subproblem cannot yield a bilevel feasible solution which is better than the current incumbent. For these two options, we set an incumbent callback function (`CPXXsetincumbentcallbackfunc()`) to prune such nodes without adding any cuts. Since the Hypercube ICs cut off bilevel feasible solutions, they require a special treatment. A bilevel feasible solution for the current leader solution is already obtained while checking the bilevel feasibility. If the objective value of this solution is better than the current incumbent, it can be stored in a list where CPLEX can reach whenever a user defined heuristic callback function is called (via `CPXXsetheuristiccallback()`). One of the solutions in the list is selected and compared to the incumbent at the time of the function call, and it is updated if necessary.

The alternating bound update mechanism in Section 5.3 requires to introduce a solve callback to the B&C algorithm via `CPXXsetsolvecallbackfunc()`. Every time a new node is selected for evaluation by CPLEX, the algorithm in Figure 5.1 is carried out for preprocessing. Note that the preprocessor of CPLEX is turned off during the B&C algorithm as it renders keeping track of the variables and subproblems during IC generation very difficult, as suggested in Fischetti *et al.* (2017b). Lastly, we set a user cut callback function via `CPXXsetusercutcallbackfunc()` to add locally valid follower upper bound cuts introduced in the same study, in order to benefit from the capabilities of the method as much as possible. As the name implies, these cuts put a lower bound on the follower objective value. It is achieved by solving a restricted follower problem which assumes the worst possible values of the leader variables within their current bounds.

# 6. MISINFORMATION SPREAD MINIMIZATION PROBLEM

In this chapter, we introduce another competitive influence spread problem which can find applications in various areas such as false news spread or spread of contagions. First, we formally define the problem and develop mathematical models under the assumptions on the rules of the diffusion process that will be explained. Next, we propose solution methods which integrate a state-of-the-art IMP technique for improving the time efficiency.[2]

## 6.1. Problem Definition and Mathematical Model

The problem we consider in this chapter is another Stackelberg game, where two players make decisions sequentially. The leader of the game, who is the first mover, wants to minimize the spread of negative influence (e.g., rumors, misinformation) on a social network by protecting (blocking) $h$ nodes in advance, while the follower aims its maximization by activating a set of $k$ nodes. The leader knows the follower's problem exactly, therefore, it becomes possible for the leader to incorporate the follower's optimization problem into his or her own problem. The protected nodes cannot be influenced at any step of the propagation process, nor can they influence any other nodes. In other words, they do not contribute to the spread of the influence at all. An example is a company or a political party which expects a misinformation diffusion (a negative influence spread) from the competitors. The party may target people who are critical for the diffusion of information to limit the spread. One strategy would be to initiate a positive spread starting from these individuals, which can be investigated with a competitive model as is the case in Budak *et al.* (2011) and He *et al.* (2012). In this chapter, however, we pursue the other strategy where the protected people agree that they do not adopt and spread the misinformation; in addition they do not spread the positive information either.

[2](Tanımmış *et al.*, 2020c) is based on the content of this chapter.

Another example that fits into the framework just described is the immunization process to prevent an epidemic. This is the case in disease spread models such as SIR (Susceptible-Infected-Removed) (Newman, 2002). Assuming that a vaccinated person neither gets infected nor transmits the disease to other people, vaccination is carried out as a precaution. If it is performed randomly, most of the individuals need to be vaccinated to prevent an epidemic. This, however, appears to be economically infeasible, and therefore a more effective way would be to select people for vaccination so that the spread of the disease is kept as low as possible. Note that the disease, which tries to spread as much as possible, corresponds to the follower in our model. Lastly, the protection of several computers in a network by means of anti-virus software against computer virus attacks is another example, where a protected computer does not spread the virus. This protection may prevent significant damage caused by malware in a computer network.

Let $G = (V, A)$ denote a directed graph representing a social network with node set $V$ and arc set $A$, and $G_X$ denote the subgraph obtained by removing from $G$ the node set $X \subset V$ and all arcs incident to the nodes in $X$. The expected number of influenced nodes, i.e. the spread, for a seed set $Y$ in graph $G$ under the diffusion model $\mathcal{M}$ is denoted by $\sigma_{\mathcal{M}}(G, Y)$. Using this notation we define the Misinformation Spread Minimization Problem (MSMP) as follows. Given a directed graph $G = (V, A)$, a diffusion model $\mathcal{M}$, and positive integers $h$ and $k$, the MSMP is the problem of finding a set of nodes $X$ of size $h$ such that $\sigma_{\mathcal{M}}(G_X, Y^*(X))$ is minimized, where $Y^*(X)$ is a set of $k$ nodes in $V \setminus X$ that will maximize the spread on $G_X$. The MSMP can thus be written as the following optimization problem:

$$\min_{\substack{X \subset V, \\ |X|=h}} \sigma_{\mathcal{M}}(G_X, Y^*(X)), \tag{6.1}$$

where

$$Y^*(X) = \arg\max_{\substack{Y \subset V \setminus X, \\ |Y|=k}} \sigma_{\mathcal{M}}(G_X, Y). \tag{6.2}$$

Stackelberg games can be modeled as bilevel mathematical programs due to their sequential structure. To this end, we assume that the influence spreads according to the widely accepted Linear Threshold (LT) model as we did for our first problem, IMPD, in Chapter 4. Before developing the formulation, we illustrate our problem in Figure 6.1 on a sample network of seven nodes and 10 arcs by assuming a deterministic LT model and choosing $h = k = 2$. The node thresholds are given next to the node names and the arc weights are shown next to each arc. We present two different leader solutions and the resulting spreads in the figures. The nodes with a dashed outline in Figure 6.1(a) and Figure 6.1(c) are the ones that are protected by the leader. Note that the protected nodes in the former are actually the optimal seed set of the IMP for that network and $k = 2$. We select this solution as the first example since it is quite intuitional to protect the most influential set of nodes. The nodes that are affected after the optimal seed selection of the follower for both leader solutions are shown with shaded circles in Figure 6.1(b) and Figure 6.1(d). The optimal set of activated nodes is $\{B, D\}$ in the former, and $\{A, G\}$ in the latter. The resulting number of affected nodes is four and two, respectively which clearly shows that protecting the optimal IMP seed may not be a good solution. Note that the second leader solution is an optimal protection decision for this example. Since the node thresholds are not known a priori and have a probabilistic nature in the LT model, the MSMP is a stochastic optimization problem that can be formulated as a stochastic bilevel mathematical program. After the parameters and decision variables are defined below, we provide the bilevel programming formulation.

*Sets and parameters:*

$V$ : set of all nodes in the network where $|V| = n$

$\theta_i$ : influence threshold of node $i \in V$

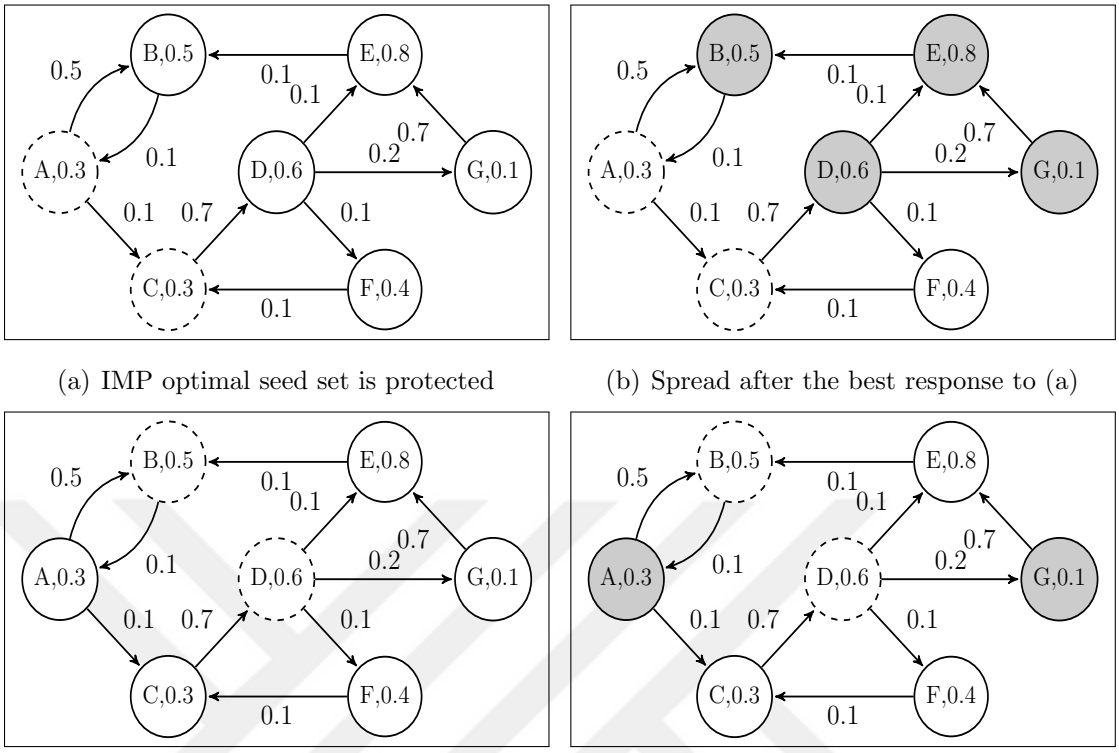
$h$ : number of nodes to protect

$k$ : number of nodes to activate

*Decision variables:*

$x_i$ : 1 if node $i$ is protected by the leader; 0 otherwise

$y_i$ : 1 if node $i$ is activated by the follower; 0 otherwise




(a) IMP optimal seed set is protected

(b) Spread after the best response to (a)

(c) MSMP optimal protection

(d) Spread after the best response to (c)

Figure 6.1. Comparison of two MSMP ULP solutions.

MSMP-S:

$$\min_{\mathbf{x}} \ \mathbb{E}_{\boldsymbol{\theta}}[g(\mathbf{x}, \mathbf{y}^*, \boldsymbol{\theta})] \tag{6.3}$$

s.t.

$$\mathbf{1}^T\mathbf{x} = h \tag{6.4}$$

$$\mathbf{x} \in \{0, 1\} \tag{6.5}$$

$$\mathbf{y}^* \in \arg\max_{\mathbf{y}} \left\{ \mathbb{E}_{\boldsymbol{\theta}}[g(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})] : \mathbf{1}^T\mathbf{y} = k, \mathbf{y} \leq \mathbf{1} - \mathbf{x}, \mathbf{y} \in \{0, 1\}^n \right\} \tag{6.6}$$

The term $g(\mathbf{x}, \mathbf{y}^\star, \boldsymbol{\theta})$ represents the number of influenced nodes when the protection decision $\mathbf{x}$, the follower's optimal response $\mathbf{y}^*$ to $\mathbf{x}$, and threshold scenario vector $\boldsymbol{\theta}$ are given. The expectation of this function over random thresholds is called the spread, and it is the objective function optimized in the opposite sense at both levels. The set in constraint (6.6) represents the rational reaction set of the follower. Note that there is no closed-form solution of the objective function. A possible approach to compute its value is developing a deterministic equivalent formulation using the

threshold scenarios or the live-arc scenarios, as is done for the IMPD. However, we see that using threshold scenarios may lead to an incorrect computation of the spread due to the loops formed among non-activated nodes in the network. This issue will be clarified after the presentation of the formulation with live-arc scenarios. The list of additional parameters and decision variables needed is provided below and followed by the bilevel formulation.

*Sets and Parameters:*

| | | |
|---|---|---|
| $R$ | : | set of all possible live-arc scenarios |
| $p_r$ | : | probability of scenario $r$ |
| $a_{ji}(\mathbf{x}, r)$ | : | 1 if node $i$ is reachable from node $j$ in scenario $r$, when the protection decision is $\mathbf{x}$; 0 otherwise |

*Decision variable:*

| | | |
|---|---|---|
| $u_{ir}$ | : | 1 if node $i$ is influenced in scenario $r$; 0 otherwise |

MSMP:

$$z^* = \min_{\mathbf{x}} z(\mathbf{x}) \tag{6.7}$$

s.t.

$$\sum_{i \in V} x_i = h \tag{6.8}$$

$$x_i \in \{0, 1\} \qquad i \in V \tag{6.9}$$

where

$$z(\mathbf{x}) = \max_{\mathbf{y},\mathbf{u}} \sum_{r \in R} \sum_{i \in V} p_r u_{ir} \tag{6.10}$$

s.t.

$$\sum_{i \in V} y_i = k \tag{6.11}$$

$$y_i \leq 1 - x_i \qquad i \in V \tag{6.12}$$

$$u_{ir} \leq 1 - x_i \qquad i \in V, r \in R \tag{6.13}$$

$$u_{ir} \leq \sum_{j \in V} a_{ji}(\mathbf{x}, r) y_j \qquad i \in V, r \in R \tag{6.14}$$

$$u_{ir} \geq 0, y_i \in \{0, 1\} \qquad i \in V, r \in R \tag{6.15}$$

In the upper level, the leader decides the set of protected nodes of size $h$ in Constraint (6.8). The decision variable $\mathbf{x}$ of the leader are forced to take binary values by Constraint (6.9). Then, the follower activates $k$ nodes to start the diffusion in constraint (6.11). A protected node can be neither activated by constraint (6.12) nor influenced in any scenario by constraint (6.13). Constraint (6.14) relates the activation decisions to the influence indicating variables. To generate these constraints we need to obtain the reachability information of all nodes from every other node in each live-arc scenario. Besides, the reachability of node $i$ from $j$ depends on the protection decision. If there is a protected node on a live-arc path from $j$ to $i$, then $i$ cannot be reached from $j$ via that particular path. Obviously, it can still be reached using other available paths. In other words, even though the live-arc scenarios are independent from the protection decisions, we need to take the protection decision into account while generating Constraint (6.14) for each scenario. If $i$ is not reachable from any of the seed nodes via the live-arc paths in that scenario under the current protection decision, then the right-hand side (RHS) of Constraint (6.14) becomes zero, which forces $u_{ir}$ to be zero. If it can be reached from at least one of the activated nodes, $u_{ir}$ is set to one in an optimal solution due to the maximization objective. Please note that $u_{ir}$ is bounded from above at value one by Constraint (6.13), and the RHS of Constraint (6.14) is always integer. Therefore, unlike variable $\mathbf{y}$ restricting variable $\mathbf{u}$ as binary is not necessary.

Now, we can illustrate why using threshold scenarios does not work correctly. A threshold based formulation can be obtained by replacing Constraint (6.14) with the inequality

$$u_{ir} \leq \sum_{j \in V} w_{ji} u_{jr} - \theta_{ir} + y_i + 1, \tag{6.16}$$

whose RHS becomes greater than one if the total incoming weight exceeds the threshold, where $w_{ji}$ is the weight of the arc $(j, i)$. Otherwise the RHS becomes less than one which forces $u_{ir}$ to be zero assuming that we have binary restrictions on these variables. Consider the network in Figure 6.2, where $w_{ij} = 1$ for all arcs. Assume that there is a single threshold scenario with $\theta_i = 0.5, \forall i$, there is no protection ($h = 0$), and $k = 1$. When node 1 is selected as a seed node, i.e., $y_1 = 1$, $y_2 = y_3 = y_4 = 0$, Constraint (6.16) yields $u_1 \leq 1.5$, $u_2 \leq u_1 + 0.5$, $u_3 \leq u_4 + 0.5$, $u_4 \leq u_3 + 0.5$. Since the objective is to maximize $\sum_i u_i$, the resulting solution becomes $u_1 = u_2 = u_3 = u_4 = 1$ even though the nodes 3 and 4 cannot be reached from the seed. To summarize, loops that are formed among non-activated nodes increase the influence spread in an inappropriate way due to the maximization objective in the LLP. Therefore, we will use the MSMP formulation in (6.7)–(6.15) in the rest of this chapter.

Figure 6.2. A sample directed graph with unit arc weights.

## 6.2. Solution Methods

The follower's problem in the MSMP formulation in (6.7)–(6.15) is the IMP on a subgraph of the original graph $G = (V, A)$, which is obtained by removing a set of nodes and the arcs incident to them, as defined in (6.2). Since we know that the IMP is $\mathcal{NP}$-hard, the MSMP is $\mathcal{NP}$-hard as it involves the IMP as a constraint in its formulation. Besides, the MSMP can be reduced to the IMP by setting the number of protected nodes $h$ to zero.

In this section, we develop algorithms for the solution of the MSMP and adapt some of the methods proposed for the IMPD in Section 4.2. First, we describe how to implement the Sample Average Approximation method on the LLP of the MSMP. Next, the adaptation of the Tabu Search based matheuristic is explained. Then, we propose a greedy heuristic which starts with an empty set and adds at each iteration

one node to the set until the required protection seed size is obtained, and a Binary Search algorithm motivated from a competitive facility location problem. For all of the mentioned methods, it is possible to integrate a heuristic algorithm for the solution of the LLP in order to reduce the time cost of the SAA method. We describe in the relevant sections how to integrate such an algorithm to the previously mentioned ULP solution methods.

### 6.2.1. Sample Average Approximation Method for the Follower's Problem

The number of live-arc scenarios increases exponentially in the number and degrees of nodes. This leads to an excessively large scenario space even for very small sized networks. Therefore, the LLP of the MSMP needs to be approximated by an appropriate method. Güney (2017) solves the IMP using Sample Average Approximation (SAA) method, and the results reveal that it yields approximately optimal solutions with very small optimality gap estimates. Therefore, we proceed with the SAA method and modify the algorithm presented in Section 4.2.1 for adapting it to the new formulation.

Suppose we are given a leader's solution $\mathbf{x}$ and $M$ scenario samples (realizations) each of size $N$. Then, the problem we need to solve is (6.10)–(6.15) except the objective function (6.10) is replaced with $\max_{\mathbf{y},\mathbf{u}} \frac{1}{N} \sum_{r=1}^{N} \sum_{i \in V} u_{ir}$ and $R$ is replaced with a sample of size $N$. Using an MILP solver, this problem can be solved for each one of the $M$ samples. The average of the optimal objective values of these problems denoted as $\bar{z}_N^M$ is an upper bound estimate on the true optimal objective value $z(\mathbf{x})$. The optimal values of the first-stage decision variables in these problems $(\tilde{\mathbf{y}}^m, m \in M)$ are candidates for the best response of the follower. To determine the best candidate, the objective value (spread) is estimated for each candidate using a scenario sample of size $N' \gg N$, and simulating the diffusion process. We denote the estimated objective values by $\hat{z}_{N'}(\mathbf{x}, \tilde{\mathbf{y}}^m)$, then the best candidate is determined as $\tilde{\mathbf{y}}^* = \arg\max_{\tilde{\mathbf{y}}^m} \hat{z}_{N'}(\mathbf{x}, \tilde{\mathbf{y}}^m)$. The last step is to re-estimate the objective value of the best candidate using an independent sample of size $N'' \gg N$. This quantity, denoted by $\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*)$, is again obtained

by numerical simulation and provides a lower bound estimate on the optimal objective value. It is actually the main output of the SAA method, which constitutes an approximation of $z(\mathbf{x})$. It is denoted simply as $\hat{z}_{SAA}(\mathbf{x})$ in the sequel. The optimality gap can be estimated using the upper and lower bound estimates by means of the expression $\bar{z}_N^M - \hat{z}_{SAA}(\mathbf{x})$ although it is an overestimation. Since the optimal objective value of the follower's problem is approximated via SAA, the estimated objective function of the leader's problem is expressed as

$$\hat{z} = \min_{\mathbf{x} \in \mathbb{X}} \hat{z}_{SAA}(\mathbf{x}), \tag{6.17}$$

where $\mathbb{X} = \{\mathbf{x} : \mathbf{1}^T\mathbf{x} = h, \mathbf{x} \in \{0,1\}\}$. The objective function $z(\mathbf{x})$ and its approximation $\hat{z}_{SAA}(\mathbf{x})$ can equivalently be expressed as the set functions $z(S)$ and $\hat{z}_{SAA}(S)$ where $S = \{i \in V : x_i = 1\}$ is the set of protected nodes. We use these notations interchangeably in the following sections. For very small problem instances, the optimal objetive value can be computed by complete enumeration of feasible leader solutions.

### 6.2.2. Tabu Search Based Matheuristic

The first method we propose to solve the MSMP is a metaheuristic method implemented on the leader's problem. The solution space of this problem is searched via Tabu Search method with a candidate list strategy. During this search, each solution visited is evaluated by implementing the SAA algorithm mentioned above on the follower's problem, i.e., the optimal response of the follower is estimated, and the resulting spread is approximated. Although we basically adopt the algorithm proposed for the IMPD in Section 4.2.3.2, the Tabu Search based matheuristic (TSM) developed here is more efficient in the sense that less SAA calls are carried out by making use of the similarity between the follower's problem and the IMP.

As before, TSM considers only feasible and unprecedented leader solutions due to the excessive computational time of implementing SAA for each solution generated. In order to avoid visiting a solution repeatedly, an explicit memory structure is used.

Besides, a candidate list strategy based on estimating the "promising neighbors" of a solution is determined to decrease the computational burden as is the case in TSM for the IMPD although the selection criterion is different here. The rationale of the selection of candidates is given in the sequel when the neighborhood structure is described in a detailed manner. The fraction of the number of evaluated neighbors to the neighborhood size is denoted as $\nu$.

*Heuristic Integration:* A standard procedure for implementing Tabu Search is to compute the estimated objective value for all (promising) neighboring solutions and select the one with the smallest objective value. However, there exists an alternative and more efficient approach, which is to compute the objective value using a fast heuristic and execute SAA only when it is necessary. We start with the calculation of the objective value $z_1$, i.e., the spread, of the first promising neighbor in the list via SAA. For the next neighbor, the follower's response is estimated with a heuristic method, and the resulting objective value is denoted by $\hat{z}_2$. If $\hat{z}_2 \geq z_1$, then there is no need to find the optimal value $z_2$ since $z_2 \geq \hat{z}_2$ because the follower's problem is maximization, and we want to obtain solutions with smaller values since the leader's problem is minimization. If $\hat{z}_2 < z_1$, then there is a likelihood that the second neighbor is better (smaller) than the first one as the objective value $z_2$ ($\geq \hat{z}_2$) associated with the optimal response of the follower may remain less than $z_1$. Thus, SAA is executed for the second neighbor. To summarize, for any promising neighbor in the candidate list, first a heuristic algorithm is executed, and depending on the comparison with the best neighbor's objective value SAA is called. Each time the SAA algorithm is executed, the best neighbor is updated if necessary.

It is important to reduce the number of SAA calls as much as possible to avoid solving time consuming MILPs. Therefore, the heuristic method used to solve the follower's problem is supposed to provide good solutions. Recall that the follower's problem is IMP on a subgraph of the original one. As a result, one of the state-of-the-art approximate solution techniques developed for the IMP can be utilized to solve the follower's problem after the protected nodes are removed from the network. We

select a recent algorithm referred to as IMM (Tang *et al.*, 2015), which is a two-phase algorithm based on martingales. While it has the same approximation guarantee as the classical greedy heuristic method by Kempe *et al.* (2003) and its improved version by Leskovec *et al.* (2007), it is much faster compared to these two methods. The IMM is scalable to networks with millions of nodes, therefore its running time is quite small for networks with a few hundred nodes that we intend to consider. Unfortunately, its output consists of only a seed set as in most of the alternative methods, and the resulting spread value must still be estimated via Monte Carlo simulations. Below, we provide the remaining implementation details of TSM.

*Neighborhood Structure:* The move operator used in TSM is 1-Swap. Let $S$ denote the current solution consisting of the set of protected nodes. A 1-Swap neighbor $S'$ of $S$ is obtained by exchanging nodes $i \in S$ and $j \in V \setminus S$, namely $S' = (S \setminus \{i\}) \cup \{j\}$. Note that the neighborhood size is $|V \setminus S||S|$. Since the leader's problem is cardinality-based implying that the number of protected nodes is fixed, the swap move operator always yields feasible solutions. Once all the neighbors are enumerated, the ones that were evaluated before are discarded. The next step is to assign a score to each neighbor $S'$ using the shortest path distances that are computed in the pre-processing step by defining the length of arc $(i, j)$ as $-\log(w_{ij})$. The rationale of using a score function is that if a node is difficult to reach from an optimal seed of the follower for a given feasible protection decision in the leader's problem, then it is highly unlikely that it needs to be protected. Protecting the nodes that are in that optimal seed or the ones that are easily reachable from them are more promising for the leader. In a sense, the leader learns the influential nodes via observing the last reaction of the follower. Therefore, the score function takes $P^*(S)$, the optimal follower seed set for the leader solution $S$, as input. Let $\lambda_{ij}$ denote the shortest path distance from node $i$ to node $j$. The score of the neighbor $S' = (S \setminus \{i\}) \cup \{j\}$ is denoted by $f(S')$, and its value is the minimum distance from set $P^*(S)$ to node $j$, i.e., $\min_{v \in P^*(S)} \lambda_{vj}$. Note that a smaller score indicates a more promising neighbor. Once the neighboring solutions are sorted in non-decreasing order with respect to their scores, the top $\nu|V \setminus S||S|$ solutions are evaluated.

*Tabu Structure:* As mentioned above, visiting a solution more than once is not allowed. Therefore all solutions evaluated before by either IMM or SAA are declared tabu, and added to the tabu list as an integer by means of a hash function.

*Termination Criterion:* A time limit parameter $t_{\max}$ is determined a priori. Moreover, if there is no new solution in the neighborhood at any iteration, then the algorithm terminates.

The pseudo-code of TSM is provided in Figure 6.3, which is followed by the pseudo-code of the 1-Swap function in Figure 6.4. Parameters $\epsilon$ and $l$ are those of the original IMM algorithm, and they are used to determine the size of the scenario set to be generated while achieving a theoretical approximation ratio of $(1 - 1/e - \epsilon)$ with probability $1 - 1/n^l$, where $n$ is the number of nodes. IMM$(S, \epsilon, l)$ function returns the expected spread of the seed obtained by the IMM algorithm when nodes in $S$ are protected and $\hat{z}_{IMM}(S)$ holds this value. Note that the expected spread is estimated over 10,000 Monte Carlo simulations. We refer the reader to the paper by Tang *et al.* (2015) for the details of IMM.

**Algorithm 6.3** TSM

```
Generate the initial solution S, and set S* ← S, ẑ* ← ẑ_SAA(S)
Choose ν, ϵ, l
while CPU time ≤ t_max do
    Set (S', ẑ_SAA(S')) ← 1-Swap(S, ν, ϵ, l)
    if S' = ∅ then
        break
    end if
    if ẑ_SAA(S') > ẑ* then
        Update the incumbent solution S* ← S', ẑ* ← ẑ_SAA(S')
    end if
end while
Return S* and ẑ*
```

Figure 6.3. Pseudo-code of Tabu Search based matheuristic (TSM) for the MSMP.

**Algorithm 6.4** 1-Swap$(S, \nu, \epsilon, l)$

Generate the set $N(S) = \{S' : S'$ is a neighbor of $S$ and was not generated before$\}$
Set $count = 0$, $maxCount = \nu \cdot |S| \cdot |V \setminus S|$, $z_{Best} \leftarrow \infty$, and $S^* \leftarrow \emptyset$
For each $S' \in N(S)$ calculate the score $f(S')$
Sort the neighbors in $N(S)$ in non-increasing order of scores
**while** $N(S) \neq \emptyset$ and $count < maxCount$ **do**
  Pick the first element $S' \in N(S)$ and set $\hat{z}_{IMM}(S') \leftarrow$ IMM$(S', \epsilon, l)$
  **if** $\hat{z}_{IMM}(S') < z_{Best}$ **then**
    Compute true objective $\hat{z}_{SAA}(S')$
    **if** $\hat{z}_{SAA}(S') < z_{Best}$ **then**
      Update the best neighbor $z_{Best} \leftarrow \hat{z}_{SAA}(S')$ and $S^* \leftarrow S'$
    **end if**
  **end if**
  Remove $S'$ from $N(S)$
**end while**
Return $S^*$ and $z_{Best}$

Figure 6.4. Pseudo-code of 1-Swap operator of TSM.

### 6.2.3. Greedy Heuristic for the Protection Decision

A constructive approach to find a set of nodes to protect is obtained by adding nodes to the protection set in a greedy fashion. Starting with an empty set $S$, the next node to protect is determined on the basis of its marginal contribution to the objective value. The objective function $z(S)$ is non-increasing in $S$, and hence at each iteration the node whose protection decreases the objective value the most is added to the current set until the desired set size $h$ is reached. The amount of decrease in the objective value, $z(S) - z(S \cup \{i\})$, can be approximated by $\hat{z}_{SAA}(S) - \hat{z}_{SAA}(S \cup \{i\})$ as before. Executing the SAA method for all $i \in V \setminus S$ for $h$ iterations results in $\sum_{i=0}^{h-1}(n-i)$ SAA calls in total. Again, this number can be reduced by estimating the follower's reaction using the IMM algorithm. SAA is implemented only when it is necessary during the greedy algorithm referred to as the Greedy Protection Heuristic (GPH) whose pseudo-code is given in Figure 6.5. In this algorithm, after each execution of the SAA method, the incumbent value is updated if necessary. The statement in line 4 handles the case where protecting none of the candidate nodes improves the incumbent, by selecting an

arbitrary node. The overall procedure terminates in $h$ main (outer) iterations.

**Algorithm 6.5** GPH

```
Set the initial solution S ← ∅, choose ϵ and l
Set z_Best ← ẑ_SAA(S), S* ← ∅
while |S| < h do
    Select any i ∈ V \ S, set S* ← S ∪ {i}
    for each node i ∈ V \ S do
        Set S' ← S ∪ {i}, compute ẑ_IMM(S') ← IMM(S', ϵ, l)
        if ẑ_IMM(S') < z_Best then
            Compute true objective ẑ_SAA(S')
            if ẑ_SAA(S') < z_Best then
                Update the best candidate S* ← S' and z_Best ← ẑ_SAA(S')
            end if
        end if
    end for
    Update the protected set S ← S*
end while
Return S and z_Best
```

Figure 6.5. Pseudo-code of the Greedy Protection heuristic (GPH).

The greedy algorithm proposed by Kempe *et al.* (2003) to solve the IMP provides a $(1 - 1/e - \epsilon)$ approximation since the objective function is maximizing a monotone submodular function (Nemhauser *et al.*, 1978). Therefore, a relevant and interesting question is whether the objective function $z(\cdot)$ is submodular or not. Recall that a submodular function $f$ satisfies $f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B)$, $i \in V$ and $A \subset B \subset V$. Since $z(\cdot)$ is a non-increasing set function, the quantities on both sides of the inequality are non-positive and $z(\cdot)$ is submodular if the following inequality holds for all $A, B$ satisfying $A \subset B \subset V$ and $i \in V \setminus B$. Hence, in our case

$$z(A) - z(A \cup \{i\}) \leq z(B) - z(B \cup \{i\}) \tag{6.18}$$

should hold for the submodularity of $z(x)$. This inequality implies that adding node $i$ to the protection set $A$ makes a smaller marginal contribution than adding it to the set $B$. However, the opposite explanation is more intuitive. If the reverse inequality is

always true, then $z(\cdot)$ is called *supermodular*. To investigate both claims, we consider the network instance displayed in Figure 6.6. For the sake of simplicity all arc weights are assumed to be one. Thus, each arc is determined as live with probability one, i.e., there is a single live-arc scenario which is identical to the original network. Now, consider the sets $A = \{1\}$, $B = \{1, 5\}$, and the nodes $i = 2$, $j = 6$. For $k = 1$, the follower would activate node 5 when nodes in $A$ are protected and node 2 when nodes in $B$ are protected. The resulting objective values are $z(A) = 4$ and $z(B) = 3$, respectively. Similarly, the objective values $z(A \cup \{i\}) = 4$, $z(B \cup \{i\}) = 2$, $z(A \cup \{j\}) = 3$ and $z(B \cup \{j\}) = 3$ can be found by inspection. Both $z(A) - z(A \cup \{i\}) \leq z(B) - z(B \cup \{i\})$ and $z(A) - z(A \cup \{j\}) \geq z(B) - z(B \cup \{j\})$ hold true, which shows that neither the inequality (6.18) nor its reverse is true for all $A, B \subset V$ and $i \in V \setminus B$. Since submodularity and supermodularity conditions are violated on a specific problem instance, we conclude that the objective function $z(\cdot)$ is neither submodular nor supermodular in general. Consequently, GPH for the MSMP does not provide an approximation guarantee. Nevertheless, it can still be used as a heuristic solution method.

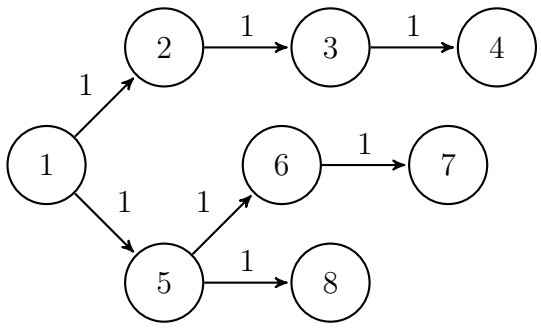



Figure 6.6. A small network with unit arc weights.

### 6.2.4. Binary Search Algorithm

Network interdiction models such as the Min-max Flow Problem in Wood (1993) and the Shortest Path Interdiction Problem in Israeli and Wood (2002) help determining the critical components of a supply/service system. The $r$-interdiction Median Problem with Fortification (RIMF) involves finding $q$ facilities to protect such that the service cost after the damage due to an intelligent opponent who will attack $r$ of the

unprotected facilities is minimized (Church and Scaparra, 2007). Scaparra and Church (2008) propose an implicit enumeration algorithm to solve the RIMF. In this algorithm, at the root of the enumeration tree, the lower level interdiction problem, which is the $r$-intediction Median (RIM) problem, is solved assuming none of the facilities is protected. The optimal interdiction decision of this problem provides an important information which is the basis of the algorithm. As quoted from Scaparra and Church (2008) "Let $I$ be the set of $r$ interdictions in the optimal solution to the lower-level RIM problem without fortification. Then the optimal set of $q$ fortifications selected by the leader must include at least one of the $r$ facilities in $I$". The reason is that if none of the facilities in $I$ is protected, the follower can always achieve the maximum disruption by targeting the facilities in $I$. Thus, the facilities in $I$ become the candidate variables to protect. By branching on one of these candidates, two new nodes are obtained and added to the tree. Suppose that facility $i$ is the current branching variable. Then two new nodes are obtained, one by fixing the fortification variable $x_i$ to 1 and the other by fixing it to 0. Then, each node is processed according to the rules that will be explained below. The algorithm terminates when all nodes are fathomed.

Although the MSMP is a stochastic problem as opposed to the RIMF, the main idea of this search algorithm can be adopted to solve the MSMP due to the similarity between the protection decisions in the MSMP and fortification in the RIMF. However, the fundamental observation about how to disrupt an optimal solution is not exactly true in our case. Let $I$ be the set of nodes that are activated by the follower in case of no protection, i.e, the optimal solution of the IMP. According to their observation, if none of the nodes in $I$ is protected, then the follower could still achieve the maximum spread. However, the expected spread (6.10) is not a function of only the seed nodes. Protecting some $i \notin I$ may significantly decrease the spread. It may even decrease the spread more than protecting a seed node as we have observed in some preliminary tests. Therefore we update the fundamental observation as follows:

*Observation:* Let $I$ denote the set of nodes that are activated by the follower when none of the nodes is protected by the leader. Then, at least one of the nodes that are

reachable from set $I$ must be protected in an optimal leader strategy.

Now, the Binary Search algorithm (BSA) can be developed. To this end, in the remainder of this section a *node* indicates a node of the enumeration tree, while a node of the MSMP problem instance is referred to as a *network node*. Each node $v$ of the enumeration tree has the following parameters. $v.\mathbf{x}$ is the protection strategy of node $v$. It is obtained by fixing $x_i$ to 1 for all $i$ that are protected on the path from the root to node $v$, and the remaining to 0. $v.size$ is the number of "protect" decisions on that path. $v.can$ denotes the list of candidate branching variables of node $v$. The algorithm starts by solving the IMP without protection at the root node. Then all network nodes that are reachable from the optimal seed are added to the candidate list. Once a candidate variable $x_i$ is picked, it is removed from the candidate list $v.can$. Two nodes that are identical to $v$ are generated and then $x_i$ is fixed to one in one of them, i.e., it is generated with a new "protect" decision, and to zero in the other. Both nodes are added to the queue. At each iteration of the algorithm the first node $v$ in the queue is picked and removed from the queue. If $v$ is obtained with a new "protect" decision, then the following steps are carried out: The SAA method is used to decide the optimal activation decision. If the number of "protect" decisions on the path from the root to $v$ is equal to $h$, $v.\mathbf{x}$ is a candidate leader solution and $v$ is called a *leaf node*. In this case, the incumbent objective is updated if necessary, then the node is fathomed. For a node which is obtained by a "do not protect" decision, the process is different. Since the protection decision $v.\mathbf{x}$ is exactly the same as its parent node, the SAA method is not needed. Its candidate list is checked. If it is empty, then the node is fathomed, otherwise a branching variable is picked and two new nodes are obtained as usual and added to the queue.

Please note that this search algorithm does not allow to fathom any node by bound since its objective value can be improved by branching further (if it is not a leaf node), whether it is better than the current incumbent or not. Consequently, order of processing does not matter in this algorithm since it does not terminate until all nodes are processed. As the value of one variable is fixed to generate a new tree node,

the maximum depth of the tree is $n$, namely the number of nodes in the network. In addition, the network nodes with zero out-degree are not included in any candidate list with the assumption that $h$ is sufficiently smaller than $n$.

The preliminary results showed us that the resulting algorithm is not efficient since a large portion of the nodes in the network are reachable from the other ones causing to a large number of branching variables. The total number of nodes requiring SAA execution can exceed the number of SAA executions carried during classical complete enumeration method in the worst case (in complete enumeration only the solutions with $h$ protected nodes are evaluated). If we leave finding an optimal solution aside, good solutions can be obtained by making an adjustment in the candidate selection process. Instead of labeling all nodes that are reachable from the current seed nodes as candidates, we introduce an elimination rule. The decision of which successors are more promising is made based on the weights of the arcs directed from the seed nodes to their successors. If the weight $w_{ij}$ on arc $(i, j)$ is less than some threshold $\beta$, then successor $j$ is not listed as candidate. Two extreme cases with $\beta = 0$ and $\beta = 1$ results in listing all successors as candidates and listing only the ones that are certainly affected by the seed (in all scenarios), respectively. The current seed nodes are in the candidate list for any value of $\beta$. Note that eliminating a successor $j$ of the seed node $i$ with $w_{ij} = 1$ is never allowed because such a node acts like a seed node in practice.

Another approach that will possibly improve the algorithm is reducing the number of SAA calls. Recall that SAA algorithm is called every time a "protect" decision is made. The IMM algorithm can be integrated into BSA as well, but not with the same structure used in the TSM. Here, the objective values are needed only in the leaf nodes where the number of protected nodes is exactly $h$. Thus, the SAA is called at only leaf nodes. In all other tree nodes which are obtained with a "protect" decision, the seed set of the follower is estimated heuristically, i.e., using the IMM algorithm. The algorithm is provided in Figure 6.7. Note that $CLIST(v.\mathbf{x}, \beta)$ is a procedure that finds a seed set via IMM algorithm and then determines the candidate branching variables based on this seed set and the value of the threshold $\beta$.

**Algorithm 6.7** BSA

1: Create root node $v_0$ with $\mathbf{x} = \mathbf{0}$ (no protection), choose $\beta$
2: Initialize the queue, $Q \leftarrow \{v_0\}$, $z_{Best} \leftarrow n$
3: **while** $Q$ is not empty **do**
4: Pick the first node $v \in Q$ and set $Q \leftarrow Q \setminus \{v\}$
5: **if** $v$ is generated with a new "protection" decision **then**
6: **if** $v.size = h$ **then**
7: Compute the objective, $z \leftarrow \hat{z}_{SAA}(v.\mathbf{x})$
8: **if** $z < z_{Best}$ **then**
9: Update $z_{Best} \leftarrow z$ and $\mathbf{x}^* \leftarrow v.\mathbf{x}$
10: **end if**
11: $v$ is a leaf node, FATHOM $v$ and go to line 3.
12: **else**
13: Obtain $v.can \leftarrow CLIST(v.\mathbf{x}, \beta)$
14: **end if**
15: **end if**
16: **if** $v.can$ is empty **then**
17: FATHOM $v$ and Continue
18: **end if**
19: Pick a branching variable $j \in v.can$, set $v.can \leftarrow v.can \setminus \{j\}$
20: Create two nodes, $v_1 \leftarrow v$, $v_2 \leftarrow v$
21: Set $v_1.x_j = 0$, $v_2.x_j = 1$ and $v_2.size \leftarrow v_2.size + 1$, add them to the end of $Q$
22: **end while**
23: Return $\mathbf{x}^*$ and $z_{Best}$

Figure 6.7. Pseudo-code of the Binary Search algorithm (BSA).

# 7. IMPROVED X-SPACE ALGORITHM FOR MIN-MAX BILEVEL PROBLEMS

In bilevel interdiction problems, the aim of the leader is to damage the objective of the follower by disabling some of the lower level decisions. We consider an existing exact algorithm proposed for min-max bilevel interdiction problems. This algorithm solves upper bound and lower bound generating problems until convergence and requires obtaining the dual of an approximation of the follower's problem to develop a formulation of the lower bound problem. We modify the algorithm by reformulating the lower bound problem using the properties of an optimal solution to the original formulation so that dualization is not needed any more. Following the reformulation, a greedy covering heuristic is integrated to the solution scheme to reduce the solution time. The new algorithm is implemented on the MSMP introduced in Chapter 6 after making some necessary modifications in its formulation.[3]

## 7.1. Improved $x$-space Algorithm

Tang *et al.* (2016) take into consideration problems that can be formulated as

$$Z^*(\mathcal{X}, \mathcal{F}) = \min_{\mathbf{x} \in \mathcal{X}} \max_{\mathbf{y} \in \mathcal{Y}} \left\{ \mathbf{p}^T \mathbf{y} : (\mathbf{x}, \mathbf{y}) \in \mathcal{F} \right\} \tag{7.1}$$

where $\mathbf{x} \in \mathbb{Z}^{n_1}$ denotes the upper level decision variables and $\mathbf{y} = (\mathbf{y_1}, \mathbf{y_2})$ with $\mathbf{y_1} \in \mathbb{R}^{n_2 - q}$ and $\mathbf{y_2} \in \mathbb{Z}^q$ denoting the lower level decision variables. The set $\mathcal{F}$ is defined as

$$\mathcal{F} = \{(\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y} : \mathbf{y} \leq \mathbf{d} - \mathbf{D}\mathbf{x}\}. \tag{7.2}$$

Here, $\mathcal{X}$ denotes the feasible region of the upper level problem while $\mathcal{Y}$ and $\mathbf{y} \leq \mathbf{d} - \mathbf{D}\mathbf{x}$ constitute the feasible region of the lower level problem. Let $\mathcal{X}^\circ = \mathcal{X} \cap \{0,1\}^{n_1}$, $\mathcal{Y}^\circ = \mathcal{Y} \cap \{0,1\}^{n_2}$ and $\mathcal{F}^\circ = \mathcal{F} \cap \{0,1\}^{n_1+n_2}$. It is shown that $Z^*(\mathcal{X}^\circ, \mathcal{F}^\circ) = Z^*(\mathcal{X}^\circ, \widetilde{\mathcal{F}^\circ})$

[3](Tanınmış *et al.*, 2020b) is based on the content of this chapter.

where $\widetilde{\mathcal{F}^\circ}$ is defined as

$$\widetilde{\mathcal{F}^\circ} = \{\mathbf{x} \in \mathbb{R}^{n_1}, \mathbf{y} \in \mathbb{R}^{n_2} : \mathbf{x} \in \mathcal{X}^\circ, \mathbf{y} \in \text{conv}(\mathcal{Y}^\circ), \mathbf{y} \leq \mathbf{d} - \mathbf{D}\mathbf{x}\} \tag{7.3}$$

(Tang *et al.*, 2016, p. 242). Notice that the follower's problem is a linear program when the feasible region is defined by $\widetilde{\mathcal{F}^\circ}$, in which case it is possible to reformulate the bilevel problem as a single-level model. However, it may be very difficult to obtain the convex hull. Therefore, Tang *et al.* (2016) propose an algorithm that involves sampling from the solution space of the lower level problem, which they call the $x$-space, and obtaining an (inner) approximation of the conv($\mathcal{Y}^\circ$) iteratively. Besides, they show that this approximation leads to a lower bound on the optimal objective value of the bilevel problem. The description of the algorithm has been presented previously in Section 2.4.

We will stick to the name "$x$-space" throughout this chapter although in our notation $\mathbf{x}$ and $\mathbf{y}$ denote the upper level and lower level variables, respectively. The original $x$-space (XS) algorithm, which requires the following assumptions of the bilevel problem, is given in Appendix C. It basically involves solving the lower bound $\text{LB}_q$ and upper bound problems $\text{UB}_q$ successively.

(i) The feasible region $\mathcal{X}^\circ$ of the leader's problem is not empty.
(ii) The follower's problem is feasible for each decision $\mathbf{x} \in \mathcal{X}^\circ$ of the leader.
(iii) The follower's problem is bounded above for at least one decision of the leader.
(iv) The trivial follower solution $\mathbf{y} = \mathbf{0}$ is feasible for each leader decision $\mathbf{x} \in \mathcal{X}^\circ$.

The upper bound problem $\text{UB}_q$ in the XS algorithm is simply the follower's problem that is defined using the most recently obtained leader solution $\mathbf{x}^q$:

$$z_q = \max_{\mathbf{y}} \{\mathbf{p}^T\mathbf{y} : (\mathbf{x}^q, \mathbf{y}) \in \mathcal{F}^\circ\} \geq z^*. \tag{7.4}$$

The lower bound problem $\text{LB}_q$

$$
\begin{aligned}
t_q &= \min_{\mathbf{x}\in\mathcal{X}^\circ} \max_{\mathbf{y}} \{\mathbf{p}^T\mathbf{y} : \mathbf{y} \in \text{conv}(S_q), \mathbf{y} \leq \mathbf{d} - \mathbf{D}\mathbf{x}\} \\
&= \min_{\mathbf{x}\in\mathcal{X}^\circ} \max_{\lambda,\mathbf{y}} \left\{ \mathbf{p}^T\mathbf{y} : \sum_{\tau\in J} \lambda_\tau = 1, \mathbf{y} - \sum_{\tau\in J} \lambda_\tau \mathbf{y}^\tau = \mathbf{0}, \mathbf{y} \leq \mathbf{d} - \mathbf{D}\mathbf{x}, \mathbf{y} \geq \mathbf{0},\ \lambda_\tau \geq 0,\ \tau \in J \right\}
\end{aligned}
\tag{7.5}
$$

is basically the bilevel problem itself, except that $\mathbf{y} \in \text{conv}(\mathcal{Y}^\circ)$ is replaced by $\mathbf{y} \in \text{conv}(S_q)$, where $S_q$ is the follower solution set at step $q$. The set $J$ contains the indices of solutions in $S_q$ and the new decision variable $\lambda$ is used to express the follower solution as a convex combination of the solutions in $S_q$.

Tang *et al.* (2016) solve the LB problem, which is a bilevel problem with an LP in the lower level, in the following manner: first, the dual formulation of the lower level problem is developed by treating $\mathbf{x}$ as a parameter. Then, the constraints in $\mathcal{X}^\circ$ are integrated into this formulation to end up with a non-linear model since $\mathbf{x}$ is actually a decision variable. After removing the nonlinearities by defining new decision variables used for linearization, the final single-level mixed-integer linear program (MILP) is solved by a commercial solver. We propose a more efficient procedure for the solution of the LB problem.

### 7.1.1. A New Lower Bound Problem Formulation

In order to describe the procedure to develop an alternative LB problem formulation, we restrict our attention without loss of generality to the case where $\mathbf{d} = \mathbf{1}$, $\mathbf{D} = \mathbf{I}$ and $n_1 = n_2 = n$. It will be clear in Section 7.2 that the method can be adapted to problems with different structures as long as some or all of the follower's variables have an interdiction relationship with a leader's decision variable. Now, let $N = \{1, ..., n\}$ denote the set of indices for both upper and lower level variables. We firstly focus on the inner optimization problem of $\text{LB}_q$ in (7.5) which we denote by $\text{LB}_q(\mathbf{x})$. Notice that $\mathbf{x}$ is not a decision variable but a parameter in $\text{LB}_q(\mathbf{x})$.

**Proposition 7.1.** *Given* $\mathbf{x} \in \mathcal{X}^\circ$ *and* $S_q$*, define the following set:*

$$\begin{aligned} B(\mathbf{x}, S_q) &= \{\tau \in J : i \in N, y_i^\tau > 1 - x_i\} \\ &= \{\tau \in J : \exists i \in N, x_i = 1, y_i^\tau = 1\}. \end{aligned} \tag{7.6}$$

*Let* $(\lambda, \mathbf{y})$ *be a feasible solution to* $LB_q(\mathbf{x})$*. Then* $\lambda_\tau = 0$ *for all* $\tau \in B(\mathbf{x}, S_q)$*.*

*Proof.* Since $(\lambda, \mathbf{y})$ is feasible, $y_i = \sum_{\tau \in J} \lambda_\tau y_i^\tau \leq 1 - x_i$ must hold for $i \in N$. Suppose there exist $\tau' \in B(\mathbf{x}, S_q)$ such that $\lambda_{\tau'} > 0$. Then, there exists at least one $i \in N$ such that $x_i = 1$ and $y_i = \sum_{\tau \in J} \lambda_\tau y_i^\tau > 0$, which contradicts $y_i \leq 1 - x_i$. □

$B(\mathbf{x}, S_q)$ defined in Proposition 7.1 contains the indices of follower solutions in $S_q$ which are *blocked* by the leader solution $\mathbf{x}$. The remaining solutions which are not blocked are denoted by $\bar{B}(\mathbf{x}, S_q) = J \setminus B(\mathbf{x}, S_q)$. Before going into the next result, note that the objective function of the inner problem can be arranged as follows:

$$\mathbf{p}^T \mathbf{y} = \sum_{i \in N} p_i y_i = \sum_{i \in N} p_i \sum_{\tau \in J} \lambda_\tau y_i^\tau = \sum_{\tau \in J} \lambda_\tau \sum_{i \in N} p_i y_i^\tau. \tag{7.7}$$

Let $z_\tau = \sum_{i \in N} p_i y_i^\tau$ denote the objective value of the follower solution $\mathbf{y}^\tau$. Then, it follows that $\mathbf{p}^T \mathbf{y} = \sum_{\tau \in J} \lambda_\tau z_\tau$.

**Proposition 7.2.** *Given* $\mathbf{x} \in \mathcal{X}^\circ$ *and* $S_q$*, the optimal objective value of the* $LB_q(\mathbf{x})$ *is*

$$z^*(\mathbf{x}, S_q) = \max_{\tau \in \bar{B}(\mathbf{x}, S_q)} z_\tau. \tag{7.8}$$

*Proof.* The objective function of $\mathrm{LB}_q(\mathbf{x})$ is $\mathbf{p}^T \mathbf{y} = \sum_{\tau \in J} \lambda_\tau z_\tau = \sum_{\tau \in \bar{B}(\mathbf{x}, S_q)} \lambda_\tau z_\tau$. The second equality follows from Proposition 7.1. If it is guaranteed that $\lambda_\tau = 0$ for all $\tau \in B(\mathbf{x}, S_q)$, then $\mathbf{y} = \sum_{\tau \in J} \lambda_\tau \mathbf{y}^\tau = \sum_{\tau \in \bar{B}(\mathbf{x}, S_q)} \lambda_\tau \mathbf{y}^\tau \leq \mathbf{1} - \mathbf{x}$ because we know

that $\mathbf{y}^\tau \leq \mathbf{1} - \mathbf{x}$ for all $\tau \in \bar{B}(\mathbf{x}, S_q)$ by definition, and $\lambda \geq 0$. Then, by fixing $\lambda_\tau = 0, \tau \in B(\mathbf{x}, S_q)$, the problem can be reduced to

$$z^*(\mathbf{x}, S_q) = \max_{\lambda} \left\{ \sum_{\tau \in \bar{B}(\mathbf{x}, S_q)} \lambda_\tau z_\tau : \sum_{\tau \in \bar{B}(\mathbf{x}, S_q)} \lambda_\tau = 1, \lambda \geq 0 \right\}. \tag{7.9}$$

As the objective value is simply a convex combination of previous objective values, its optimal value is $\max_{\tau \in \bar{B}(\mathbf{x}, S_q)} z_\tau$. □

Using the results presented above, the bilevel problem in (7.5) is reformulated as

$$t_q = \min_{\mathbf{x} \in \mathcal{X}} \max_{\tau \in \bar{B}(\mathbf{x}, S_q)} z_\tau \tag{7.10}$$

$$= \min_{\mathbf{x}, z} z \tag{7.11}$$

s.t.

$$\mathbf{x} \in \mathcal{X} \tag{7.12}$$

$$z \geq z_\tau, \tau \in \bar{B}(\mathbf{x}, S_q) \tag{7.13}$$

Instead of generating the set $B(\mathbf{x}, S_q)$ for all $\mathbf{x} \in \mathcal{X}$, we define a new set $C_\tau = \{i \in N : y_i^\tau = 1\}$, the indices of the leader variables that cause the solution $\mathbf{y}^\tau$ to be blocked when at least one of them is positive. We refer to $C_\tau$ as the *blocker set* of $\tau$. The following proposition establishes the relation between $C_\tau$ and $B(\mathbf{x}, S_q)$.

**Proposition 7.3.** *If $\sum_{i \in C_\tau} x_i \geq 1$, then $\tau \in B(\mathbf{x}, S_q)$, i.e., $\mathbf{y}^\tau$ is blocked by $\mathbf{x}$.*

*Proof.* If $\sum_{i \in C_\tau} x_i \geq 1$, then there exists $i' \in C_\tau$ such that $x_{i'} = 1$. By definition of $C_\tau$, $\mathbf{y}_{i'}^\tau = 1$. Recall that $B(\mathbf{x}, S_q) = \{\tau \in J : x_i = 1, y_i^\tau = 1, i \in N\}$. Since $x_{i'} = 1$ and $y_{i'}^\tau = 1$, $\tau$ belongs to $B(\mathbf{x}, S_q)$. □

Notice that constraint (7.13) is stated for only unblocked solutions. We introduce a binary decision variable $\alpha_\tau$ to the model for each $\tau \in J$ to indicate whether a follower solution is blocked or not. In the new LB problem formulation, $\alpha_\tau$ takes value of zero when $\mathbf{y}^\tau$ cannot be blocked so that constraint (7.16) becomes $z \geq z_\tau$.

$$LB'_q : t_q = \min z \tag{7.14}$$

$$\text{s.t.}$$

$$\mathbf{x} \in \mathcal{X} \tag{7.15}$$

$$z \geq (1 - \alpha_\tau) z_\tau \qquad \tau \in J \tag{7.16}$$

$$\alpha_\tau \leq \sum_{i \in C_\tau} x_i \qquad \tau \in J \tag{7.17}$$

$$z \geq 0,\ \alpha_\tau \in \{0, 1\} \qquad \tau \in J \tag{7.18}$$

The new formulation can be improved by using the information on which solutions must be blocked. The following proposition helps develop a stronger LB problem formulation.

**Proposition 7.4.** *Let $\bar{Z}$ be an upper bound on the optimal objective value $Z^*(\mathcal{X}, \mathcal{F})$ of the leader. Then, any solution $\mathbf{y}^\tau$ with a greater objective value than $\bar{Z}$ must be blocked in an optimal leader solution, i.e., $\alpha_\tau = 1$ for all $\tau$ such that $z_\tau > \bar{Z}$, in an optimal solution to $LB'_q$.*

*Proof.* Let $(\mathbf{x}^*, \alpha^*)$ be an optimal solution to $LB_q$-2. Then $t_q \geq (1 - \alpha^*_\tau) z_\tau$, $\tau \in J$. Suppose there exist $\tau' \in J$ such that $\alpha^*_{\tau'} = 0$ and $z_{\tau'} > \bar{Z}$. Then $t_q \geq z_{\tau'} > \bar{Z}$, which contradicts with $t_q \leq Z^*(\mathcal{X}, \mathcal{F}) \leq \bar{Z}$. □

Recall that each solution in $S_q$ can be obtained by solving the follower's problem to optimality with the exception of the trivial solution ($\tau = 0$) that is feasible for all feasible leader solutions. Therefore, each $z_\tau, \tau \in J \setminus \{0\}$ constitutes an upper bound on $Z^*(\mathcal{X}^\circ, \mathcal{F}^\circ)$, i.e., $z_\tau \geq Z^*(\mathcal{X}^\circ, \mathcal{F}^\circ), \tau \in J \setminus \{0\}$. From now on, $\bar{Z} = \min_{\tau \in J \setminus \{0\}} z_\tau$

denotes the current upper bound on the optimal objective value. Now, we define $J^B = \{\tau \in J : z_\tau > \bar{Z}\}$ as the index set of the follower solutions that must be blocked according to Proposition 7.4. Then, an alternative and stronger MILP formulation is obtained as follows:

$$LB''_q : t_q = \min z \tag{7.19}$$

s.t.

$$\mathbf{x} \in \mathcal{X} \tag{7.20}$$

$$z \geq (1 - \alpha_\tau) z_\tau \qquad \tau \in J \setminus J^B \tag{7.21}$$

$$\alpha_\tau \leq \sum_{i \in C_\tau} x_i \qquad \tau \in J \setminus J^B \tag{7.22}$$

$$1 \leq \sum_{i \in C_\tau} x_i \qquad \tau \in J^B \tag{7.23}$$

$$z \geq 0,\ \alpha_\tau \in \{0, 1\} \qquad \tau \in J \setminus J^B \tag{7.24}$$

Consider $z_0$ which is the objective value of the trivial follower solution $\mathbf{y}^0$. It is associated with a feasible (but not necessarily optimal) follower reaction for any leader solution, therefore $z_0 \leq \max_{\mathbf{y}}\{\mathbf{p}^T\mathbf{y} : (\mathbf{x}, \mathbf{y}) \in \mathcal{F}^\circ\} \leq Z^*(\mathcal{X}^\circ, \mathcal{F}^\circ)$ for all $\mathbf{x} \in \mathcal{X}^\circ$. Furthermore, since $\mathbf{y}^0$ is feasible for all $\mathbf{x} \in \mathcal{X}$, it cannot be blocked by the leader, i.e., $\alpha_0 = 0$ and $z \geq z_0$ in an optimal solution to $LB''_q$. It follows that $z_0 \leq t_q \leq \bar{Z}$. Next proposition reduces the domain of $t_q$ further and it is a direct consequence of Proposition 7.4.

**Proposition 7.5.** *Either $t_q = z_0$ or $t_q = \bar{Z}$ for any $q$.*

*Proof.* Firstly, $z_0 \leq \bar{Z}$ as explained above. Now, notice that the set $\{z_\tau : \tau \in J \setminus J^B\}$ has only two distinct values, $z_0$ and $\bar{Z}$ since $J \setminus J^B = \{\tau \in J : z_\tau \leq \bar{Z}\}$ and $\bar{Z} = \min_{\tau \in J \setminus \{0\}} z_\tau$. As a result, constraint type (7.21) includes: $z \geq z_0$ and $z \geq (1 - \alpha_\tau)\bar{Z}$ for $\tau \in J \setminus (J^B \cup \{0\})$. If there is a feasible leader solution that can block all $\tau \in J \setminus (J^B \cup \{0\})$, then $\alpha_\tau = 1$ and $t_q = z_0$; otherwise $\alpha_\tau = 0$ and $t_q = \bar{Z}$. $\square$

If $t_q = \bar{Z}$, then $\bar{Z} \leq Z^*(\mathcal{X}^\circ, \mathcal{F}^\circ) \leq \bar{Z}$ and the algorithm terminates. Otherwise, the lower bound remains the same as $z_0$.

### 7.1.2. Greedy Maximum Covering

In an interdiction problem, the upper level feasible region is typically associated with cardinality/budget constraints and logical restrictions on interdiction decisions. In this section, we focus on the case with cardinality constraints only, i.e., $\mathcal{X}^\circ = \{\mathbf{x} \in \{0,1\}^n : \sum_i x_i \leq p\}$. Given the index set $J$ for $S_q$, consider the maximum covering problem:

$$c_q = \max \sum_{\tau \in J \setminus \{0\}} \alpha_\tau \tag{7.25}$$

s.t.

$$\sum_{i \in N} x_i \leq p \tag{7.26}$$

$$\alpha_\tau \leq \sum_{i \in C_\tau} x_i \qquad \tau \in J \setminus \{0\} \tag{7.27}$$

$$x_i \in \{0,1\},\ \alpha_\tau \in \{0,1\} \qquad i \in N, \tau \in J \setminus \{0\} \tag{7.28}$$

If $c_q = |J| - 1$, then it indicates the existence of a feasible $\mathbf{x}$ which covers (blocks in our context) all $\tau \in J \setminus \{0\}$. Then, the optimal objective value $t_q$ of the LB problem for $S_q$ is $z_0$ due to Proposition 7.5. Now let $\hat{c}_q$ denote the objective value that a greedy maximum covering algorithm yields for the same problem and $\hat{\mathbf{x}}^q$ denote the corresponding solution. If $\hat{c}_q = |J| - 1$, then $\hat{\mathbf{x}}^q$ is an optimal solution to $\text{LB}_q$ and $t_q = z_0$. This identity allows us to first solve the LB problem heuristically, and then solve the formulation in (7.19)–(7.24) in case $\hat{c}_q < |J| - 1$. The modified version of the $x$-space algorithm (IXS) is given as Algorithm 7.1.

**Algorithm 7.1** IXS

**Step 0: Initialization**

Define $J = \{-(\rho - 1), ..., 0\}$ as an index set of initial solutions where $\rho \in \mathbb{Z}_+$.

Select $\mathbf{y}^0$ such that it is a feasible follower solution for all feasible leader solutions.

**for** $\tau \in J \setminus \{0\}$ **do**

  Choose a feasible $\mathbf{x}^\tau$ and $\mathbf{y}^\tau \in \arg\max_{\mathbf{y}} \left\{\mathbf{p}^T\mathbf{y} : (\mathbf{x}^\tau, \mathbf{y}) \in \mathcal{F}^\circ\right\}$

  Set $z_\tau \leftarrow \mathbf{p}^T\mathbf{y}^\tau$ and $C_\tau \leftarrow \{i \in N : y_i^\tau = 1\}$

**end for**

Let $S_1 = \bigcup_{\tau \in J}\{\mathbf{y}^\tau\}$.

Set $q = 1$, $\bar{Z} \leftarrow \min_{\tau \in J \setminus \{0\}} z_\tau$ and $\mathbf{x}^* \in \{\mathbf{x}^\tau : z_\tau = \bar{Z}\}$

**Step q:** $(q = 1, 2, ...)$

  **Step q$_0$:** Use the Greedy Covering algorithm to obtain $\hat{\mathbf{x}}^q$ and $\hat{c}_q$

    **if** $\hat{c}_q < |J|$ **then**

      Go to Step q-1

    **else**

      Set $\mathbf{x}^q \leftarrow \hat{\mathbf{x}}^q$, $t_q \leftarrow \hat{c}_q$, go to Step q-2

  **Step q$_1$:** Solve $LB''_q$ to obtain an optimal solution $\mathbf{x}^q$ and optimal value $t_q$.

  **Step q$_2$:** Solve $\mathrm{UB}_q$ to obtain an optimal solution $\mathbf{y}^q$ and optimal value $z_q$.

    **if** $z_q < \bar{Z}$ **then**

      Set $\bar{Z} \leftarrow z_q$ and $\mathbf{x}^* \leftarrow \mathbf{x}^q$

  **Step q$_3$:** **if** $t_q < \bar{Z}$ **then**

      Expand $S_{q+1} = S_q \cup \{\mathbf{y}^q\}$ and update $J = J \cup q$.

      **Go to** Step $q + 1$.

    **else**

      **Return** optimal solution $(\mathbf{x}^*, \mathbf{y}^*) = (\mathbf{x}^q, \mathbf{y}^q)$ and $z^* = \bar{Z}$.

Figure 7.1. Pseudo-code of the Improved $x$-space (IXS) algorithm.

## 7.2. Implementation on the Misinformation Spread Minimization Problem

Recall that the Misinformation Spread Minimization Problem (MSMP) is a Stackelberg game between two players. The leader of the game protects a subset of nodes and then the follower activates a set of unprotected nodes (seed set) to start a diffusion process. The aim of the follower is to maximize the expected number of influenced nodes, i.e., the spread, while the leader wants to minimize the same measure. Although the leader decisions are named "protection" due to the application example, the problem is a stochastic interdiction problem where the leader interdicts the activation of a set of nodes by the follower. In this section, we aim to implement the IXS algorithm on the MSMP which does not have the required mathematical model in present. Besides, the number of possible live-arc scenarios $R$ which we have used to develop a deterministic equivalent formulation can be excessive. In this section we define $R$ as a live-arc scenario set of any size. Therefore, the spread distribution may be different than the one in the LT model.

### 7.2.1. Reformulation of the Problem

The upper bound problem in the IXS algorithm corresponds to the follower's problem of the MSMP formulation in (6.10)–(6.15) and it needs to be updated and solved for each $\mathbf{x}^q$. Recall the constraint (6.14) given as $u_{ir} \leq \sum_{j \in V} a_{ji}(\mathbf{x}, r) y_j$, $i \in V, r \in R$. Here, the term $a_{ji}(\mathbf{x}, r)$ is a parameter for given $\mathbf{x}$ and can be computed via a graph traversal algorithm, or it can be defined explicitly as

$$a_{ji}(\mathbf{x}, r) = \begin{cases} \prod_{k \in b^r_{ji}} (1 - x_k) & j \in a^r_i \\ 0 & j \notin a^r_i, \end{cases} \tag{7.29}$$

where $a^r_i$ denotes the set of nodes that can reach node $i$ via the arcs in scenario $r$ including $i$ itself. Parameter $b^r_{ji}$ represents the set of nodes on the path from $j$ to $i$ in scenario $r$ excluding $i$ and $j$ if such a path exists. In other words, there is an eligible path from $j$ to $i$ if none of the nodes between them is protected. By its definition

$a_{ji}(\mathbf{x}, r) = 1$ if $i = j$ or $j$ is the direct predecessor of $i$ in scenario $r$, which implies $b^r_{ji} = \emptyset$. Note that there can be at most one path from one node to another in a scenario (a subgraph) due to the arc selection scheme of the live-arc technique for the LT model. Therefore, we are not concerned with a path selection decision. Let $\bar{a}^r_i$ denote the subset of $a^r_i$ excluding $i$ and its direct predecessor in scenario $r$, if it exists. Now, to linearize the right-hand side of (6.14), we define a binary decision variable $m^r_{ji}$ for each $i \in V$, $j \in \bar{a}^r_i$ and $r \in R$, whose value is one if node $j$ is active and influences node $i$ in scenario $r$. In short $m^r_{ji} = y_j \prod_{k \in b^r_{ji}} (1 - x_k)$. Then, the following constraints are added to the lower level problem (LLP).

$$m^r_{ji} \leq y_j \qquad i \in V, j \in \bar{a}^r_i, r \in R \tag{7.30}$$

$$m^r_{ji} \leq 1 - x_k \qquad i \in V, j \in \bar{a}^r_i, k \in b^r_{ji}, r \in R \tag{7.31}$$

$$m^r_{ji} \geq y_j - \sum_{k \in b^r_{ji}} x_k \qquad i \in V, j \in \bar{a}^r_i, r \in R \tag{7.32}$$

We can replace constraint (6.14) with

$$u_{ir} \leq \sum_{j \in a^r_i \setminus \bar{a}^r_i} y_j + \sum_{j \in \bar{a}^r_i} m^r_{ji} \qquad i \in V, r \in R. \tag{7.33}$$

Although (7.32) is required to define $m^r_{ji}$, it is easy to observe that the optimal objective value remains the same when this constraint is relaxed due to the sense of optimization. So, we discard it and the final LLP includes (6.10)–(6.13), (6.15), (7.30)–(7.31), and (7.33). It was also possible to formulate (7.33) as $u_{ir} \leq \sum_{j \in a^r_i} m^r_{ji}$ which would be still correct despite the fact that it leads to a larger MILP. Note that the type of the cardinality constraint in (6.11) is changed from being equality to a "$\leq$" constraint to ensure that $\mathbf{y} = \mathbf{0}$ is a feasible follower solution for any leader solution.

### 7.2.2. Improved $x$-space Algorithm Implementation

Recall that the LB problem in (7.5) includes the convex combination constraints and the constraints that relate the lower and upper level variables. Let $\bar{\mathbf{y}} = (\mathbf{y}, \mathbf{u}, \mathbf{m})$

denote the combined decision variables of the LLP of the MSMP. The initial LB formulation of the $x$-space algorithm is developed below, where $\mathcal{X}^\circ$ is defined by (6.8) and (6.9).

$$t_q = \min_{\mathbf{x}\in\mathcal{X}^\circ} \max_{\lambda,\bar{\mathbf{y}}} \sum_{r\in R}\sum_{i\in V} p_r u_{ir} \tag{7.34}$$

s.t.

$$\sum_{\tau\in J} \lambda_\tau = 1 \tag{7.35}$$

$$\bar{\mathbf{y}} - \sum_{\tau\in J} \lambda_\tau \bar{\mathbf{y}}^\tau = 0 \tag{7.36}$$

$$y_i \leq 1 - x_i \qquad i \in V \tag{7.37}$$

$$u_{ir} \leq 1 - x_i \qquad i \in V, r \in R \tag{7.38}$$

$$m^r_{ji} \leq 1 - x_k \qquad i \in V, j \in a^r_i, k \in b^r_{ji}, r \in R \tag{7.39}$$

$$\lambda \geq 0, \bar{\mathbf{y}} \geq 0 \tag{7.40}$$

Notice that the formulation does not possess the special structure that we mention in Section 7.1.1, i.e., $\mathbf{d} = \mathbf{1}$, $\mathbf{D} = \mathbf{I}$ and $n_1 = n_2 = n$. Nevertheless, the alternative LB problem formulations $LB'_q$ and $LB''_q$ can be developed for the MSMP by redefining the set $B(\mathbf{x}, S_q)$ as a first step. Then, we can define $C_\tau$ which is used directly in these formulations.

**Proposition 7.6.** *For the MSMP,* $B(\mathbf{x}, S_q) = \{\tau \in J : \exists i \in V, r \in R, x_i = 1, u^\tau_{ir} = 1\}$.

*Proof.* Recall that $B(\mathbf{x}, S_q)$ is the index set of the follower solutions in $S_q$ that $\mathbf{x}$ blocks, i.e., renders infeasible. A follower solution becomes infeasible for $\mathbf{x}$ if one of the constraints in (7.37)–(7.39) is violated. Let $B_l(\mathbf{x}, S_q)$ be the set of follower solutions that violate $l^{th}$ of these constraints. Then, $B(\mathbf{x}, S_q) = \cup_{l=1}^3 B_l(\mathbf{x}, S_q)$ by definition. We need to show that $B(\mathbf{x}, S_q) = B_2(\mathbf{x}, S_q) = \{\tau \in J : \exists i \in V, r \in R, x_i = 1, u^\tau_{ir} = 1\}$. Consider $B_1(\mathbf{x}, S_q) = \{\tau \in J : \exists i \in V, x_i = 1, y^\tau_i = 1\}$. In an optimal follower solution, if $y_i = 1$ then $u_{ir} = 1, \forall r$. This leads to the relation $B_1(\mathbf{x}, S_q) \subset B_2(\mathbf{x}, S_q)$, since the

solutions in $S_q$ are optimal follower solutions except $\bar{\mathbf{y}}^0$. Now consider $B_3(\mathbf{x}, S_q) = \{\tau \in J \mid \exists i \in V, j \in a_i^r, k \in b_{ji}^r, r \in R, x_k = 1, m_{ji}^\tau(r) = 1\}$. Take $\tau \in B_3(\mathbf{x}, S_q)$. If $m_{ji}^\tau(r) = 1$, i.e., if node $j$ can influence node $i$ under scenario $r$ in solution $\tau$, then all the nodes on the path from $j$ to $i$ must be influenced as well. In other words, if $m_{ji}^\tau(r) = 1$ then $u_{kr}^\tau = 1$ for all $k \in b_{ji}^r$. Since also $x_k = 1$ for some $k \in b_{ji}^r$, $\tau$ is contained in $B_2(\mathbf{x}, S_q)$ and it shows that $B_3(\mathbf{x}, S_q) \subset B_2(\mathbf{x}, S_q)$. $\square$

To summarize, any solution in $\text{conv}(S_q)$ that violates (7.37) or (7.39) also violates (7.38). This allows us to define the blocker set of a follower solution by using only constraint (7.38). Thus, we obtain $C_\tau = \{i \in V : \exists r \in R, u_{ir}^\tau = 1\}$. Since $C_\tau$ is explicitly defined now, we can use the $LB_q'$ and $LB_q''$ formulations in Section 7.1.1 to solve the lower bound problem, as well as the IXS algorithm in Figure 7.1. Since we will compare the results of XS and IXS in Section 8.5, we need to develop the necessary mathematical models used in the original algorithm. We describe the procedure and present the resulting LB problem in Appendix C. It is worth mentioning that (7.37) and (7.39) are removed from the lower bound problem formulation of the XS algorithm as a result of Proposition 7.6, for a fair comparison of the original and improved algorithms.

# 8. COMPUTATIONAL RESULTS

## 8.1. Experimental Settings and Instance Generation

All of the algorithms proposed in this thesis as well as those in the literature used for comparison have been coded in C++ using Microsoft Visual Studio 2015. The computer with which we realize our experiments is a workstation with an Intel® Xeon® E5-2687W CPU, 3.10 GHz processor and 64GB RAM. The operating system is Microsoft Windows 7 Professional. Two MILP solvers are used depending on the problem and the method: CPLEX Optimization Studio (versions 12.7 and 12.9) and Gurobi (version 8.0). The solver choice will be indicated at the beginning of each section.

Two classes of test instances are used. The first class includes networks that are randomly generated according to one of the following network generation models: Watts-Strogatz (Watts and Strogatz, 1998) and Erdős-Rényi (Erdos and Rényi, 1960) models. The Watts-Strogatz (WS) model produces networks with small average distances between node pairs, and a relatively high clustering coefficient, i.e, a small-world network reflecting the general structure of real social networks. The parameters of the WS model are the number of nodes $n$, the number of arcs $m$, and the rewire probability $p$. The rewire probability is determined as $p = 0.05$ for $n \leq 100$ and $p = 0.01$ for $n > 100$ based on the preliminary experiments performed. In these experiments, we investigate on networks with similar sizes the change in the clustering coefficient and at the average path length as $p$ changes. The selected values yield a good balance between these measures for the corresponding sizes. The `sample_smallworld` function of the `igraph` package is employed to obtain the WS networks. Completely random networks are generated following the Erdős-Rényi (ER) model where each arc between any pair of nodes has the same chance to be generated, by means of the `sample_gnm` function of the `igraph` package whose parameters are $n$ and $m$. As is the case with the WS model, the ER networks also have the property of having small average distance

between node pairs. However, the clustering coefficient of the resulting network is not as large as in the small-world networks obtained by the WS model. The arc weights for WS and ER instances are generated uniformly in the $[0, 1]$ interval and normalized to satisfy $\sum_{j \in V} w_{ji} \leq 1$, $i \in V$.

The second class includes networks which are obtained from a real collaboration network, i.e. the co-authorship relations in the physics section of the e-print repository `arXiv` (Cornell University, 1991). While extracting a network of size $n$, a subset of $n$ nodes is selected (along with the arcs among them) such that the average degree of the nodes in the resulting network is close to the average degree of the original network. The selection algorithm is presented in Appendix D. The arc weights $w_{ij}$ are set to $1/d_j$ where $d_j$ is the in-degree of node $j$ as done in the related studies.

## 8.2. Results for the Influence Maximization Problem with Deactivation

In this section, we present the results of a computational study on the matheuristic methods proposed for the IMPD in Chapter 4. The performances of the SAM and TSM methods depend heavily on the SAA method because the spread corresponding to a leader's decision is estimated using the SAA method (Algorithm 4.2) with a certain optimality gap. Hence, the optimality gaps attained basically determine the quality of the final solution. Therefore, we first assess the performance of the implemented SAA method. Then, we investigate the results of the matheuristic methods in terms of solution quality on small problem instances. Lastly, we conduct experiments on larger instances using both of the formulations proposed for IMPD, and analyze the effect of the formulation choice on the results. CPLEX 12.7 is used to solve the MILPs in the SAA method.

### 8.2.1. Sample Average Approximation Method Results

We consider two measures to evaluate the quality of a follower solution found by the SAA algorithm: the estimate of the optimality gap and its variance. Before going

into a detailed analysis, we conduct some preliminary experiments to observe the effect of the sampling methods described in Section 4.2.1. To this end, the lower level problem of IMPD-t is approximated via SAA for randomly generated leader solutions since the goal of these experiments is to analyze the quality of the solutions obtained for the LLP. We generate one WS and one ER network instance for each of the $(n, d)$ combinations considered. The number of samples used in the first stage of SAA, $M$, is determined as 20. The sample sizes in stage 2 and stage 3 are $N' = 1000$ and $N'' = 5000$, respectively. Unit activation and deactivation costs ($c_i = e_i = 1, i \in V$) are assumed. Therefore the budget levels actually represent the seed sizes of the players. The random leader solutions are generated using an activation budget of $0.1n$ and each problem is solved for three follower's budget levels which is denoted by $E$. The average results are reported in Table 8.1.

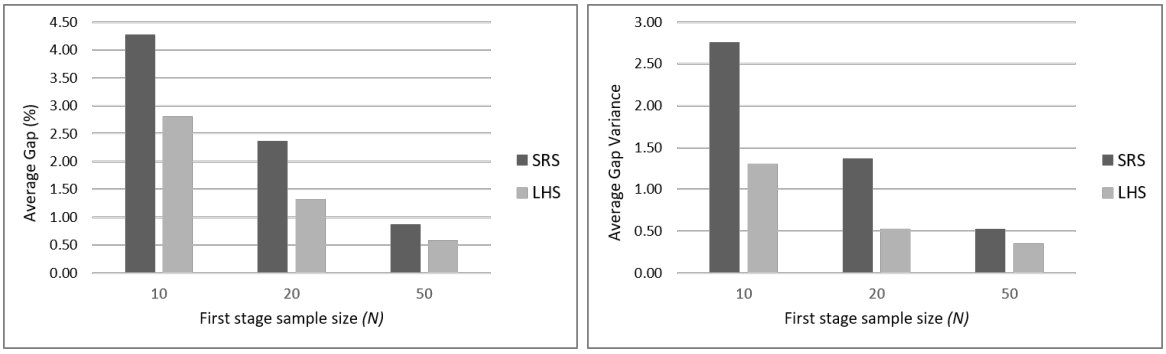
SRS indicates that all samples used in the SAA algorithm are generated using simple random sampling. In $k$-means++, for each sample of size $N$ in stage 1, first $10N$ random points (threshold vectors) are generated. Then, they are partitioned into $N$ clusters and a random member of each cluster is used in the approximate problem formulation ALLP-t, which is an MILP. In the LHS method, each sample of stage 1 are generated using Latin Hypercube sampling. The methods are applied only in the first stage, since the sample size of this phase is small compared to the others and it is observed that the main source of the large optimality gaps are poor LB estimates which are the outputs of stage 1. Percent gap estimates show that LHS outperforms the other methods for especially small sample size and SRS is the worst performer. The effect of $k$-means++ is quite small compared to LHS. As the sample size increases, the gap estimates decrease and the benefit of LHS decreases, but it still seems significant. The solution times using each method are not reported here since they are quite close to each other and dominated by MILP solution times.

Another advantage of the LHS method is smaller variance of the gap estimator. In Figure 8.1(a) and Figure 8.1(b), we compare the average gap estimates and their variances for SRS and LHS methods for the instances in Table 8.1. The improvement

Table 8.1. Gap(%) Estimates of SAA algorithm for arbitrary leader strategy.

| | | $N = 10$ | | | $N = 20$ | | | $N = 50$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $(n, d)$ | $(C, E)$ | SRS | $k$-means++ | LHS | SRS | $k$-means++ | LHS | SRS | $k$-means++ | LHS |
| (500,0.002) | (50,5) | 1.02 | 0.22 | 0.40 | 0.5 | 0.17 | 0.12 | -0.13 | -0.07 | 0.16 |
| | (50,6) | 1.03 | 0.25 | 0.46 | 0.53 | 1.2 | 0.16 | -0.08 | -0.09 | 0.14 |
| | (50,8) | 0.97 | 0.22 | 0.34 | 0.57 | 0.13 | 0.12 | -0.14 | 0.21 | 0.1 |
| (500,0.003) | (50,5) | 0.91 | 1.25 | 0.47 | 0.4 | 0.25 | 0.26 | 0.04 | 0.15 | 0.19 |
| | (50,6) | 1.11 | 1.38 | 0.49 | 0.55 | 1.13 | 0.4 | 0.13 | 0.23 | 0.23 |
| | (50,8) | 1.51 | 1.46 | 0.63 | 0.86 | 1.91 | 0.42 | 0.19 | 0.16 | 0.16 |
| (500,0.004) | (50,5) | 4.59 | 4.90 | 4.09 | 3.41 | 3.03 | 2.34 | 1.59 | 1.42 | 1.61 |
| | (50,6) | 4.99 | 5.62 | 4.39 | 3.44 | 2.4 | 2.68 | 1.77 | 1.92 | 1.86 |
| | (50,8) | 4.87 | 5.70 | 4.09 | 3.85 | 3.79 | 2.26 | 1.48 | 1.55 | 1.66 |
| (1000,0.002) | (100,10) | 4.2 | 3.99 | 1.94 | 2.28 | 1.92 | 1.04 | 0.81 | 0.72 | 0.31 |
| | (100,12) | 3.88 | 4.48 | 1.94 | 2.05 | 1.14 | 0.93 | 0.87 | 0.66 | 0.35 |
| | (100,17) | 4.87 | 5.43 | 1.84 | 2.62 | 2.9 | 1.12 | 1.27 | 1.12 | 0.52 |
| (1000,0.003) | (100,10) | 4.92 | 5.91 | 3.85 | 1.69 | 2.86 | 1.73 | 0.4 | 0.42 | 0.32 |
| | (100,12) | 5.45 | 7.44 | 4.17 | 2.92 | 1.75 | 1.63 | 0.45 | 0.51 | 0.49 |
| | (100,17) | 7.25 | 7.96 | 7.16 | 3.5 | 4.86 | 2.95 | 1.02 | 1.04 | 0.99 |
| (1500,0.001) | (150,15) | 2.74 | 2.07 | 1.40 | 1.21 | 1.09 | 0.57 | 0.59 | 0.52 | 0.29 |
| | (150,19) | 3.3 | 2.79 | 1.48 | 1.8 | 1.5 | 0.72 | 0.87 | 0.73 | 0.5 |
| | (150,25) | 4.2 | 3.24 | 1.82 | 2.39 | 1.88 | 0.71 | 0.98 | 0.89 | 0.61 |
| (1500,0.002) | (150,15) | 7.78 | 5.84 | 5.08 | 4.09 | 3.15 | 2.06 | 0.83 | 0.75 | 0.42 |
| | (150,19) | 9.62 | 7.28 | 5.89 | 5.28 | 3.95 | 2.62 | 0.79 | 0.76 | 0.53 |
| | (150,25) | 10.56 | 9.49 | 7.01 | 5.7 | 5.63 | 2.91 | 1.53 | 1.22 | 0.84 |
| Average | | 4.27 | 4.14 | 2.81 | 2.36 | 2.22 | 1.32 | 0.87 | 0.79 | 0.58 |

on the variance is even larger than that on the gap estimate. As a consequence, we decided to implement the SAA algorithm with LHS in all our future experiments.




(a) Average gap estimate (%) (b) Average variance of the gap estimator

Figure 8.1. Comparison of LHS and SRS methods.

8.2.1.1. Optimality Gaps. For the analysis of the SAA algorithm, we consider 10 instances for each of the WS and ER models where the number of nodes $n$ takes five values from the set $\{20, 40, 60, 80, 100\}$. $m$ is set to two different values given as $m = 2n$ and $m = 4n$. As before, the leader's solutions, i.e., seed sets are generated randomly. The seed size is fixed to $\lfloor 0.15\, n \rfloor$. The first part of the experiments is devoted to the cardinality-based deactivation where the deactivation costs are assumed to be unity (i.e., $e_i = 1$ for all $i$). This implies that the deactivation budget $E$ represents the number of nodes in the seed set that can be deactivated. In the second part of the experiments, we use cost-based deactivation in which deactivation costs $e_i$ are generated uniformly from the set $\{10, 15, 20\}$ corresponding to low, medium, and high unit deactivation cost levels. The sample sizes are determined as $M = 20$, $N = 50$, $N' = 2000$ and $N'' = 10,000$.

The results are presented in Figure 8.2 and Figure 8.3 for cardinality-based and cost-based instances, respectively. Since we are primarily interested in the quality of the solutions that SAA produces, we display confidence intervals (CI) on the true gap using the average lower and upper bound estimates over 20 replications of SAA for the same leader solution. The instance characteristics including the network size $n$, the number of arcs $m$ and the values of the budget parameters ($C$ for the leader's activation budget and $E$ for the follower's deactivation budget) are shown next to the horizontal

axis. Recall that the objective value $\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*)$ that SAA yields is an upper bound estimate on the true optimal objective, and $\bar{z}_N^M(\mathbf{x})$ which is the output of its first stage is a lower bound estimate. The percent optimality gaps and a conservative 95% CI on the true gap as a percentage of the average upper bound estimate are plotted on the graphs. Remark that the confidence interval computed by the expression

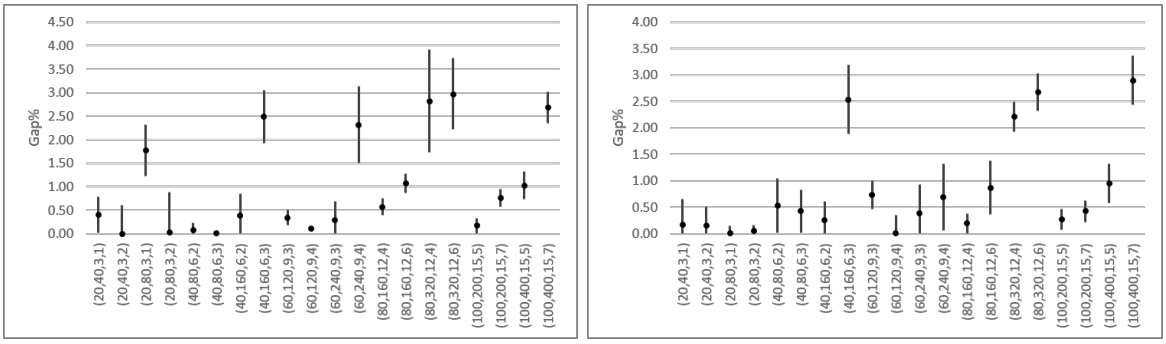
$$100 \times \frac{\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*) - \bar{z}_N^M(\mathbf{x}) \pm t_{\alpha/2,\nu}\sqrt{V(\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*)) + V(\bar{z}_N^M(\mathbf{x}))}}{\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*)} \tag{8.1}$$

represents the variance of the optimality gap, where $\nu$ is the approximate degrees of freedom (Montgomery and Runger, 2010) that can be determined by the formula

$$\nu = \frac{\left[V(\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^\star)) + V(\bar{z}_N^M(\mathbf{x}))\right]^2}{\frac{V(\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^\star))^2}{N''-1} + \frac{V(\bar{z}_N^M(\mathbf{x}))^2}{M-1}}. \tag{8.2}$$

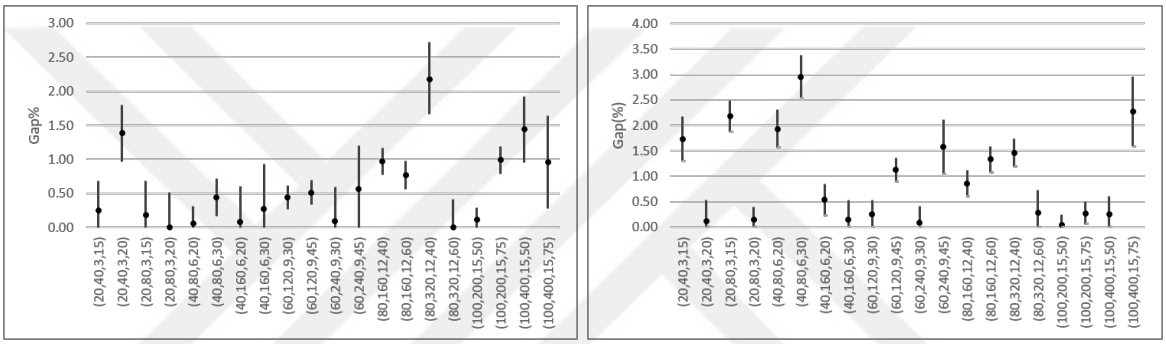
Due to the variance in the bound estimates, sometimes the (percent) optimality gap estimate $\hat{z}_{N''}(\mathbf{x}, \tilde{\mathbf{y}}^*) - \bar{z}_N^M(\mathbf{x})$ may turn out to be negative. This is also the case with CI limits, in particular with the lower CI limit. However, as the sample size used in the SAA method increases, negative gap estimates disappear. As suggested by Mak *et al.* (1999), a conservative CI formula can be utilized to get rid of the negative limits. In the experiments of this section, we do not come across a negative gap estimate that is computed by taking the average over 20 replications. As there exist negative lower limits, however, we adopt the conservative approach of Mak *et al.* (1999), and calculate the lower limit $l$ of the CIs plotted in Figure 8.2 and Figure 8.3 as $l^+ = \max\{l, 0\}$.

It can be observed in these two figures that instances with relatively larger optimality gap estimates also have larger variances of the gaps with wider CIs. As can be seen in Figure 8.2(a) and Figure 8.2(b), the gap estimates increase as the number of arcs $m$ and the deactivation budget $E$ increase for the instances with $n = 80$ and $n = 100$. Our additional experiments carried out with larger instances support this result. There is not a notable difference between the results obtained for cost-based




(a) ER networks (b) WS networks

Figure 8.2. SAA gap estimates for cardinality-based instances.




(a) ER networks (b) WS networks

Figure 8.3. SAA gap estimates for cost-based instances.

and cardinality-based instances. Most of the optimality gap estimates are in the range of $[0, 1.5]$. It is worthwhile to mention that the same follower response is obtained in all replications of the SAA algorithm executed for the same instance. An analysis about the CPU times of the algorithm will be presented in the next section.

<u>8.2.1.2. Effect of the Formulation.</u> In this section, we investigate the performance of the SAA for each one of the proposed IMPD formulations, in terms of solution quality and CPU time. We generate 40 new instances in addition to 10 instances that are used in Section 8.2.1.1 for both network types (i.e., WS and ER). This results in 50 instances for each network type giving rise to 100 instances for cost-based and cardinality-based scenarios separately. For each instance, we generate a random leader seed set of size $C$ and solve the problem with IMPD-t and IMPD-La formulations for two different $E$ values. The results are reported in Table 8.2 and Table 8.3, where each number represents an average over 10 instances. Gap(%) and Var denote the averages of SAA

gap estimates and variances of the gap estimates, and the last columns show the CPU time in seconds.

Table 8.2. Effect of the IMPD formulation on SAA results for cardinality-based instances.

| $(n, m, C, E)$ | IMPD-t | | | IMPD-La | | |
|---|---|---|---|---|---|---|
| | Gap(%) | Var | Time | Gap(%) | Var | Time |
| (20,40,3,1) | 0.15 | 0.00 | 0.22 | 0.39 | 0.00 | 0.28 |
| (20,40,3,2) | 0.82 | 0.00 | 0.20 | 2.00 | 0.00 | 0.28 |
| (20,80,3,1) | 0.48 | 0.01 | 0.81 | 2.17 | 0.01 | 0.38 |
| (20,80,3,2) | 1.29 | 0.00 | 0.69 | 1.57 | 0.00 | 0.38 |
| (40,80,6,2) | 0.34 | 0.00 | 0.44 | 1.35 | 0.00 | 0.93 |
| (40,80,6,3) | 0.78 | 0.00 | 0.34 | 1.93 | 0.00 | 0.91 |
| (40,160,6,2) | 0.87 | 0.01 | 2.87 | 2.30 | 0.02 | 1.42 |
| (40,160,6,3) | 0.76 | 0.01 | 1.90 | 1.66 | 0.01 | 1.42 |
| (60,120,9,3) | 0.41 | 0.00 | 0.60 | 1.56 | 0.00 | 2.08 |
| (60,120,9,4) | 0.43 | 0.00 | 0.55 | 1.40 | 0.00 | 2.11 |
| (60,240,9,3) | 0.89 | 0.03 | **19.93** | 1.53 | 0.04 | 3.52 |
| (60,240,9,4) | 1.45 | 0.03 | **18.16** | 2.13 | 0.03 | 3.40 |
| (80,160,12,4) | 0.51 | 0.01 | 0.89 | 0.40 | 0.01 | 3.65 |
| (80,160,12,6) | 0.88 | 0.00 | 0.75 | 2.11 | 0.00 | 3.73 |
| (80,320,12,4) | 0.96 | 0.05 | **38.00** | 1.71 | 0.07 | 6.17 |
| (80,320,12,6) | 2.36 | 0.02 | **29.26** | 2.45 | 0.03 | 6.28 |
| (100,200,15,5) | 0.75 | 0.01 | 1.10 | 0.63 | 0.01 | 5.69 |
| (100,200,15,7) | 0.68 | 0.01 | 1.00 | 1.49 | 0.01 | 5.67 |
| (100,400,15,5) | 2.24 | 0.05 | **89.36** | 1.19 | 0.07 | 9.75 |
| (100,400,15,7) | 2.93 | 0.04 | **70.34** | 2.81 | 0.05 | 9.73 |
| Average | 1.00 | 0.01 | 13.87 | 1.64 | 0.02 | 3.39 |

It is observed that the average of the gap estimates and their variances become slightly larger when IMPD-La formulation is used. An important portion of the variance of the gap estimate comes from the lower bound estimate due to limited sample size used in the first stage of the SAA algorithm (81.52% and 77.96% of the total variance for IMPD-La and IMPD-t, respectively). However, the first stage dominates the overall solution time as the network becomes larger (higher values of $n$) and denser (higher values of $m$). This is the reason why increasing the sample size ($N$) or the number of batches ($M$) to obtain better lower bound and gap estimates is not practical. On the other hand, as can be seen from the tables there are some rapid increases

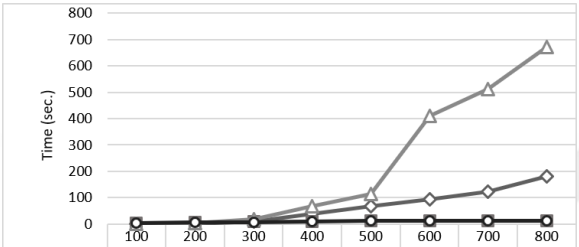
Table 8.3. Effect of the IMPD formulation on SAA results for cost-based instances.

| $(n, m, C, E)$ | IMPD-t | | | IMPD-La | | |
|---|---|---|---|---|---|---|
| | Gap(%) | Var | Time | Gap(%) | Var | Time |
| (20,40,3,15) | 0.21 | 0.00 | 0.17 | 0.41 | 0.00 | 0.28 |
| (20,40,3,20) | 0.25 | 0.00 | 0.16 | 1.20 | 0.00 | 0.28 |
| (20,80,3,15) | 0.60 | 0.00 | 0.43 | 0.34 | 0.01 | 0.39 |
| (20,80,3,20) | 0.94 | 0.00 | 0.65 | 0.81 | 0.01 | 0.38 |
| (40,80,6,20) | 0.18 | 0.01 | 0.35 | 0.35 | 0.00 | 0.91 |
| (40,80,6,30) | 0.53 | 0.00 | 0.45 | 0.47 | 0.00 | 0.95 |
| (40,160,6,20) | 0.59 | 0.02 | 1.84 | 0.26 | 0.03 | 1.43 |
| (40,160,6,30) | 0.15 | 0.02 | 2.14 | 1.39 | 0.02 | 1.48 |
| (60,120,9,30) | 0.43 | 0.01 | 0.63 | 0.59 | 0.01 | 2.14 |
| (60,120,9,45) | 0.72 | 0.01 | 0.76 | 0.96 | 0.01 | 2.16 |
| (60,240,9,30) | 1.36 | 0.05 | **12.25** | 0.89 | 0.06 | 3.39 |
| (60,240,9,45) | 1.07 | 0.05 | **12.88** | 1.75 | 0.05 | 3.49 |
| (80,160,12,40) | 0.42 | 0.01 | 0.94 | 0.67 | 0.01 | 3.66 |
| (80,160,12,60) | 0.64 | 0.01 | 0.90 | 0.53 | 0.01 | 3.82 |
| (80,320,12,40) | 0.91 | 0.07 | **23.46** | 1.07 | 0.08 | 6.16 |
| (80,320,12,60) | 0.87 | 0.04 | **32.19** | 1.65 | 0.05 | 6.36 |
| (100,200,15,50) | 0.36 | 0.01 | 1.29 | 0.72 | 0.02 | 5.98 |
| (100,200,15,75) | 0.73 | 0.01 | 1.30 | 0.60 | 0.01 | 5.75 |
| (100,400,15,50) | 1.24 | 0.08 | **55.91** | 1.00 | 0.08 | 9.86 |
| (100,400,15,75) | 1.69 | 0.08 | **62.40** | 1.22 | 0.10 | 9.72 |
| Average | 0.70 | 0.02 | 10.55 | 0.84 | 0.03 | 3.43 |

in average solution times when we use IMPD-t to solve the follower's problem, in both cost-based and cardinality-based cases. In these experiments, the CPU times of the overall SAA algorithm are in the range of $[0.11, 160.63]$ and $[0.13, 168.06]$ seconds for ER and WS, respectively, with threshold scenarios. The same ranges are computed as $[0.15, 10.73]$ and $[0.16, 9.77]$ with live-arc scenarios.

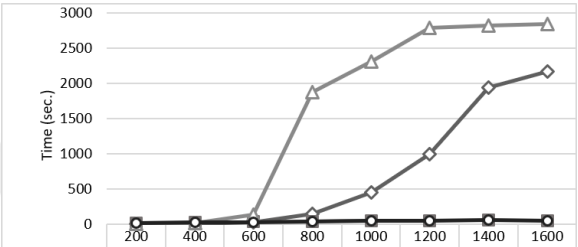
The significant differences in solution times motivate us to conduct more experiments with different $(n, m)$ combinations. In the new test set we have $n \in \{100, 200, 300, 400, 500, 1000\}$ and $m \in \{2n, 4n, 6n, 8n\}$. We consider additional values of $m \in \{n, 3n, 5n, 7n\}$ for $n \in \{100, 200\}$. For the cardinality-based instances, two activation budget levels $C \in \{0.05n, 0.1n\}$ and for each value of $C$, two deactivation budget levels $E \in \{0.2C, 0.4C\}$ are determined. For the cost-based instances, four budget lev-

els are selected in such a way that the average number of nodes that can be activated or deactivated is compatible with the numbers in the cardinality-based case, for the same $n$. 5 instances are generated for each of the parameter settings described above. The resulting number of test instances is 2560. Here, we present the average SAA times obtained during the execution of the proposed matheuristic methods whose results are presented in Section 8.2.2.4.



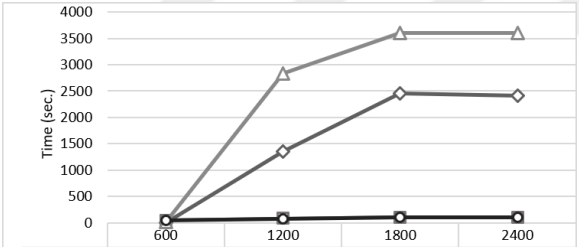

| | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 |
|---|---|---|---|---|---|---|---|---|
| WS-Th | 0.4 | 1.5 | 8.2 | 38.3 | 67.9 | 92.7 | 123.1 | 180.6 |
| WS-La | 2.9 | 6.3 | 8.8 | 10.2 | 11.4 | 11.9 | 12.1 | 12.8 |
| ER-Th | 0.5 | 3.9 | 17.8 | 67.0 | 113.1 | 410.9 | 512.6 | 670.4 |
| ER-La | 2.9 | 5.7 | 7.9 | 9.7 | 10.9 | 11.8 | 12.4 | 13.1 |

(a) $n = 100$



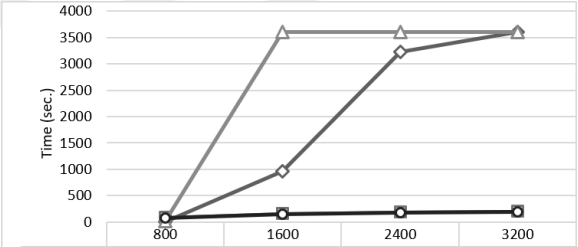

| | 200 | 400 | 600 | 800 | 1000 | 1200 | 1400 | 1600 |
|---|---|---|---|---|---|---|---|---|
| WS-Th | 0.8 | 3.1 | 26.9 | 149.4 | 447.4 | 994.8 | 1941.7 | 2168.2 |
| WS-La | 10.3 | 23.5 | 33.8 | 40.2 | 48.6 | 46.6 | 52.9 | 49.8 |
| ER-Th | 1.3 | 8.7 | 131.3 | 1872.0 | 2315.5 | 2795.1 | 2823.9 | 2845.9 |
| ER-La | 10.4 | 21.4 | 29.3 | 36.1 | 45.7 | 43.9 | 52.6 | 48.8 |

(b) $n = 200$



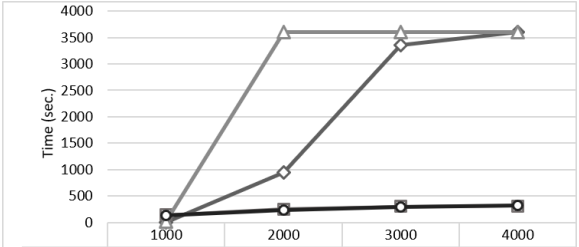

| | 600 | 1200 | 1800 | 2400 |
|---|---|---|---|---|
| WS-Th | 3.2 | 1358.8 | 2458.7 | 2417.8 |
| WS-La | 51.7 | 88.5 | 103.8 | 109.2 |
| ER-Th | 12.4 | 2827.5 | 3600.0 | 3600.0 |
| ER-La | 46.1 | 80.6 | 99.8 | 109.3 |

(c) $n = 300$



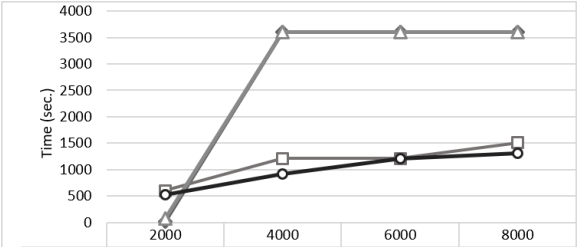

| | 800 | 1600 | 2400 | 3200 |
|---|---|---|---|---|
| WS-Th | 5.4 | 967.4 | 3225.0 | 3600.0 |
| WS-La | 88.8 | 157.4 | 185.9 | 200.3 |
| ER-Th | 12.3 | 3600.0 | 3600.0 | 3600.0 |
| ER-La | 78.4 | 144.7 | 179.3 | 197.6 |

(d) $n = 400$




| | 1000 | 2000 | 3000 | 4000 |
|---|---|---|---|---|
| WS-Th | 6.7 | 941.3 | 3360.0 | 3600.0 |
| WS-La | 146.5 | 255.8 | 300.6 | 311.4 |
| ER-Th | 21.8 | 3600.0 | 3600.0 | 3600.0 |
| ER-La | 132.3 | 232.8 | 299.4 | 325.2 |

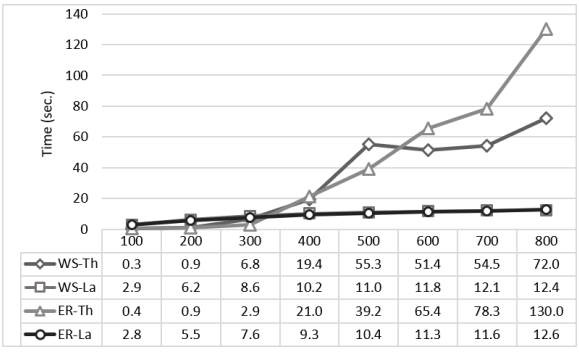
(e) $n = 500$



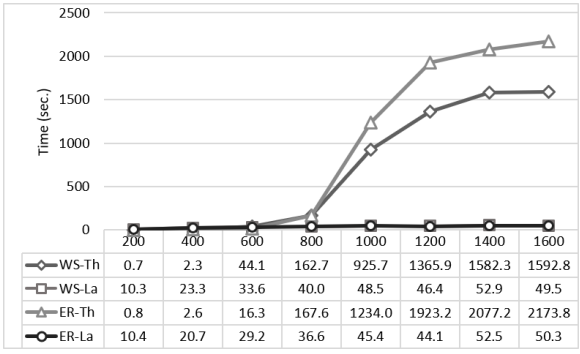

| | 2000 | 4000 | 6000 | 8000 |
|---|---|---|---|---|
| WS-Th | 13.5 | 3600.0 | 3600.0 | 3600.0 |
| WS-La | 602.2 | 1209.2 | 1208.6 | 1511.3 |
| ER-Th | 81.2 | 3600.0 | 3600.0 | 3600.0 |
| ER-La | 520.5 | 920.3 | 1208.3 | 1302.4 |

(f) $n = 1000$

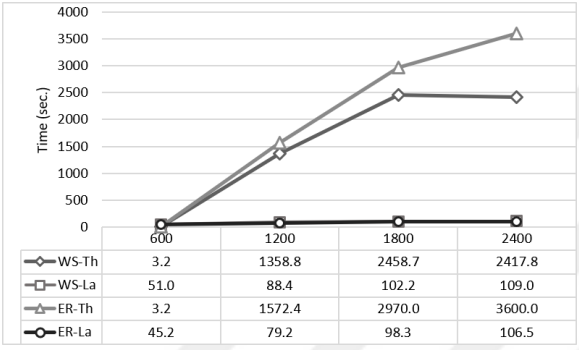
Figure 8.4. Average SAA times for cardinality-based instances.



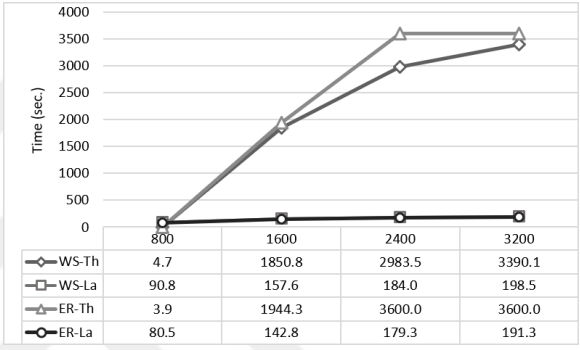

(a) $n = 100$



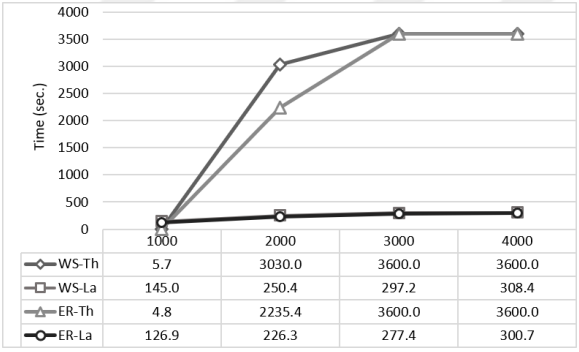

(b) $n = 200$



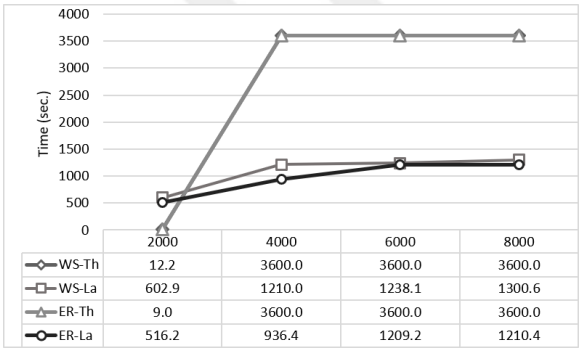

(c) $n = 300$




(d) $n = 400$




(e) $n = 500$




(f) $n = 1000$

Figure 8.5. Average SAA times for cost-based instances.

The average SAA time for an instance is computed as the ratio of total solution time to the number of leader solutions for which the SAA implementation is completed. If the time limit of one hour is reached before the first SAA output is obtained, then the average SAA time is recorded as 3600 seconds for that instance. The results are presented in Figure 8.4 and Figure 8.5. The number of arcs is shown on the horizontal axes. The results are aggregated over the budget levels and the instances with the same parameter settings. In consequence, each data point shows the average computation

time over 20 instances. The legends denote the network model (WS or ER) and the formulation type: threshold (Th) or live-arc (La). Under both the cost-based and the cardinality based setting, it is clear that the average SAA time increases rapidly as the arc density increases in the experiments with the threshold formulation. Although the threshold formulation yields smaller solution times with relatively small $m$ values for each $n$, the live-arc formulation is barely affected by the increase in $m$. Moreover, it can be observed in the tables that with the live-arc formulation the rate of increase in the average SAA computation time is reduced significantly as $m$ increases.

### 8.2.2. Matheuristic Method Results

8.2.2.1. Assessment of the Performances of the Matheuristics. Since we observe optimality gap estimates having quite small values in the SAA experiments, we focus now on the assessment of the performances of the proposed matheuristics SAM and TSM. We first consider small test instances that can also be solved by complete enumeration. Networks are generated for two different network sizes with $n = 20, 30$ and two different number of arcs $m$ for each $n$ using the WS and ER models. Five distinct networks are considered for each setting giving rise to 40 instances in total. For the cardinality-based IMPD instances, the seed selection and deactivation costs are unity. Hence $C$ represents the number of nodes selected as the seed, and $E$ designates the number of deactivated nodes. For the cost-based IMPD instances, the activation costs $c_i$ and the deactivation costs $e_i$ are generated randomly from the set $\{10, 15, 20\}$ independent from each other. Note that the threshold formulation IMPD-t is adopted in the experiments of this section as it yields slightly smaller SAA gaps and times for the network sizes considered, as is explained in Section 8.2.1.2, and the aim of the experiments of this section is to assess the matheuristic methods.

Two approaches are used for the generation of the initial solutions as explained before. We find out that for the cost-based IMPD instances, the second approach that uses only the costs yields better results. For the cardinality-based IMPD instances, the initial solution method takes into account only the individual spread values since the

costs are equal in this type of instances. In all of our experiments, the same parameter values given in Section 8.2.1.1 are used for the SAA algorithm. The $t_{\max}$ parameter is set to one hour and the best solution found is reported at the end of every 20 minutes in both matheuristics. Moreover, each of the three move types is equally likely to be selected in both SAM and TSM for the cost-based IMPD instances. Only 1-Swap is implemented for SAM and TSM when cardinality-based IMPD instances are solved. The remaining parameter values in SAM are set as $p_0 = 0.8$, $r = 0.9$, $\eta = 0.2$, and $\phi = 0.5$. In TSM, the proportion of the neighboring solutions examined, or equivalently the candidate list size of TSM is set to $\nu = 10/N$. The penalty coefficient $\mu$ for the long-term frequency-based memory is taken equal to one. All the parameter values of the two matheuristic methods are determined based on preliminary numerical analysis. The results are displayed in Table 8.4 for cardinality-based IMPD instances and in Table 8.5 for cost-based IMPD instances in terms of averages calculated over five different network instances with three replications for each one.

Table 8.4. Results obtained by SAM and TSM for cardinality-based IMPD instances.

| | $(n, m, C, E)$ | SAM Gap(%) | TSM Gap(%) | $\hat{z}$ | CE CPU Time(s) |
|---|---|---|---|---|---|
| ER | (20,40,3,1) | 0.00 | 0.00 | 7.51 | 200.27 |
| | (20,80,3,1) | 0.12 | 0.00 | 10.34 | 1357.79 |
| | (30,90,4,2) | 0.00 | 0.00 | 10.24 | 26249.00 |
| | (30,180,4,2) | 0.74 | 0.00 | 11.62 | 81564.40 |
| WS | (20,40,3,1) | 0.00 | 0.00 | 9.52 | 239.10 |
| | (20,80,3,1) | 0.11 | 0.00 | 11.61 | 493.18 |
| | (30,90,4,2) | 0.00 | 0.00 | 11.48 | 7760.87 |
| | (30,180,4,2) | 0.00 | 0.00 | 14.37 | 8339.00 |
| Average | | 0.12 | 0.00 | 10.84 | 15775.45 |

The first columns in Table 8.4 and Table 8.5 indicate the network type. The second columns show the network size in terms of $n$ and $m$ as well as the activation and deactivation budget levels $C$ and $E$. Please note that budget levels are determined in such a way that the solution time of complete enumeration is acceptable. In the next two columns, the average optimality gaps of the best solutions found by SAM and TSM are provided as a percentage of the optimal objective value $\hat{z}$ associated with each

Table 8.5. Results obtained by SAM and TSM for cost-based IMPD instances.

| | $(n, m, C, E)$ | SAM Gap(%) | TSM Gap(%) | $\hat{z}$ | CE CPU Time(s) |
|---|---|---|---|---|---|
| ER | (20,40,40,20) | 0.00 | 0.42 | 7.12 | 77.68 |
| | (20,80,40,20) | 0.00 | 0.00 | 10.28 | 459.27 |
| | (30,90,60,30) | 0.76 | 0.00 | 14.21 | 25394.17 |
| | (30,180,60,30) | 0.25 | 0.00 | 15.47 | 79144.14 |
| WS | (20,40,40,20) | 0.00 | 0.00 | 8.07 | 72.04 |
| | (20,80,40,20) | 0.00 | 0.00 | 11.23 | 317.10 |
| | (30,90,60,30) | 0.09 | 0.00 | 15.10 | 17341.01 |
| | (30,180,60,30) | 1.09 | 0.83 | 18.70 | 33465.82 |
| Average | | 0.27 | 0.16 | 12.52 | 19533.90 |

instance. The averages of $\hat{z}$ values and complete enumeration times over five instances are provided in the last two columns.

Before evaluating the results of the matheuristics we see that for the same size and budget values, WS instances always yield a larger average objective value. We attribute this outcome to the highly clustered structure of the WS networks since the number of arcs is the same in WS and ER networks. On the basis of the optimality gaps reported in both tables, we can observe that they are quite satisfactory. Especially in the case of cardinality-based IMPD instances, using only 1-Swap moves seems to indicate a better performance in terms of searching the solution space. In addition, TSM yields smaller optimality gaps within the time limit. One possible reason may be that the actual running time of SAM is less than one hour on the average, because it terminates before $t_{\max}$ when it gets stuck at a local optimal solution. To analyze this issue, we compare the intermediate optimality gaps of all instances with $n = 30$, which are presented in Table 8.6. The third-to-fifth columns are the end-of-$\frac{1}{3}$hour average optimality gaps over 20 instances and three replications. The last column gives the average running time of the algorithms. It can be seen that SAM terminates before TSM. However, TSM outperforms SAM at all epochs, even at the end of the first 1/3 hour in all instance types. Besides, the average gaps at the end of 1/3 hour are below 2% for TSM, whereas the solution of these instances via complete enumeration takes hours especially for ER instances.

Table 8.6. Results of SAM and TSM for $n = 30$.

| Instance | Matheuristic | $\text{Gap}_1$ | $\text{Gap}_2$ | Final Gap | CPU time |
|---|---|---|---|---|---|
| Cardinality-based | TSM | 1.71 | 0.00 | 0.00 | 3600.02 |
| | SAM | 6.65 | 4.79 | 0.19 | 2409.36 |
| Cost-based | TSM | 1.73 | 0.55 | 0.21 | 3590.25 |
| | SAM | 10.39 | 5.50 | 0.55 | 1734.76 |

8.2.2.2. Comparison of the Outputs of the Matheuristics. In this section, we compare the performances of our matheuristics on IMPD instances the optimal objective values of which cannot be attained by complete enumeration due to excessive solution times. To this end, we consider the same 100 instances (separately for cardinality-based and cost-based cases) used in Section 8.2.1.2, where $n \in \{20, 40, 60, 80, 100\}$ and $m \in \{2n, 4n\}$. The activation budget values are determined in such a way that the leader can select at most 10% of the nodes as the seed set in the network and the follower can deactivate at most half of the nodes in the seed set. The SAM objective values are used as the baseline and the relative performance of TSM is reported as $\Delta = 100 \times (\hat{z}^*_{TSM} - \hat{z}^*_{SAM})/\hat{z}^*_{SAM}$. Here, $\hat{z}^*_{SAM}$ ($\hat{z}^*_{TSM}$) denotes the average objective value that SAM (TSM) yields over ten instances with the same $n$ value. A positive $\Delta$ value indicates that TSM finds better solutions on the average.

In Table 8.7 we report the results by presenting the end-of-$\frac{1}{3}$hour performances. The $t_{\max}$ parameter is set to one hour except for the instances with $n \in \{80, 100\}$ and $m = 2n$ because they require significantly larger solution times by the SAA method compared to others. For these instances, $t_{\max}$ is set to three hours and $\Delta_1$, $\Delta_2$, and $\Delta_3$ correspond to the results at the end of each hour. The results for the cost-based instances show that TSM outperforms SAM in most of the instances, even though its superiority is not consistent in terms of elapsed time. For both network types, average relative differences are positive at all epochs. The superiority of TSM is more apparent for those with $n = 100$. This suggests that TSM is more advantageous for larger problem instances. For the cardinality-based instance results on the other hand, it is seen that the relative differences are not positive at all epochs. Especially for the WS instances, SAM performs better in the first two epochs. Nevertheless, the average

values in the last column of cardinality-based block are positive, which implies that SAM is outperformed by TSM as more CPU time is allotted.

Table 8.7. Relative difference of the objective values found in SAM and TSM.

| | | Cardinality-based | | | Cost-based | | |
|---|---|---|---|---|---|---|---|
| Model | $n$ | $\Delta_1$ | $\Delta_2$ | $\Delta_3$ | $\Delta_1$ | $\Delta_2$ | $\Delta_3$ |
| ER | 20 | 0.00 | 0.00 | 0.00 | 0.79 | 0.79 | 0.79 |
| | 40 | 1.55 | 0.42 | 0.60 | 4.82 | 4.44 | 4.44 |
| | 60 | 4.38 | 4.86 | 5.30 | 8.18 | 1.39 | -0.42 |
| | 80 | -2.40 | -4.28 | 0.80 | 1.34 | 2.04 | 1.24 |
| | 100 | 8.78 | 4.45 | 2.00 | 9.80 | 12.31 | 14.10 |
| | Average | 2.46 | 1.09 | 1.74 | 4.99 | 4.19 | 4.03 |
| WS | 20 | 0.00 | 0.00 | 0.00 | 0.06 | 0.06 | 0.06 |
| | 40 | 1.02 | 0.80 | 0.81 | 0.25 | 0.34 | 0.34 |
| | 60 | -10.57 | -8.37 | 0.95 | 2.03 | 0.85 | 1.54 |
| | 80 | -11.79 | -2.24 | 0.13 | 2.06 | 1.03 | 2.69 |
| | 100 | -9.08 | -2.69 | -0.14 | 7.71 | 9.01 | 9.16 |
| | Average | -6.08 | -2.50 | 0.35 | 2.42 | 2.26 | 2.76 |
| Average | | -1.81 | -0.71 | 1.04 | 3.70 | 3.23 | 3.40 |

8.2.2.3. Results Obtained by Simple Heuristics. In this section, we consider the same problem instances considered in the previous section and solve them using the simple heuristic methods described in Section 4.2.3.3 to better assess the benefit of SAM and TSM. The results are presented in Table 8.8 and Table 8.9 in terms of the average percent difference between the best objective value obtained via a simple heuristic method and the final objective value that each matheuristic yields within the time limit. The values are calculated by taking the averages over 50 instances of the relevant network type. For the cardinality-based instances, the benefit of the matheuristics turns out to be larger for the WS instances as presented in Table 8.8. In case of the cost-based instances the results of which can be seen in Table 8.9, it is clear that the degree-based heuristic SH-OD is the worst strategy, while SH-SP produces the best solutions. The most probable reason for this outcome seems to be the property of SH-SP that computes the spread values for each node individually before selecting them as seed nodes and also relies on the unit activation cost values.

Table 8.8. Comparison of the simple heuristics with SAM and TSM for cardinality-based instances.

| | Matheuristic | SH-OD | SH-SP |
|---|---|---|---|
| ER | SAM | 27.37 | 35.77 |
| | TSM | 29.30 | 37.71 |
| WS | SAM | 46.27 | 86.66 |
| | TSM | 46.73 | 85.70 |
| Average | SAM | 36.82 | 61.22 |
| | TSM | 38.01 | 61.71 |

Table 8.9. Comparison of the simple heuristics with SAM and TSM for cost-based instances.

| | Matheuristic | SH-AC | SH-DC | SH-OD | SH-SP |
|---|---|---|---|---|---|
| ER | SAM | 46.07 | 48.63 | 108.77 | 26.74 |
| | TSM | 49.86 | 51.77 | 112.79 | 30.14 |
| WS | SAM | 46.02 | 47.59 | 141.65 | 27.54 |
| | TSM | 49.02 | 50.54 | 144.79 | 29.93 |
| Average | SAM | 46.04 | 48.11 | 125.21 | 27.14 |
| | TSM | 49.44 | 51.16 | 128.79 | 30.03 |

8.2.2.4. The Effect of the Formulation on Tabu Search Based Matheuristic Results. After observing that the TSM performs slightly better than the SAM method on the average (especially the cost-based and relatively larger instances), now we are interested in the effect of the choice of IMPD formulation on the TSM results. We present in Table 8.10 and Table 8.11 the results related to the best objective values found by TSM with both of the developed formulations, IMPD-t and IMPD-La. The *Nfs* column provides the number of instances among the 20 instances with the same $n$ and $m$ values for which no feasible solution can be obtained by the threshold model IMPD-t within time limit allotted. Note that we provide the *Nfs* column only for the threshold formulation since the live-arc formulation is able to generate within the time limit feasible leader solutions for all 20 instances at a given $n$ and $m$ combination. The columns titled "Threshold" and "Live-arc" contain the averages of the best objective values found over the instances for which the threshold formulation is able to yield a feasible solution. For example, the threshold formulation can provide a feasible solution for 14 of the 20 ER instances with $n = 200$ and $m = 800$, in cardinality-based

case. Therefore, the average objective values are computed over those 14 instances. For the WS instances with $n = 1000$ and $m = 4000, 6000, 8000$, $Nfs = 20$ implies that an average cannot be calculated for the threshold model, and thus we present for the live-arc model the average objective value over the 20 instances.

Note that in the experiments with live-arc scenarios, the objective value of the best solution found is re-evaluated with a sample of $10,000$ threshold scenarios and that objective value is used in the analysis of the results. Since the same sample is used to evaluate the final objective values, if both models yield the same leader and follower solution pair, then their objective value estimates are the same. Lastly, in the column named $\Delta$, we present the average of the percent changes in the objective values when the live-arc formulation is used. The change is calculated as $100 \times (z_{La} - z_{Th})/z_{Th}$ for a single instance where $z_{La}$ and $z_{Th}$ denote the best objective value with the live-arc and threshold models, respectively. If an objective value is not available for the threshold formulation, then the change is recorded as 100% for that individual instance.

The numbers show that the average change (or improvement) in the objective value starts at a negative value for the smallest value of $m$ considered for a given value of $n$, and increases as the value of $m$ increases except for a few fluctuations. As the live-arc model renders the SAA algorithm stable and robust to the changes in $m$ and the network structure, it is possible to carry out a more comprehensive search by extending the time limit, if desired. A modest increase in the time limit would not help in the threshold case except for the smallest $m$ values considered, as the SAA times increase exponentially.

Next, we analyze the effect of the budget amount on the spread improvement. As described in Section 8.2.1.2, four different budget levels are determined for each of the test instances depending on the number of nodes, $n$. While the first level corresponds to the smallest $C$ and $E$ values, the fourth is associated with the largest budget values considered for that $n$ value. We refer to these levels as budget classes (BC) independent of $n$, and examine the relationship between the budget class and

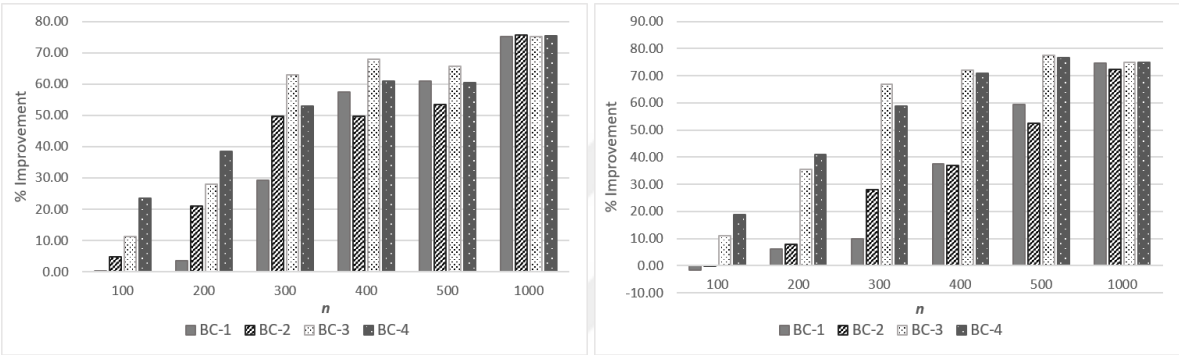
Table 8.10. Results of the TSM for cardinality-based instances.

| | | WS | | | | ER | | | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | $m$ | #Nfs | Threshold | Live-arc | $\Delta$ | #Nfs | Threshold | Live-arc | $\Delta$ |
| 100 | 100 | 0 | 15.72 | 15.30 | -1.97 | 0 | 16.76 | 16.67 | -0.32 |
| | 200 | 0 | 28.06 | 27.43 | -2.17 | 0 | 25.58 | 25.72 | 0.77 |
| | 300 | 0 | 37.74 | 38.41 | 2.13 | 0 | 32.53 | 34.38 | 5.45 |
| | 400 | 0 | 46.54 | 54.24 | 22.10 | 0 | 39.89 | 45.00 | 13.42 |
| | 500 | 0 | 50.22 | 58.53 | 22.47 | 0 | 43.56 | 46.30 | 6.43 |
| | 600 | 0 | 52.40 | 61.93 | 22.96 | 1 | 49.90 | 51.55 | 8.32 |
| | 700 | 0 | 53.10 | 64.41 | 27.55 | 0 | 49.23 | 52.71 | 7.51 |
| | 800 | 0 | 55.97 | 65.01 | 20.30 | 0 | 51.49 | 53.68 | 4.33 |
| *Average* | | 0.00 | 42.47 | 48.16 | 14.17 | 0.13 | 38.62 | 40.75 | 5.74 |
| 200 | 200 | 0 | 29.81 | 26.53 | -9.59 | 0 | 30.71 | 27.89 | -8.70 |
| | 400 | 0 | 43.74 | 36.38 | -16.98 | 0 | 45.23 | 41.69 | -8.48 |
| | 600 | 0 | 49.88 | 51.06 | 2.52 | 0 | 53.08 | 58.98 | 11.96 |
| | 800 | 0 | 54.70 | 59.05 | 10.11 | 6 | 78.27 | 84.89 | 36.62 |
| | 1000 | 1 | 62.20 | 73.72 | 25.35 | 11 | 81.56 | 86.66 | 57.91 |
| | 1200 | 1 | 74.87 | 84.26 | 19.35 | 11 | 86.82 | 94.26 | 58.97 |
| | 1400 | 1 | 80.33 | 87.34 | 14.05 | 14 | 107.73 | 109.35 | 70.65 |
| | 1600 | 3 | 89.16 | 97.32 | 24.30 | 15 | 114.44 | 119.42 | 76.08 |
| *Average* | | 0.75 | 60.59 | 64.46 | 8.64 | 7.13 | 74.73 | 77.89 | 36.88 |
| 300 | 600 | 0 | 56.85 | 52.49 | -9.03 | 0 | 60.99 | 60.18 | -1.91 |
| | 1200 | 1 | 72.45 | 78.06 | 13.43 | 14 | 110.77 | 114.67 | 71.15 |
| | 1800 | 6 | 89.16 | 95.05 | 34.81 | 20 | NA | 152.91 | 100.00 |
| | 2400 | 16 | 152.10 | 158.69 | 80.90 | 20 | NA | 168.40 | 100.00 |
| *Average* | | 5.75 | 92.64 | 96.07 | 30.03 | 13.50 | 85.88 | 124.04 | 67.31 |
| 400 | 800 | 0 | 63.96 | 59.03 | -9.41 | 0 | 75.48 | 72.62 | -4.81 |
| | 1600 | 2 | 81.89 | 91.73 | 22.09 | 19 | 158.45 | 163.97 | 95.17 |
| | 2400 | 14 | 74.71 | 83.13 | 73.64 | 20 | NA | 201.55 | 100.00 |
| | 3200 | 19 | 72.22 | 74.29 | 95.14 | 20 | NA | 223.69 | 100.00 |
| *Average* | | 8.75 | 73.20 | 77.04 | 45.37 | 14.75 | 116.96 | 165.46 | 72.59 |
| 500 | 1000 | 0 | 75.87 | 72.13 | -6.71 | 0 | 91.43 | 91.04 | -1.23 |
| | 2000 | 1 | 89.98 | 92.52 | 7.60 | 20 | NA | 182.48 | 100.00 |
| | 3000 | 16 | 81.03 | 87.65 | 81.59 | 20 | NA | 251.27 | 100.00 |
| | 4000 | 20 | NA | 155.19 | 100.00 | 20 | NA | 285.62 | 100.00 |
| *Average* | | 9.25 | 82.29 | 101.87 | 45.62 | 15.00 | 91.43 | 202.60 | 74.69 |
| 1000 | 2000 | 0 | 133.45 | 134.25 | 0.88 | 0 | 177.45 | 180.19 | 1.69 |
| | 4000 | 20 | NA | 224.92 | 100.00 | 20 | NA | 361.24 | 100.00 |
| | 6000 | 20 | NA | 241.24 | 100.00 | 20 | NA | 514.99 | 100.00 |
| | 8000 | 20 | NA | 248.01 | 100.00 | 20 | NA | 589.64 | 100.00 |
| *Average* | | 15.00 | 133.45 | 212.10 | 75.22 | 15.00 | 177.45 | 411.52 | 75.42 |

Table 8.11. Results of the TSM for cost-based instances.

| | | WS | | | | ER | | | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | $m$ | #Nfs | Threshold | Live-arc | $\Delta$ | #Nfs | Threshold | Live-arc | $\Delta$ |
| 100 | 100 | 0 | 11.42 | 10.57 | -8.33 | 0 | 9.87 | 8.82 | -10.84 |
| | 200 | 0 | 21.56 | 19.86 | -7.50 | 0 | 16.97 | 15.34 | -9.72 |
| | 300 | 0 | 34.58 | 33.82 | -2.81 | 0 | 21.35 | 21.13 | -0.90 |
| | 400 | 0 | 37.10 | 41.40 | 16.16 | 0 | 28.70 | 31.18 | 11.80 |
| | 500 | 0 | 43.64 | 50.79 | 20.28 | 0 | 32.84 | 35.77 | 10.56 |
| | 600 | 0 | 42.04 | 45.47 | 12.94 | 0 | 33.54 | 37.40 | 12.09 |
| | 700 | 0 | 38.54 | 43.66 | 18.44 | 0 | 37.19 | 41.23 | 11.35 |
| | 800 | 0 | 39.69 | 48.13 | 25.62 | 0 | 36.42 | 40.82 | 13.41 |
| *Average* | | 0.00 | 33.57 | 36.71 | 9.35 | 0.00 | 27.11 | 28.96 | 4.72 |
| 200 | 200 | 0 | 18.15 | 16.94 | -5.63 | 0 | 16.07 | 14.97 | -7.61 |
| | 400 | 0 | 32.45 | 29.01 | -10.26 | 0 | 28.27 | 24.40 | -13.90 |
| | 600 | 0 | 53.91 | 55.41 | 2.97 | 0 | 34.90 | 33.20 | -5.15 |
| | 800 | 0 | 51.73 | 57.36 | 10.66 | 0 | 44.77 | 47.05 | 6.10 |
| | 1000 | 3 | 59.98 | 68.34 | 27.44 | 6 | 46.73 | 51.23 | 36.51 |
| | 1200 | 6 | 58.87 | 72.09 | 48.46 | 10 | 51.91 | 60.09 | 58.91 |
| | 1400 | 6 | 59.19 | 73.39 | 46.84 | 10 | 64.09 | 69.82 | 54.75 |
| | 1600 | 7 | 54.88 | 72.49 | 56.93 | 10 | 66.08 | 72.34 | 55.08 |
| *Average* | | 2.75 | 48.65 | 55.63 | 22.18 | 4.50 | 44.10 | 46.64 | 23.09 |
| 300 | 600 | 0 | 47.22 | 42.58 | -10.62 | 0 | 35.83 | 31.15 | -14.62 |
| | 1200 | 5 | 82.17 | 83.49 | 27.46 | 6 | 56.66 | 60.31 | 35.10 |
| | 1800 | 12 | 86.67 | 92.08 | 63.53 | 14 | 98.64 | 99.96 | 70.55 |
| | 2400 | 11 | 81.89 | 89.41 | 60.10 | 19 | 114.47 | 121.53 | 95.31 |
| *Average* | | 7.00 | 74.49 | 76.89 | 35.12 | 9.75 | 76.40 | 78.24 | 46.58 |
| 400 | 800 | 0 | 60.53 | 53.74 | -12.21 | 0 | 44.13 | 39.23 | -12.13 |
| | 1600 | 8 | 87.79 | 94.49 | 44.95 | 10 | 62.56 | 66.45 | 53.14 |
| | 2400 | 16 | 79.49 | 79.16 | 79.95 | 20 | NA | 133.14 | 100.00 |
| | 3200 | 16 | 87.57 | 90.87 | 80.68 | 20 | NA | 161.56 | 100.00 |
| *Average* | | 10.00 | 78.84 | 79.57 | 48.34 | 12.50 | 53.35 | 100.10 | 60.25 |
| 500 | 1000 | 0 | 71.28 | 65.40 | -9.20 | 0 | 51.14 | 47.42 | -8.78 |
| | 2000 | 14 | 118.76 | 118.90 | 70.02 | 10 | 80.57 | 83.76 | 51.80 |
| | 3000 | 19 | 96.24 | 96.24 | 95.00 | 20 | NA | 166.46 | 100.00 |
| | 4000 | 20 | NA | 161.27 | 100.00 | 20 | NA | 198.66 | 100.00 |
| *Average* | | 13.25 | 95.43 | 110.45 | 63.96 | 12.50 | 65.86 | 124.08 | 60.75 |
| 1000 | 2000 | 0 | 129.99 | 129.39 | -0.46 | 0 | 84.61 | 83.75 | -1.15 |
| | 4000 | 20 | NA | 296.05 | 100.00 | 19 | 116.21 | 118.04 | 95.08 |
| | 6000 | 20 | NA | 339.57 | 100.00 | 20 | NA | 349.27 | 100.00 |
| | 8000 | 20 | NA | 310.15 | 100.00 | 20 | NA | 435.20 | 100.00 |
| *Average* | | 15.00 | 129.99 | 268.79 | 74.88 | 14.75 | 100.41 | 246.56 | 73.48 |

the percent improvement in the objective values due to the live-arc formulation. The results are displayed in Figure 8.6. The improvements are calculated as the average improvement of all the instances sharing the same $n$ value and the budget class. The results imply that as the budgets become larger, the improvement in the spread due to the use of the live-arc model increases. However, this relation becomes insignificant for larger $n$ values.




(a) Cardinality-based instances (b) Cost-based instances

Figure 8.6. Budget class vs. average improvement in the spread.

We aggregate the results over the number of nodes ($n$) and present them in Table 8.12 for both the cardinality-based and the cost-based instances. The increase in the average improvement is an indicator for the benefit of the live-arc formulation as the activation and the deactivation budgets increase.

Table 8.12. Average percent improvement in the spread based on the budget classes.

| | Cardinality-based | | Cost-based | |
|---|---|---|---|---|
| Budget Class | WS | ER | WS | ER |
| 1 | 23.47 | 34.24 | 23.49 | 25.27 |
| 2 | 25.73 | 44.33 | 24.63 | 27.50 |
| 3 | 34.80 | 52.61 | 45.54 | 51.31 |
| 4 | 36.92 | 56.44 | 49.01 | 51.89 |

Lastly, the TSM is tested on nine networks that we extract from `arXiv`, for the cardinality-based case and using the IMPD-La formulation. As described in Section 8.1, we fix $n$ at nine different values and obtain the networks with the number of arcs

$m$ given in Table 8.13. The table presents the result in terms of the best objective value obtained at the end of every two-hours, with a time limit of eight hours. We repeat the experiments for two values of proportion of evaluated neighbors in TSM, $\nu \in \{5N^{-1}, 10N^{-1}\}$. While a significant difference is not observed due to the difference in $\nu$, $\nu = 5N^{-1}$ yields slightly better objective values at each epoch which may be interpreted as evaluating more solutions at one iteration does not necessarily lead to better solutions when running time is limited. Another observation is that the objective value is improved significantly for $n \in [200, 400]$ within the time limit. It gets more difficult to find better solutions as networks get larger, whereas the solution space is searched extensively in the first few hours for the smallest two instances.

Table 8.13. Results on the co-authorship network from `arXiv`.

| | | | $\nu = 5N^{-1}$ | | | | $\nu = 10N^{-1}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $(n, m)$ | $C$ | $E$ | $z_2$ | $z_4$ | $z_6$ | $z_8$ | $z_2$ | $z_4$ | $z_6$ | $z_8$ |
| (100, 418) | 10 | 5 | 27.71 | 27.71 | 27.71 | 27.71 | 27.71 | 27.71 | 27.71 | 27.71 |
| (150, 620) | 10 | 5 | 34.78 | 34.78 | 34.78 | 34.78 | 34.78 | 34.78 | 34.78 | 35.14 |
| (200, 867) | 20 | 10 | 51.34 | 55.43 | 57.37 | 58.83 | 48.70 | 54.82 | 60.76 | 61.74 |
| (250, 1037) | 20 | 10 | 51.03 | 53.88 | 62.70 | 63.83 | 43.98 | 51.03 | 53.88 | 57.55 |
| (300, 1314) | 30 | 15 | 59.90 | 62.93 | 68.20 | 70.09 | 59.93 | 59.93 | 62.93 | 67.29 |
| (350, 1428) | 30 | 15 | 48.38 | 54.88 | 54.88 | 60.10 | 48.38 | 48.38 | 48.38 | 56.01 |
| (400, 1730) | 40 | 20 | 63.70 | 63.70 | 63.70 | 70.13 | 63.70 | 63.70 | 63.70 | 68.65 |
| (450, 1882) | 40 | 20 | 88.45 | 88.45 | 88.45 | 91.45 | 88.45 | 88.45 | 88.45 | 92.00 |
| (500, 2214) | 50 | 25 | 93.30 | 93.30 | 93.30 | 98.13 | 93.86 | 93.86 | 93.86 | 97.88 |
| Average | | | 57.62 | 59.45 | 61.23 | 63.89 | 56.61 | 58.07 | 59.38 | 62.66 |

## 8.3. Results of the Branch-and-Cut Algorithm with Intersection Cuts

In this section we present the results of the experiments when the branch-and-cut algorithm described in Chapter 5 is used to solve the IMPD assuming a finite number of live-arc scenarios. We investigate the effect of the Alternating Bound Update mechanism and the choice of bilevel-free set on solution time and quality. As is described within the implementation details in Section 5.4, the B&C algorithm of CPLEX is modified via callback functions. The version 12.9 is used to benefit from new Callable Library functions. Note that we restrict our attention to small-world networks, therefore we use only the Watts-Strogatz model to generate the problem instances. We

sample the live-arc scenarios with Latin Hypercube Sampling method.

### 8.3.1. Solution Time Reduction Due to Alternating Bound Update

We generate five WS networks for each $n \in \{20, 30, 40\}$ and $d \in \{0.1, 0.2\}$ combination where $n$ is the number of nodes and $d$ is the density of the network. Four different values of the number of scenarios $R \in \{1, 10, 25, 50\}$ are determined and it is assumed that the scenarios have equal probabilities, i.e., $p_r = 1/R$. The time limit is determined as one hour. In all of the instances, it is assumed that $k = 4$ and $h = 2$. The preliminary experiments have revealed that including the bilevel feasibility constraints (5.9) and (5.10) in the HPR formulation results in $-41\%$, $-32\%$, $5\%$ and $8\%$ change in average solution times for $R = 1, 10, 25$ and 50, respectively, for BFS1 which is considered as the basic BFS. The number of optimal solutions found within the time limit increases from 84 to 90. Therefore, we include these constraints in HPR in all of the experiments while the BFS definitions remain the same as described in Section 5.2.

We present the percent decrease in average solution times as a result of Alternating Bound Update (ABU) procedure in Table 8.14 for each of the bilevel-free sets considered. The numbers are calculated by the formula $100 \times (t_1 - t_2)/t_1$, where $t_1$ and $t_2$ represent the average solution times (in seconds) without and with ABU, respectively, over ten instances with the same $n$ and $R$ values. The bold written numbers indicate that either both $t_1$ and $t_2$ or only $t_1$ is equal to 3600 seconds, i.e., the time limit. In the former case, the improvement is calculated as zero. In the latter, the computed reduction is actually a lower bound on the true value because if the algorithm is allowed to run until termination, the average solution time $t_1$ would be larger. It is clear that the bilevel-free sets (BFS) 4 and 5 benefit from ABU most. Although the reductions seem significant for all BFS's, the average improvement is larger than the others for these two alternatives, while BFS6 yields the smallest average improvement. The values of $t_1$ and $t_2$ are provided in Table B.1 and Table B.2. It is possible to see there that BFS6 yields the smallest average times both with and without ABU implementation.

Table 8.14. Percent reduction in solution times due to ABU.

| | | Bilevel Free Set (BFS) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $n$ | $R$ | 1 | 2 | 3 | 4 | 5 | 6 | HC |
| 20 | 1 | 13.27 | 18.75 | 19.44 | 37.21 | 14.17 | 13.47 | 20.62 |
| | 10 | 15.86 | 32.94 | 35.23 | 73.55 | 74.07 | 6.14 | 17.46 |
| | 25 | 35.47 | 40.19 | 43.46 | 82.43 | 86.12 | 23.22 | 37.30 |
| | 50 | 43.84 | 42.54 | 39.84 | ***54.12*** | ***62.28*** | 25.98 | 36.90 |
| *Average* | | 27.11 | 33.60 | 34.49 | 61.83 | 59.16 | 17.20 | 28.07 |
| 30 | 1 | 9.60 | 36.50 | 19.24 | 58.06 | 39.64 | 10.01 | 25.05 |
| | 10 | 38.64 | 40.39 | 38.68 | ***78.35*** | ***76.59*** | 16.06 | 36.17 |
| | 25 | ***0.00*** | ***0.00*** | ***1.21*** | ***1.48*** | ***5.51*** | ***2.34*** | ***1.98*** |
| | 50 | ***0.00*** | ***0.00*** | ***0.00*** | ***0.00*** | ***0.00*** | ***0.00*** | ***0.00*** |
| *Average* | | 12.06 | 19.22 | 14.78 | 34.48 | 30.44 | 7.10 | 15.80 |
| 40 | 1 | 14.88 | 30.06 | 25.43 | 45.94 | 37.33 | 12.61 | 9.88 |
| | 10 | 19.87 | 15.72 | 10.40 | 56.14 | 61.30 | 8.37 | 3.35 |
| | 25 | ***0.00*** | ***0.00*** | ***0.00*** | ***0.00*** | ***7.80*** | ***2.07*** | ***0.00*** |
| | 50 | ***0.00*** | ***0.00*** | ***0.00*** | ***0.00*** | ***0.00*** | ***0.00*** | ***0.00*** |
| *Average* | | 8.69 | 11.45 | 8.96 | 25.51 | 26.61 | 5.76 | 3.31 |

### 8.3.2. The Comparison of Bilevel-free Sets

In Table 8.15 and Table 8.16, the total number of optimally solved instances is presented as aggregated over $R$ and $n$, respectively. The optimal objective values are obtained via complete enumeration. The numbers in parenthesis show the relevant data for the case where ABU is not used. The results show that the set BFS6 we propose in Section 5.2 yields the most effective separation alternative with and without ABU. The effect of facet removals can be understood from the pairwise differences between BFS1, BFS2 and BFS3, as well as the difference between BFS4 and BFS5. Recall that the BFS1 and BFS4 are defined using different follower solutions, while the BFS2 BFS3 and BFS5 are obtained through the removal of some facets from them. It turns out that removing unnecessary facets improve the performance. Another implication is that the use of ABU improves the algorithm especially for BFS4 and BFS5, which is in line with the findings presented in Table 8.14. Please see Table B.3 and Table B.4 in Appendix B for the number of instances for which the optimality is proved by CPLEX.

Table 8.15. Number of optimally solved instances.

| $n$ | BFS1 | BFS2 | BFS3 | BFS4 | BFS5 | BFS6 | HC |
|---|---|---|---|---|---|---|---|
| 20 | 40 (40) | 40 (40) | 40 (40) | 40 (30) | 40 (30) | 40 (40) | 40 (39) |
| 30 | 29 (24) | 30 (27) | 31 (27) | 29 (13) | 29 (12) | 35 (28) | 31 (26) |
| 40 | 28 (25) | 28 (26) | 28 (30) | 29 (22) | 32 (24) | 34 (31) | 32 (29) |
| Total | 97 (89) | 98 (93) | 99 (97) | 98 (65) | 101 (66) | 109 (99) | 103 (94) |

Table 8.16. Number of optimally solved instances.

| $R$ | BFS1 | BFS2 | BFS3 | BFS4 | BFS5 | BFS6 | HC |
|---|---|---|---|---|---|---|---|
| 1 | 30 (30) | 30 (30) | 30 (30) | 30 (30) | 30 (30) | 30 (30) | 30 (30) |
| 10 | 30 (30) | 30 (30) | 30 (30) | 30 (22) | 30 (21) | 30 (30) | 30 (30) |
| 25 | 24 (17) | 24 (21) | 24 (22) | 24 (10) | 26 (12) | 30 (25) | 22 (20) |
| 50 | 13 (12) | 14 (12) | 15 (15) | 14 (3) | 15 (3) | 19 (14) | 21 (14) |
| Total | 97 (89) | 98 (93) | 99 (97) | 98 (65) | 101 (66) | 109 (99) | 103 (94) |

Another important measure is the objective value of the best solution found for the instances that cannot be solved optimally within the time limit. The average optimality gaps are presented in Table 8.17 and Table 8.18. As before, the value in parenthesis shows the value when ABU is not used. While all of the average gaps are small when ABU is used, BFS6 yields the best results. It is observed that the increase of $R$ from 25 to 50 result in the largest deterioration in solution quality.

Table 8.17. Average optimality gaps.

| $n$ | BFS1 | BFS2 | BFS3 | BFS4 | BFS5 | BFS6 | HC |
|---|---|---|---|---|---|---|---|
| 20 | 0.00 (0.00) | 0.00 (0.00) | 0.00 (0.00) | 0.00 (2.19) | 0.00 (1.53) | 0.00 (0.00) | 0.00 (0.08) |
| 30 | 1.35 (2.40) | 0.94 (2.08) | 0.97 (1.66) | 1.03 (14.49) | 1.40 (14.84) | 0.54 (1.71) | 0.90 (1.65) |
| 40 | 3.00 (2.97) | 2.33 (3.25) | 2.02 (2.12) | 3.04 (8.99) | 2.21 (8.70) | 1.05 (1.55) | 1.39 (2.65) |
| Avg. | 1.45 (1.79) | 1.09 (1.78) | 1.00 (1.26) | 1.36 (8.55) | 1.20 (8.36) | 0.53 (1.08) | 0.76 (1.46) |

### 8.3.3. Assessment of the Branch-and-Cut Algorithm with BFS6

The results of the experiments of the previous section have revealed that BFS6 is superior to the other bilevel-free sets considered and the use of ABU has a significant impact on the performance of the algorithm. Therefore, we conduct more experiments while holding to the choice of BFS and the use of ABU. In addition to the previously generated instances we generate five networks for each combination of $n \in \{50, 60\}$

Table 8.18. Average optimality gaps.

| $R$ | BFS1 | BFS2 | BFS3 | BFS4 | BFS5 | BFS6 | HC |
|---|---|---|---|---|---|---|---|
| 1 | 0.00 (0.00) | 0.00 (0.00) | 0.00 (0.00) | 0.00 (0.00) | 0.00 (0.00) | 0.00 (0.00) | 0.00 (0.00) |
| 10 | 0.00 (0.00) | 0.00 (0.00) | 0.00 (0.00) | 0.00 (1.85) | 0.00 (1.67) | 0.00 (0.00) | 0.00 (0.00) |
| 25 | 0.99 (2.11) | 1.31 (1.82) | 0.59 (1.06) | 0.77 (8.79) | 0.55 (8.86) | 0.00 (0.80) | 1.00 (1.73) |
| 50 | 4.81 (5.04) | 3.06 (5.29) | 3.40 (3.98) | 4.66 (23.57) | 4.26 (22.92) | 2.12 (3.53) | 2.04 (4.11) |
| Avg. | 1.45 (1.79) | 1.09 (1.78) | 1.00 (1.26) | 1.36 (8.55) | 1.20 (8.36) | 0.53 (1.08) | 0.76 (1.46) |

and $d \in \{0.1, 0.2\}$. All of the seed sizes are determined as $k = 6$ and $h = 3$. Since the best bounds found by CPLEX are not tight, we use the complete enumeration (CE) algorithm with one-hour time limit as the benchmark method. The results are presented in Table 8.19. $z_{CE}$ and $z_{BC}$ denote the average best objective values found by complete enumeration and B&C algorithms, respectively. The last two columns hold the corresponding solution times. It is observed that while the algorithms perform similarly for $n \in \{20, 30\}$ and $R = 1$, B&C solution time is affected by the increase in the value of $R$ more, for these $n$ values. Consequently, it is slightly outperformed by CE for $n = 20, 30$. However, as $n$ increases it performs significantly better, even though the time limit is reached in most of the instances, with both CE and B&C.

In Table 8.20 we present the average percent difference in the objective values found by CE and B&C. The difference for a single problem instance $i$ is calculated using the following formula: $100 \times (z^i_{BC} - z^i_{CE})/z^i_{CE}$. The numbers show that while B&C algorithm cannot beat the CE for relatively small $n$ and large $R$, there is a threshold size after which CE becomes inefficient and B&C yields larger improvement in the objective value while the values of $n$ and $R$ increase.

## 8.4. Results for the Misinformation Spread Minimization Problem

This section involves the numerical results of the experiments for the Misinformation Spread Minimization Problem (MSMP) proposed in Chapter 6. Firstly, the effect of integrating the IMM algorithm (Tang *et al.*, 2015) into our methods, i.e., Tabu Search based matheuristic (TSM), Greedy Protection Heuristic (GPH) and Binary Search algorithm (BSA), is investigated. After an assessment of these methods

Table 8.19. Assessment of the B&C algorithm.

| $n$ | $R$ | $z_{CE}$ | $z_{BC}$ | Time CE | Time BC |
|---|---|---|---|---|---|
| 20 | 1 | 8.50 | 8.50 | 86.74 | 69.17 |
| | 10 | 8.46 | 8.46 | 147.92 | 306.45 |
| | 25 | 8.56 | 8.56 | 211.92 | 2436.33 |
| | 50 | 8.51 | 8.48 | 295.94 | 3600.19 |
| *Average* | | *8.51* | *8.50* | *185.63* | *1603.04* |
| 30 | 1 | 12.70 | 12.70 | 1036.44 | 1153.42 |
| | 10 | 13.29 | 13.12 | 1838.17 | 3600.01 |
| | 25 | 13.47 | 12.64 | 2657.44 | 3600.19 |
| | 50 | 12.95 | 11.85 | 3600.29 | 3600.70 |
| *Average* | | *13.10* | *12.58* | *2283.08* | *2988.58* |
| 40 | 1 | 15.80 | 16.80 | 3602.63 | 2924.93 |
| | 10 | 13.17 | 17.23 | 3601.92 | 3600.20 |
| | 25 | 11.38 | 16.30 | 3602.01 | 3601.18 |
| | 50 | 9.54 | 14.51 | 3604.28 | 3600.59 |
| *Average* | | *12.47* | *16.21* | *3602.71* | *3431.72* |
| 50 | 1 | 14.10 | 20.10 | 3607.84 | 3600.00 |
| | 10 | 9.99 | 20.91 | 3608.59 | 3600.04 |
| | 25 | 9.12 | 19.15 | 3608.90 | 3600.24 |
| | 50 | 8.30 | 15.32 | 3608.72 | 3601.80 |
| *Average* | | *10.38* | *18.87* | *3608.51* | *3600.52* |
| 60 | 1 | 10.90 | 23.10 | 3624.95 | 3600.02 |
| | 10 | 9.21 | 24.03 | 3626.77 | 3600.12 |
| | 25 | 8.22 | 22.11 | 3625.94 | 3600.64 |
| | 50 | 7.75 | 16.99 | 3624.97 | 3601.17 |
| *Average* | | *9.02* | *21.56* | *3625.66* | *3600.49* |

Table 8.20. Average percent differences in the objective values of B&C and CE.

| | $n$ | | | | |
|---|---|---|---|---|---|
| $R$ | 20 | 30 | 40 | 50 | 60 |
| 1 | 0.00 | 0.00 | 6.56 | 44.50 | 112.42 |
| 10 | 0.00 | -1.28 | 32.08 | 113.24 | 163.09 |
| 25 | 0.00 | -6.23 | 44.88 | 113.07 | 171.92 |
| 50 | -0.41 | -8.43 | 52.60 | 84.25 | 122.72 |
| Average | -0.10 | -3.99 | 34.03 | 88.77 | 142.54 |

based on the optimality gaps that are obtained through complete enumeration, they are investigated on relatively larger problem instances as well as on subgraphs extracted from a real social network.

The experimental details common to all algorithms considered in this section can be summarized as follows. The MILP solver is determined as CPLEX 12.7. The sample size parameters of the SAA algorithm are set to values used to solve the lower-level of the Influence Maximization Problem with Deactivation in Section 8.2 after preliminary tests, i.e., $M = 20$, $N = 50$, $N' = 2000$, and $N'' = 10,000$. Although Güney (2017) reports that the SAA method yields very small optimality gap estimates for IMP when slightly larger sample sizes ($M$ and $N$) are used, we opt for the aforementioned sample sizes since the average optimality gap estimate of the follower's objective has been found around 1.1% in our experiments. Please recall that the SAA method overestimates the optimality gap as is described in Section 6.2.1. Moreover, it is not practical for us to increase the sample sizes because the problem is modelled by a bilevel-integer programming formulation. Another implementation detail about the SAA algorithm is the sampling method. As is the case for the IMPD, we observed that using Latin Hypercube Sampling (LHS) helps reducing the estimated gap as well as its variance. Therefore, LHS is preferred to simple random sampling while sampling live-arc scenarios. For the sake of fairness, the same sample of size 10,000 is used to evaluate the objective values of the solutions that IMM algorithm produces. The parameters of IMM are set to the following values as recommended by Tang *et al.* (2015): $\epsilon = 0.5$ and $l = 1$.

### 8.4.1. IMM Algorithm Integration Results

In the first part of the experiments, three network sizes $n = 20, 30, 40$ and two density values $d = 0.1, 0.2$ are determined. For each size and density combination five distinct networks are generated according to the WS model, so as to obtain 30 instances in total. The size of the protected set $h$ and the size of the seed set $k$ are set equal to each other assuming that the players have balanced resources. The approximate

optimal objective $\hat{z}$ is found via complete enumeration of the leader solutions for each instance. TSM is executed with a time limit of one hour and the fraction of the neighborhood solutions evaluated is determined as $\nu = 10/n$ in all experiments. The initial leader solution in the TSM is determined as the set of $h$ nodes with largest total weights on their outgoing arcs. GPH is run without a time limit since it terminates when a feasible solution is obtained. BSA is run until termination as well, for the values of threshold parameter $\beta \in \{0, 0.25, 0.50, 0.75, 1\}$. First, we investigate the benefit of using IMM in solving the follower's problem. The results are shown in Table 8.21 as averages over 10 instances having the same $n$ and $h$ values. For BSA, they correspond to $\beta = 0.50$.

The number of necessary SAA calls in the complete enumeration method is actually the number of feasible leader solutions, $\binom{n}{h}$ when the IMM algorithm is not applied. The number of SAA calls in GPH would be equal to $\sum_{i=0}^{h-1}(n-i)$, if IMM or another heuristic method were not used. In BSA without IMM integration, SAA is implemented completely in only leaf nodes while the last stage of it is discarded in the other nodes since an objective value is not required there. We only consider the complete executions of SAA here. These numbers are given in the columns titled "IMM-". In the columns next to them, the average numbers of SAA calls realized in our experiments with IMM integration are presented.

Table 8.21. Change in the number of SAA calls.

| | | Complete Enum. | | GPH | | BSA | |
|---|---|---|---|---|---|---|---|
| n | $\boldsymbol{h}$ | IMM− | IMM+ | IMM− | IMM+ | IMM− | IMM+ |
| 20 | 2 | 190 | 21.60 | 39 | 9.60 | 14.40 | 3.40 |
| 30 | 3 | 4060 | 122.50 | 87 | 16.70 | 58.90 | 3.80 |
| 40 | 3 | 9880 | 209.90 | 117 | 14.80 | 1613.90 | 5.70 |
| Average | | 4710.00 | 118.00 | 81.00 | 13.70 | 562.40 | 4.30 |

It is clear that less SAA calls has the advantage of reducing the CPU time. In the experiments which are used to prepare Table 8.21, it is observed that the average time reductions due to IMM are as follows: 55%, 74% and 88% in complete enumeration; 43%, 62% and 62% in GPH; and 51%, 71% and 56% in BSA; for the instances with $n = 20$, $n = 30$ and $n = 40$, respectively. Since TSM is run for a predetermined

duration of time, i.e., $t_{\max}$, we can evaluate the effect of IMM by the number of solutions visited in the same time interval. The results indicate that the number of solutions visited in one hour increases by 124% on the average.

### 8.4.2. Assessment of Tabu Search Based Matheuristic and Greedy Protection Heuristic

The results of the experiments described in the previous section are displayed in Table 8.22. The network size and cardinality parameters are given in the first column. In the next block consisting of columns two and three, the running times and optimal objective values associated with complete enumeration are shown. The remaining columns present the running times and the optimality gaps of the final solutions obtained by TSM and GPH.

Table 8.22. Assessment of TSM and GPH.

| | Complete Enum. | | TSM | | GPH | |
|---|---|---|---|---|---|---|
| $(n, d, h, k)$ | Time(s) | $\hat{z}$ | Time(s) | Gap (%) | Time(s) | Gap (%) |
| (20,0.1,2,2) | 201.00 | 6.99 | 276.00 | 0.00 | 54.60 | 3.28 |
| (20,0.2,2,2) | 232.20 | 8.90 | 351.60 | 0.00 | 69.20 | 0.00 |
| (30,0.1,3,3) | 6481.00 | 13.87 | 3600.00 | 0.00 | 265.20 | 0.96 |
| (30,0.2,3,3) | 6782.80 | 13.55 | 3600.00 | 0.00 | 217.40 | 0.95 |
| (40,0.1,3,3) | 22097.60 | 18.84 | 3600.00 | 0.00 | 344.80 | 0.19 |
| (40,0.2,3,3) | 18992.60 | 15.69 | 3600.00 | 0.00 | 243.20 | 0.22 |
| Average | 9131.20 | 12.97 | 2504.60 | 0.00 | 199.07 | 0.93 |

An interesting result is that the running times of TSM for the instances with $n = 20$ are larger than those of complete enumeration. Notice that TSM algorithm terminates before the time limit in those instances as there is not any unprecedented neighbor at the last iteration. The reason is that TSM calls IMM while deciding the best neighbor of the current solution at each iteration, thus SAA is called at least once at each iteration. Besides, the solutions in the neighborhood are pre-evaluated to be assigned a score for the candidate list. Thus, exploring even a very small feasible region takes longer in TSM. However, it pays off and as can be seen in the next column TSM finds the optimal solutions within one hour in all of the instances while the complete

enumeration takes hours for most of them. The average optimality gap of GPH is 0.93%, which is acceptable, especially when the running times are considered. It is observed that both methods produce optimal or near optimal solutions for the small test instances with only one exception, which occurs for GPH in solving an instance with parameters $(20, 0.1, 2, 2)$. GPH proves itself as more time efficient than TSM.

### 8.4.3. Assessment of the Binary Search algorithm

Recall that, in this method the value of the parameter $\beta$ determines the length of the candidate branching variable list. To observe the impact of $\beta$ on the performance of Algorithm 6.7, experiments have been conducted for five different values of $\beta$ as mentioned in Section 8.4.1. Table 8.23 shows the results of the instances described in Section 8.4.1 in terms of the number of SAA calls ($\#_{SAA}$) and the number of tree nodes ($\#tn$) generated during the search. The numbers represent the averages over 10 instances with the same $n$ value. It is clear that listing only the seed nodes of the current follower solution as candidates for branching ($\beta = 1$) results in a very small search tree. In only a few of the leaf nodes SAA is needed as the average value of 2.53 indicates. As we decrease the threshold $\beta$, the number of tree nodes increases quickly. Especially for $n = 40$ this increase is remarkable. The number of SAA calls is not affected by the threshold decrease as much as the number of tree nodes is affected. If we roughly assume that the IMM algorithm is called for half of the tree nodes (at every branching one of the nodes is created with a "protect" decision), it is clear that the complexity of the Binary Search algorithm (BSA) depends mostly on the complexity of the IMM, not the SAA for small $\beta$.

Table 8.23. Number of SAA calls and tree nodes in BSA for different $\beta$.

| | $\beta = 1.00$ | | $\beta = 0.75$ | | $\beta = 0.50$ | | $\beta = 0.25$ | | $\beta = 0.00$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $\#_{SAA}$ | $\#tn$ | $\#_{SAA}$ | $\#tn$ | $\#_{SAA}$ | $\#tn$ | $\#_{SAA}$ | $\#tn$ | $\#_{SAA}$ | $\#tn$ |
| 20 | 2.30 | 11.60 | 2.50 | 16.40 | 3.40 | 36.00 | 4.50 | 70.20 | 6.20 | 126.80 |
| 30 | 3.10 | 44.60 | 2.80 | 62.40 | 3.80 | 140.80 | 7.70 | 649.20 | 10.60 | 2915.00 |
| 40 | 2.20 | 43.20 | 4.50 | 818.40 | 5.70 | 3331.60 | 6.90 | 4270.20 | 7.30 | 4406.20 |
| Avg. | 2.53 | 33.13 | 3.27 | 299.07 | 4.30 | 1169.47 | 6.37 | 1663.20 | 8.03 | 2482.67 |

Figure 8.7 and Figure 8.8 display the average percent gaps and the average running times of the BSA for all $\beta$ values considered. As it is projected, branching only on the seed nodes and almost-seed nodes (described before) turns out to be a poor strategy. Even though it is very fast, the average optimality gap $\beta = 1$ is 10.28%. Reducing $\beta$ to 0.50 seems to provide a significant decrease in the optimality gaps with an average of 4.19% while the running times are still small compared to complete enumeration times which are indicated below the chart. Decreasing the threshold value more results in smaller gaps on the average. However, for the cases with $n = 40$ it is observed that the improvement is not distinct after $\beta = 0.50$, despite the fact that 30% more tree nodes are generated when $\beta = 0.25$. We cannot compute optimality gaps for larger network sized due to excessive complete enumeration times. The instances that were solved showed that considering all nodes that are reachable from the seed for branching is not necessary to obtain a good solution. However, both GPH and TSM perform better than BSA in terms of solution quality and time. Therefore, we do not consider the BSA for further experiments.

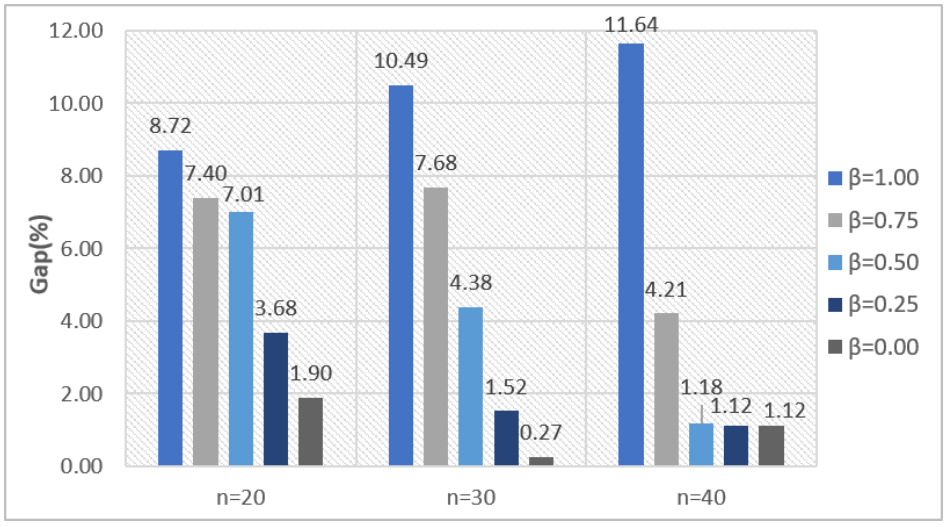



Figure 8.7. Effect of BSA parameter $\beta$ on optimality gap.

### 8.4.4. Comparison of Tabu Search Based Matheuristic and Greedy Protection Heuristic

In this section, we run TSM and GPH on relatively larger instances and compare their results in terms of running time and objective value. To this end, networks of

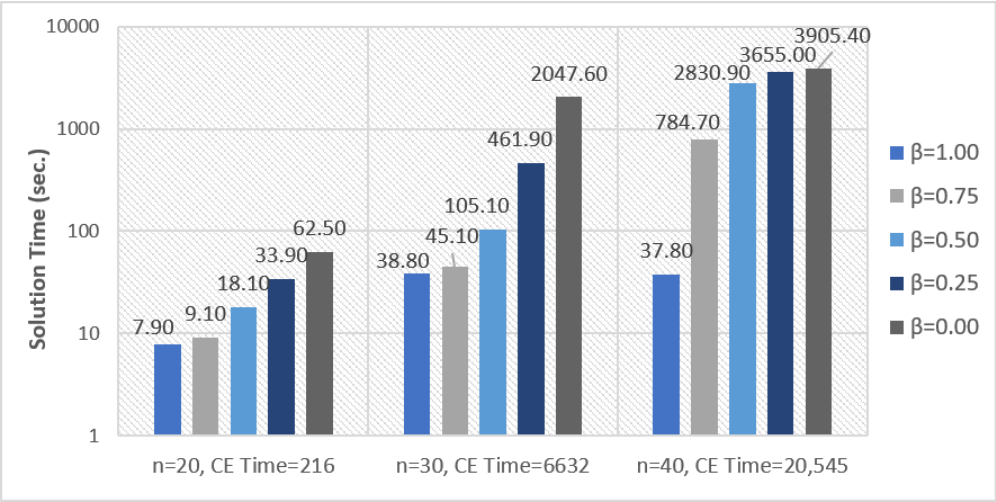



Figure 8.8. Effect of BSA parameter $\beta$ on solution times.

four different sizes $n = 40, 60, 80, 100$ and two density values $d = 0.1, 0.2$ are generated according to Watts-Strogatz model. For each network size, two values are selected as the number of protected nodes, namely $h = 0.05n, 0.1n$, and for each value of $h$ three levels are determined for the number of seed nodes, namely $k = h, h+1, h+2$. Once the GPH results are attained, $t_{\max}$ of TSM in the unit of seconds is set to $\max(t_{\text{GPH}}, 28800)$ where $t_{\text{GPH}}$ denotes the CPU time required by GPH in seconds. Table 8.24 and Table 8.25 present the results in terms of the final objective values.

The columns titled "$z_t$" and "$z_{\text{GPH}}$" in Tables 8.24 and 8.25 denote the objective value at the end of $t$ hours given by TSM and the objective value of the solution provided by GPH, respectively. For instances with $n = 40$ and $n = 60$, the incumbent (best) solution of TSM does not improve after one hour. Therefore, only $z_1$ values are displayed in Table 8.24. If one heuristic produces a better objective value than the other, the former is printed in bold typeface. It can be observed that in only two out of 24 instances GPH finds a better solution, whereas in 12 instances both methods yield the same solution. When we take $z_{\text{GPH}}$ as the baseline objective value, the relative difference between the objectives can be measured as $100 \times (z_1 - z_{\text{GPH}})/z_{\text{GPH}}$. Thus, this quantity becomes negative when TSM finds a better solution. We remark that the average relative difference is calculated as $-0.72\%$.

Table 8.24. Comparison of the performances of TSM and GPH.

| | | | | TSM | GPH | |
|---|---|---|---|---|---|---|
| $n$ | $d$ | $h$ | $k$ | $z_1$ | $z_{\text{GPH}}$ | Time (sec.) |
| 40 | 0.1 | 2 | 2 | 11.03 | 11.03 | 276 |
| | | 2 | 3 | 14.91 | 14.91 | 366 |
| | | 2 | 4 | 18.77 | 18.77 | 443 |
| | | 4 | 4 | **14.62** | 15.32 | 814 |
| | | 4 | 5 | **17.07** | 17.67 | 896 |
| | | 4 | 6 | **19.51** | 20.04 | 1114 |
| 40 | 0.2 | 2 | 2 | 16.41 | 16.41 | 314 |
| | | 2 | 3 | 21.77 | 21.77 | 431 |
| | | 2 | 4 | **26.00** | 26.59 | 536 |
| | | 4 | 4 | **18.99** | 19.27 | 936 |
| | | 4 | 5 | **21.92** | 22.23 | 1106 |
| | | 4 | 6 | **24.43** | 24.71 | 1389 |
| 60 | 0.1 | 3 | 3 | 13.42 | 13.42 | 1362 |
| | | 3 | 4 | 17.25 | 17.25 | 1612 |
| | | 3 | 5 | 20.90 | 20.90 | 1772 |
| | | 6 | 6 | 19.26 | 19.26 | 4189 |
| | | 6 | 7 | 21.88 | 21.88 | 4526 |
| | | 6 | 8 | 25.36 | **24.45** | 4828 |
| 60 | 0.2 | 3 | 3 | 27.02 | 27.02 | 1920 |
| | | 3 | 4 | **32.34** | 33.40 | 2082 |
| | | 3 | 5 | 36.08 | 36.08 | 2365 |
| | | 6 | 6 | **29.39** | 29.58 | 4983 |
| | | 6 | 7 | **32.09** | 32.26 | 5307 |
| | | 6 | 8 | 34.69 | **34.61** | 5709 |

Table 8.25. Comparison of the performances of TSM and GPH.

| | | | | TSM | | | | | GPH | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $d$ | $h$ | $k$ | $z_1$ | $z_2$ | $z_4$ | $z_8$ | $z_{\text{TSM}}$ | $z_{\text{GPH}}$ | Time (sec.) |
| 80 | 0.1 | 4 | 4 | 18.61 | 18.61 | 18.61 | 18.61 | 18.61 | 18.61 | 4464 |
| | | 4 | 5 | 22.70 | 22.70 | 22.70 | 22.70 | **22.70** | 22.72 | 5332 |
| | | 4 | 6 | 26.57 | 26.57 | 26.57 | 26.57 | 26.57 | **26.50** | 5827 |
| | | 8 | 8 | 27.27 | 26.37 | 26.37 | 26.27 | 26.37 | **26.34** | 14,301 |
| | | 8 | 9 | 29.39 | 28.93 | 28.93 | 28.92 | **28.93** | 29.22 | 15,268 |
| | | 8 | 10 | 31.48 | 31.13 | 31.13 | 31.13 | **31.13** | 31.14 | 16,858 |
| 80 | 0.2 | 4 | 4 | 35.16 | 35.16 | 35.16 | 34.82 | **35.16** | 36.39 | 7537 |
| | | 4 | 5 | 40.42 | 40.42 | 40.42 | 40.42 | **40.42** | 40.90 | 8174 |
| | | 4 | 6 | 44.39 | 44.39 | 44.24 | 44.24 | **44.39** | 44.68 | 7869 |
| | | 8 | 8 | 40.53 | 37.72 | 37.72 | 37.72 | 37.72 | **37.26** | 21,283 |
| | | 8 | 9 | 43.42 | 41.50 | 41.35 | 40.62 | 40.95 | **40.01** | 24,300 |
| | | 8 | 10 | 45.26 | 45.01 | 43.94 | 43.94 | 43.94 | **42.54** | 25,461 |
| 100 | 0.1 | 5 | 5 | 30.05 | 28.34 | 27.28 | 26.52 | 27.28 | **26.52** | 12,224 |
| | | 5 | 6 | 32.87 | 31.78 | 31.78 | 31.49 | 31.78 | **31.00** | 13,810 |
| | | 5 | 7 | 36.38 | 36.20 | 36.20 | 35.50 | 36.20 | **35.19** | 14,391 |
| | | 10 | 10 | 39.46 | 39.46 | 39.02 | 38.42 | 38.42 | **35.22** | 41,369 |
| | | 10 | 11 | 43.85 | 42.48 | 42.22 | 40.71 | 39.46 | **38.14** | 46,797 |
| | | 10 | 12 | 46.28 | 45.08 | 44.82 | 44.03 | 42.33 | **41.42** | 52,228 |
| 100 | 0.2 | 5 | 5 | 50.00 | 46.09 | 45.27 | 44.36 | 45.04 | **44.77** | 17,520 |
| | | 5 | 6 | 51.54 | 50.80 | 50.77 | 50.77 | 50.77 | **50.70** | 23,767 |
| | | 5 | 7 | 57.15 | 55.80 | 55.80 | 54.45 | **54.45** | 55.90 | 31,060 |
| | | 10 | 10 | 57.91 | 55.52 | 54.00 | 51.72 | 49.16 | **49.09** | 71,025 |
| | | 10 | 11 | 60.60 | 60.60 | 58.96 | 55.25 | 54.31 | **53.76** | 93,248 |
| | | 10 | 12 | 63.48 | 63.48 | 61.26 | 57.84 | 55.67 | **54.70** | 81,472 |

In Table 8.25 which contains the results for $n = 80$ and $n = 100$, the incumbent is reported at four different epochs in addition to $z_{\text{TSM}}$ which represents the best objective value that TSM produces when it runs for the same duration as GPH. The minimum of $z_{\text{TSM}}$ and $z_{\text{GPH}}$ values is written in bold typeface. In only one out of 24 instances both TSM and GPH yield the same objective value in $t_{\text{GPH}}$ seconds, while GPH finds a better solution in 16 of the remaining instances. Especially for $n = 100$, GPH performs better in most of the instances even though the differences between objective values do not seem large.

The average of the relative differences $100 \times (z_{\text{TSM}} - z_{\text{GPH}})/z_{\text{GPH}}$ between TSM and GPH objective values found in $t_{\text{GPH}}$ hours is 1.04%, i.e., TSM produces 1.04% worse solutions with higher objective values than GPH on the average. This average is computed as 6.15%, 3.92%, 3.01% and 1.40% when the term $z_{\text{TSM}}$ is replaced with $z_1$, $z_2$, $z_4$ and $z_8$, respectively. When the number of the protected nodes, $h$, is large, due to excessive CPU time of GPH, TSM becomes more advantageous with a small sacrifice from solution quality. For larger problem sizes, GPH turns out to be impractical since it does not provide a feasible solution before the termination of the algorithm as is the case for most of the greedy heuristic.

Lastly, we test the algorithms on nine networks that we extract from `arXiv` (which were also used in Section 8.2.2.4, and summarized in Table 8.13). The time limit is set to 24 hours for TSM, and the incumbent (best) objective value is recorded at four different time instants as can be observed in Table 8.26. GPH performs slightly better in the instance with $n = 100$ whereas it is outperformed by TSM for $n = 150, 200, 250, 300$ despite a running time of 24 hours. In the last four instances GPH does not terminate within one week, and thus it is terminated. This result is in line with the earlier observation that GPH is not efficient for large networks even for moderate $h$ values.

Table 8.26. Results on the co-authorship network from `arXiv`.

| | | | TSM | | | | GPH | |
|---|---|---|---|---|---|---|---|---|
| $(n, m)$ | $h$ | $k$ | $z_4$ | $z_8$ | $z_{12}$ | $z_{24}$ | $z_{\text{GPH}}$ | Time (sec.) |
| (100,418) | 5 | 5 | 28.13 | 28.10 | 28.10 | 28.10 | **27.94** | 9507 |
| (150,620) | 5 | 5 | **37.26** | **37.26** | **37.26** | **37.26** | 37.30 | 20710 |
| (200,867) | 10 | 10 | 64.36 | 63.72 | **63.62** | **63.62** | 63.64 | 206,490 |
| (250,1037) | 10 | 10 | 73.23 | 73.23 | **71.40** | **71.40** | 72.43 | 345718 |
| (300,1314) | 10 | 10 | 82.23 | 78.27 | 78.27 | **76.23** | 76.39 | 590,682 |
| (350,1428) | 10 | 10 | 83.77 | 83.77 | 79.35 | 77.14 | – | > 604,800 |
| (400,1730) | 15 | 15 | 111.89 | 111.89 | 111.89 | 109.25 | – | > 604,800 |
| (450,1882) | 15 | 15 | 123.36 | 123.36 | 120.93 | 120.14 | – | > 604,800 |
| (500,2214) | 15 | 15 | 126.85 | 126.85 | 126.85 | 125.41 | – | > 604,800 |

### 8.4.5. A Solution Approach Based on Centrality Measures

Some of the studies which address a node blocking problem and referred in Section 3.1 such as the one by Yu *et al.* (2008) use various graph theoretic measures to select the nodes that will be blocked. To this end, they rank the nodes according to the measure of interest, and block (equivalent to protection in the MSMP) the highest-ranking first $k$ nodes. In this section, we consider six such measures to determine the protected nodes. For the sake of completeness, we briefly provide the descriptions of these measures. Betweenness of a node is the ratio of the shortest paths going through that node to the number of pairwise shortest paths in the network. Degree of node $i$ is the number of adjacent edges to $i$ and out-degree of node $i$ is the number of outgoing arcs from $i$. Closeness of $i$ is the inverse of the average shortest path length from $i$ to all other nodes. PageRank is the score computed according to Google's ranking algorithm for web pages. Lastly, clustering coefficient of $i$ is the ratio of edges between the neighbors of $i$ to all possible number of edges among them.

To assess the quality of the solutions obtained by means of centrality measures, we consider the instances used in the previous section. The ranking of the nodes is determined for each measure. After the highest-ranking $h$ nodes are protected, the follower response is obtained using the SAA method. Let $z_\mu$ denote the spread that measure $\mu$ yields. The comparison between $z_\mu$ and $z_{\text{GPH}}$ is presented in Figure 8.9 as

a percent deviation from $z_{\text{GPH}}$ (i.e., $100 \times (z_\mu - z_{\text{GPH}})/z_{\text{GPH}}$) for each network size $n$ considered in the previous experiments. Additionally, we provide for each measure the average percent deviation over all network sizes. A randomly chosen leader solution is also evaluated which is called "Random Protection". The overall average increase in the spread in comparison with GPH, which implies a deterioration in the quality of the solution in the leader's minimization problem, is 31.98%, 25.61%, 23.79%, 28.34%, 25.12%, 37.06%, and 40.88% for random protection, betweenness, degree, out-degree, closeness, PageRank, and clustering coefficient, respectively. Parallel to the findings of Yu *et al.* (2008), the degree measure is the best one on the average with 23.79% while the clustering coefficient is the worst one among the measures considered with 40.88%. The results indicate that the performance of simple centrality measures is not comparable with intelligent methods such as GPH and the matheuristic methods.

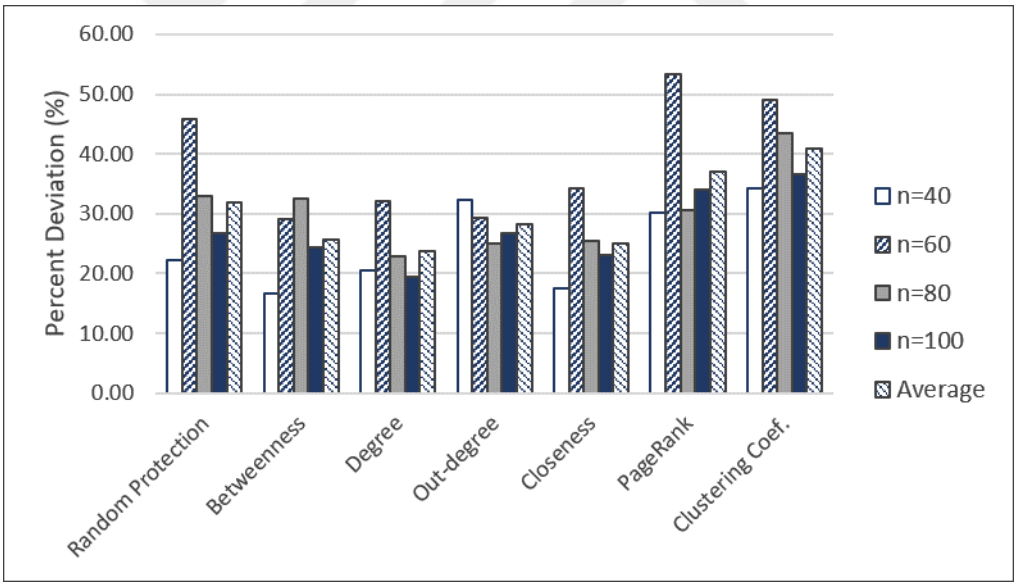



Figure 8.9. Assessment of the results obtained by means of centrality measures.

## 8.5. Results of the Improved $x$-space Algorithm

In this section, we first compare the performance of the original $x$-space algorithm with that of the improved $x$-space algorithm proposed in Chapter 7, on the test instances of two interdiction problems solved by Tang *et al.* (2016) before. Then, we present the numerical results related to the implementation of these algorithms on the MSMP described in Section 7.2. Besides, we compare the results to those obtained via

the state-of-the-art MIBLP algorithm proposed by Fischetti *et al.* (2017b) using its default setting MIX++. The MIBLP solver which is available at (Fischetti *et al.*, 2017a) is run on a virtual machine with 12 GB RAM allocation and Ubuntu 20.04 operating system using CPLEX 12.7, on the workstation which we run all of our experiments. The experiments of XS and IXS algorithms are carried out with two MILP solvers: CPLEX 12.7 and Gurobi 8.0.

### 8.5.1. Comparison of the Algorithms on Bilevel Knapsack Problem and Bilevel Clique Problem Instances

Tang *et al.* (2016) consider two types of interdiction problems, the bilevel knapsack problem (BKP) which is a min-max 0-1 knapsack problem and the bilevel maximum clique problem (BCP) which is a min-max clique problem to evaluate the performance of the $x$-space algorithm. In this section we compare the improved $x$-space algorithm with the original one as well as the MIX++ algorithm by solving the instances solved by Tang *et al.* (2016). They can be reached at (Smith, 2016).

The formulation of the BKP is given as

$$\min_{\mathbf{x}\in X} \max_{\mathbf{y}} \{p^T\mathbf{y} : a^T\mathbf{y} \leq b, \mathbf{y} \leq \mathbf{1} - \mathbf{x}, \mathbf{y} \in \{0,1\}^n\} \tag{8.3}$$

where $X = \{\mathbf{x} \in \{0,1\}^n : \mathbf{1}^T\mathbf{x} \leq k\}$. Recall that it is necessary to define $C_\tau$, the blocker set of solution $\tau$, to develop the $LB''_q$ formulation of the improved algorithm. For the BKP, the set of solutions that $\mathbf{x}$ blocks is $B(\mathbf{x}, S_q) = \{\tau \in J : \exists i \in \{1,...,n\}, x_i = 1, y_i^\tau = 1\}$. Using this set, we can define $C_\tau$ as

$$C_\tau = \{i \in \{1,...,n\} : y_i^\tau = 1\}. \tag{8.4}$$

The second problem, BCP, is formulated by Tang *et al.* (2016) initially as

$$\min_{\mathbf{x}\in X} \max \sum_{i\in V} y_i \tag{8.5}$$

$$\text{s.t. } y_i + y_j \leq 1 \qquad (i,j) \in \bar{E} \tag{8.6}$$

$$y_i + y_j \leq 2 - x_{ij} \qquad (i,j) \in E \tag{8.7}$$

$$y_i \in \{0,1\} \qquad i \in V. \tag{8.8}$$

Here $V$ and $E$ represent the set of vertices and edges, respectively, $\bar{E} = \{(i,j) : (i,j) \notin E\}$, and $X = \{\mathbf{x} \in \{0,1\}^{|E|} : \sum_{(i,j)\in E} x_{ij} \leq k\}$. Then, the formulation is modified as follows since the XS algorithm requires a special structure, i.e., an interdiction relation in the form of the constraint $\mathbf{y} \leq \mathbf{d} - \mathbf{D}\mathbf{x}$, where $\mathbf{x}$ and $\mathbf{y}$ stand for the leader and follower variables, respectively.

$$\min_{x\in X} \max \sum_{i\in V} y_i \tag{8.9}$$

$$\text{s.t. } y_i + y_j \leq 1 \qquad (i,j) \in \bar{E} \tag{8.10}$$

$$y_i + y_j \leq 1 + w_{ij} \qquad (i,j) \in E \tag{8.11}$$

$$w_{ij} \leq 1 - x_{ij} \qquad (i,j) \in E \tag{8.12}$$

$$y_i \in \{0,1\},\ w_{ij} \in \{0,1\}, \qquad i \in V, (i,j) \in E. \tag{8.13}$$

The development of the LB problem formulation for the BCP is more complicated than that of the BKP, when the latter formulation (which will be referred to as BCP′) is used. Defining the set of blocked solutions according to the interdiction constraint (8.12) as $B(\mathbf{x}, S_q) = \{\tau \in J : \exists (i,j) \in E, x_{ij} = 1, w_{ij}^{\tau} = 1\}$ gives rise to the following blocker set of edges as a result of the fact that the interdiction variables are related to the edges of the network:

$$C_\tau = \{(i,j) \in E : w_{ij}^{\tau} = 1\}. \tag{8.14}$$

However, this set may become unnecessarily large under the current BCP formulation due to the following reason. In an optimal follower solution to the interdiction decision $\mathbf{w}$, it is possible that $y_i + y_j \leq 1, x_{ij} = 0$ and $w_{ij} = 1$ for some $(i, j) \in E$. In other words, $w_{ij} = 1$ does not necessarily indicate the existence of edge $(i, j)$ in the maximum clique. To force $w_{ij} = 1$ only if edge $(i, j)$ belongs to the maximum clique we modify the follower's objective function by adding a penalty term so as to obtain $\sum_{i \in V} y_i - \mu \sum_{(i,j) \in E} w_{ij}$ where $\mu$ is a small positive real number. Thus, if $y_i + y_j \leq 1$, then $w_{ij} = 0$ in an optimal follower solution. We conduct the experiments for both versions of the BCP$'$, with and without the penalty term.

On the other hand, it is possible to use the first and simpler formulation in (8.5)–(8.8) within the IXS algorithm (as well as MIX++). If the set of blocked solutions is defined according to the constraint (8.7) as $B(\mathbf{w}, S_q) = \{\tau \in J : \exists (i, j) \in E, w_{ij} = 1, y_i^\tau + y_j^\tau = 2\}$, it leads to the blocker set

$$C_\tau = \{(i, j) \in E : y_i^\tau + y_j^\tau = 2\} \tag{8.15}$$

of edges as a result of the fact that the interdiction variables are related to the edges of the graph. In other words, if nodes $i$ and $j$ are included in the clique of solution $\mathbf{y}^\tau$, then interdicting the edge $(i, j)$ blocks that solution.

While obtaining the single-level LB formulation used in the $x$-space algorithm, it is required to dualize the inner problem in (7.5) and then linearize the resulting bilevel terms. To this end, bounds on the dual variables of the LB problem are needed. For both BKP and BCP we compute the bounds as suggested in the paper of Tang *et al.* (2016). Besides, we use the same initial follower solution set $S_1$ which contains the optimal follower solutions to all possible unit vectors as leader solutions, the trivial follower solution $\mathbf{y} = \mathbf{0}$, and a heuristic feasible solution obtained via relaxing the integer variables of the follower's problem. We refer the reader to Tang *et al.* (2016) for details.

Test problem set includes 150 BKP instances for 15 different $(n, k)$ combinations. The results are given in Table 8.27. Columns XS-G and IXS-G show the results of our implementation of the $x$-space algorithm and improved $x$-space algorithm with Gurobi 8.0. Their counterparts with CPLEX 12.7 are given in columns XS-C and IXS-C. Lastly, column MIX++ shows the results of the branch-and-cut solver in Fischetti *et al.* (2017b) with its default settings where two types of intersection cuts are used along with a bilevel specific preprocessing procedure and follower upper bound cuts. The results are presented in terms of the number of unsolved instances over 10 instances in the same $(n, k)$ setting, and the average CPU time for those instances with a one-hour CPU time limit.

Table 8.27. Results obtained on BKP instances.

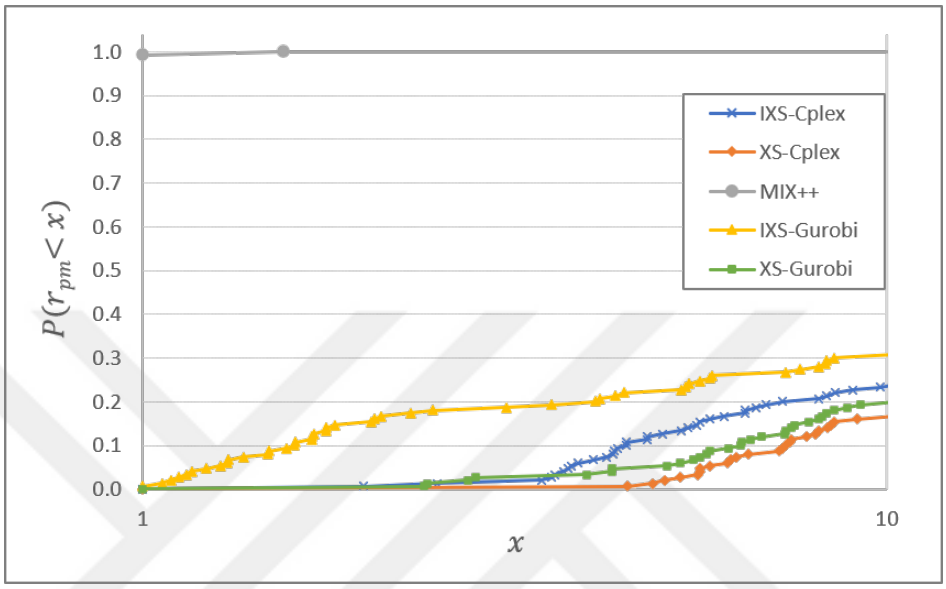
| | | # unsolved | | | | | Time(sec.) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $k$ | XS-G | IXS-G | XS-C | IXS-C | MIX++ | XS-G | IXS-G | XS-C | IXS-C | MIX++ |
| 20 | 5 | 0 | 0 | 0 | 0 | 0 | 65.4 | 20.0 | 215.2 | 52.9 | 13.3 |
| | 10 | 4 | 0 | 1 | 0 | 0 | 2486.4 | 213.4 | 1451.4 | 456.9 | 4.4 |
| | 15 | 0 | 0 | 0 | 0 | 0 | 8.6 | 1.3 | 9.3 | 5.8 | 0.3 |
| Average | | 1.3 | 0.0 | 0.3 | 0.0 | 0.0 | 853.5 | 78.2 | 558.6 | 171.8 | 6.0 |
| 22 | 6 | 0 | 0 | 4 | 0 | 0 | 1248.3 | 228.1 | 2724.9 | 610.9 | 21.5 |
| | 11 | 8 | 1 | 8 | 1 | 0 | 3405.9 | 1376.7 | 3470.4 | 2074.3 | 6.7 |
| | 17 | 0 | 0 | 0 | 0 | 0 | 20.9 | 1.8 | 12.4 | 7.3 | 0.3 |
| Average | | 2.7 | 0.3 | 4.0 | 0.3 | 0.0 | 1558.4 | 535.5 | 2069.3 | 897.5 | 9.5 |
| 25 | 7 | 10 | 5 | 10 | 8 | 0 | 3601.9 | 2609.0 | 3601.6 | 3344.5 | 74.4 |
| | 13 | 10 | 10 | 10 | 10 | 0 | 3601.3 | 3600.5 | 3600.6 | 3600.3 | 23.8 |
| | 19 | 0 | 0 | 0 | 0 | 0 | 576.8 | 26.1 | 305.2 | 58.8 | 0.6 |
| Average | | 6.7 | 5.0 | 6.7 | 6.0 | 0.0 | 2593.3 | 2078.5 | 2502.5 | 2334.5 | 32.9 |
| 28 | 7 | 10 | 10 | 10 | 10 | 0 | 3600.3 | 3600.6 | 3601.7 | 3601.2 | 191.7 |
| | 14 | 10 | 10 | 10 | 10 | 0 | 3600.5 | 3600.4 | 3600.3 | 3600.5 | 63.3 |
| | 21 | 8 | 0 | 5 | 0 | 0 | 3523.9 | 1042.2 | 3044.2 | 1194.4 | 1.2 |
| Average | | 9.3 | 6.7 | 8.3 | 6.7 | 0.0 | 3574.9 | 2747.7 | 3415.4 | 2798.7 | 85.4 |
| 30 | 8 | 10 | 10 | 10 | 10 | 0 | 3600.3 | 3601.1 | 3600.6 | 3601.8 | 429.5 |
| | 15 | 10 | 10 | 10 | 10 | 0 | 3600.6 | 3601.4 | 3600.3 | 3600.9 | 96.2 |
| | 23 | 10 | 3 | 8 | 3 | 0 | 3600.6 | 1995.3 | 3472.0 | 1980.1 | 1.3 |
| Average | | 10.0 | 7.7 | 9.3 | 7.7 | 0.0 | 3600.5 | 3066.0 | 3557.6 | 3060.9 | 175.7 |

It can be observed that IXS performs better than XS in terms of the total number of unsolved instances with both MILP solvers. When the improved algorithm is used, the number of unsolved instances reduces from 86 to 62 with CPLEX and from 90 to 59 with Gurobi. Note that the number of unsolved instances for XS-C is less than the value

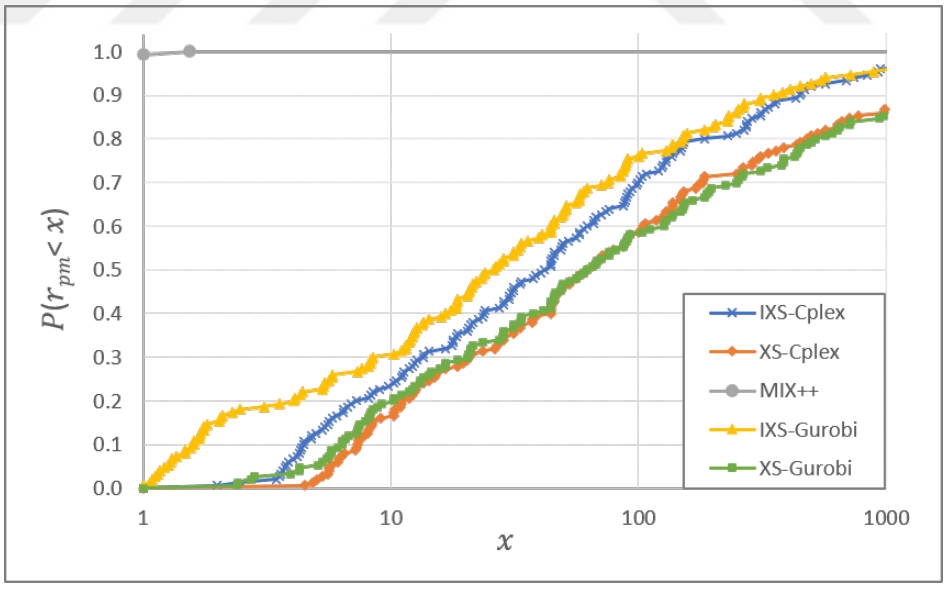
reported in Tang *et al.* (2016) which can be attributed to the MILP solver version (it is CPLEX 12.5 in the paper), to the implementation or to hardware related differences. When the CPU times are considered, the differences between the two algorithms are apparent especially for the $(n, k)$ settings for which most of the instances can be solved optimally within one hour. For example, the average CPU time for $(25, 19)$ instances is reduced from 577 to 26 seconds with Gurobi, and from 305 to 59 seconds with CPLEX whereas this number is 1175 in Tang *et al.* (2016). The number of unsolved instances of settings $(28, 21)$ and $(30, 23)$ decreases significantly for both solvers. Nevertheless, MIX++ can solve all of the BKP instances optimally within one hour. The average CPU time is 61.9 seconds in our test environment. It shows that, although it is effective, the IXS algorithm is outperformed by the state-of-the-art method in solving BKP instances.

Figure 8.10 shows the performance profiles of the five solution settings as described in Dolan and Moré (2002) for benchmarking of optimization software. In this approach, a performance ratio $r_{pm}$ of method $m \in M$ on problem $p \in P$ is defined as the ratio of the solution time of problem $p$ with method $m$ to the minimum solution time for that problem instance, i.e. $r_{pm} = t_{pm}/\min_m\{t_{pm}\}$, when the performance measure of interest is the solution time. The performance profile of each method $m \in M$ refers to the cumulative distribution function of $r_{pm}$. Following the approach in Fischetti *et al.* (2017b), we update the definition of $r_{pm}$ as follows to reduce the effect of the problems that can be solved in a very short time (in seconds): $r_{pm} = (t_{pm} + 1)/(\min_m\{t_{pm}\} + 1)$. For the instances that a method fail to solve in the time limit, $t_{pm}$ is accepted as 3600 seconds. Both figures use logarithmic scales with different ranges and it is clear that MIX++ is the best performer. Since it yields the minimum solution time in almost all BKP instances implying that $P(r_{pm} \leq 1) \approx 1$, it gives a straight line on the top of the chart. Then, we see that IXS-G yields the next best setting while XS-G and XS-C are very close to each other and they yield the worst two performances. The probability that IXS-G (IXS-C) solves a BKP problem at most 10 times slower than the best setting is around 30% (23%). Figure 8.11 shows the performance profiles of XS and IXS algorithms when MIX++ is not considered. IXS-G performs better than

the others in 95% of the instances while the remaining instances are solved faster by IXS-C. The probability that XS-G (XS-C) time is at most 10 times worse than the best method is 85% (75%).




(a) $x \in [1, 10]$




(b) $x \in [1, 1000]$

Figure 8.10. Comparison of the methods for BKP instances.

The BCP instances in the data set belong to eight different $(n, d)$ combinations where $d$ denotes the network density. The number of interdicted edges is fixed to $k = \lceil m \rceil$ with $m$ representing the number of edges. There are 10 instances for each

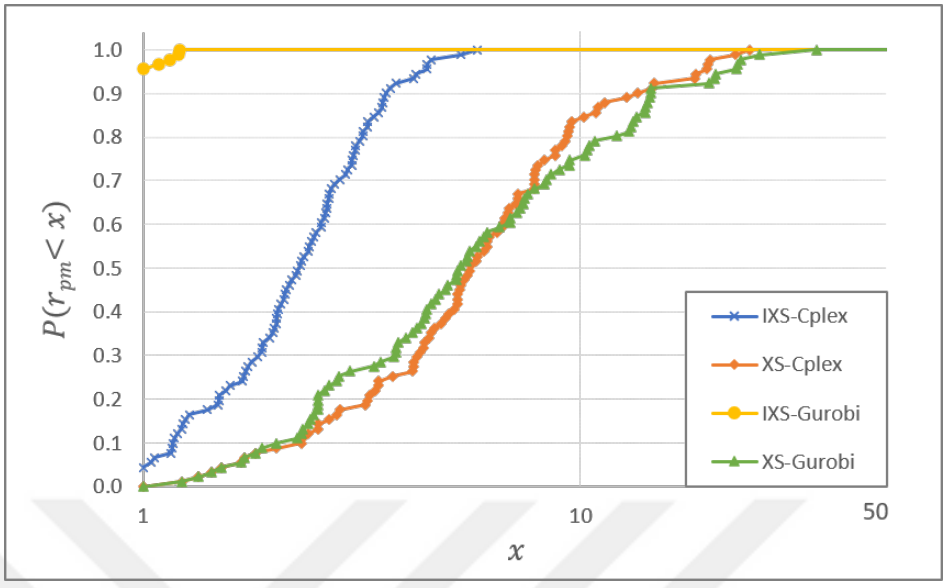



Figure 8.11. Comparison of IXS and XS for BKP instances.

$(n, d)$ combination resulting in 80 instances in total. The time limit is again one hour. When BCP′ formulation is used without the penalty term, none of the instances except the ones corresponding to smallest size setting $(n, d) = (8, 0.7)$ can be solved optimally by any algorithm and solver combination. In other words only 10 out of 80 instances are solved by XS or IXS. The average solution time for those instances is reduced from 83.4 seconds to 53.4 seconds by the modified algorithm when CPLEX is used. With Gurobi it is reduced from 1577.5 to 505.2. On the other hand, adding the penalty term to the same formulation improves not only IXS but also XS remarkably as the figures in Table 8.28 suggest. By means of the new objective function, the IXS algorithm can solve all of the instances optimally within the time limit. Actually, it takes only a few seconds except the last setting with $(n, d) = (15, 0.9)$, for which it requires about two minutes on the average. The XS algorithm is now able to solve 60 (50) of the instances instead of 10, when the MILP solver is CPLEX (Gurobi). However, as the problem size increases, it is significantly outperformed by the IXS algorithm even for the instances that are solved optimally by both algorithms.

Table 8.29 presents the results for XS, IXS and MIX++ algorithms when XS solves the BCP′ formulation as is in Tang *et al.* (2016), and IXS and MIX++ solves the original formulation in (8.5)–(8.8). As above, the IXS algorithm can solve all

Table 8.28. Results obtained on BCP instances using BCP′ formulation with penalty term in the objective function.

| | | # unsolved | | | | Time(sec.) | | | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | $d$ | XS-C | IXS-C | XS-G | IXS-G | XS-C | IXS-C | XS-G | IXS-G |
| 8 | 0.7 | 0 | 0 | 0 | 0 | 0.3 | 0.3 | 0.1 | 0.3 |
| | 0.9 | 0 | 0 | 0 | 0 | 1.1 | 0.5 | 1.1 | 0.2 |
| Average | | 0.0 | 0.0 | 0.0 | 0.0 | 0.7 | 0.4 | 0.6 | 0.2 |
| 10 | 0.7 | 0 | 0 | 0 | 0 | 1.4 | 0.6 | 1.4 | 0.3 |
| | 0.9 | 0 | 0 | 0 | 0 | 24.6 | 1.3 | 29.5 | 0.8 |
| Average | | 0.0 | 0.0 | 0.0 | 0.0 | 13.0 | 0.9 | 15.5 | 0.5 |
| 12 | 0.7 | 0 | 0 | 0 | 0 | 43.3 | 1.2 | 54.3 | 0.8 |
| | 0.9 | 0 | 0 | 10 | 0 | 2348.5 | 4.7 | 3602.0 | 3.6 |
| Average | | 0.0 | 0.0 | 5.0 | 0.0 | 1195.9 | 2.9 | 1828.2 | 2.2 |
| 15 | 0.7 | 10 | 0 | 10 | 0 | 3603.4 | 3.8 | 3605.0 | 3.1 |
| | 0.9 | 10 | 0 | 10 | 0 | 3601.3 | 114.3 | 3600.7 | 131.5 |
| Average | | 10.0 | 0.0 | 10.0 | 0.0 | 3602.3 | 59.0 | 3602.9 | 67.3 |

of the instances optimally, within a duration even smaller than the ones reported in Table 8.28. MIX++ can solve all of the instances optimally as well, although it yields a slightly larger average solution time than IXS-C and IXS-G. It is worth to mention that MIX++ solution times are larger than the ones reported in Fischetti *et al.* (2017b) which are obtained via CPLEX 12.6.3 while we run the experiments with version 12.7 for which the code is publicly available.

The performance profiles are shown in Figure 8.12 on a logarithmic scale as before. It is seen that, the proportion of instances for which IXS-G, IXS-C and MIX++ have the minimum solution time are 80%, 10% and 10%, respectively. The minimum performance ratios of XS-G and XS-C are 15 and 27, therefore their lines coincide with the horizontal axis in Figure 8.12(a). While IXS-C can solve 90% of the instances within a duration at most 1.5 times larger than the minimum time, this ratio is 2.9 for MIX++. The proportion of instances for which the solution time is very close to the minimum is larger for MIX++ than IXS-C. However, the proportion of instances with performance ratios larger than 2 is also larger for this method. In fact, IXS-C yields a ratio larger than 2 in only 3 of the instances and MIX++ does so in 17 of them.

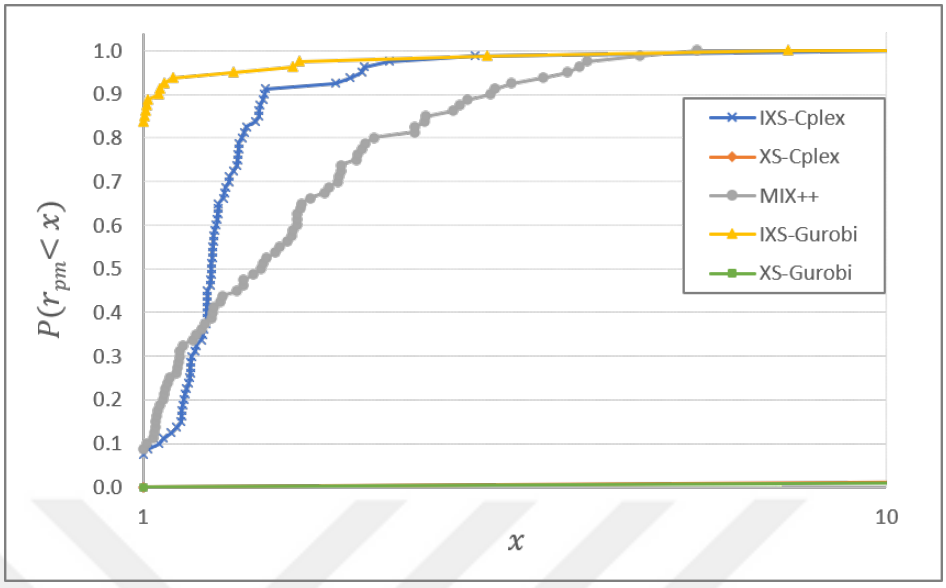
Table 8.29. Results obtained on BCP instances.

| | | # unsolved | | | | | Time(sec.) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $d$ | XS-G | IXS-G | XS-C | IXS-C | MIX++ | XS-G | IXS-G | XS-C | IXS-C | MIX++ |
| 8 | 0.7 | 0 | 0 | 0 | 0 | 0 | 1577.5 | 0.1 | 83.4 | 0.3 | 0.1 |
| | 0.9 | 10 | 0 | 10 | 0 | 0 | 3600.4 | 0.1 | 3600.4 | 0.3 | 0.3 |
| Average | | 5.0 | 0.0 | 5.0 | 0.0 | 0.0 | 2588.9 | 0.1 | 1841.9 | 0.3 | 0.2 |
| 10 | 0.7 | 10 | 0 | 10 | 0 | 0 | 3600.5 | 0.1 | 3600.5 | 0.4 | 0.3 |
| | 0.9 | 10 | 0 | 10 | 0 | 0 | 3600.5 | 0.4 | 3600.4 | 0.7 | 1.3 |
| Average | | 10.0 | 0.0 | 10.0 | 0.0 | 0.0 | 3600.5 | 0.3 | 3600.5 | 0.5 | 0.8 |
| 12 | 0.7 | 10 | 0 | 10 | 0 | 0 | 3600.6 | 0.3 | 3600.5 | 0.7 | 1.2 |
| | 0.9 | 10 | 0 | 10 | 0 | 0 | 3600.4 | 2.2 | 3600.6 | 2.4 | 4.2 |
| Average | | 10.0 | 0.0 | 10.0 | 0.0 | 0.0 | 3600.5 | 1.3 | 3600.5 | 1.6 | 2.7 |
| 15 | 0.7 | 10 | 0 | 10 | 0 | 0 | 3600.7 | 1.1 | 3600.6 | 1.8 | 5.4 |
| | 0.9 | 10 | 0 | 10 | 0 | 0 | 3600.5 | 95.2 | 3600.6 | 113.1 | 168.5 |
| Average | | 10.0 | 0.0 | 10.0 | 0.0 | 0.0 | 3600.6 | 48.2 | 3600.6 | 57.5 | 87.0 |

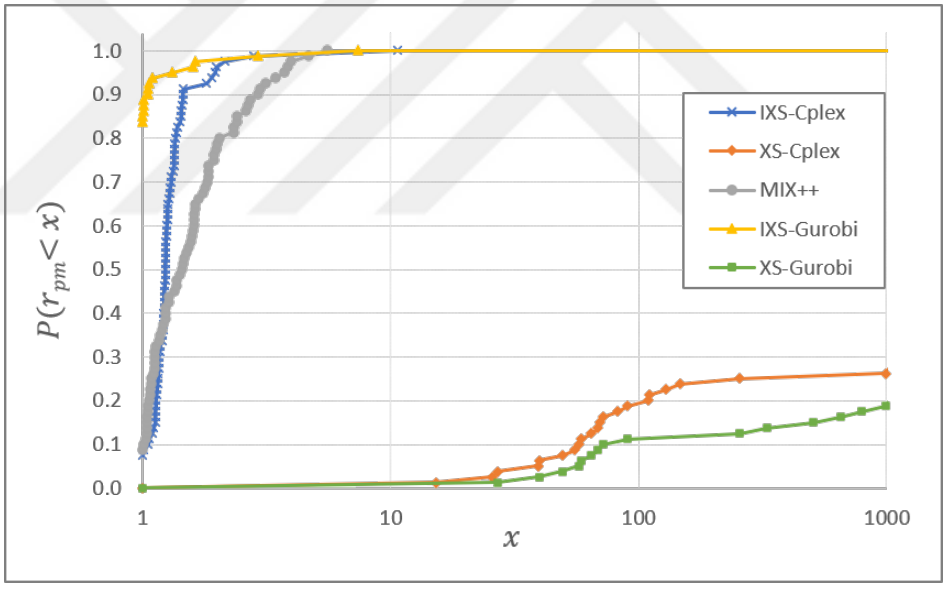
### 8.5.2. Instance Generation and Results Obtained on the Misinformation Spread Minimization Problem Instances

For the experiments of this section, in addition to the Watts-Strogatz networks with $n = 20, 30, 40$ and $d = 0.1, 0.2$ used in Section 8.4.2, five more networks are generated for each combination of $n = 25, 35$ and $d = 0.1, 0.2$ resulting in 50 networks. The values of the seed sizes are determined as $h = k = 3$. The live-arc scenario samples are generated using Latin Hypercube Sampling method and it is assumed that probability of a scenario $p_r = 1/|R|$, $r \in R$.

In all experiments, the trivial follower solution selected in Step 0 of the IXS algorithm (see Figure 7.1) is determined as $\bar{\mathbf{y}}^0 = \mathbf{0}$. Since any node in $V$ can be protected by the leader, this is the only solution that is feasible for all leader solutions. The initial set of the follower solutions is obtained by solving the follower's problem for randomly generated leader solutions. The preliminary experiments show that choosing the leader solutions by simple heuristic methods instead of random selection does not have a significant impact on the results. The value of $a_{ji}(\mathbf{x}, r)$ is computed via a depth-first search on the reverse of the associated scenario sub-




(a) $x \in [1, 10]$




(b) $x \in [1, 1000]$

Figure 8.12. Comparison of the methods for BCP instances.

graph, which is an in-tree, since at most one of the incoming arcs to a node can be live (active) in a scenario. For MIX++, the high-point relaxation problem is $\min_{\mathbf{x},\bar{\mathbf{y}}} \left\{ \sum_{r \in R} \sum_{i \in V} u_{ir} : (6.8), (6.9), (6.11) - (6.13), (6.15), (7.30), (7.31), (7.33) \right\}$.

Table 8.30 displays the results of 200 problem instances. There are 10 instances for each of the 20 $(n, R)$ combinations. The CPU time limit is one hour as before. Both XS and IXS algorithms (regardless of the MILP solver used) can solve all instances

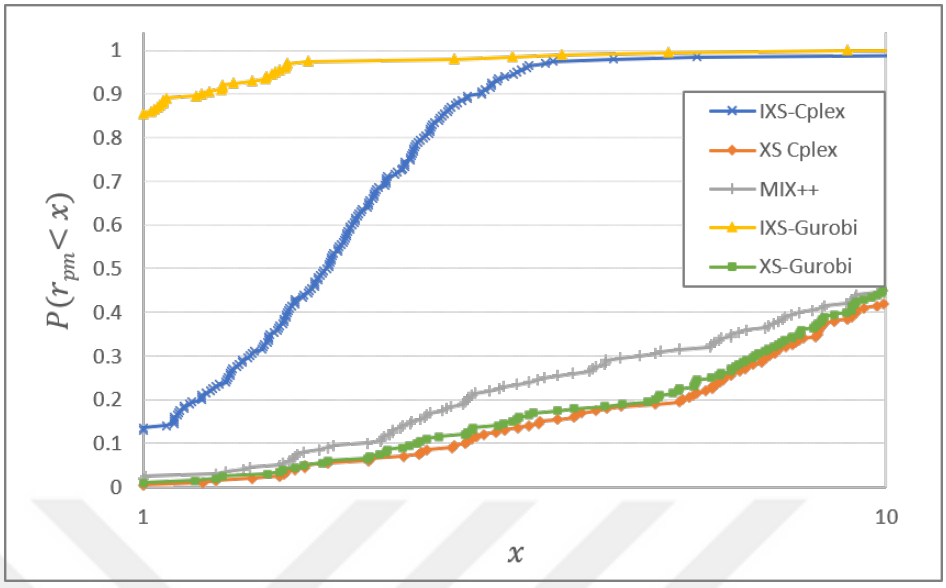
with 20 nodes within the time limit, with a significant difference in average solution times in favor of IXS. For the same size MIX++ fails to solve 15 problems and leads to very long solution times. For $n \in \{25, 30, 35\}$, IXS-C and IXS-G can still solve all of the instances in one hour while XS-C fails to solve 68 and XS-G fails to solve 77 out of 120. It is observed that for these values of $n$, MIX++ yields smaller solution times than XS when $R = 1$, while the total number of unsolved instances is 76. When the instance set with $n = 40$ is considered, XS-G and XS-C can solve only 9 and 10 of them, respectively, while IXS-G and IXS-C are able to solve 39 and 29 out of 40 instances optimally. When $n = 40, R = 1$, the average solution time of MIX++ is slightly smaller than IXS-C, but it is outperformed by IXS-G, as is the case for all of the other problem sizes.

The performance profiles of each method over the MSMP instances are displayed in Figure 8.13. The intersections of the lines with the vertical axis indicate that IXS-G yields the smallest CPU time in 85% of the instances and IXS-C needs the smallest time in 13.5% of them. MIX++, XS-G and XS-C solves only 2%, 1% and 0.5% of all instances faster than the other methods, respectively. The proportion of the instances that require at most 5 times of the minimum solution time is 99.5% with IXS-G, 98% with IXS-C, 31% with MIX++, 21% with XS-G and 19% with XS-C. It can be seen in Figure 8.13(b) that while MIX++ line is above the XS-C and XS-G lines for small values of $x$, it gets worse after $x = 15$. In other words, while the probability of yielding a solution time that is at most 15 times worse than the minimum solution time is larger for MIX++ than XS-C and XS-G, the probability of yielding much worse performance ratios is also larger (e.g., $P(r_{pm} \leq 5)$ is 31% for MIX++ and 19% for XS-C; $P(r_{pm} \leq 100)$ is 86% for MIX++ and 97% for XS-C).

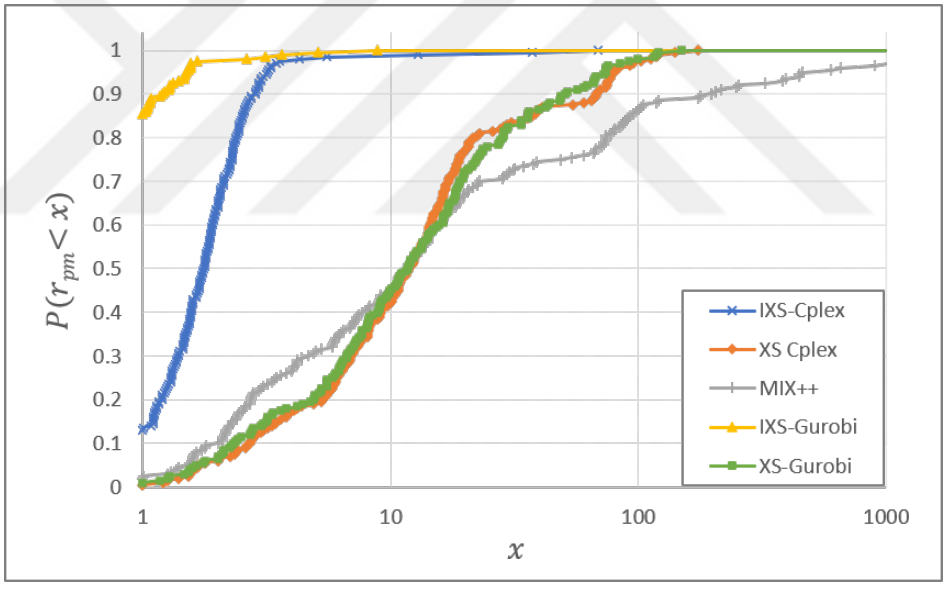
The results suggest that the IXS algorithm is less sensitive to the increase in the number of scenarios. This result is in line with the findings of Section 8.5.1, which indicates that IXS is more advantageous when the interdiction relations are not straightforward. In the MSMP, protection (interdiction) decisions do not only affect the nodes that are protected but all nodes that are linked to them via some path. The

Table 8.30. Results obtained on MSMP instances.

| | | # unsolved | | | | | CPU Time (sec.) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $R$ | XS-G | IXS-G | XS-C | IXS-C | MIX++ | XS-G | IXS-G | XS-C | IXS-C | MIX++ |
| 20 | 1 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0.0 | 1.3 | 0.3 | 2.1 |
| | 10 | 0 | 0 | 0 | 0 | 0 | 9.3 | 0.9 | 11.1 | 1.6 | 127.7 |
| | 25 | 0 | 0 | 0 | 0 | 6 | 21.0 | 2.1 | 27.0 | 3.6 | 2309.5 |
| | 50 | 0 | 0 | 0 | 0 | 9 | 46.3 | 7.6 | 52.3 | 8.7 | 3330.4 |
| Average | | 0.0 | 0.0 | 0.0 | 0.0 | 3.8 | 19.4 | 2.6 | 23.0 | 3.5 | 1442.4 |
| 25 | 1 | 0 | 0 | 0 | 0 | 0 | 33.1 | 2.7 | 41.0 | 6.3 | 7.7 |
| | 10 | 0 | 0 | 5 | 0 | 1 | 1090.6 | 19.2 | 1914.3 | 32.4 | 1060.6 |
| | 25 | 3 | 0 | 5 | 0 | 6 | 2007.2 | 28.2 | 2525.3 | 54.0 | 2816.4 |
| | 50 | 5 | 0 | 7 | 0 | 10 | 2288.9 | 40.0 | 2868.7 | 89.6 | 3600.1 |
| Average | | 2.0 | 0.0 | 4.3 | 0.0 | 4.3 | 1354.9 | 22.5 | 1837.3 | 45.6 | 1871.2 |
| 30 | 1 | 0 | 0 | 0 | 0 | 0 | 786.1 | 29.7 | 365.3 | 50.2 | 53.9 |
| | 10 | 10 | 0 | 10 | 0 | 10 | 3600.0 | 209.6 | 3604.4 | 436.1 | 3600.0 |
| | 25 | 10 | 0 | 10 | 0 | 10 | 3600.0 | 237.4 | 3605.4 | 462.5 | 3600.2 |
| | 50 | 10 | 0 | 10 | 0 | 10 | 3600.0 | 380.9 | 3604.7 | 706.0 | 3600.4 |
| Average | | 7.5 | 0.0 | 7.5 | 0.0 | 7.5 | 2896.5 | 214.4 | 2794.9 | 413.7 | 2713.6 |
| 35 | 1 | 0 | 0 | 0 | 0 | 0 | 518.8 | 22.8 | 278.2 | 19.3 | 46.1 |
| | 10 | 10 | 0 | 10 | 0 | 9 | 3600.0 | 314.3 | 3604.2 | 837.3 | 3534.0 |
| | 25 | 10 | 0 | 10 | 0 | 10 | 3600.0 | 362.0 | 3604.6 | 878.3 | 3600.1 |
| | 50 | 10 | 0 | 10 | 0 | 10 | 3600.0 | 418.8 | 3610.3 | 1099.1 | 3600.2 |
| Average | | 7.5 | 0.0 | 7.5 | 0.0 | 7.3 | 2829.7 | 279.5 | 2774.3 | 708.5 | 2695.1 |
| 40 | 1 | 1 | 0 | 0 | 0 | 0 | 1813.4 | 67.7 | 1074.7 | 133.4 | 114.2 |
| | 10 | 10 | 0 | 10 | 4 | 10 | 3600.0 | 1204.1 | 3602.7 | 2399.2 | 3600.1 |
| | 25 | 10 | 0 | 10 | 4 | 10 | 3600.0 | 1175.7 | 3606.3 | 2192.3 | 3600.2 |
| | 50 | 10 | 1 | 10 | 3 | 10 | 3600.0 | 1396.2 | 3607.2 | 2454.2 | 3600.3 |
| Average | | 7.8 | 0.3 | 7.5 | 2.8 | 7.5 | 3153.3 | 960.9 | 2972.7 | 1794.8 | 2728.7 |




(a) $x \in [1, 10]$




(b) $x \in [1, 1000]$

Figure 8.13. Comparison of the methods for MSMP instances.

increase in the number of scenarios only changes the number of nodes in the set $C_\tau$ since it is defined as the set of nodes that are affected in any scenario in solution $\tau$. The poor performance of MIX++ can be attributed to the size of the problems solved, since IXS makes it possible to discard a significant number of constraints as a result of Proposition 7.6. This result indicates the necessity of algorithms tailored for the problem structure.

# 9. CONCLUSION

## 9.1. Summary of the Contributions

In this thesis, two novel competitive influence spread problems are introduced: Influence Maximization Problem with Deactivation (IMPD) and Misinformation Spread Minimization Problem (MSMP). Both problems are in the form of a zero-sum Stackelberg game where two decision makers act sequentially and the first player (leader) anticipates the reaction of the second player (follower). There are only a few studies in the literature which develop mathematical programs for influence spread related problems. We show that competitive spread problems involving sequential decision making can be formulated as stochastic bilevel programming models. Furthermore, deterministic equivalent formulations can be obtained via enumeration of all diffusion scenarios of the adopted model. The findings about the equivalence between widely studied diffusion models can be used to represent the uncertainty in different ways which allows to develop alternative formulations for the same problem. In this study, we use the equivalence of triggering set and linear threshold models under certain assumptions and propose two deterministic equivalent formulations for the IMPD, which turns out to have a significant impact on the results obtained. For the MSMP, it is not possible to develop a formulation without using an exponential number of constraints, unless the equivalence between the two models is exploited. Thus, a single bilevel program is formulated using the triggering model, i.e., live-arc scenarios.

Complex optimization problems such as the ones addressed in this thesis require integrated solution approaches. The lower-levels of the bilevel models we propose are two-stage stochastic programs with an objective value which is hard to compute. Therefore, designing even a heuristic method can be very complicated. In order to search the solution space of the leader via metaheuristic methods, we use the Sample Average Approximation (SAA) method for the approximation of the follower's objective value. Our experiments show that the success of the SAA method is considerably

affected by the sampling procedure as well as the sample size. We experiment with several sampling methods and find out that Latin Hypercube Sampling helps the most to obtain better estimates of the objective value, i.e., yields smaller gap and variance. Another finding about the SAA method is that using live-arc scenarios in the LLP of IMPD formulation results in a much more robust formulation than using threshold scenarios. While the latter yields smaller solution times for sparse networks of a given size, they increase rapidly as the number of arcs increases. Live-arc formulation is barely affected by the increase in the density as well as by the change in the network structure: the solution times remain almost the same with Watts-Strogatz and Erdős-Rényi networks, which is not the case for the threshold formulation.

In the newly proposed matheuristics, we include rules which helps to avoid excessive computations due to the time cost of the SAA method used for the lower level. For example, a candidate list strategy is used in the Tabu Search based matheuristic TSM, for both IMPD and MSMP. We aim to eliminate the neighbors which are not promising based on a preprocessing procedure which takes into account the chances of nodes to affect each other. Likewise, an explicit memory structure is used and only the solutions that we call eligible are considered for evaluation, in both TSM and Simulated Annealing based matheuristic (SAM). Note that SAM is proposed for the IMPD only, while a greedy heuristic (GPH) and a Binary Search algorithm (BSA) with a tuneable parameter are developed for the MSMP in addition to the TSM. It should be noted that, a significant improvement on the performances of the MSMP methods is achieved via integrating a state-of-the-art IMP method, IMM, into their algorithms. This approach helps to reduce the number of SAA implementations significantly. As a result the solution times decreases.

When evaluated on small problem instances whose optimal objective value can be learned via complete enumeration, all methods yield very small optimality gaps. For relatively larger instances, we compare the performances of the algorithms to each other. The results show that TSM performs better than SAM especially for the cost-based instances with heterogeneous activation and deactivation costs, for the IMPD.

Among the MSMP solution methods, GPH performs slightly better than TSM when solution time and quality are considered together, for small networks and small number of protected nodes, while it fails to find a feasible solution within a reasonable amount of time for larger ones. The success of BSA depends on its threshold parameter which determines the extent of the search. We show that decreasing its value results in significantly better solutions with the expense of increased solution time. Although performs satisfactorily, it falls behind TSM and GPH.

While we need approximation methods due the excessive number of diffusion scenarios, it is possible to solve the problems exactly by assuming a limited number of scenarios. The branch-and-cut algorithm with Intersection Cuts proposed by Fischetti *et al.* (2016) is implemented on the IMPD with live-arc scenarios. The use of different bilevel-free set definitions to obtain the ICs is shown to have a significant effect on the success of the algorithm. Another improvement is due to the alternating bound update (ABU) procedure we propose, which aims to reduce the size of the subproblems and prune them without solving if possible. The ABU procedure successfully decreases average solution times. Another contribution of this thesis in terms of an exact method is the Improved $x$-space (IXS) algorithm which follows the algorithm structure proposed for bilevel interdiction problems by Tang *et al.* (2016). Their $x$-space (XS) algorithm is improved by developing a new lower bound problem formulation, which does not require any dualization and linearization as opposed to the original one. The new algorithm also integrates a set covering heuristic to avoid unnecessary MILP solutions. Its performance is examined on three problem families, BKP, BCP and MSMP. The results show that IXS can solve most of the instances that XS fails to solve within the time limit while yielding much smaller solution times on the average.

## 9.2. Future Research Directions

The problems introduced in this thesis involve two players deciding sequentially and a single diffusion process. One possible future research direction is addressing two competing diffusion processes which unfold on the same network simultaneously.

While the objective of the players is to maximize their own influence spread, a time lag between the decisions of the players can be considered as well. Although there are studies considering similar settings in the literature, none of them provides an explicit mathematical formulation or a solution approach involving the perspectives of both players, to the best of our knowledge.

Another possible research subject involves dynamic actions of two competitors, such as dynamic activation, deactivation, and protection decisions. In this setting, the players would add new seed nodes, or isolate more network nodes as the diffusion continues. Instead of using all of the resources at once, allocating separate budgets for each time period and each decision type would be a reasonable extension of the problems considered in this thesis. In this case, the leader needs to take the response of the follower to the most recent leader action into consideration.

Finally, exact solution methods can be developed for the spread blocking problems where the decision maker seeks to minimize the spread of a rumor or a disease for a given seed set, i.e., initially infected nodes, by blocking nodes or edges. Note that this problem setting leads to single-level mathematical formulation. In most of the related studies, simple heuristic approaches such as centrality-based selection are proposed. An exact method can serve as a benchmark for the evaluation of such heuristic methods.

# REFERENCES


Aksen, D., S. Ş. Akca, and N. Aras, 2014, "A bilevel partial interdiction problem with capacitated facilities and demand outsourcing", *Computers & Operations Research*, Vol. 41, pp. 346–358.

Alizadeh, S., P. Marcotte, and G. Savard, 2013, "Two-stage stochastic bilevel programming over a transportation network", *Transportation Research Part B: Methodological*, Vol. 58, pp. 92–105.

Alon, N., M. Feldman, A. D. Procaccia, and M. Tennenholtz, 2010, "A note on competitive diffusion through social networks", *Information Processing Letters*, Vol. 110(6), pp. 221–225.

Anagnostopoulos, A., R. Kumar, and M. Mahdian, 2008, "Influence and correlation in social networks", in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM.

Angelo, J. S., and H. J. Barbosa, 2015, "A study on the use of heuristics to solve a bilevel programming problem", *International Transactions in Operational Research*, Vol. 22(5), pp. 861–882.

Aral, S., L. Muchnik, and A. Sundararajan, 2009, "Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks", *Proceedings of the National Academy of Sciences*, Vol. 106(51), pp. 21544–21549.

Arthur, D., and S. Vassilvitskii, 2006, "k-means++: The advantages of careful seeding", Tech. Rep., Stanford University.

Balas, E., 1971, "Intersection cuts—a new type of cutting planes for integer programming", *Operations Research*, Vol. 19(1), pp. 19–39.

Bard, J. F., 1991, "Some properties of the bilevel programming problem", *Journal of optimization theory and applications*, Vol. 68(2), pp. 371–378.

Bard, J. F., and J. T. Moore, 1990, "A branch and bound algorithm for the bilevel programming problem", *SIAM Journal on Scientific and Statistical Computing*, Vol. 11(2), pp. 281–292.

Bard, J. F., and J. T. Moore, 1992, "An algorithm for the discrete bilevel programming problem", *Naval Research Logistics (NRL)*, Vol. 39(3), pp. 419–435.

Ben-Ayed, O., and C. E. Blair, 1990, "Computational difficulties of bilevel linear programming", *Operations Research*, Vol. 38(3), pp. 556–560.

Bharathi, S., D. Kempe, and M. Salek, 2007, "Competitive influence maximization in social networks", in *International Workshop on Web and Internet Economics*, Springer.

Bialas, W. F., and M. H. Karwan, 1984, "Two-level linear programming", *Management science*, Vol. 30(8), pp. 1004–1020.

Borgs, C., M. Brautbar, J. Chayes, and B. Lucier, 2014, "Maximizing social influence in nearly optimal time", in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, SIAM.

Borodin, A., Y. Filmus, and J. Oren, 2010, "Threshold models for competitive influence in social networks.", in *International workshop on internet and network economics*, Springer.

Boschetti, M. A., V. Maniezzo, M. Roffilli, and A. B. Röhler, 2009, "Matheuristics: Optimization, simulation and control", in *International Workshop on Hybrid Metaheuristics*, Springer.

Bracken, J., and J. T. McGill, 1973, "Mathematical programs with optimization problems in the constraints", *Operations Research*, Vol. 21(1), pp. 37–44.

Budak, C., D. Agrawal, and A. El Abbadi, 2011, "Limiting the spread of misinformation in social networks", in *Proceedings of the 20th international conference on World wide web*, ACM.

Caprara, A., M. Carvalho, A. Lodi, and G. J. Woeginger, 2016, "Bilevel knapsack with interdiction constraints", *INFORMS Journal on Computing*, Vol. 28(2), pp. 319–333.

Caramia, M., and R. Mari, 2015, "Enhanced exact algorithms for discrete bilevel linear problems", *Optimization Letters*, Vol. 9(7), pp. 1447–1468.

Carnes, T., C. Nagarajan, S. M. Wild, and A. Van Zuylen, 2007, "Maximizing influence in a competitive social network: a follower's perspective", in *Proceedings of the ninth international conference on Electronic commerce*, ACM.

Chen, N., 2009, "On the approximability of influence in social networks", *SIAM Journal on Discrete Mathematics*, Vol. 23(3), pp. 1400–1415.

Chen, W., A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan, 2011, "Influence maximization in social networks when negative opinions may emerge and propagate", in *Proceedings of the 2011 SIAM International Conference on Data Mining*, SIAM.

Chen, W., L. V. Lakshmanan, and C. Castillo, 2013, "Information and influence propagation in social networks", *Synthesis Lectures on Data Management*, Vol. 5(4), pp. 1–177.

Chen, W., C. Wang, and Y. Wang, 2010a, "Scalable influence maximization for prevalent viral marketing in large-scale social networks", in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM.

Chen, W., Y. Wang, and S. Yang, 2009, "Efficient influence maximization in social networks", in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM.

Chen, W., Y. Yuan, and L. Zhang, 2010b, "Scalable influence maximization in social networks under the linear threshold model", in *2010 IEEE International Conference on Data Mining*, IEEE.

Church, R. L., and M. P. Scaparra, 2007, "Protecting critical assets: the r-interdiction median problem with fortification", *Geographical Analysis*, Vol. 39(2), pp. 129–146.

Clark, A., and R. Poovendran, 2011, "Maximizing influence in competitive environments: a game-theoretic approach", in *International Conference on Decision and Game Theory for Security*, Springer.

Cohen, R., S. Havlin, and D. Ben-Avraham, 2003, "Efficient immunization strategies for computer networks and populations", *Physical review letters*, Vol. 91(24), p. 247901.

Colson, B., P. Marcotte, and G. Savard, 2005, "Bilevel programming: A survey", *4OR: A Quarterly Journal of Operations Research*, Vol. 3(2), pp. 87–107.

Colson, B., P. Marcotte, and G. Savard, 2007, "An overview of bilevel optimization", *Annals of operations research*, Vol. 153(1), pp. 235–256.

Conforti, M., G. Cornuéjols, and G. Zambelli, 2011, "Corner polyhedron and intersection cuts", *Surveys in Operations Research and Management Science*, Vol. 16(2), pp. 105–120.

Cooke, K. L., 1979, "Stability analysis for a vector disease model", *The Rocky Mountain Journal of Mathematics*, Vol. 9(1), pp. 31–42.

Cornell University, 1991, *ArXiv.org High Energy Physics - Theory*, `https://arxiv.org/archive/hep-th`, accessed in August 2018.

Cornuejols, G., M. L. Fisher, and G. L. Nemhauser, 1977, "Exceptional paper—location of bank accounts to optimize float: An analytic study of exact and approximate algorithms", *Management science*, Vol. 23(8), pp. 789–810.

Dempe, S., 2018, *Bilevel optimization: theory, algorithms and applications*, TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik.

Dempe, S., S. Ivanov, and A. Naumov, 2017, "Reduction of the bilevel stochastic optimization problem with quantile objective function to a mixed-integer problem", *Applied Stochastic Models in Business and Industry*, Vol. 33(5), pp. 544–554.

DeNegre, S., and T. Ralphs, 2009, "A branch-and-cut algorithm for integer bilevel linear programs", in J. Chinneck, B. Kristjansson, and M. Saltzman, editors, *Operations Research and Cyber-Infrastructure. Operations Research/Computer Science Interfaces*, Springer, Boston, MA.

Dijkstra, E. W., 1959, "A note on two problems in connexion with graphs", *Numerische Mathematik*, Vol. 1(1), pp. 269–271, `https://doi.org/10.1007/BF01386390`.

Dolan, E. D., and J. J. Moré, 2002, "Benchmarking optimization software with performance profiles", *Mathematical Programming*, Vol. 91(2), pp. 201–213.

Domingos, P., and M. Richardson, 2001, "Mining the network value of customers", in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM.

Emelogu, A., S. Chowdhury, M. Marufuzzaman, L. Bian, and B. Eksioglu, 2016, "An enhanced sample average approximation method for stochastic optimization", *International Journal of Production Economics*, Vol. 182, pp. 230–252.

Erdos, P., and A. Rényi, 1960, "On the evolution of random graphs", *Publ. Math. Inst. Hung. Acad. Sci*, Vol. 5(1), pp. 17–60.

Fampa, M., L. Barroso, D. Candal, and L. Simonetti, 2008, "Bilevel optimization applied to strategic pricing in competitive electricity markets", *Computational Optimization and Applications*, Vol. 39(2), pp. 121–142.

Fan, L., Z. Lu, W. Wu, B. Thuraisingham, H. Ma, and Y. Bi, 2013, "Least cost rumor blocking in social networks", in *2013 IEEE 33rd International Conference on Distributed Computing Systems*, IEEE.

Fischetti, M., M. Kahr, M. Leitner, M. Monaci, and M. Ruthmair, 2018, "Least cost influence propagation in (social) networks", *Mathematical Programming*, Vol. 170(1), pp. 293–325.

Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl, 2016, "Intersection cuts for bilevel optimization", in *International Conference on Integer Programming and Combinatorial Optimization*, Springer.

Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl, 2017a, *Instances and solver software for mixed-integer bilevel linear problems*, `https://msinnl.github.io/pages/bilevel.html`, accessed in May 2020.

Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl, 2017b, "A new general-purpose algorithm for mixed-integer bilevel linear programs", *Operations Research*, Vol. 65(6), pp. 1615–1637.

Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl, 2019, "Interdiction games and monotonicity, with application to knapsack problems", *INFORMS Journal on Computing*, Vol. 31(2), pp. 390–410.

Fortuny-Amat, J., and B. McCarl, 1981, "A representation and economic interpretation of a two-level programming problem", *Journal of the operational Research Society*, Vol. 32(9), pp. 783–792.

Furini, F., I. Ljubić, S. Martin, and P. San Segundo, 2019, "The maximum clique interdiction problem", *European Journal of Operational Research*, Vol. 277(1), pp. 112–127.

Galhotra, S., A. Arora, and S. Roy, 2016, "Holistic influence maximization: Combining scalability and efficiency with opinion-aware models", in *Proceedings of the 2016 International Conference on Management of Data*, ACM.

Gendreau, M., P. Marcotte, and G. Savard, 1996, "A hybrid tabu-ascent algorithm for the linear bilevel programming problem", *Journal of Global Optimization*, Vol. 8(3), pp. 217–233.

Glover, F., 1974, "Polyhedral convexity cuts and negative edge extensions", *Zeitschrift für Operations Research*, Vol. 18(5), pp. 181–186.

Goyal, A., F. Bonchi, and L. V. Lakshmanan, 2010, "Learning influence probabilities in social networks", in *Proceedings of the third ACM international conference on Web search and data mining*, ACM.

Goyal, S., H. Heidari, and M. Kearns, 2019, "Competitive contagion in networks", *Games and Economic Behavior*, Vol. 113, pp. 58–79.

Güney, E., 2017, "On the optimal solution of budgeted influence maximization problem in social networks", *Operational Research*, pp. 1–15.

Güney, E., 2019, "An efficient linear programming based method for the influence maximization problem in social networks", *Information Sciences*, Vol. 503, pp. 589–605.

Güney, E., M. Leitner, M. Ruthmair, and M. Sinnl, 2019, "Large-scale influence maximization via maximal covering location", Tech. Rep., University of Vienna, `http://www.optimization-online.org/DB_FILE/2018/12/6985.pdf`.

Günneç, D., S. Raghavan, and R. Zhang, 2019, "Least-cost influence maximization on social networks", *INFORMS Journal on Computing*.

Gursoy, F., and D. Gunnec, 2018, "Influence maximization in social networks under deterministic linear threshold model", *Knowledge-Based Systems*, Vol. 161, pp. 111–123.

He, X., G. Song, W. Chen, and Q. Jiang, 2012, "Influence blocking maximization in social networks under the competitive linear threshold model", in *Proceedings of the 2012 SIAM International Conference on Data Mining*, SIAM.

Hejazi, S. R., A. Memariani, G. Jahanshahloo, and M. M. Sepehri, 2002, "Linear bilevel programming solution by genetic algorithm", *Computers & Operations Research*, Vol. 29(13), pp. 1913–1925.

Hemmati, M., J. C. Smith, and M. T. Thai, 2014, "A cutting-plane algorithm for solving a weighted influence interdiction problem", *Computational Optimization and Applications*, Vol. 57(1), pp. 71–104.

Holme, P., 2004, "Efficient local strategies for vaccination and network attack", *EPL (Europhysics Letters)*, Vol. 68(6), p. 908.

Huang, K., S. Wang, G. Bevilacqua, X. Xiao, and L. V. Lakshmanan, 2017, "Revisiting the stop-and-stare algorithms for influence maximization", *Proceedings of the VLDB Endowment*, Vol. 10(9), pp. 913–924.

Israeli, E., and R. K. Wood, 2002, "Shortest-path network interdiction", *Networks: An International Journal*, Vol. 40(2), pp. 97–111.

Jeroslow, R. G., 1985, "The polynomial hierarchy and a simple model for competitive analysis", *Mathematical programming*, Vol. 32(2), pp. 146–164.

Jung, K., W. Heo, and W. Chen, 2012, "Irie: Scalable and robust influence maximization in social networks", in *2012 IEEE 12th International Conference on Data Mining*, IEEE.

Kahr, M., M. Leitner, M. Ruthmair, and M. Sinnl, 2020, "Benders decomposition for competitive influence maximization in (social) networks", .

Kaur, H., and J. He, 2017, "Blocking negative influential node set in social networks: from host perspective", *Transactions on Emerging Telecommunications Technologies*, Vol. 28(4), p. e3007.

Kempe, D., J. Kleinberg, and É. Tardos, 2003, "Maximizing the spread of influence through a social network", in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM.

Kempe, D., J. Kleinberg, and E. Tardos, 2015, "Maximizing the spread of influence through a social network", *Theory of Computing*, Vol. 11(4), pp. 105–147.

Keskin, M. E., and M. G. Güler, 2018, "Influence maximization in social networks: an integer programming approach", *Turkish Journal of Electrical Engineering & Computer Sciences*, Vol. 26(6), pp. 3383–3396.

Kimura, M., K. Saito, and H. Motoda, 2008a, "Minimizing the spread of contamination by blocking links in a network", in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*.

Kimura, M., K. Saito, and H. Motoda, 2008b, "Solving the contamination minimization problem on networks for the linear threshold model", in *Pacific Rim International Conference on Artificial Intelligence*, Springer.

Kimura, M., K. Saito, and H. Motoda, 2009, "Blocking links to minimize contamination spread in a social network", *ACM Transactions on Knowledge Discovery from Data (TKDD)*, Vol. 3(2), pp. 1–23.

Kimura, M., K. Saito, and R. Nakano, 2007, "Extracting influential nodes for information diffusion on a social network", in *AAAI*.

Kleywegt, A. J., A. Shapiro, and T. Homem-de Mello, 2002, "The sample average approximation method for stochastic discrete optimization", *SIAM Journal on Optimization*, Vol. 12(2), pp. 479–502.

Kostka, J., Y. A. Oswald, and R. Wattenhofer, 2008, "Word of mouth: Rumor dissemination in social networks", in *International Colloquium on Structural Information and Communication Complexity*, Springer.

Kosuch, S., P. Le Bodic, J. Leung, and A. Lisser, 2012, "On a stochastic bilevel programming problem", *Networks*, Vol. 59(1), pp. 107–116.

Kuhlman, C. J., V. A. Kumar, M. V. Marathe, S. Ravi, and D. J. Rosenkrantz, 2010, "Finding critical nodes for inhibiting diffusion of complex contagions in social networks", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer.

Kuhlman, C. J., G. Tuli, S. Swarup, M. V. Marathe, and S. Ravi, 2013, "Blocking simple and complex contagion by edge removal", in *2013 IEEE 13th International Conference on Data Mining*, IEEE.

Kuo, R., and C. Huang, 2009, "Application of particle swarm optimization algorithm for solving bi-level linear programming problem", *Computers & Mathematics with Applications*, Vol. 58(4), pp. 678–685.

Laporte, G., and F. V. Louveaux, 1993, "The integer l-shaped method for stochastic integer programs with complete recourse", *Operations research letters*, Vol. 13(3), pp. 133–142.

Leskovec, J., A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, 2007, "Cost-effective outbreak detection in networks", in *Proceedings of the 13th ACM SIGKDD International conference on Knowledge discovery and data mining*, ACM.

Li, X., J. D. Smith, T. N. Dinh, and M. T. Thai, 2019, "Tiptop:(almost) exact solutions for influence maximization in billion-scale networks", *IEEE/ACM Transactions on Networking*, Vol. 27(2), pp. 649–661.

Li, Y., J. Fan, Y. Wang, and K.-L. Tan, 2018, "Influence maximization on social graphs: A survey", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 30(10), pp. 1852–1872.

Linderoth, J., A. Shapiro, and S. Wright, 2006, "The empirical behavior of sampling methods for stochastic programming", *Annals of Operations Research*, Vol. 142(1), pp. 215–241.

Lozano, L., and J. C. Smith, 2016, "A backward sampling framework for interdiction problems with fortification", *INFORMS Journal on Computing*, Vol. 29(1), pp. 123–139.

Lozano, L., and J. C. Smith, 2017, "A value-function-based exact approach for the bilevel mixed-integer programming problem", *Operations Research*, Vol. 65(3), pp. 768–786.

Lu, J., J. Han, Y. Hu, and G. Zhang, 2016, "Multilevel decision-making: A survey", *Information Sciences*, Vol. 346, pp. 463–487.

Lu, Z., W. Zhang, W. Wu, B. Fu, and D. Du, 2011, "Approximation and inapproximation for the influence maximization problem in social networks under deterministic linear threshold model", in *2011 31st International Conference on Distributed Computing Systems Workshops*, IEEE.

Lu, Z., W. Zhang, W. Wu, J. Kim, and B. Fu, 2012, "The complexity of influence maximization problem in the deterministic linear threshold model", *Journal of combinatorial optimization*, Vol. 24(3), pp. 374–378.

Mak, W.-K., D. P. Morton, and R. K. Wood, 1999, "Monte carlo bounding techniques for determining solution quality in stochastic programs", *Operations Research Letters*, Vol. 24(1), pp. 47–56.

McKay, M. D., R. J. Beckman, and W. J. Conover, 1979, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code", *Technometrics*, Vol. 21(2), pp. 239–245.

Montgomery, D. C., and G. C. Runger, 2010, *Applied statistics and probability for engineers*, John Wiley & Sons.

Moore, J. T., and J. F. Bard, 1990, "The mixed integer linear bilevel programming problem", *Operations research*, Vol. 38(5), pp. 911–921.

Nemhauser, G. L., L. A. Wolsey, and M. L. Fisher, 1978, "An analysis of approximations for maximizing submodular set functions—i", *Mathematical Programming*, Vol. 14(1), pp. 265–294.

Newman, M. E., 2002, "Spread of epidemic disease on networks", *Physical review E*, Vol. 66(1), p. 016128.

Nguyen, H. T., M. T. Thai, and T. N. Dinh, 2016, "Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks", in *Proceedings of the 2016 International Conference on Management of Data*, ACM.

Nguyen, N. P., G. Yan, M. T. Thai, and S. Eidenbenz, 2012, "Containment of misinformation spread in online social networks", in *Proceedings of the 4th Annual ACM Web Science Conference*, ACM.

Nishizaki, I., and M. Sakawa, 2005, "Computational methods through genetic algorithms for obtaining stackelberg solutions to two-level integer programming problems", *Cybernetics and Systems: An International Journal*, Vol. 36(6), pp. 565–579.

Norkin, V. I., G. C. Pflug, and A. Ruszczyński, 1998, "A branch and bound method for stochastic global optimization", *Mathematical Programming*, Vol. 83(1), pp. 425–450, `http://dx.doi.org/10.1007/BF02680569`.

Ohlmann, J., and B. Thomas, 2007, "A compressed annealing approach to the traveling salesman problem with time windows", *Informs Journal on Computing*, Vol. 19(1), pp. 80–90.

Özaltın, O. Y., O. A. Prokopyev, and A. J. Schaefer, 2010, "The bilevel knapsack problem with stochastic right-hand sides", *Operations Research Letters*, Vol. 38(4), pp. 328–333.

Richardson, M., and P. Domingos, 2002, "Mining knowledge-sharing sites for viral marketing", in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM.

Scaparra, M. P., and R. L. Church, 2008, "A bilevel mixed-integer program for critical infrastructure protection planning", *Computers & Operations Research*, Vol. 35(6), pp. 1905–1923.

Scott, J., 2012, *Social network analysis*, Sage.

Shen, S., J. C. Smith, and R. Goli, 2012, "Exact interdiction models and algorithms for disconnecting networks via node deletions", *Discrete Optimization*, Vol. 9(3), pp. 172–188.

Sherali, H. D., and C. H. Tuncbilek, 1992, "A squared-euclidean distance location-allocation problem", *Naval Research Logistics (NRL)*, Vol. 39(4), pp. 447–469.

Shirazipourazad, S., B. Bogard, H. Vachhani, A. Sen, and P. Horn, 2012, "Influence propagation in adversarial setting: how to defeat competition with least amount of investment", in *Proceedings of the 21st ACM international conference on Information and knowledge management*, ACM.

Smith, J. C., 2016, *Test instances*, `http://jcsmith.people.clemson.edu/Test_Instances.html`, accessed in November 2019.

Smith, J. C., and Y. Song, 2020, "A survey of network interdiction models and algorithms", *European Journal of Operational Research*, Vol. 283(3), pp. 797–811.

Stackelberg, H. V., 1952, *Theory of the market economy*, Oxford University Press.

Sumith, N., B. Annappa, and S. Bhattacharya, 2018, "Influence maximization in large social networks: Heuristics, models and parameters", *Future Generation Computer Systems*, Vol. 89, pp. 777–790.

Sun, J., and J. Tang, 2013, "Models and algorithms for social influence analysis", in *Proceedings of the sixth ACM international conference on Web search and data mining*, ACM.

Tan, Z., D. Wu, T. Gao, I. You, and V. Sharma, 2019, "Aim: Activation increment minimization strategy for preventing bad information diffusion in osns", *Future Generation Computer Systems*, Vol. 94, pp. 293–301.

Tang, J., J. Sun, C. Wang, and Z. Yang, 2009, "Social influence analysis in large-scale networks", in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM.

Tang, Y., J.-P. P. Richard, and J. C. Smith, 2016, "A class of algorithms for mixed-integer bilevel min–max optimization", *Journal of Global Optimization*, Vol. 66(2), pp. 225–262.

Tang, Y., Y. Shi, and X. Xiao, 2015, "Influence maximization in near-linear time: A martingale approach", in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ACM.

Tang, Y., X. Xiao, and Y. Shi, 2014, "Influence maximization: Near-optimal time complexity meets practical efficiency", in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, ACM.

Tanınmış, K., N. Aras, and I. K. Altınel, 2019, "Influence maximization with deactivation in social networks", *European Journal of Operational Research*, Vol. 278(1), pp. 105–119.

Tanınmış, K., N. Aras, and I. K. Altınel, 2020a, "Live-arc formulation for the competitive influence maximization problem", Tech. Rep. No: FBE/IE-02/2020-02, Department of Industrial Engineering, Boğaziçi University, İstanbul, Turkey,

Tanınmış, K., N. Aras, and I. K. Altınel, 2020b, "Improved x-space algorithm for min-max bilevel integer programming with an application to misinformation spread in social networks", Tech. Rep. No: FBE/IE-03/2020-03, Department of Industrial Engineering, Boğaziçi University, İstanbul, Turkey,

Tanınmış, K., N. Aras, I. K. Altınel, and E. Güney, 2020c, "Minimizing the misinformation spread in social networks", *IISE Transactions*, Vol. 52(8), pp. 850–863.

Tsai, J., T. H. Nguyen, and M. Tambe, 2012, "Security games for controlling contagion", in *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Tzoumas, V., C. Amanatidis, and E. Markakis, 2012, "A game-theoretic analysis of a competitive diffusion process over social networks", in *International Workshop on Internet and Network Economics*, Springer.

Venkatramanan, S., and A. Kumar, 2014, "Competition for content spread over multiple social networks", in *Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on*, IEEE.

Verweij, B., S. Ahmed, A. J. Kleywegt, G. Nemhauser, and A. Shapiro, 2003, "The sample average approximation method applied to stochastic routing problems: a computational study", *Computational Optimization and Applications*, Vol. 24(2), pp. 289–333.

Vicente, L., G. Savard, and J. Júdice, 1994, "Descent approaches for quadratic bilevel programming", *Journal of Optimization Theory and Applications*, Vol. 81(2), pp. 379–399.

Wang, B., G. Chen, L. Fu, L. Song, and X. Wang, 2017, "Drimux: Dynamic rumor influence minimization with user experience in social networks", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 29(10), pp. 2168–2181.

Wang, C., W. Chen, and Y. Wang, 2012, "Scalable influence maximization for independent cascade model in large-scale social networks", *Data Mining and Knowledge Discovery*, Vol. 25(3), p. 545.

Wang, L., and P. Xu, 2017, "The watermelon algorithm for the bilevel integer linear programming problem", *SIAM Journal on Optimization*, Vol. 27(3), pp. 1403–1430.

Wang, S., X. Zhao, Y. Chen, Z. Li, K. Zhang, and J. Xia, 2013, "Negative influence minimizing by blocking nodes in social networks", in *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Wasserman, S., and K. Faust, 1994, *Social network analysis: Methods and applications*, Vol. 8, Cambridge university press.

Watts, D. J., and S. H. Strogatz, 1998, "Collective dynamics of 'small-world'networks", *nature*, Vol. 393(6684), pp. 440–442.

Wen, U., and A. Huang, 1996, "A simple tabu search method to solve the mixed-integer linear bilevel programming problem", *European Journal of Operational Research*, Vol. 88(3), pp. 563–571.

Wood, R. K., 1993, "Deterministic network interdiction", *Mathematical and Computer Modelling*, Vol. 17(2), pp. 1–18.

Wood, R. K., 2010, "Bilevel network interdiction models: Formulations and solutions", *Wiley encyclopedia of operations research and management science*.

Wu, H.-H., and S. Küçükyavuz, 2018, "A two-stage stochastic programming approach for influence maximization in social networks", *Computational Optimization and Applications*, Vol. 69(3), pp. 563–595.

Wu, P., and L. Pan, 2017, "Scalable influence blocking maximization in social networks under competitive independent cascade models", *Computer Networks*, Vol. 123, pp. 38–50.

Xu, P., and L. Wang, 2014, "An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions", *Computers & Operations Research*, Vol. 41, pp. 309–318.

Yao, Q., R. Shi, C. Zhou, P. Wang, and L. Guo, 2015, "Topic-aware social influence minimization", in *Proceedings of the 24th International Conference on World Wide Web*, ACM.

Yu, Y., T. Y. Berger-Wolf, J. Saia, *et al.*, 2008, "Finding spread blockers in dynamic networks", in *International Workshop on Social Network Mining and Analysis*, Springer, pp.

# APPENDIX A: k-means++ Algorithm

**Algorithm A.1** $k$-means

Choose the number of clusters $k$
Initialize the cluster centers $\mu \leftarrow \emptyset$ and $C_i$, the set of points in cluster $i$, $C_i \leftarrow \emptyset$, $i = 1, ..., k$
Generate $T$, a set of threshold scenarios of size $N_C$
Select a point in $T$ as the first cluster center $\mu_1$, set $\mu \leftarrow \mu \cup \mu_1$
**while** the number of cluster centers chosen $|\mu| < k$ **do**
    Calculate the minimum squared Euclidean distance $d(\theta, \mu)$, from each point $\theta \in T$ to the closest cluster center
    Select $\theta^* \in T$ with probability $\frac{d(\theta^*,\mu)}{\sum_{\theta \in T} d(\theta,\mu)}$ as the new cluster center, $\mu \leftarrow \mu \cup \theta^*$
**end while**
**while** $iterCount < maxIter$ **do**
    Set $C_i \leftarrow \emptyset$, $i = 1, ..., k$
    Assign each point $\theta \in T$ to the closest cluster, $C_i \leftarrow C_i \cup \theta$
    Update the cluster centers $\mu_i$ as the center of gravity of $C_i$ for $i = 1, ..., k$
**end while**
Return $C_i$, $i = 1, .., k$

Figure A.1. Separation of an Intersection Cut.

# APPENDIX B: Branch-and-Cut Method with Intersection Cuts

## B.1. Illustration of the Method

Fischetti *et al.* (2016) demonstrates how their method works on the following example.

$$\begin{aligned}
&\min_{x\in\mathbb{Z}} -x-10y \\
&y \in \arg\min_{y'\in\mathbb{Z}}\{y' : \quad -25x+20y' \le 30 \\
&\qquad\qquad\qquad\qquad\qquad\qquad x+2y' \le 10 \\
&\qquad\qquad\qquad\qquad\qquad\qquad 2x-y' \le 15 \\
&\qquad\qquad\qquad\qquad\qquad\qquad 2x+10y' \ge 15\}.
\end{aligned}$$

The optimal solution of the LP relaxation of HPR, $A = (2, 4)$, is integral but not bilevel feasible. The corresponding bilevel-free set is obtained as $S^+(\hat{y} = 2) = \{(x, y) : 9/25 \le x \le 2, y \ge 2\}$ using the definition in (5.11). Then, using this set the intersection cut (IC) $y \le 2$ (intersection of corner polyhedron and $S^+(\hat{y})$) is generated and added to the problem. The next LP solution $B = (6, 2)$ is still bilevel infeasible. The corresponding bilevel-free set $S^+(\hat{y} = 1) = \{(x, y) : 2 \le x \le 17/2, y \ge 1\}$ yields the IC: $x + 6y \le 14$. In the third iteration, solution $(2, 2)$ obtained and it is bilevel feasible. It is the optimal solution. The feasible region and the generated cuts are shown in Figure B.1. (Note that the problem is solved at the root node of the B&C tree in this example.)

## B.2. Obtaining Intersection Cuts

While generating the ICs, the approach in Glover (1974) is followed by Fischetti *et al.* (2017b). Once the LP at the current B&C node is reformulated in standard form as $\max\{c^T\xi : \hat{A}\xi = \hat{b}, \xi \ge 0\}$ where $\xi$ is the whole variable vector including

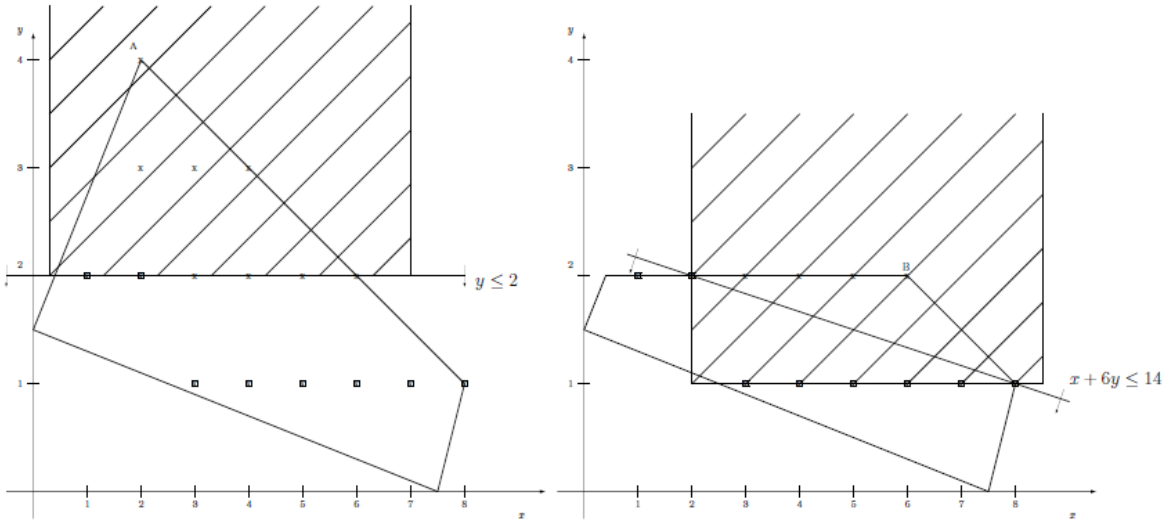



Figure B.1. IC example using $S^+(\hat{y})$.

slack/surplus variables, and the corresponding bilevel-free set $S$ is generated as $S = \{\xi : g_k^T \xi \leq g_{k0}, k = 1, ..., K\}$, the IC can be generated following the steps in Figure B.2. Note that the set $S$ can be any convex set that contains the optimal solution $\xi^*$ of the current LP and does not contain any bilevel feasible point in its interior (except the set $\text{HC}^+(x^*)$ in Hypercube IC method). In order to cut off $\xi^*$ from the feasible region of the current subproblem, we make use of the fact that all the inequalities in the following disjunction are violated by $\xi^*$:

$$\bigvee_{k=1}^{K} g_k^T \xi \geq g_{k0}. \tag{B.1}$$

Using the set $S^+(\hat{\mathbf{y}}, \hat{\mathbf{u}})$ defined for our problem in (5.13), the following disjunction is obtained:

$$\begin{aligned} \left( \sum_{r \in R} \sum_{i \in V} \hat{u}_{ir} - \sum_{r \in R} \sum_{i \in V} u_{ir} \geq 0 \right) \vee \left( \hat{y}_1 - x_1 \geq 1 \right) \vee \left( x_1 - \hat{y}_1 - \hat{u}_{1,1} \geq 1 \right) \vee \\ \ldots \vee \left( \hat{y}_{n_1} - x_{n_1} \geq 1 \right) \vee \left( x_{n_1} - \hat{y}_{n_1} - \hat{u}_{n_1,R} \geq 1 \right). \end{aligned} \tag{B.2}$$

The steps of the separation algorithm in Figure B.2 guarantee to obtain a single inequality that is valid for each term of the disjunction and violated by $\xi^*$.

**Algorithm B.2** IC Separation

**Input**: An LP vertex $\xi^*$ along with its associated LP basis $\hat{B}$, the feasible-free polyhedron $S = \{\xi : g_k^T \xi \leq g_{k0}, k = 1, ..., K\}$ and the associated valid disjunction $\bigvee_{k=1}^{K} g_k^T \xi \geq g_{k0}$ whose members are violated by $\xi^*$;

**Output**: A valid IC violated by $\xi^*$;

```
for k = 1 to K do
    (ḡ_k^T, ḡ_k0) = (g_k^T, g_k0) − u_k^T(Â, b̂) where u_k^T = (g_k)_B̂^T B̂^{−1}
end for
for j = 1 to n do
    γ_j = max{g_kj/g_k0 : k ∈ {1, ..., K}}
end for
if γ ≥ 0 then
    for j = 1 to n do
        if ξ_j is integer constrained then
            γ_j = min{γ_j, 1}
        end if
    end for
end if
return the violated cut: γ^T ξ ≥ 1.
```

Figure B.2. Separation of an Intersection Cut.

Since it is not allowed to include terms with slack/surplus variables in a lazy constraint in the Callable Library environment, the LHS of the IC is rewritten as

$$\begin{aligned}\gamma^T \xi &= \sum_{j=1}^{n_1+n_2} \gamma_j \xi_j + \sum_{i=1}^{m} \gamma_{n_1+n_2+i} \big(\hat{b}_i - \sum_{j=1}^{n_1+n_2} \hat{A}_{ij} \xi_j\big) \\ &= \sum_{j=1}^{n_1+n_2} \big(\gamma_j - \sum_{i=1}^{m} \gamma_{n_1+n_2+i} \hat{A}_{ij}\big) \xi_j + \sum_{i=1}^{m} \gamma_{n_1+n_2+i} \hat{b}_i\end{aligned}$$

where $m$ stands for the number of constraints in the current subproblem. Therefore, the cut inequality becomes

$$\sum_{j=1}^{n_1+n_2} \big(\gamma_j - \sum_{i=1}^{m} \gamma_{n_1+n_2+i} \hat{A}_{ij}\big) \xi_j \geq 1 - \sum_{i=1}^{m} \gamma_{n_1+n_2+i} \hat{b}_i. \tag{B.3}$$

Let $\bar{\gamma}_j = \gamma_j - \sum_{i=1}^{m} \gamma_{n_1+n_2+i}\hat{A}_{ij}$ denote the new cut coefficient of $\xi_j, j = 1, ..., n_1 + n_2$. Then, the final cut can be expressed as

$$\sum_{j=1}^{n_1+n_2} \bar{\gamma}_j \xi_j \geq 1 - \sum_{i=1}^{m} \gamma_{n_1+n_2+i}\hat{b}_i. \tag{B.4}$$

### B.3. The Separation Mixed-integer Linear Program

The following MILP is solved in separation methods BFS4 and BFS5 to determine the follower solution which maximizes the number of removable facets (equivalently minimizes the number of unremovable facets) in the BFS (5.13).

$$\begin{aligned}
&\min \sum_{k=1}^{m} w_k \\
&\text{s.t.} \\
&\quad \sum_{i=1}^{n}\sum_{r=1}^{R} u_{ir} \leq \sum_{i=1}^{n}\sum_{r=1}^{R} u^*_{ir} - 1 \\
&\quad \sum_{i=1}^{n} y_i = l \\
&\quad -y_i - u_{ir} + u_{\delta_r(i),r} \leq 0 && i = 1, ..., n,\ r = 1, ..., R \\
&\quad y_i + s_i = 0 && i = 1, ..., n \\
&\quad -y_i - u_{ir} + s_k = 0 && i = 1, ..., n,\ r = 1, ..., R,\ k = n + (i-1)R + r \\
&\quad s_k + (L_k^{\max} - L_k^*)w_k \geq L_k^{\max} && k = 1, ..., m \\
&\quad y_i \in \{0, 1\}, && i = 1, ..., n,\ r = 1, ..., R \\
&\quad w_k \in \{0, 1\},\ s_k \text{ free} && k = 1, ..., m,
\end{aligned}$$

where $m = n + n.R$ is the number of follower constraints excluding the cardinality restrictions. $L^*$ and $L^{\max}$ denote the current value and maximum possible value of the terms with leader variables in each follower constraint when they are written as $Ax + By \leq b$, respectively. They are obtained as follows: for $k = 1, ..., n$, $L_k^* = -x_k^*$

and $L_k^{\max} = -\underline{x_k}$. For $i = 1, ..., n$, $r = 1, ..., R$, $k = n + (i-1)R + r$, $L_k^* = x_i^*$ and $L_k^{\max} = \overline{x_i}$. $s_k$ represent the slacks such that if $s_k \geq L_k^{\max}$ then that facet is removable and the corresponding indicator variable $w_k$ is set to zero due to the objective function. The objective is to minimize the number of unremovable facets in the bilevel-free set.

## B.4. Additional Results

Table B.1. Solution times with Alternating Bound Update.

| $n$ | $R$ | BFS1 | BFS2 | BFS3 | BFS4 | BFS5 | BFS6 | HC | CE Time |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 1 | 18.29 | 11.44 | 11.03 | 10.31 | 15.27 | 9.69 | 14.34 | 10.63 |
| | 10 | 91.22 | 71.83 | 48.79 | 90.00 | 96.47 | 42.78 | 65.65 | 19.43 |
| | 25 | 606.92 | 532.16 | 448.42 | 527.48 | 410.69 | 252.72 | 408.30 | 24.76 |
| | 50 | 1413.53 | 1365.87 | 1148.61 | 1651.85 | 1358.20 | 790.39 | 1404.96 | 32.74 |
| *Average* | | *532.49* | *495.32* | *414.21* | *569.91* | *470.16* | *273.89* | *473.31* | *21.89* |
| 30 | 1 | 123.43 | 72.16 | 73.10 | 42.59 | 68.54 | 58.18 | 96.30 | 62.53 |
| | 10 | 1003.44 | 828.94 | 527.28 | 779.36 | 842.74 | 397.93 | 644.14 | 118.63 |
| | 25 | 3600.25 | 3600.28 | 3556.71 | 3546.90 | 3401.86 | 3516.22 | 3528.93 | 158.98 |
| | 50 | 3600.34 | 3600.53 | 3600.66 | 3600.72 | 3600.51 | 3601.02 | 3600.66 | 308.79 |
| *Average* | | *2081.86* | *2025.48* | *1939.44* | *1992.39* | *1978.41* | *1893.34* | *1967.51* | *162.23* |
| 40 | 1 | 165.64 | 112.80 | 104.71 | 82.69 | 99.64 | 98.66 | 148.33 | 218.08 |
| | 10 | 1464.62 | 1264.57 | 823.11 | 1152.12 | 1048.65 | 575.68 | 1031.76 | 367.60 |
| | 25 | 3600.35 | 3600.34 | 3600.36 | 3600.17 | 3319.32 | 3422.80 | 3600.34 | 436.19 |
| | 50 | 3600.55 | 3600.76 | 3600.37 | 3602.22 | 3600.99 | 3600.46 | 3600.62 | 854.80 |
| *Average* | | *2207.79* | *2144.62* | *2032.14* | *2109.30* | *2017.15* | *1924.40* | *2095.26* | *469.17* |
| *Overall Avg.* | | *1607.38* | *1555.14* | *1461.93* | *1557.20* | *1488.57* | *1363.88* | *1512.03* | *217.76* |

Table B.2. Solution times without Alternating Bound Update.

| $n$ | $R$ | BFS1 | BFS2 | BFS3 | BFS4 | BFS5 | BFS6 | HC | CE Time |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 1 | 21.09 | 14.08 | 13.69 | 16.41 | 17.79 | 11.20 | 18.06 | 10.63 |
| | 10 | 108.41 | 107.10 | 75.32 | 340.19 | 372.01 | 45.58 | 79.53 | 19.43 |
| | 25 | 940.56 | 889.74 | 793.13 | 3002.78 | 2959.61 | 329.15 | 651.18 | 24.76 |
| | 50 | 2516.77 | 2376.87 | 1909.25 | 3600.48 | 3600.36 | 1067.74 | 2226.49 | 32.74 |
| *Average* | | *896.71* | *846.95* | *697.85* | *1739.96* | *1737.44* | *363.42* | *743.81* | *21.89* |
| 30 | 1 | 136.53 | 113.64 | 90.51 | 101.55 | 113.54 | 64.65 | 128.49 | 62.53 |
| | 10 | 1635.28 | 1390.67 | 859.82 | 3600.02 | 3600.01 | 474.08 | 1009.15 | 118.63 |
| | 25 | 3600.30 | 3600.25 | 3600.22 | 3600.33 | 3600.21 | 3600.32 | 3600.22 | 158.98 |
| | 50 | 3600.69 | 3600.27 | 3600.47 | 3602.08 | 3601.10 | 3600.55 | 3600.79 | 308.79 |
| *Average* | | *2243.20* | *2176.21* | *2037.75* | *2725.99* | *2728.71* | *1934.90* | *2084.66* | *162.23* |
| 40 | 1 | 194.60 | 161.27 | 140.43 | 152.96 | 159.00 | 112.90 | 164.59 | 218.08 |
| | 10 | 1827.88 | 1500.39 | 918.69 | 2626.78 | 2709.53 | 628.28 | 1067.56 | 367.60 |
| | 25 | 3600.22 | 3600.30 | 3600.34 | 3600.16 | 3600.18 | 3495.23 | 3600.32 | 436.19 |
| | 50 | 3601.08 | 3601.63 | 3600.81 | 3601.39 | 3600.59 | 3600.23 | 3600.74 | 854.80 |
| *Average* | | *2305.94* | *2215.90* | *2065.06* | *2495.32* | *2517.33* | *1959.16* | *2108.31* | *469.17* |
| *Overall Avg.* | | *1815.28* | *1746.35* | *1600.22* | *2320.43* | *2327.83* | *1419.16* | *1645.59* | *217.76* |

Table B.3. Number of proven optimalities with (without) Alternating Bound Update.

| $n$ | BFS1 | BFS2 | BFS3 | BFS4 | BFS5 | BFS6 | HC |
|---|---|---|---|---|---|---|---|
| 20 | 40 (38) | 40 (39) | 40 (40) | 40 (24) | 40 (24) | 40 (40) | 40 (39) |
| 30 | 20 (20) | 20 (20) | 21 (20) | 21 (10) | 21 (10) | 22 (20) | 21 (20) |
| 40 | 20 (20) | 20 (20) | 20 (20) | 20 (18) | 24 (16) | 22 (22) | 20 (20) |
| Total | 80 (78) | 80 (79) | 81 (80) | 81 (52) | 85 (50) | 84 (82) | 81 (79) |

Table B.4. Number of proven optimalities with (without) Alternating Bound Update.

| $R$ | BFS1 | BFS2 | BFS3 | BFS4 | BFS5 | BFS6 | HC |
|---|---|---|---|---|---|---|---|
| 1 | 30 (30) | 30 (30) | 30 (30) | 30 (30) | 30 (30) | 30 (30) | 30 (30) |
| 10 | 30 (30) | 30 (30) | 30 (30) | 30 (18) | 30 (16) | 30 (30) | 30 (30) |
| 25 | 10 (10) | 10 (10) | 11 (10) | 11 (4) | 15 (4) | 14 (12) | 11 (10) |
| 50 | 10 (8) | 10 (9) | 10 (10) | 10 (0) | 10 (0) | 10 (10) | 10 (9) |
| Total | 80 (78) | 80 (79) | 81 (80) | 81 (52) | 85 (50) | 84 (82) | 81 (79) |

# APPENDIX C: $x$-space Algorithm

**Algorithm C.1** XS

**Step 0: Initialization**
Define $J = \{-(\rho - 1), ..., 0\}$ as an index set of initial solutions where $\rho \in \mathbb{Z}_+$.
Select $\mathbf{y}^0$ such that $\mathbf{y}^0$ is a feasible follower solution for all feasible leader solutions.
Choose a feasible $\mathbf{x}^\tau$ and $\mathbf{y}^\tau \in \arg\max_{\mathbf{y}}\{\mathbf{p}^T\mathbf{y} : (\mathbf{x}^\tau, \mathbf{y}) \in \mathcal{F}^\circ\}$ for $\tau \in J \setminus \{0\}$.
Let $S_1 = \bigcup_{\tau \in J}\{\mathbf{y}^\tau\}$.
Set $q = 1$.
**Step** $q$**:** $(q = 1, 2, ...)$
    **Step** $q_1$**:** Obtain an optimal solution $\mathbf{x}^q$ and optimal value $t_q$ to the approximate leader problem $(LB_q)$.
    **Step** $q_2$**:** Obtain an optimal solution $\mathbf{y}^q$ and optimal value $z_q$ to the follower problem $(UB_q)$.
    **Step** $q_3$**: if** $t_q \neq z_q$ **then**
        Expand $S_{q+1} = S_q \cup \{\mathbf{y}^q\}$ and update $J = J \cup q$.
        **Go to** Step $q + 1$.
    **else**
        Return optimal solution $(\mathbf{x}^*, \mathbf{y}^*) = (\mathbf{x}^q, \mathbf{y}^q)$ and $z^* = z_q$.

Figure C.1. The $x$-space (XS) algorithm.

In the implementation of the original XS algorithm on the MSMP, the LB problem is formulated as follows:

$$
\begin{aligned}
&(LB_q): \\
t_q =& \min_{x \in \mathcal{X}} \max_{y,u} \left\{ \frac{1}{|R|} \sum_{r \in R} \sum_{i \in V} u_{ir} \mid (y, u) \in \text{conv}(S_q), y_i \leq 1 - x_i, u_{ir} \leq 1 - x_i, i \in V, r \in R \right\} \\
=& \min_{x \in \mathcal{X}} \max_{\lambda, y, u} \left\{ \frac{1}{|R|} \sum_{r \in R} \sum_{i \in V} u_{ir} \;\middle|\; \begin{array}{l} \sum_{k \in J} \lambda_k = 1 \\ y_i - \sum_{k \in J} \lambda_k y_i^k = 0,\ i \in V \\ u_{ir} - \sum_{k \in J} \lambda_k u_{ir}^k = 0,\ i \in V, r \in R \\ y_i \leq 1 - x_i,\ i \in V \\ u_{ir} \leq 1 - x_i,\ i \in V, r \in R \\ y_i \geq 0,\ u_{ir} \geq 0,\ \lambda_k \geq 0,\ i \in V, r \in R, k \in J \end{array} \right\}.
\end{aligned}
\tag{C.1}
$$

By treating $\mathbf{x}$ as a parameter of the inner maximization problem, the dual of the inner problem becomes:

$$\min\ \pi_0 + \sum_{i \in V} \phi'_i(1 - x_i) + \sum_{i \in V} \sum_{r \in R} \phi''_{ir}(1 - x_i) \tag{C.2}$$

s.t.

$$\pi_0 - \sum_{i \in V} y_i^k \psi'_i - \sum_{i \in V} \sum_{r \in R} u_{ir}^k \psi''_{ir} \geq 0, \qquad k \in J \tag{C.3}$$

$$\psi'_i + \phi'_i \geq 0, \qquad i \in V \tag{C.4}$$

$$\psi''_{ir} + \phi''_{ir} \geq \frac{1}{R}, \qquad i \in V, r \in R \tag{C.5}$$

$$\phi'_i \geq 0, \phi''_{ir} \geq 0, \qquad i \in V, r \in R. \tag{C.6}$$

Now, the non-linear terms $\phi'_i(1 - x_i)$ and $\phi''_{ir}(1 - x_i)$ in the dual formulation need to be linearized. We define new decision variables $C_i = \phi'_i(1 - x_i)$ and $D_{ir} = \phi''_{ir}(1 - x_i)$. In order to set the relationship between the new and old decision variables, bounds on variables $\phi'_i$ and $\phi''_{ir}$ are needed. As they are non-negative variables, lower bounds are zero. Upper bounds can be set from the primal (maximization) problem by examining the change in the objective value when the relevant RHS values change by one unit. An upper bound on this value is the difference between the maximum and minimum values that the objective function can take, which is simply $n$. The final and linear lower bound problem ($LB_q$) is obtained by integrating the dual problem in the outer minimization problem of the leader:

$LB_q$ :

$$t_q = \min\ \pi_0 + \sum_{i \in V} (C_i + \sum_{r \in R} D_{ir}) \tag{C.7}$$

s.t.

$$\sum_{i \in V} x_i = h \tag{C.8}$$

$$\pi_0 - \sum_{i \in V} y_i^k \psi'_i - \sum_{i \in V} \sum_{r \in R} u_{ir}^k \psi''_{ir} \geq 0, \qquad k \in J \tag{C.9}$$

$$\psi'_i + \phi'_i \geq 0, \qquad i \in V \tag{C.10}$$

$$\psi''_{ir} + \phi''_{ir} \geq \frac{1}{R}, \qquad i \in V, r \in R \tag{C.11}$$

$$C_i \geq \phi'_i - nx_i, \qquad i \in V \tag{C.12}$$

$$D_{ir} \geq \phi''_{ir} - nx_i, \qquad i \in V, r \in R \tag{C.13}$$

$$C_i, D_{ir}, \phi'_i, \phi''_{ir} \geq 0, x_i \in \{0, 1\}, \qquad i \in V, r \in R. \tag{C.14}$$

The UB problem formulation remains the same since it is simply the follower's problem given with (6.10)–(6.15).

# APPENDIX D: Instance Generation

**Algorithm D.1** `arXiv` instance generation

1: Set $degree \leftarrow 0$, $V \leftarrow \emptyset$, $A \leftarrow \emptyset$, choose an interval $[a, b]$
2: Sort all nodes in non-increasing order of their out-degrees and add the first $5n$ nodes to $V$
3: **while** $degree$ is not in $[a, b]$ **do**
4: Select a random node set $V' \subset V$, $|V'| = n$
5: Determine the arc set $A' \subset A$ such that $(i, j) \in A' \iff i \in V', j \in V'$ (without duplicates and self loops)
6: Generate the graph $G'(V', A')$
7: Set $degree \leftarrow$ the average out-degree of $G'$
8: **end while**

Figure D.1. Selection of a subgraph of the co-authorship network.