GUARD: Constructing Realistic Two-Player Matrix and Security Games for Benchmarking Game-Theoretic Algorithms

Noah Krever - Columbia

Jakub Černý - Columbia

Moïse Blanchard - Georgia Tech

Christian Kroer - Columbia

What we did:

We proved randomized security games are **degenerate**, and introduced a framework to build **realistic** ones.

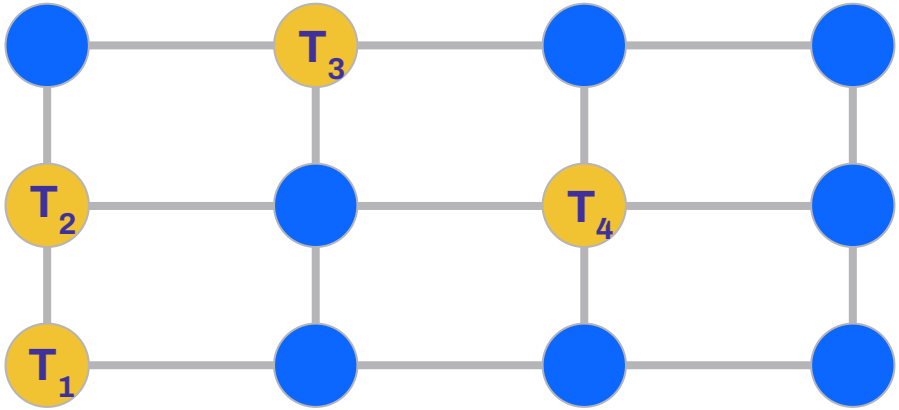
Security Games

Security games model a defender establishing protection over a set of targets that an attacker tries to capture.

We consider two different types of action spaces for these games: **schedules** and **paths.**

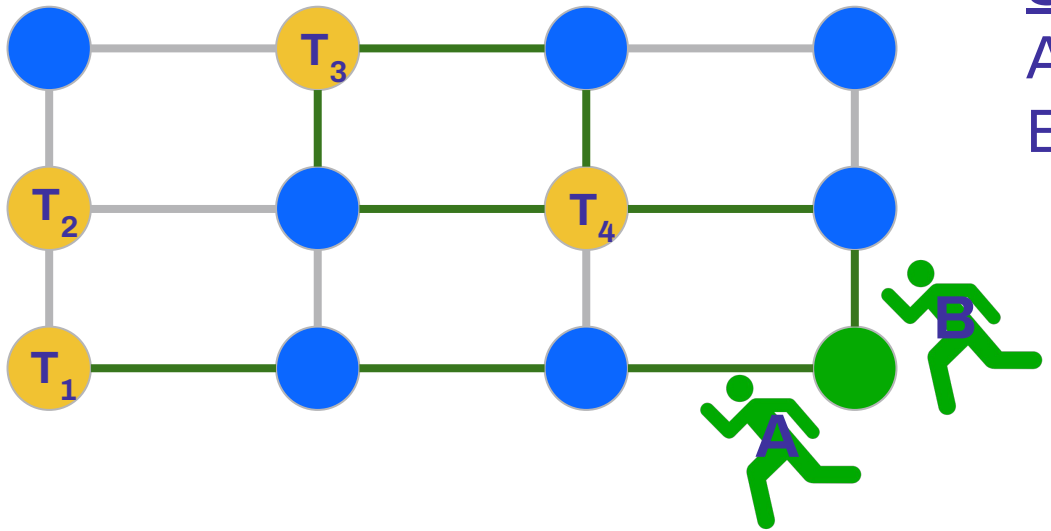
Schedule-Form Security Games

Consider a traversable environment with a set of **targets** T, each differing in value.



Schedule-Form Security Games

Defender resource(s) establish protection over **T**, assigning a **schedule** of simultaneously coverable targets to each resource. Subject to constraints.



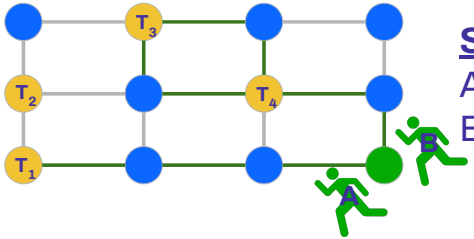
Schedules

A: $\{T_1\}$

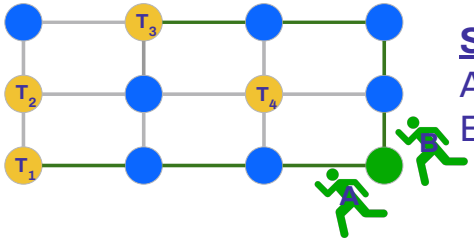
B: $\{T_3, T_4\}$

Schedule-Form Security Games

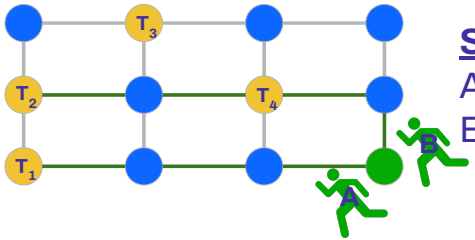
Over time, the defender strategy becomes a distribution of possible coverages.



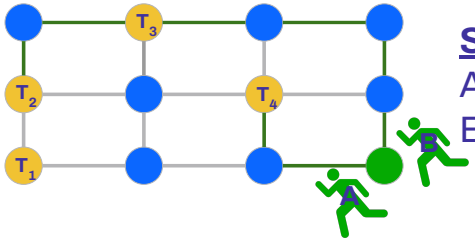
Schedules
A: $\{T_1\}$
B: $\{T_3, T_4\}$



Schedules
A: $\{T_1\}$
B: $\{T_3\}$

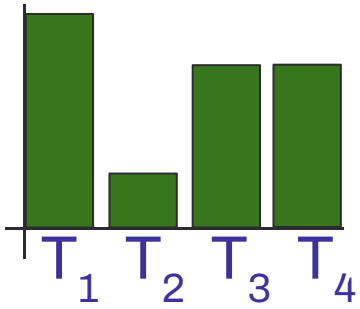


Schedules
A: $\{T_1\}$
B: $\{T_2, T_4\}$



Schedules
A: $\{T_1, T_4\}$
B: $\{T_3\}$

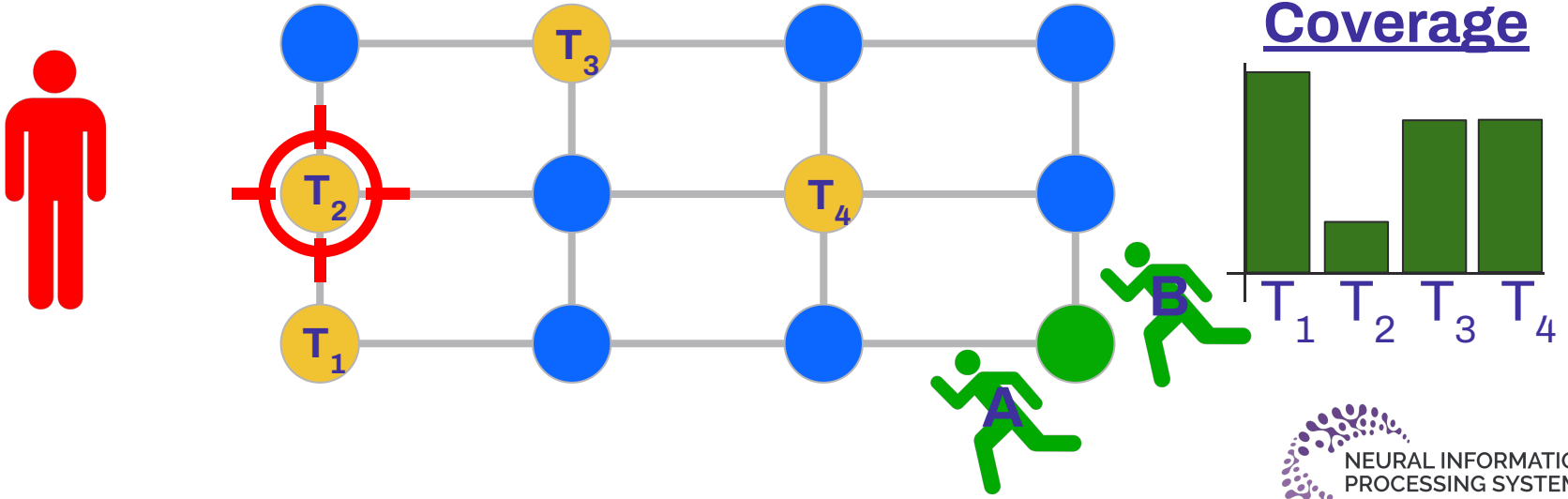
Coverage



NEURAL INFORMATION PROCESSING SYSTEMS

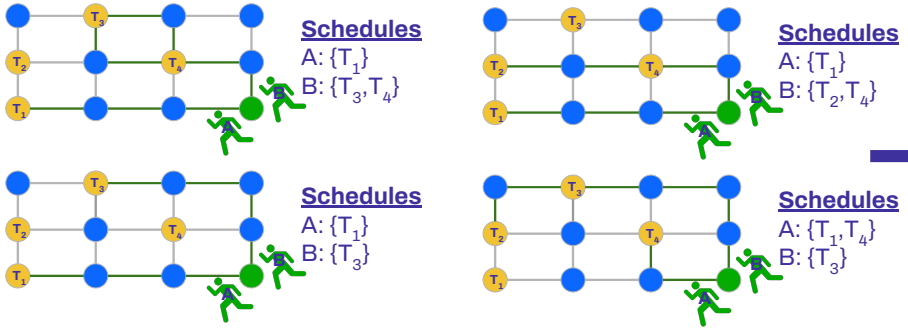
Schedule-Form Security Games

Attacker observes the protection strategy, and chooses a target to capture. They receive a utility based on if the target is covered or uncovered.



Schedule Form → Matrix Form

Schedule-form security games can be converted to matrix games by expanding schedule coverages into a defender action set, and setting possible attacked targets as the attacker's actions.



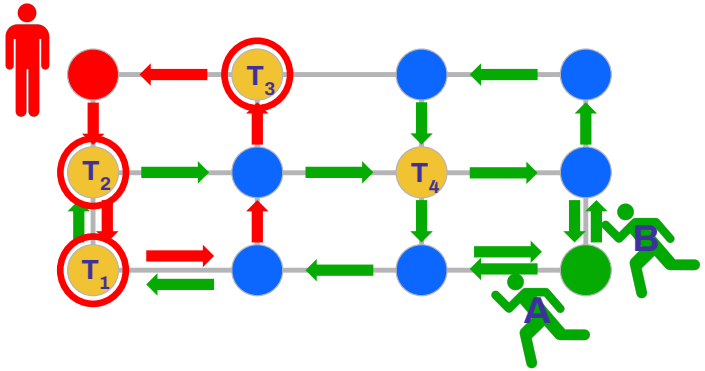
Attacker

Defender

	Coverage$_1$	Coverage$_2$	Coverage$_3$	. . .
T_1	Utility$_{1,1}$	Utility$_{1,2}$	Utility$_{1,3}$	
T_2	Utility$_{2,1}$	Utility$_{2,2}$	Utility$_{2,3}$	
T_3	Utility$_{3,1}$	Utility$_{3,2}$	Utility$_{3,3}$	
T_4	Utility$_{4,1}$	Utility$_{4,2}$	Utility$_{4,3}$	

Path-Space Matrix Form Games

Security games can also be represented as **path-space matrix games**. Instead of schedules of covered targets and a chosen attack, actions are different **paths** that visit nodes.



Defender

Attacker	$Path(s)_D1$	$Path(s)_D2$	$Path(s)_D3$	. . .
$Path(s)_A1$	$Utility_{1,1}$	$Utility_{1,2}$	$Utility_{1,3}$	
$Path(s)_A2$	$Utility_{2,1}$	$Utility_{2,2}$	$Utility_{2,3}$	
$Path(s)_A3$	$Utility_{3,1}$	$Utility_{3,2}$	$Utility_{3,3}$	
.				
.				
.				

Game is formulated as a matrix of payoffs which can be solved with game-theoretic algorithms.

Security Games

Model interactions like **anti-poaching**, and **urban counter-terrorism**.

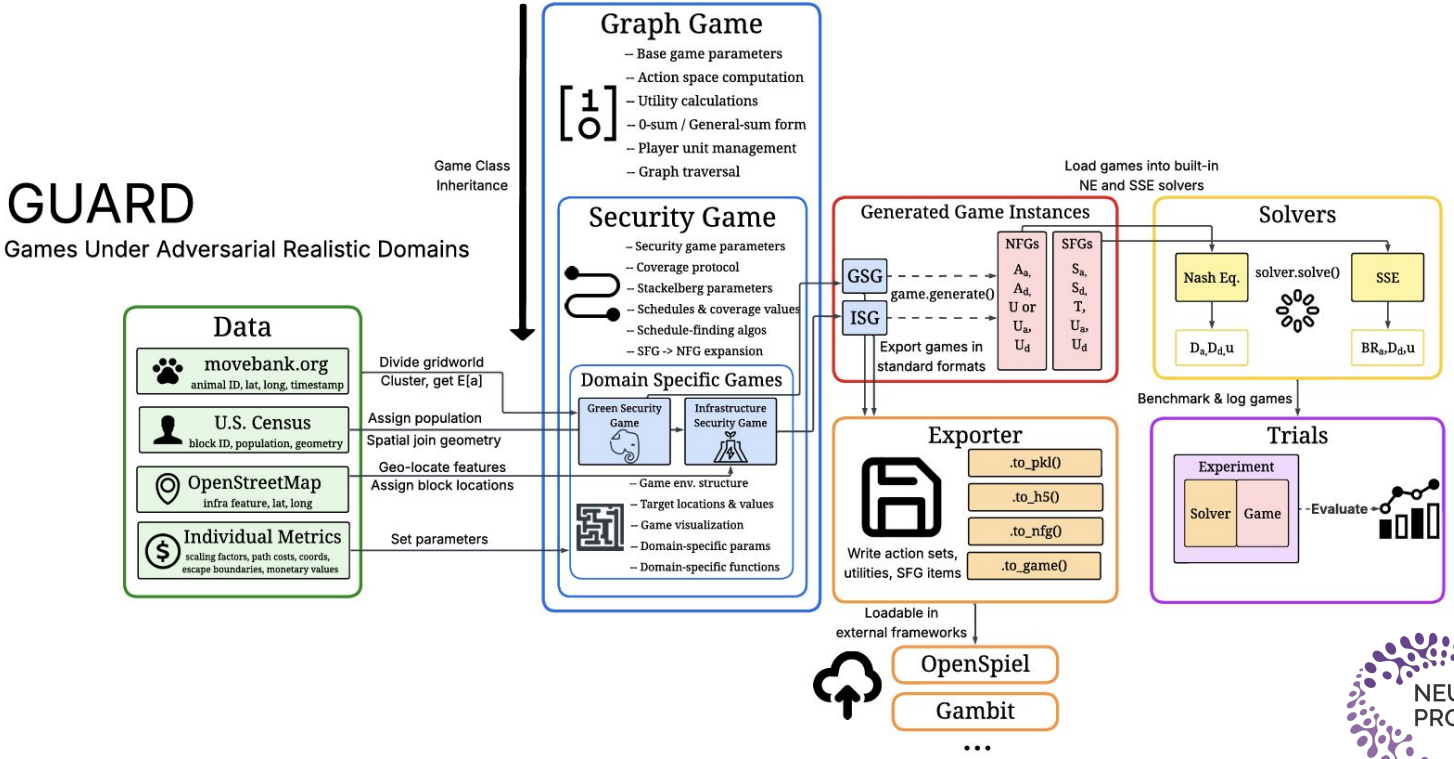
In AGT, algorithms are often benchmarked on randomized instances. **Target locations** and **payoffs** are random.

Limitations of Random Games

Thms 1 & 2: Randomized general sum games result in **degeneracy**: sparse strategies and trivial utility capture.

GUARD

GAMES UNDER ADVERSARIAL REALISTIC DOMAINS

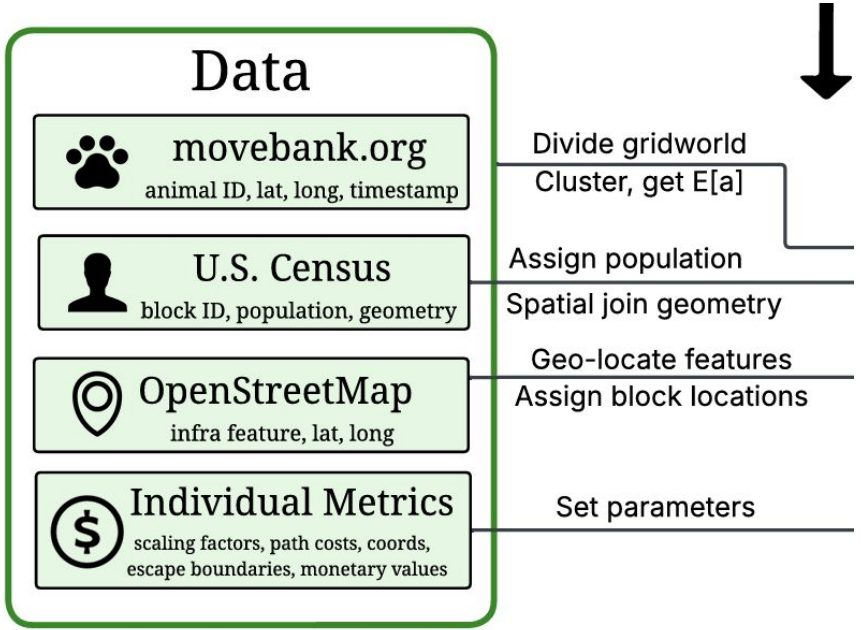


NEURAL INFORMATION PROCESSING SYSTEMS

How does it work?

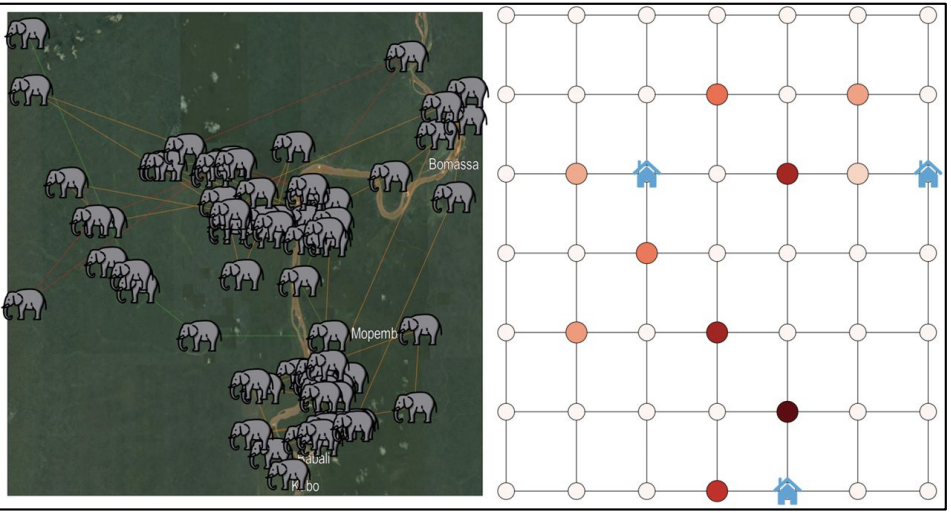
Use **real data** to build realistic game instances: GSGs and ISGs.

Target locations and values are informed by smart heuristics: animal density, population, etc.

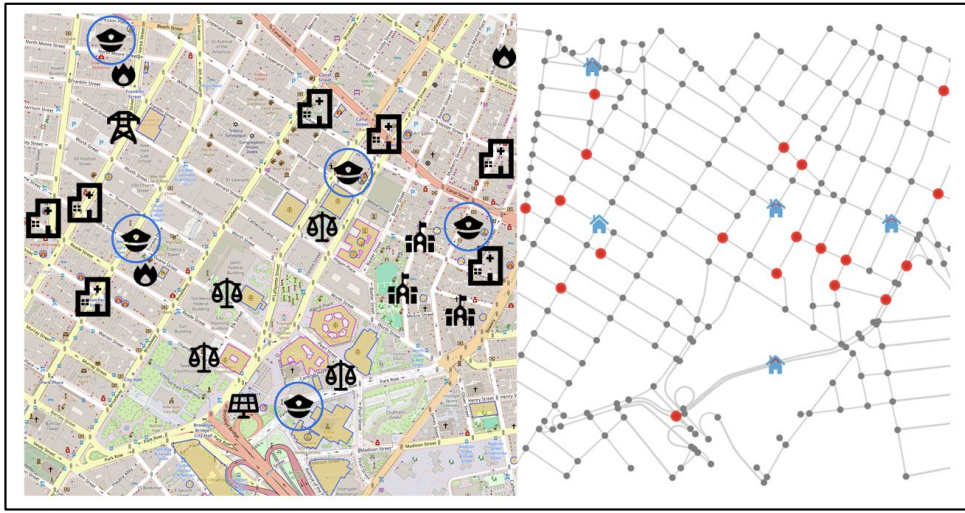


NEURAL INFORMATION PROCESSING SYSTEMS

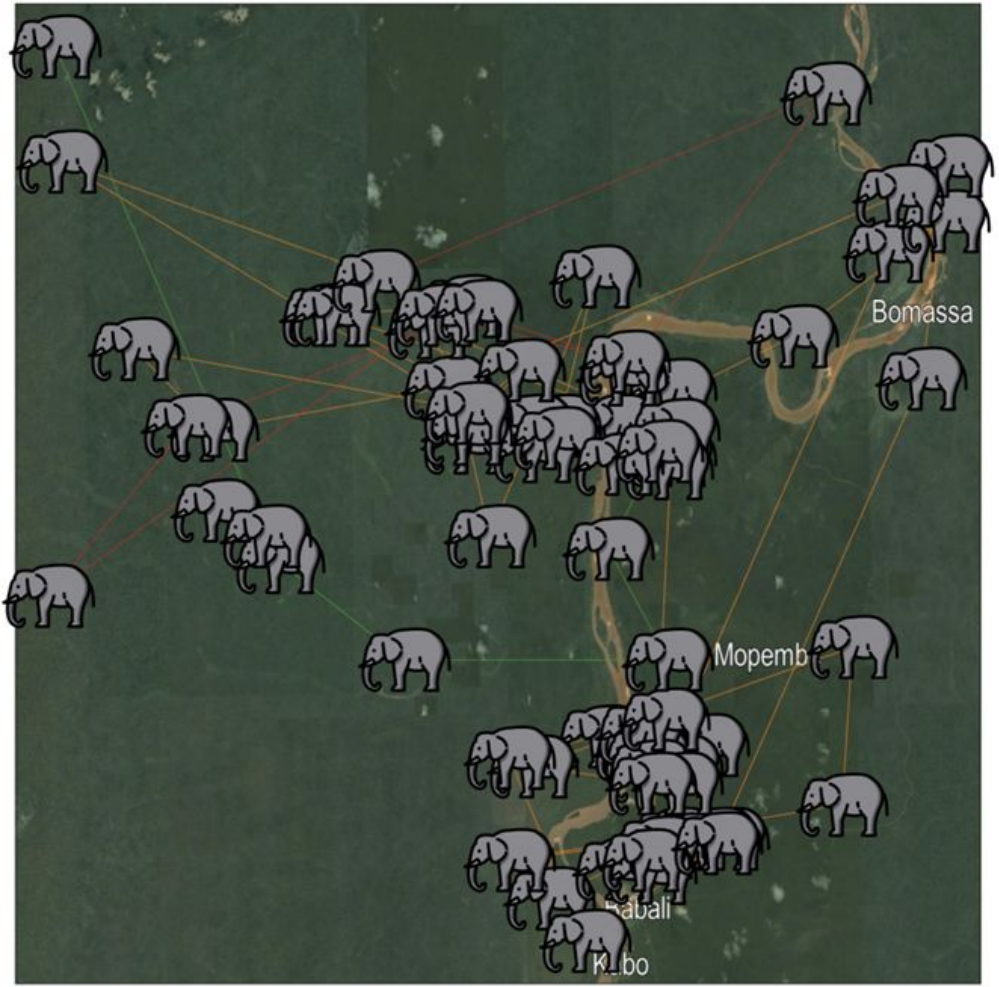
Examples

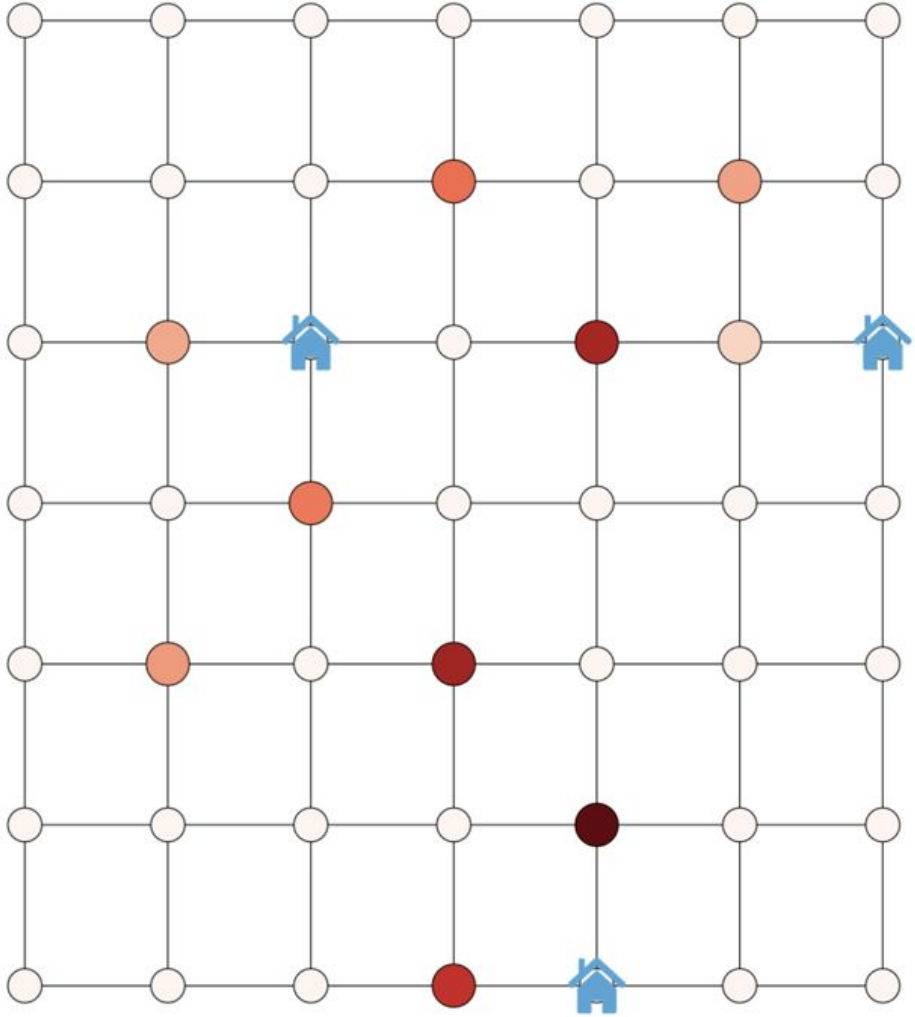


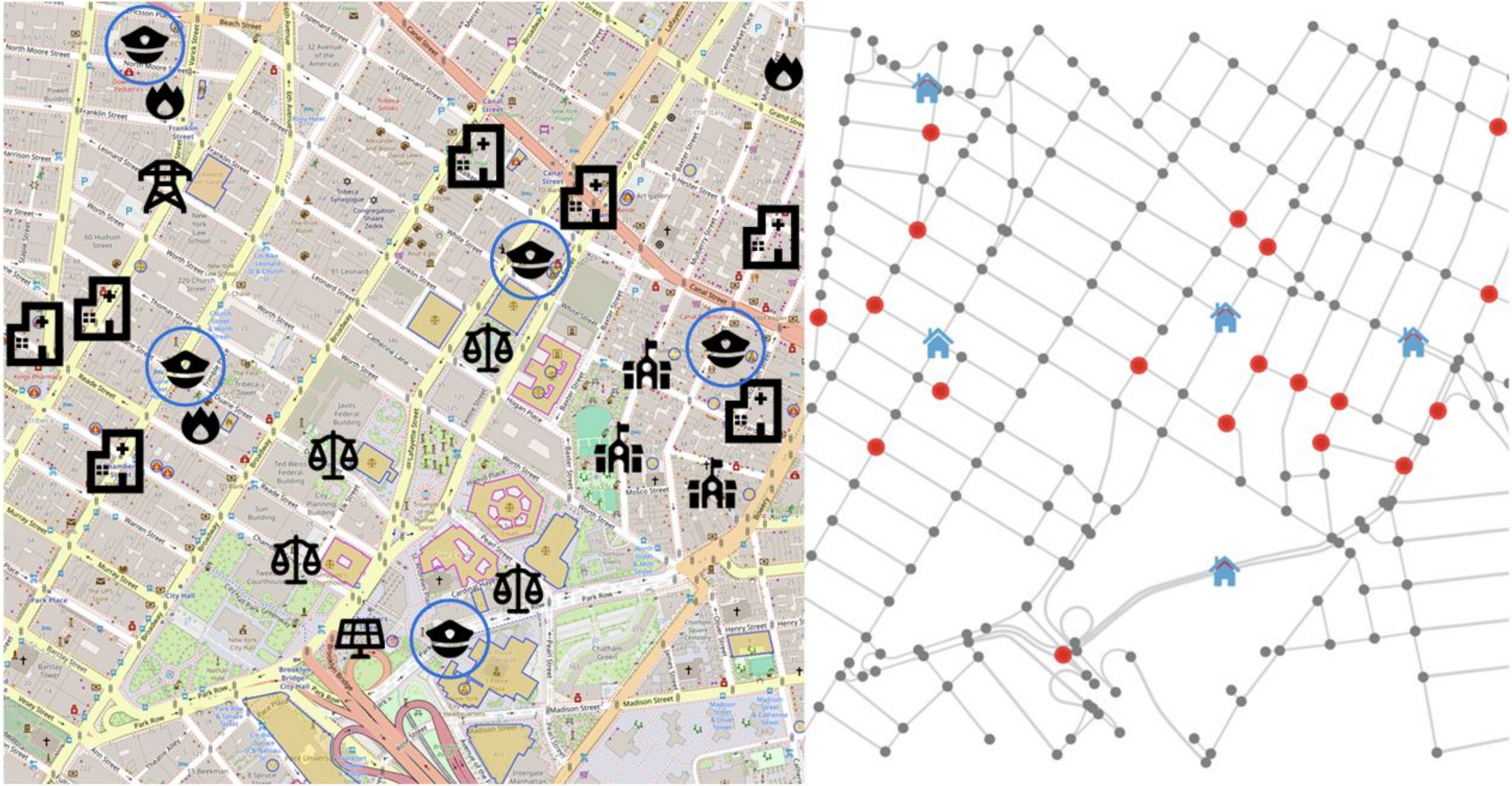
Left: Down-sampled elephant movements in Lobéké National Park. **Right**: Corresponding GSG model.



Left: Civil infrastructure in Manhattan's Chinatown. **Right**: Corresponding ISG model.


Bomassa
Mopemb
Kabali
Kubo



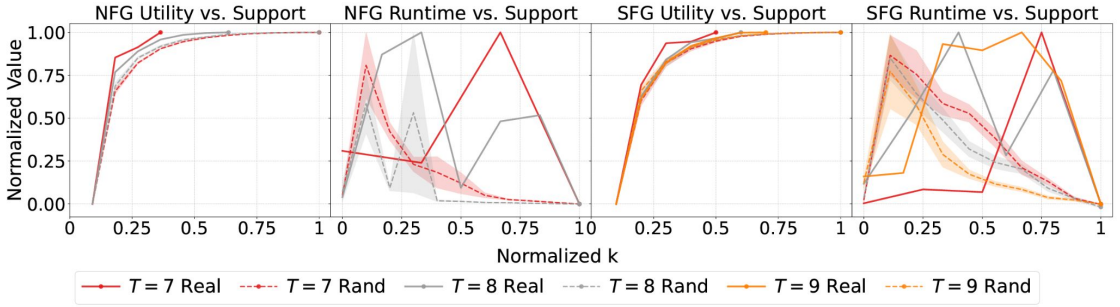


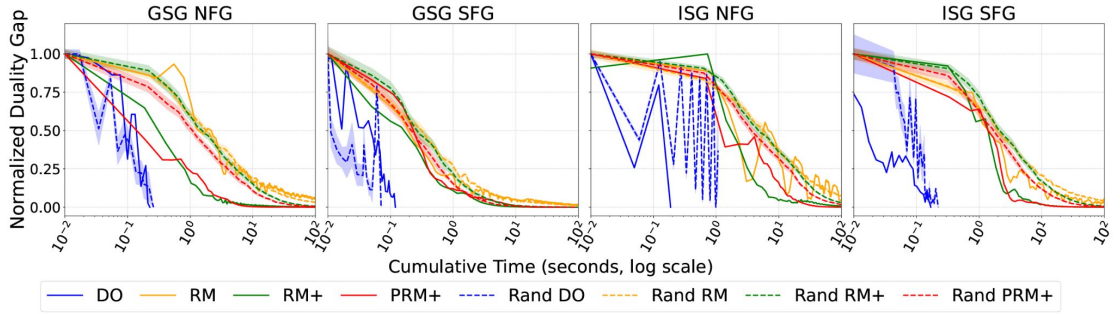
Beach Street
North Moore Street
Franklin Street
Leonard Street
Worth Street
Thomas Street
Reade Street
Warren Street
Murray Street
Park Place
Canal Street
Lispenard Street
White Street
Walker Street
Broadway
Lafayette Street
Centre Street
Baxter Street
Mulberry Street
Mosco Street
Park Row
Pearl Street
Madison Street
Henry Street
Grand Street
Hester Street
Catherine Lane
Hogan Place
City Hall

Experiments

3 experiments across random and GUARD game instances, matrix games and security games, GSG and ISG settings.

- Sparsity and Runtime
- Iterative Convergence
- General Sum SSE



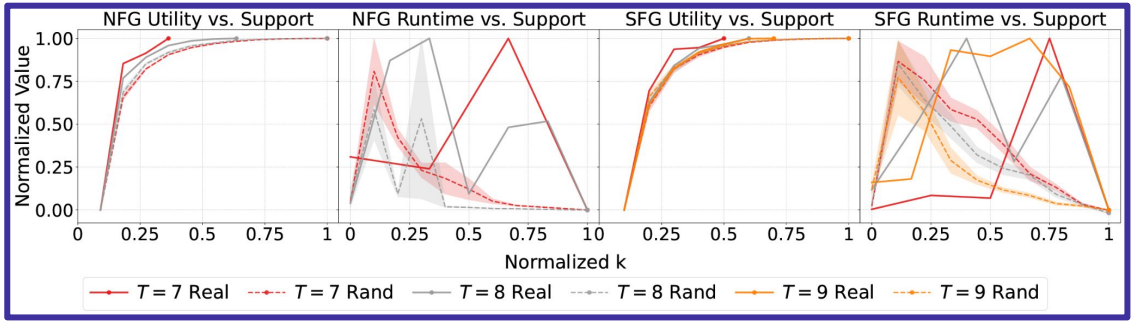


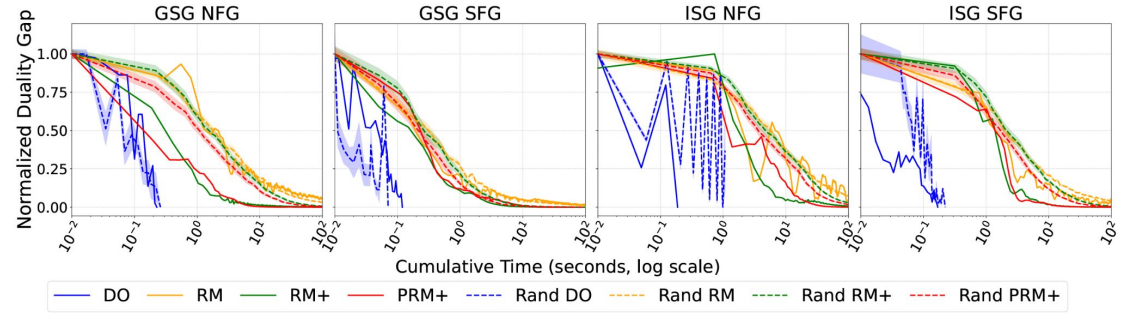
Form	Trial	GSG			ISG		
		u_d	R	S	u_d	R	S
Simple	Real	-0.390	0.015s	6	-0.429	0.030s	11
	RT	-0.298 ± 0.03	0.01s ± 0	5.0 ± 0.52	-0.214 ± 0.05	0.04s ± 0	7.5 ± 1.20
General	Real	-0.207	0.46s	9	-0.408	838s	14
	RM	-0.035 ± 0	0.52s ± 0.02	2 ± 0.15	-0.026 ± 0	541.5s ± 1.98	1.7 ± 0.15
	RT	-0.253 ± 0	0.35s ± 0.01	1.7 ± 0.26	-0.013 ± 0	116.2s ± 1.87	1 ± 0
	RTS	-0.270 ± 0	0.45s ± 0.03	1.4 ± 0.16	-0.032 ± 0	1.57± 0.04	1 ± 0

Experiments

Sparsity and Runtime

- Randomized games produced artificially dense equilibrium strategies.
- GUARD games had simpler, but non-trivial strategies.
- Solving random games takes longer.



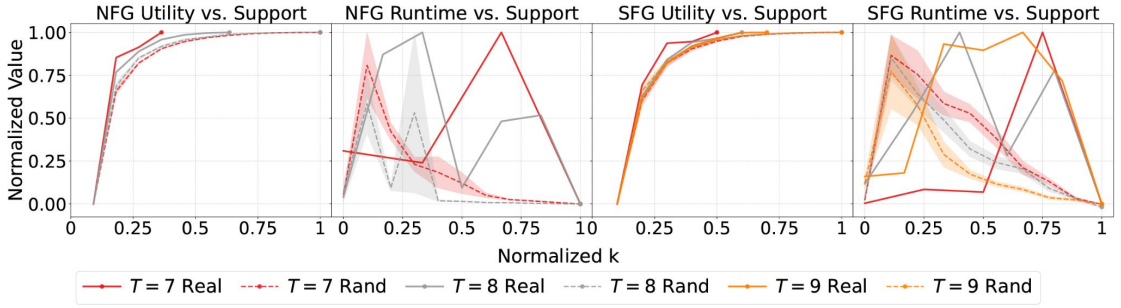


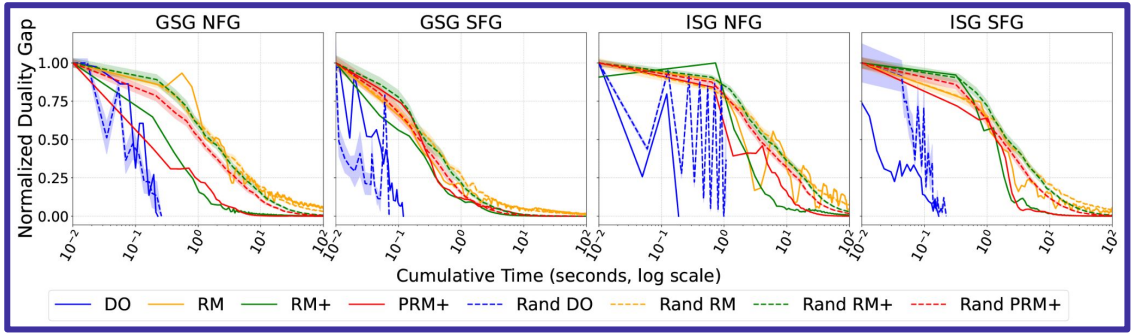
Form	Trial	GSG			ISG		
		u_d	R	S	u_d	R	S
Simple	Real	-0.390	0.015s	6	-0.429	0.030s	11
	RT	-0.298 ± 0.03	0.01s ± 0	5.0 ± 0.52	-0.214 ± 0.05	0.04s ± 0	7.5 ± 1.20
General	Real	-0.207	0.46s	9	-0.408	838s	14
	RM	-0.035 ± 0	0.52s ± 0.02	2 ± 0.15	-0.026 ± 0	541.5s ± 1.98	1.7 ± 0.15
	RT	-0.253 ± 0	0.35s ± 0.01	1.7 ± 0.26	-0.013 ± 0	116.2s ± 1.87	1 ± 0
	RTS	-0.270 ± 0	0.45s ± 0.03	1.4 ± 0.16	-0.032 ± 0	1.57± 0.04	1 ± 0

Experiments

Iterative Convergence

- Random games took longer to converge.
- Random games converged more erratically.
- GUARD games exhibited more reasonable convergence profiles.



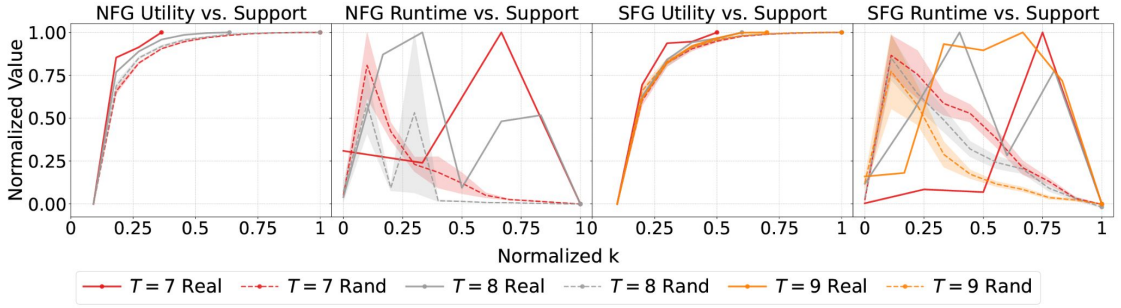


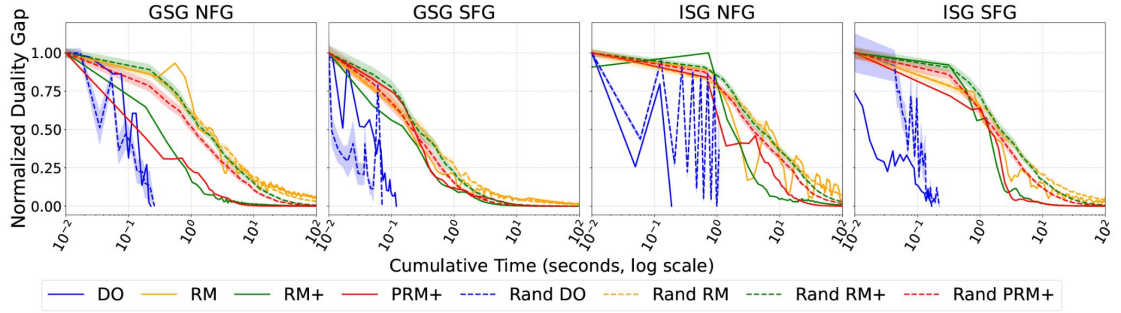
Form	Trial	GSG			ISG		
		u_d	R	S	u_d	R	S
Simple	Real	-0.390	0.015s	6	-0.429	0.030s	11
	RT	-0.298 ± 0.03	0.01s ± 0	5.0 ± 0.52	-0.214 ± 0.05	0.04s ± 0	7.5 ± 1.20
General	Real	-0.207	0.46s	9	-0.408	838s	14
	RM	-0.035 ± 0	0.52s ± 0.02	2 ± 0.15	-0.026 ± 0	541.5s ± 1.98	1.7 ± 0.15
	RT	-0.253 ± 0	0.35s ± 0.01	1.7 ± 0.26	-0.013 ± 0	116.2s ± 1.87	1 ± 0
	RTS	-0.270 ± 0	0.45s ± 0.03	1.4 ± 0.16	-0.032 ± 0	1.57± 0.04	1 ± 0

Experiments

General Sum SSE

- Random games were completely degenerate in sparsity and utility capture.
- GUARD games had richer strategies and realistic utility capture.
- Directly illustrates our theorems' tenets.





Form	Trial	GSG			ISG		
		u_d	R	S	u_d	R	S
Simple	Real	-0.390	0.015s	6	-0.429	0.030s	11
	RT	-0.298 ± 0.03	0.01s ± 0	5.0 ± 0.52	-0.214 ± 0.05	0.04s ± 0	7.5 ± 1.20
General	Real	-0.207	0.46s	9	-0.408	838s	14
	RM	-0.035 ± 0	0.52s ± 0.02	2 ± 0.15	-0.026 ± 0	541.5s ± 1.98	1.7 ± 0.15
	RT	-0.253 ± 0	0.35s ± 0.01	1.7 ± 0.26	-0.013 ± 0	116.2s ± 1.87	1 ± 0
	RTS	-0.270 ± 0	0.45s ± 0.03	1.4 ± 0.16	-0.032 ± 0	1.57± 0.04	1 ± 0

Reproducibility

GUARD can be used to generate new realistic games, but also comes with a predefined suite of instances.

It also integrates with existing game theory frameworks, and is completely open source, with continual improvements in progress.

github.com/CoffeeAndConvexity/GUARD