Locating Optimal Destabilization Strategies

Il-Chul Moon
PhD student
School of Computer Science
Carnegie Mellon University

Center for Computational Analysis of Social and Organizational Systems

http://www.casos.cs.cmu.edu/

Problem statement

- Network destabilization is an important tactic.
  - Counter terrorism [9,11], network centric warfare [10], computer network security [1].
- However, we don't have good answers for following answers
  - How to find an efficient network destabilization strategy (or scenario) ?
    - Minimum intervention, maximum destabilization
  - If we remove a node (possibly, agent, resource, knowledge),
    - Which node to target?
      - Node with many resources and knowledge vs. node at the center of an organization
    - When to remove the node?
  - How to assess the located strategy?
    - Big damage, but still able to recover
    - Or, small damage, but unable to recover
    - Or, big damage and unable to recover
- Some use multi-agent models. However, still there are problems
  - Analysts should build strategies and scenarios (possible human error, bias, etc)
  - Running complex multi-agent models often require vast amounts of time and storage capacity

Introduction

- We limit ourselves to
  - Destabilization of an organization represented in a network structure
  - Only agent removal strategic intervention
  - Only one agent removal for a single intervention
  - Limited number of interventions
- We develop a framework
  - With automatic (optimal) destabilization scenario by using machine learning technique
  - Using a multi-agent model, Dynet, as a test-bed for the developed scenarios
  - Assessing different impacts of interventions
- We expect to see
  - Automatically generated interesting destabilization
  - Better destabilization result compared to random destabilization tactics

Previous research

- Importance of network destabilization
  - Networks and Netwars [2]
    - Terrorist or criminal groups are leaderless, but still effective
- Theoretical background
  - Social network analysis [4]
    - Measures, tools, multi-agent simulations
- Previous practical projects
  - Netwatch [16]
    - Multi-agent simulation tool, provide destabilization scenario setup and estimated results
    - Too simple scenario setup capability
    - Simulation of two opposing groups. More useful to understand the nature of destabilization process. Not intended to generate and estimate an intervention scenario or strategy
  - NetAttacker
  - KeyPlayer
    - Network analysis tool.
    - Not a stochastic model. Static network analysis. No dynamic changes

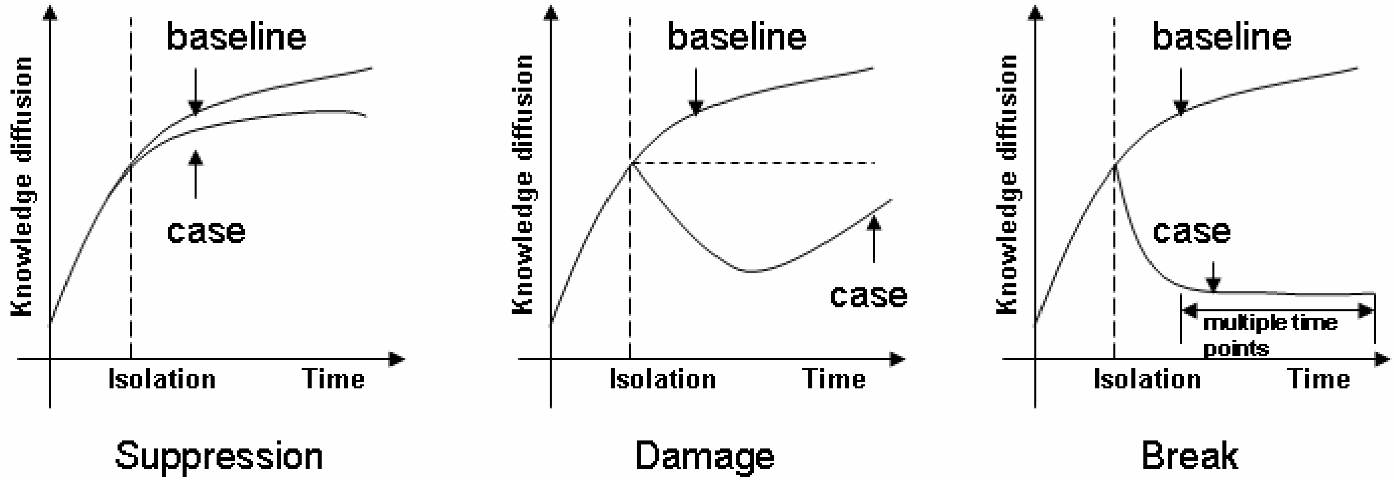
Method
- Near-Term Analysis and Dynet

- Dynet [7]
  - Multi-agent simulation
    - Agent interact based on probability of interaction which is determined by agent-to-agent network, relative similarity, relative experty, etc.
  - Functions to isolate nodes
  - Performance metrics
- Near-Term Analysis [12]
  - A wrapping function for Dynet
    - GUI front-end for Dynet and callable for ORA [13], dynamic network analysis tool
  - Provides a function to setup a sophisticated what-if scenarios
  - Easy control of parameters for Dynet

- We will use this combination for
  - A small training set to train a learning algorithm used for automatic scenarios generation
  - Tests showing the destabilization results

Method
- evaluation criteria for destabilization events

- We use a knowledge diffusion [5,12] output to see the performance changes
- Three classes of events
  - Suppression
    - Diffusion rate goes up, but not as much as baseline without intervention
  - Damage
    - Diffusion rate goes down, but can recover in the next time point
  - Break
    - Diffusion rate goes down, and the damage sustained for multiple time points



Method
- definition of isolation sequence

- In this context, a destabilization scenario is equivalent to an isolation sequence for agents
- ten isolations and one agent removal for each isolation
  - Test data has 16 agents
- The first isolation happens at time 2, and the next isolation happens after a gap of two time periods.
  - Start at time 2 and end at time 20

Method
- generation of an isolation sequence (1)

- We create a training set by brief searching in the possible sequence space
  - Record the result of intervention, metrics for node positions, metrics for network topology
- We train a machine learning algorithm, a variant of Support Vector Machine [14,15]
  - Result of intervention is a dependent variable
  - Metrics for nodes and networks are an independent variables
- We use the trained learning algorithm and create possible sequences
  - Get estimates for result by supplying the node and network metrics
  - Synthesize the sequence by choosing the agents with the highest damage estimates

Training set
(Random sequence)

- Randomly generated isolation sequences

- 1024 random cases and 10 isolations for each case

- Features for training include isolation timing, network level statistics and node level statistics from social network analysis.

→

Machine Learning

- Used a variant of Support Vector Machine

- Applied a non-linear kernel, RBF.

- Accept training instances with network measures and a boolean value for the success of suppression.

→

Test set
(Selected sequence)

- Test all the nodes with the same measures in the training set.

- Get the estimate value from the learning algorithm

- Select the top two nodes showing the high estimates for the isolation of the time.

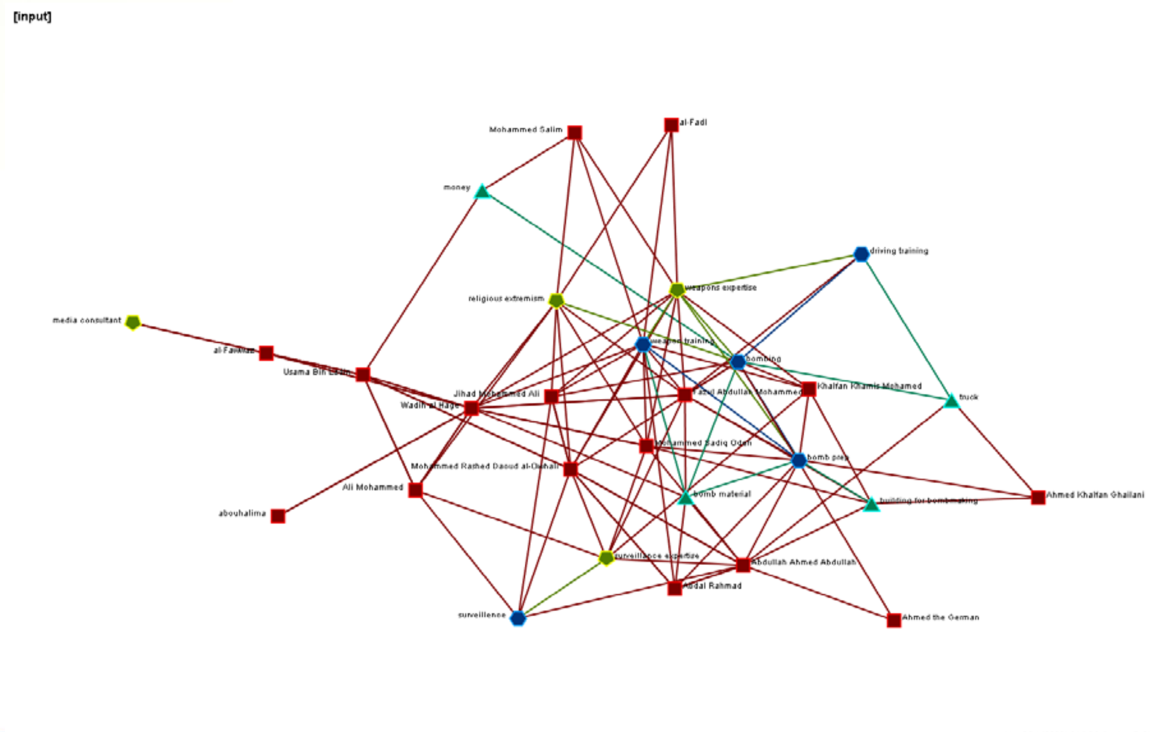
Method
- generation of an isolation sequence (2)

- Network and node metrics are based on the social network analysis of the input organizational structure
- Metrics are responsible for training the learning algorithm
- Metrics are calculated by ORA [13]

	Used measures
Network measure (27 measures)	knowledge task completion, knowledge under supply, overall task completion, performance as accuracy, average distance, average speed, betweenness centralization, closeness centralization, clustering coefficient, communicative need, connectedness, density, diameter, efficiency, fragmentation, global efficiency, hierarchy, in degree centralization, lateral edge count, minimum speed, network levels, out degree centralization, reciprocal edge count, sequential edge count, span of control, strong component count, weak component count
Node measure (11 measures)	cognitive demand, total degree centrality, clique count, row degree centrality, eigen vector centrality, betweenness centrality, high betweenness and low degree, task exclusivity, knowledge exclusivity, resource exclusivity, workload

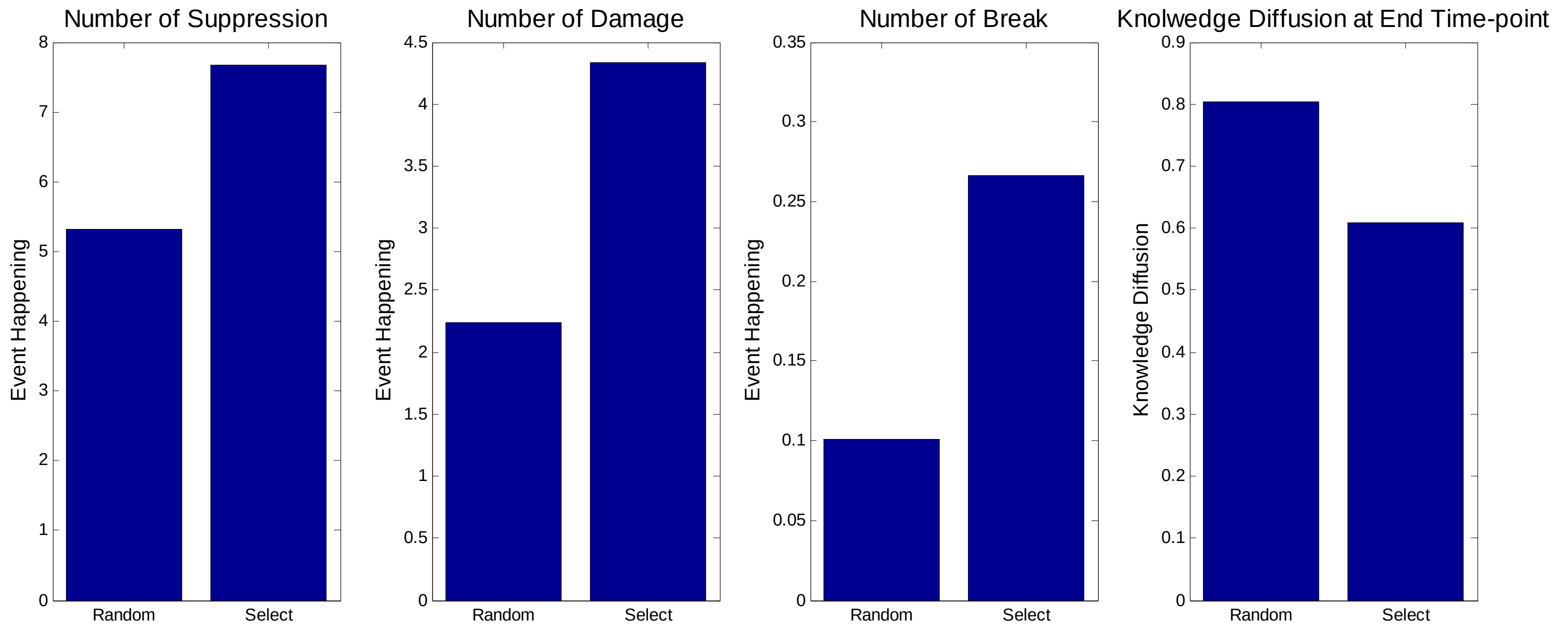
Test data

- Tanzania dataset [8]
  - 16 agents, 4 knowledge pieces, 4 resources, 5 tasks
  - Small dataset
  - Short computation time for learning algorithm
  - Too small, but good enough to test the proposed process



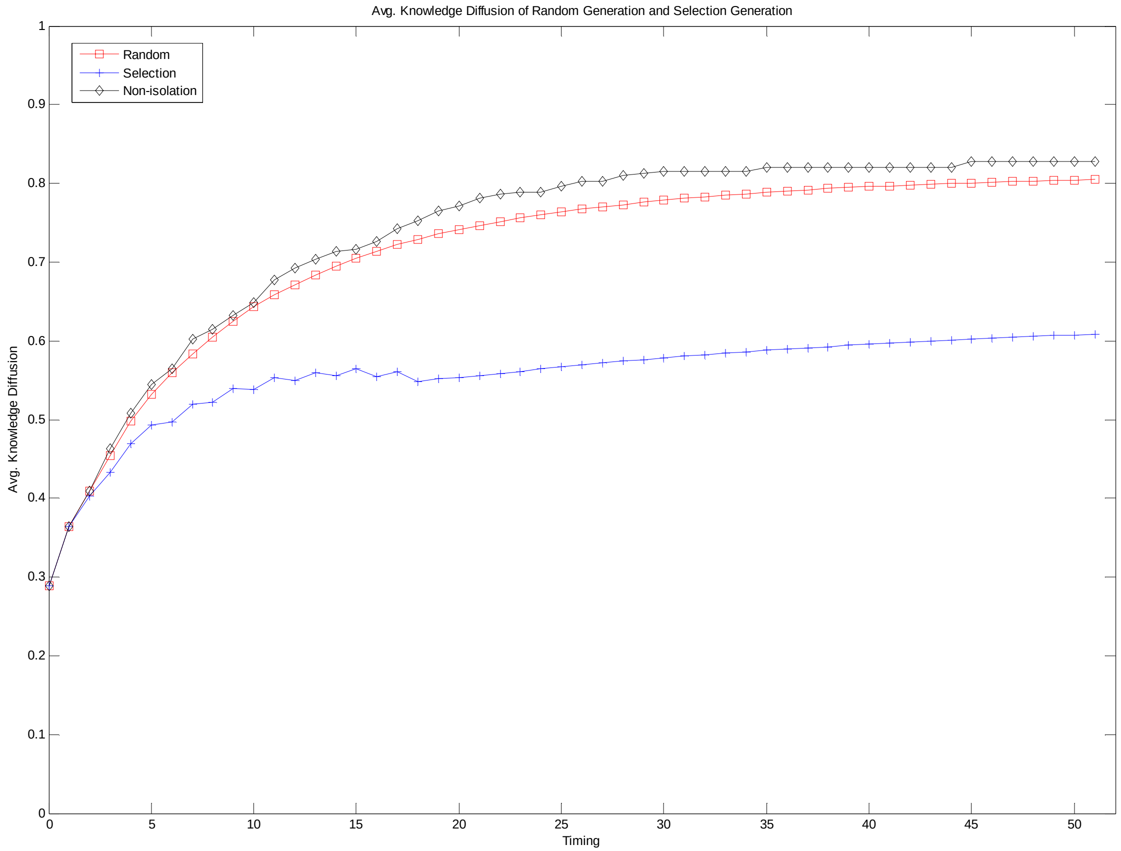
Result (1) - average destabilization performance

- Randomly generated isolation sequence vs. learning algorithm generated isolation sequence
- The learning algorithm generated sequences show more destabilization events and lower overall knowledge diffusion rates.
- High level comparison of two isolation sequence generation schemes



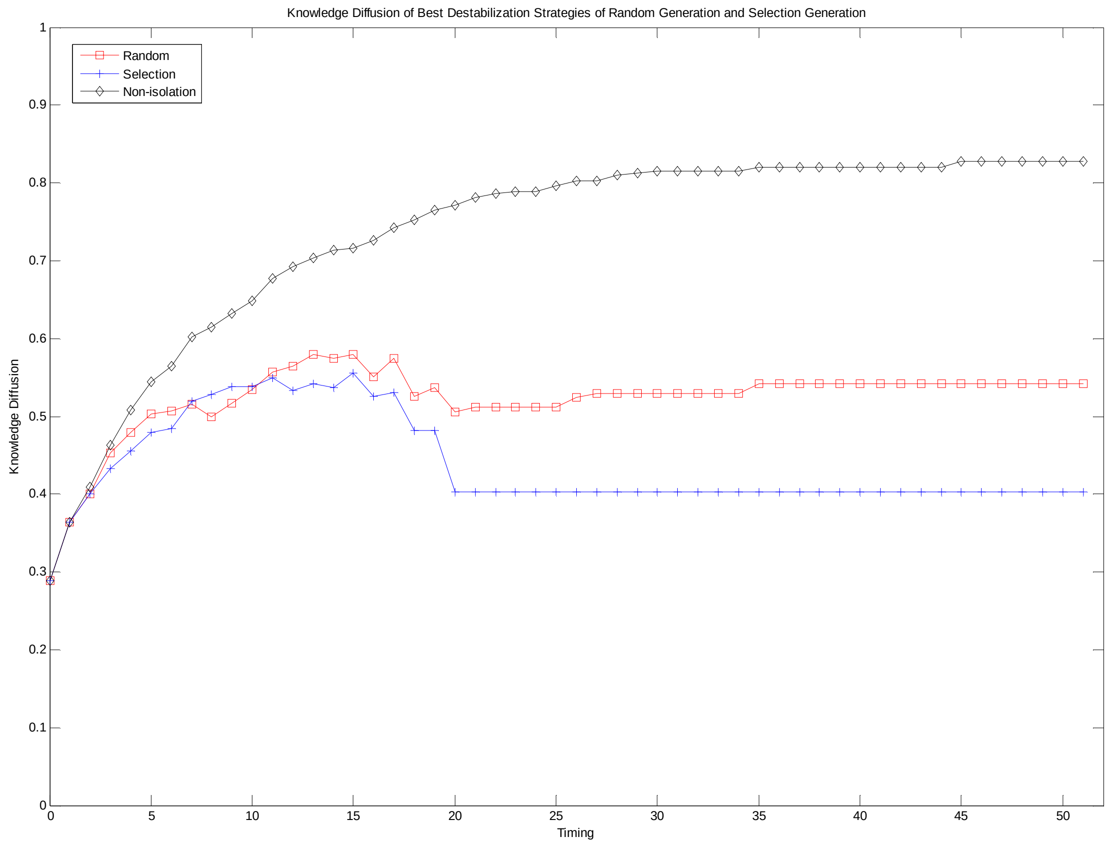
Result (2)
- average over time destabilization result

- Baseline, a case without intervention, shows highest knowledge diffusion rate.
- Random isolation sequence shows somewhat damaged diffusion rate.
- Learning algorithm shows very lower diffusion rate.



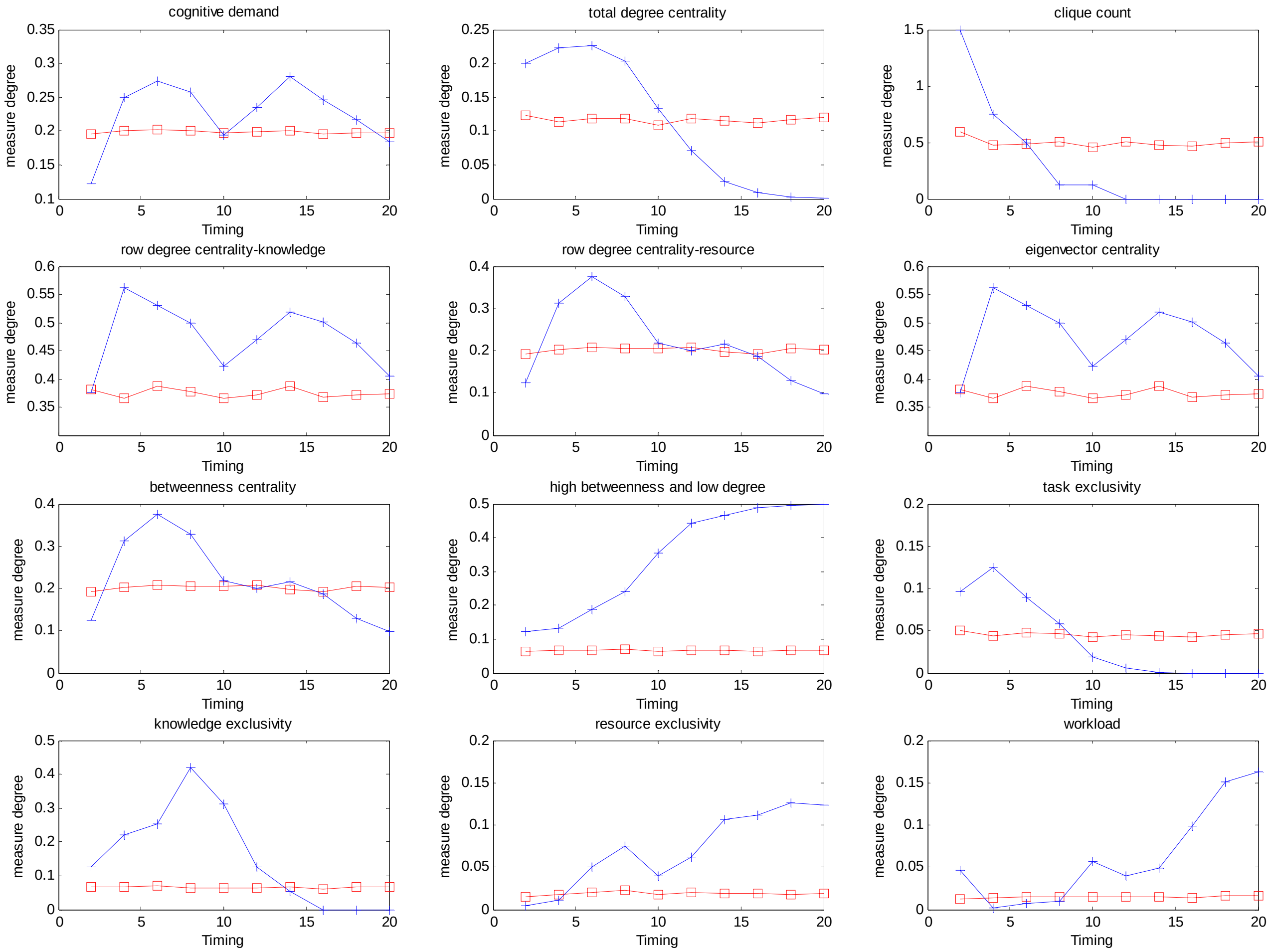
Result (3)
- best over time destabilization result

- Baseline, a case without intervention, shows highest knowledge diffusion rate.
  - Same to the previous slide
- Random isolation sequence shows pretty damaged diffusion rate, but the organization is still able to recover.
- Learning algorithm shows total break-down of the organization in terms of knowledge diffusion



Result (4) - who to target and when

- First waves of isolations
  - Target nodes with high-degree centrality, clique count, betweenness centrality, etc
- Next waves of isolations
  - Target nodes with high betweennes and low degree, meaning connecting nodes
- Isolations of agents with exclusive knowledge are not the first priority.
  - It happens after initial isolation of high degree centrality agents



Conclusion

- We demonstrated that
  - Machine learning based destabilization scenario creation
  - Destabilization scenario test result based on a multi-agent simulation
  - Better destabilization performance compared to random isolations
- We examined and found out that
  - Trained learning algorithm have a certain preference in choosing the target
    - Initial attacks, target nodes at the center of the network
    - Last attacks, target nodes at bridging points
    - Isolation of agents with exclusive knowledge may not be a priority, and they can be isolated after the nodes with high degree centrality.
  - This tendency implies that
    - Destabilize the network first
    - Isolate the exclusive knowledge or resource later

Limitation & Future work

- Too small dataset, need extensive tests
- Need to find out the performance changes when we limit the initial training set size.
- Need to test the robustness of this framework when the network is not fully uncovered.
- Need to test the scalability in terms of computation time

- Any improvements in three related areas will enhance the performance of this framework
  - Better social network metrics to represent the network structure accurately
  - Better multi-agent models with better usability, confidence, validation, etc.
  - Better machine learning technique

Reference

[1] R. Albert, H. Jeong and A.-L. Barabasi (2000), Error and attack tolerance of complex networks, Nature, Vol 406, pp 378-382

[2] J. Arquilla and D. Ronfeldt (editors) (2001), Networks and Netwars: The Future of Terror, Crime, and Militancy. Santa Monica, Calif.: RAND, MR-1382-OSD. www.rand.org/ publications/ MR/MR1382/

[3] J. Brockner (1992), The escalation of commitment to a failing course of action: Toward theoretical progress, Academy of Management Review, Vol 17, pp 39-61.

[4] K. M. Carley, J. S. Lee and D. Krackhardt (2001) Destabilizing Networks. Connections, Vol 24(3), pp 31-44.

[5] K. M. Carley and C. Schreiber (2002), Information Technology and Knowledge Distribution in C3I teams, Proceedings of the 2002 Command and Control Research and Technology Symposium, Naval Postgraduate School, Monterey, CA:

[6] K. M. Carley (2003), Dynamic Network Analysis, Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers, Eds. Ronald Breiger, Kathleen Carley, and Philippa Pattison, Committee on Human Factors, National Research Council, National Research Council, pp 133-145

[7] K. M. Carley (2004), Estimating Vulnerabilities in Large Covert Networks Using Multi-Level Data, In Proceedings of the 2004 International Symposium on Command and Control Research and Technology. Conference held in June, San Diego, CA., Evidence Based Research, Presented during Track 1, Electronic Publication, Vienna, VA.

[8] K. M. Carley and K. Y. Natalia (2004), A Network Optimization Approach for Improving Organizational Design, Carnegie Mellon University, School of Computer Science, Institute for Software Research International, Technical Report CMU-ISRI-04-102.

[9] K. M. Carley (2006), Destabilization of covert networks, Computational & Mathematical Organization Theory, Vol 12. Num 1., pp 51-66

[10] A. H. Dekker and B. D. Colbert (2004), Network robustness and graph topology, Proceedings of the 27th Australasian conference on Computer science, Vol 26, pp 359-368

[11] J. S. McIllwain (1999), Organized crime: A social network approach, Crime, Law and Social Change, Vol 32. Num 4., pp 301-323

[12] I. C. Moon and K. M. Carley (2006), Estimating the near-term changes of an organization with simulations, AAAI Fall Symposium, Arlington, VA

[13] J. Reminga and K. M. Carley (2004), ORA:Organization Risk Analyzer, Tech Report, CMU-ISRI-04-106, CASOS. Carnegie Mellon University. Pittsburgh PA, http://www.casos.cs.cmu.edu/projects/ora/index.html

[14] S. K. Shevade, S. S. Keerthi, C. Bhattacharyya and K. R. K. Murthy (1999), Improvements to SMO Algorithm for SVM Regression. Technical Report CD-99-16, Control Division Dept of Mechanical and Production Engineering, National University of Singapore.

[15] A. J. Smola and B. Scholkopf (1998), A Tutorial on Support Vector Regression, NeuroCOLT2 Technical Report Series - NC2-TR-1998-030.

[16] M. Tsvetovat (2005), Social structure simulation and inference using artificial intelligence techniques, Ph. D. Thesis, Carnegie Mellon University, CMU-ISRI-05-115

Acknowledgements

- This work was supported in part by the Office of Naval Research (ONR N0001140210973-NAVY, N000140610921 and N00014-06-0104), the National Science Foundation (SES-0452487), the Army Research Lab, and the AirForce Office of Sponsored Research (MURI: Cultural Modeling of the Adversary, 600322) and the Department of Defense for research in the area of dynamic network analysis. Additional support was provided by CASOS - the center for Computational Analysis of Social and Organizational Systems at Carnegie Mellon University. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research, the National Science Foundation, the Army Research Lab or the U.S. government.