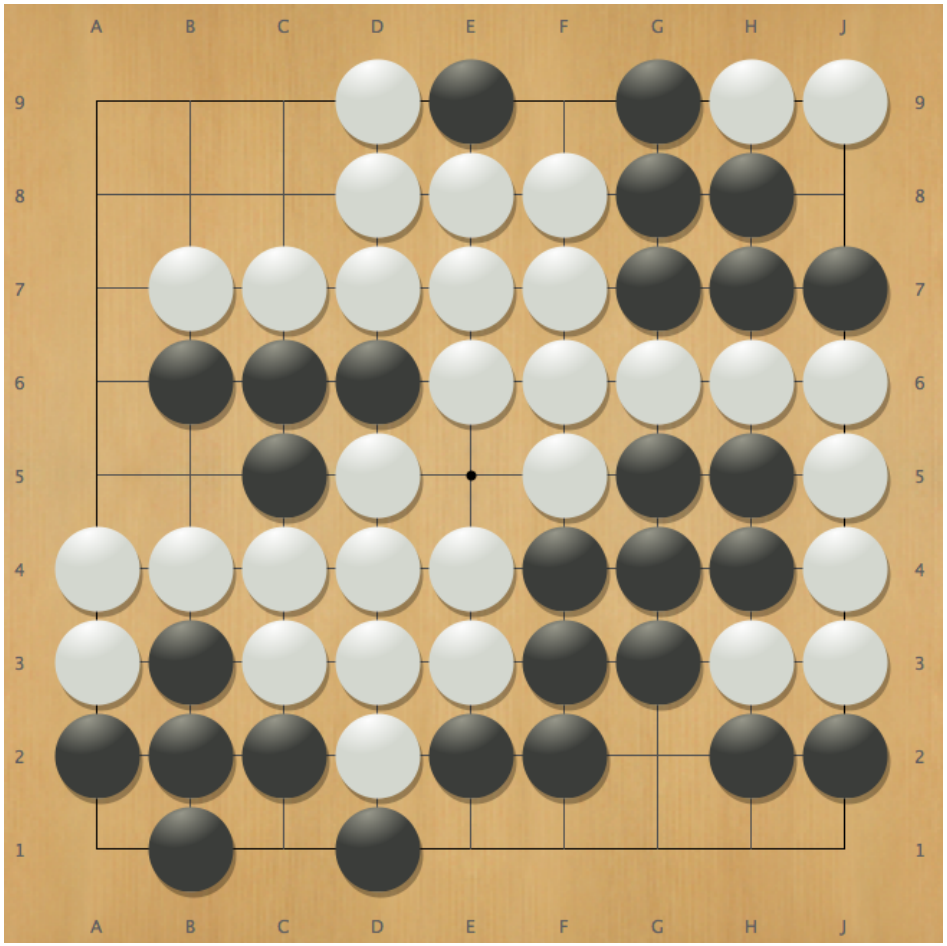
Modelling Go Positions with Planar CRFs

Dmitry Kamenetsky, Nicol N. Schraudolph,
Simon Günter, S.V. N. Vishwanathan

NICTA, Australian National University, Australia

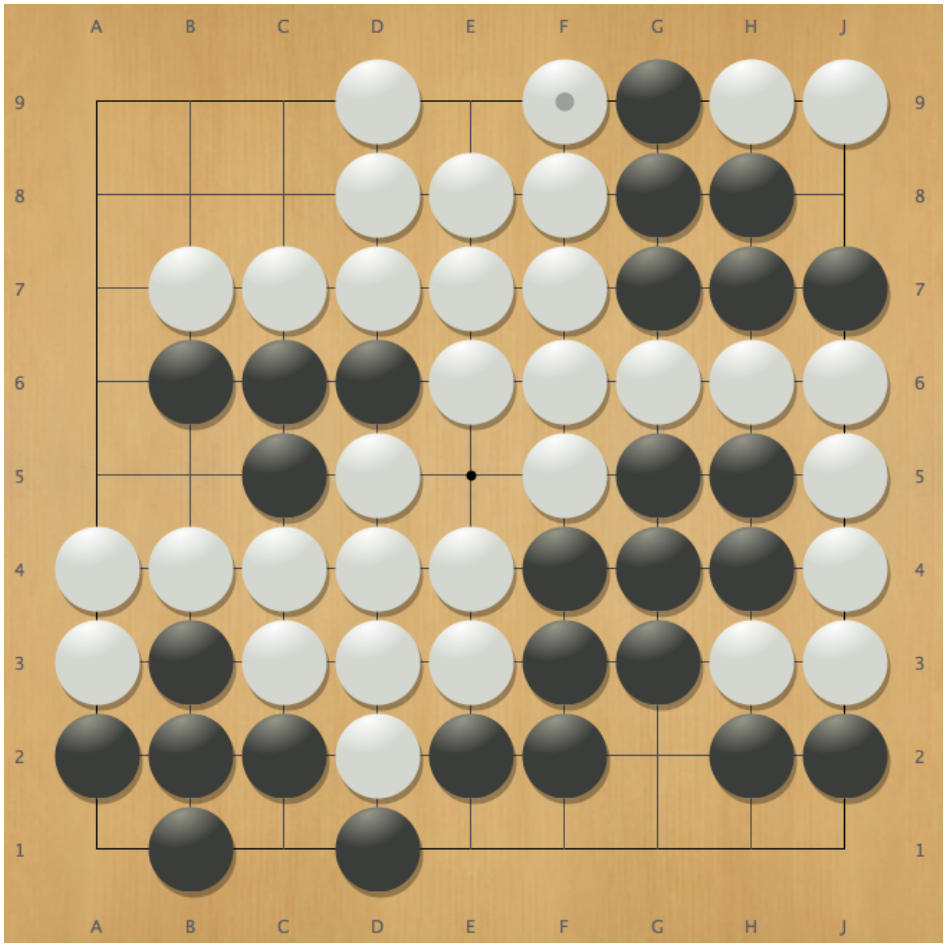
Machine Learning and Games Workshop, December 2007

What is Go?



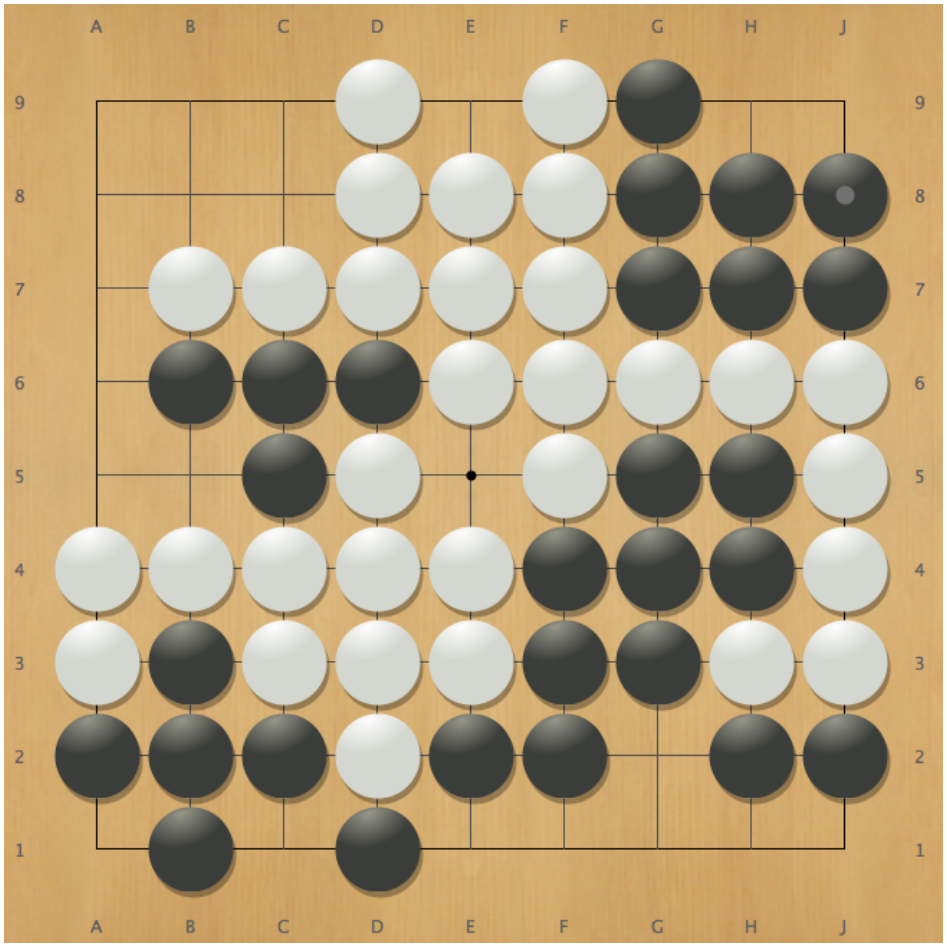
- Two players alternate in placing stones on the intersections of a grid
- Neighbouring stones of the same colour form a contiguous *block*
- A block can be *captured* if all its empty neighbours are occupied by opponent stones

What is Go?



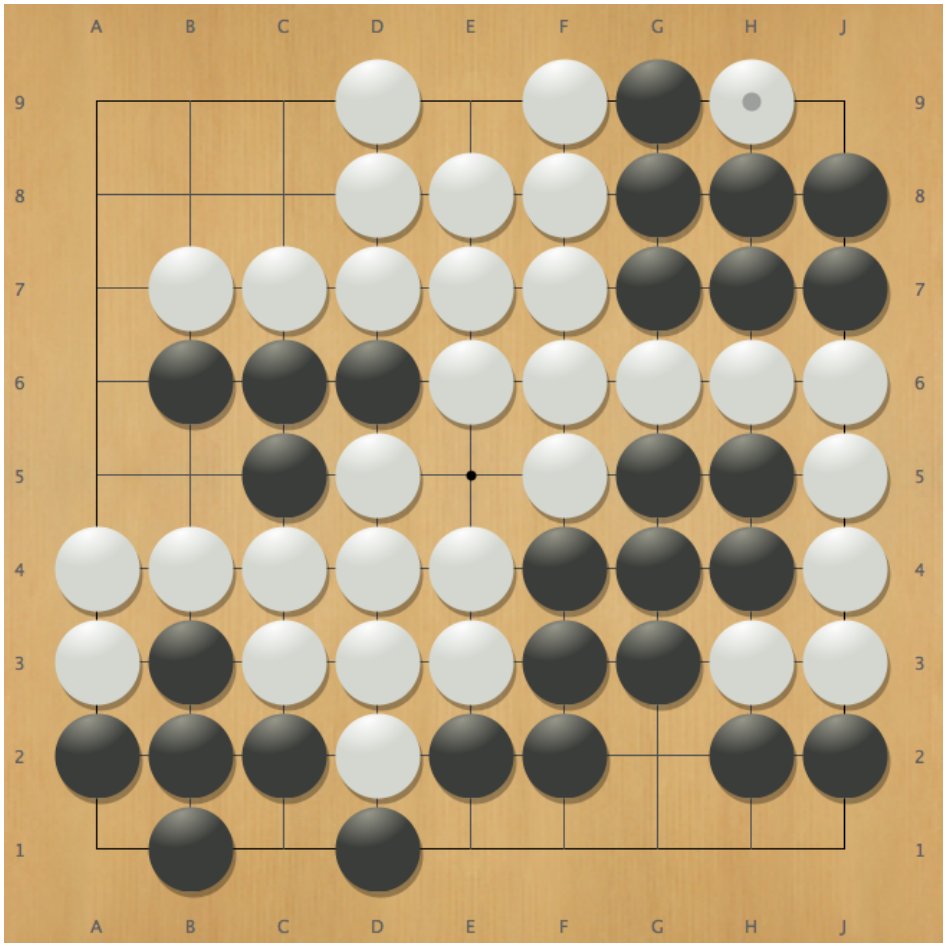
- Two players alternate in placing stones on the intersections of a grid
- Neighbouring stones of the same colour form a contiguous *block*
- A block can be *captured* if all its empty neighbours are occupied by opponent stones

What is Go?



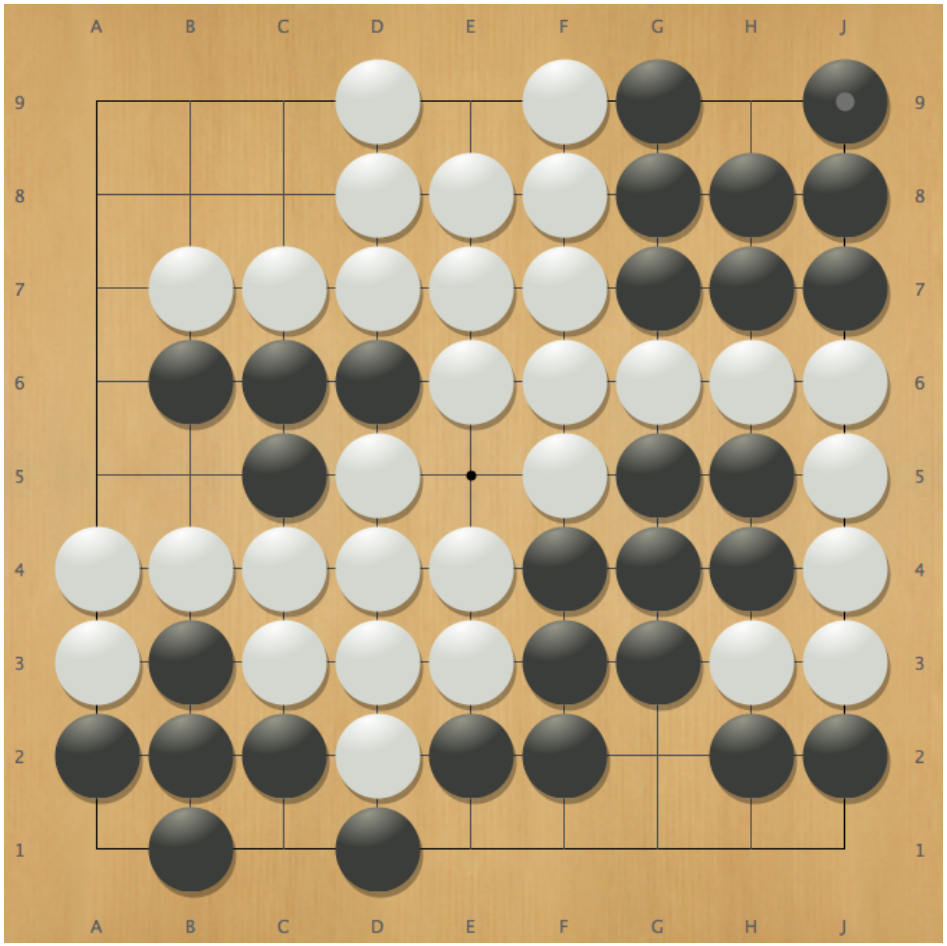
- Two players alternate in placing stones on the intersections of a grid
- Neighbouring stones of the same colour form a contiguous *block*
- A block can be *captured* if all its empty neighbours are occupied by opponent stones

What is Go?



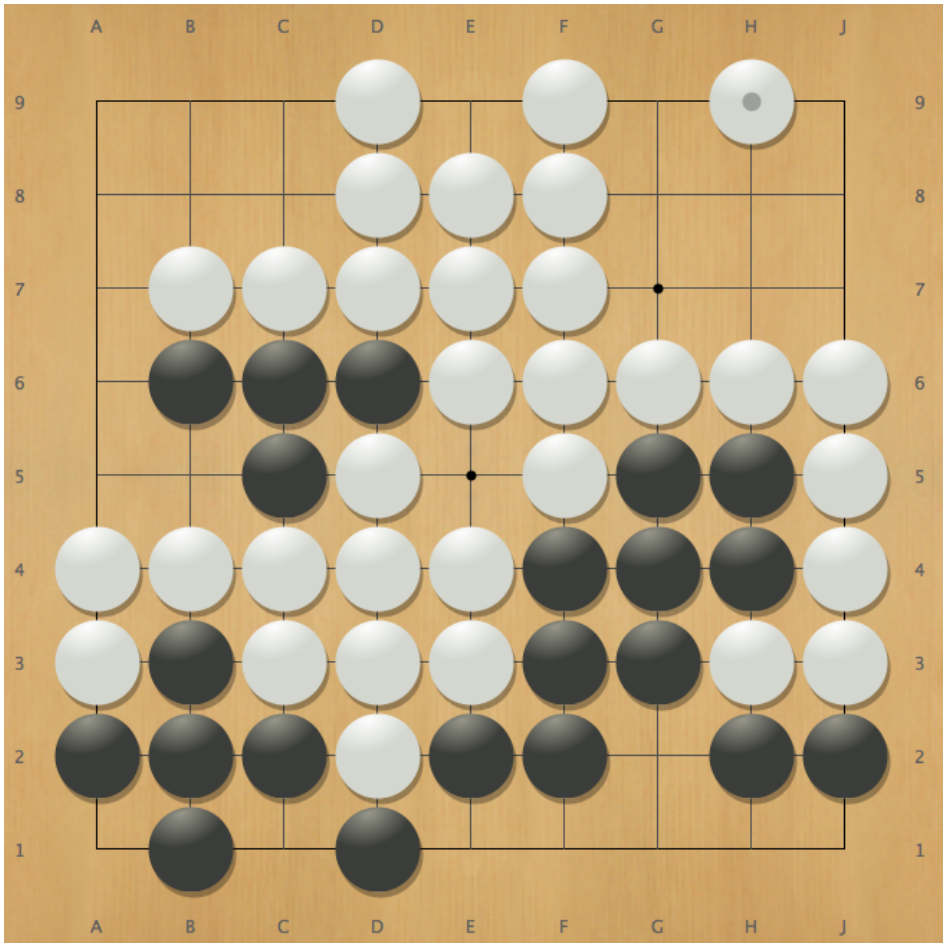
- Two players alternate in placing stones on the intersections of a grid
- Neighbouring stones of the same colour form a contiguous *block*
- A block can be *captured* if all its empty neighbours are occupied by opponent stones

What is Go?



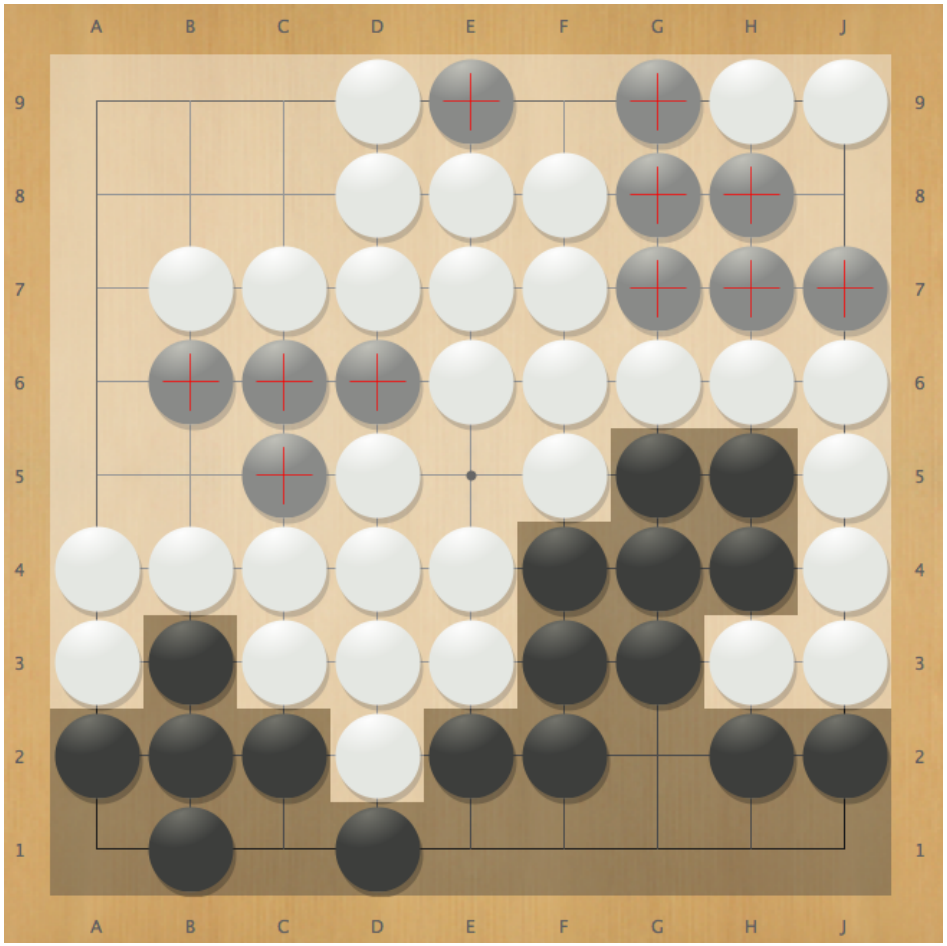
- Two players alternate in placing stones on the intersections of a grid
- Neighbouring stones of the same colour form a contiguous *block*
- A block can be *captured* if all its empty neighbours are occupied by opponent stones

What is Go?



- Two players alternate in placing stones on the intersections of a grid
- Neighbouring stones of the same colour form a contiguous *block*
- A block can be *captured* if all its empty neighbours are occupied by opponent stones

What is Go?



- The game terminates once players agree on the life status of blocks
- The blocks and their surrounding area count towards *territory*
- **Territory prediction:** Given a board position predict the owner of each intersection
- Challenging problem for ML!

Modelling Go

- Go is played on a grid graph G, so it is natural to model it with a graphical model such as CRF
- If we want to perform exact inference we can use the Junction Tree Algorithm (G is loopy)
- But... Junction Tree's complexity is exponential in the tree-width, which is N for $N \times N$ grid

Solution

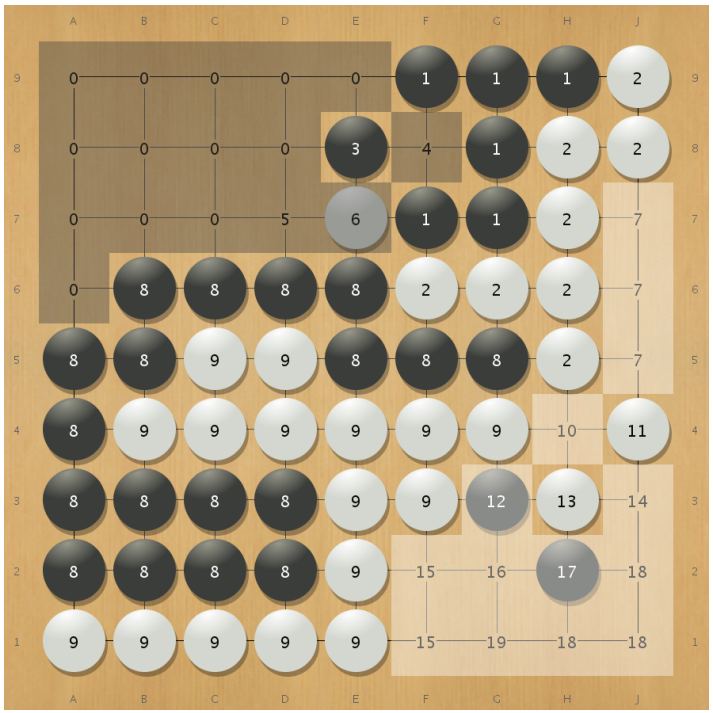
- Globerson & Jaakkola 2006 presented an exact polynomial-time algorithm for computing the partition function
- Restrictions:
  - Graph is planar: can be drawn without crossing edges
  - Binary-valued labels
  - Only edge potentials, no node potentials

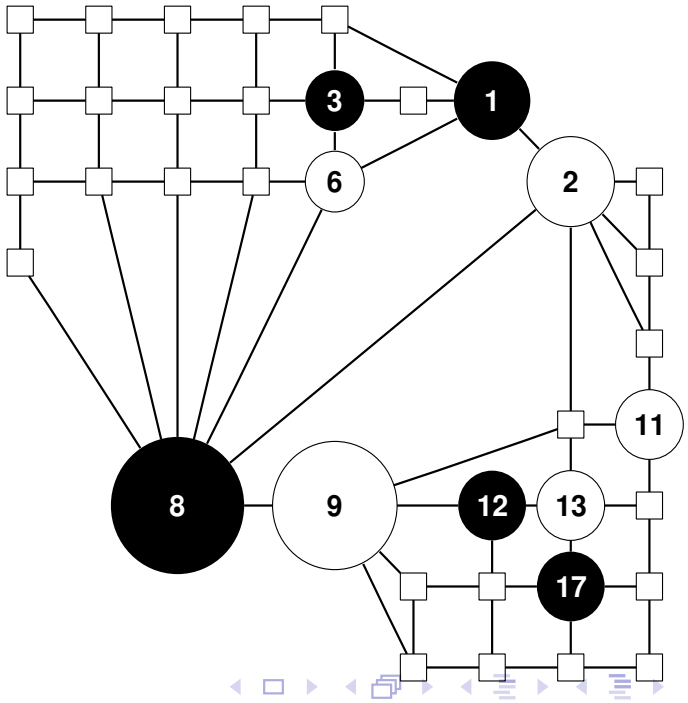
Our work - overview

- Faster and simpler version of Globerson and Jaakkola algorithm
- No need to compute the expanded dual of the original graph
- Showed how to compute gradients and thus perform parameter estimation
- Applied to territory prediction in Go

Graph abstraction: common fate graph

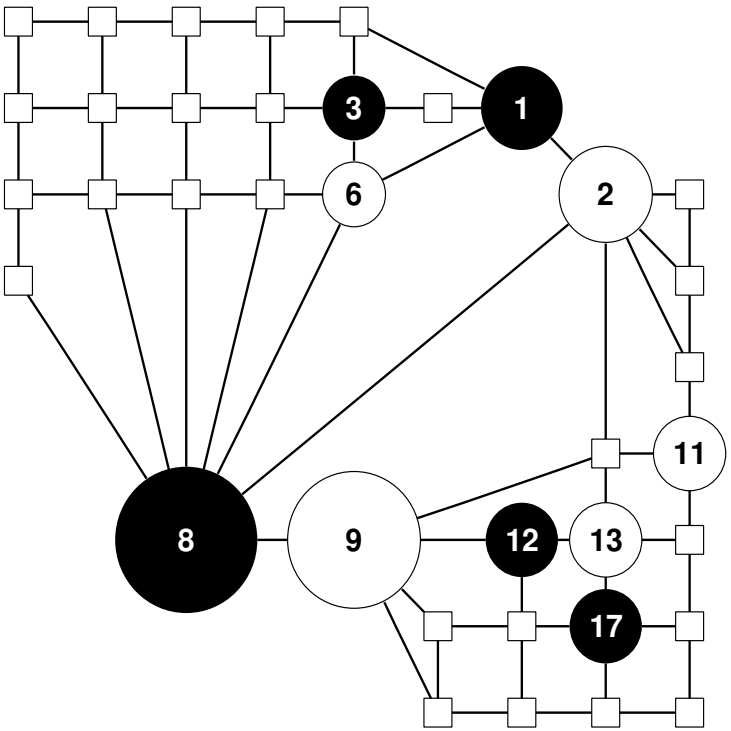
- Blocks always live or die as a unit; Grid graph G does not capture this
- *Common fate graph* G_f (Graepel et al., 2001) merges all stones of a block into a single node

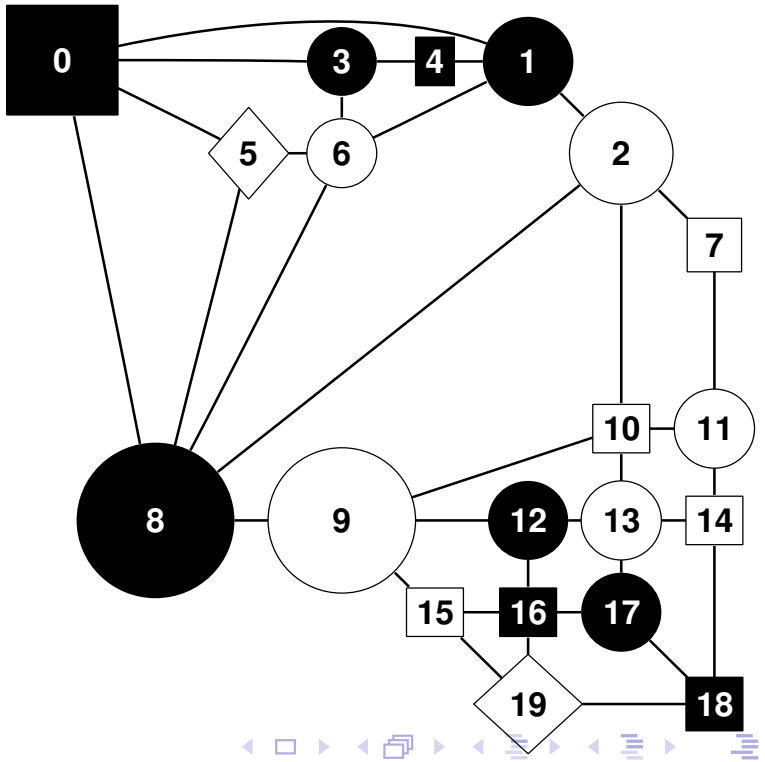




Graph abstraction: block graph

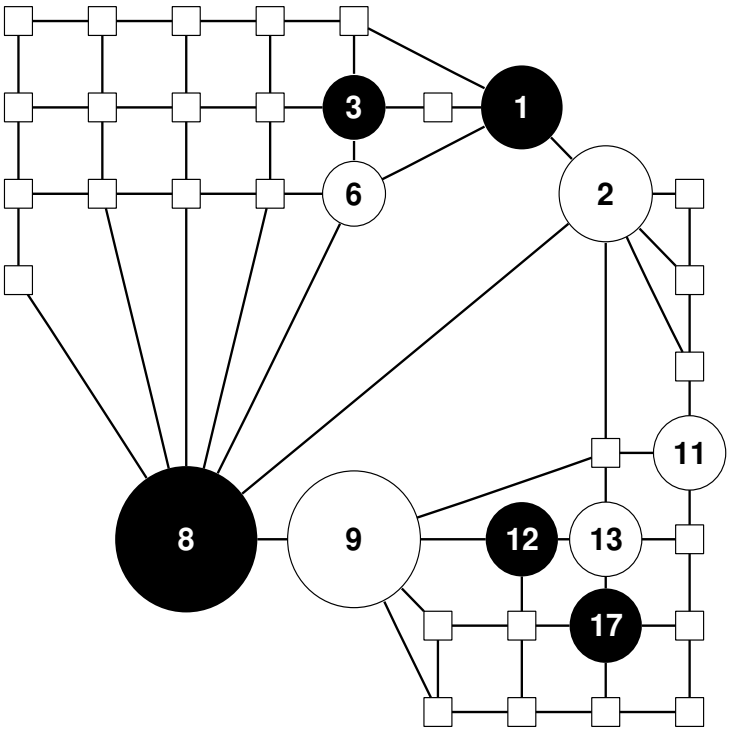
- Use Manhattan distance to classify empty regions into 3 types: *black surround* (■), *neutral*(◇) and *white surround*(□)
- Collapse empty regions of G_f to form the *block graph* G_b

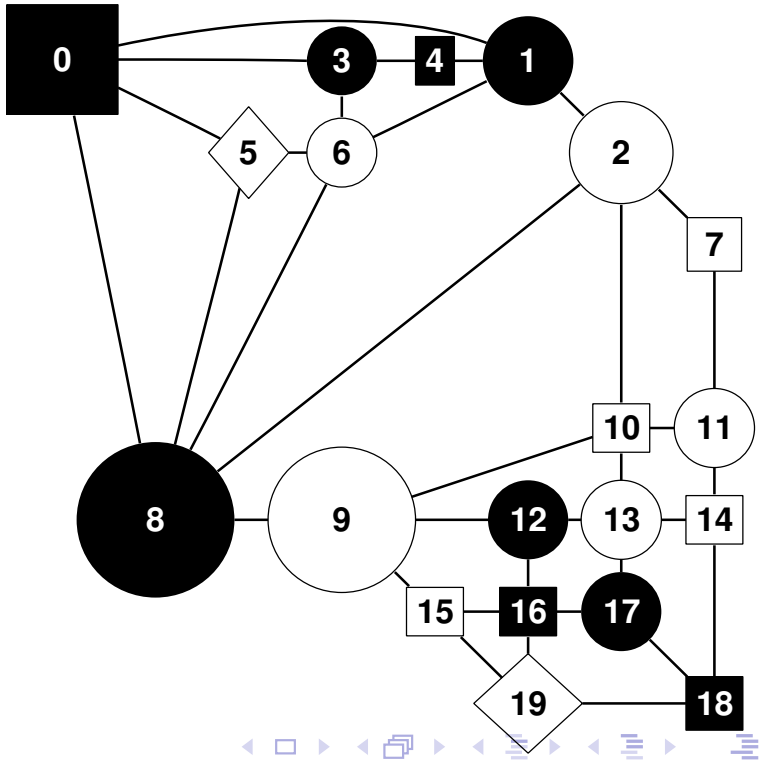




Graph abstraction: block graph

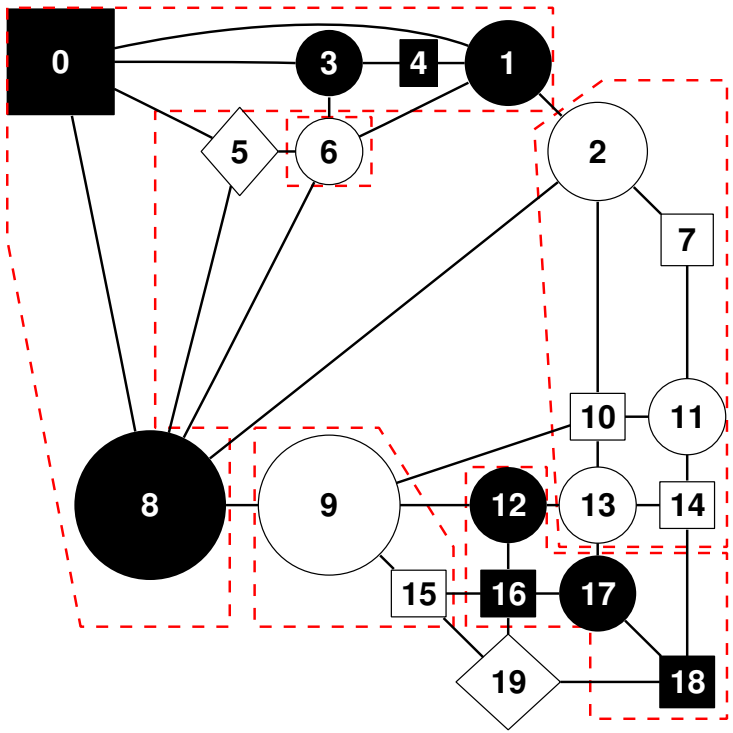
- Surrounds encode the possibility for obtaining territory
- G_b is more concise than G_f, but preserves the kind of information required for predicting territory





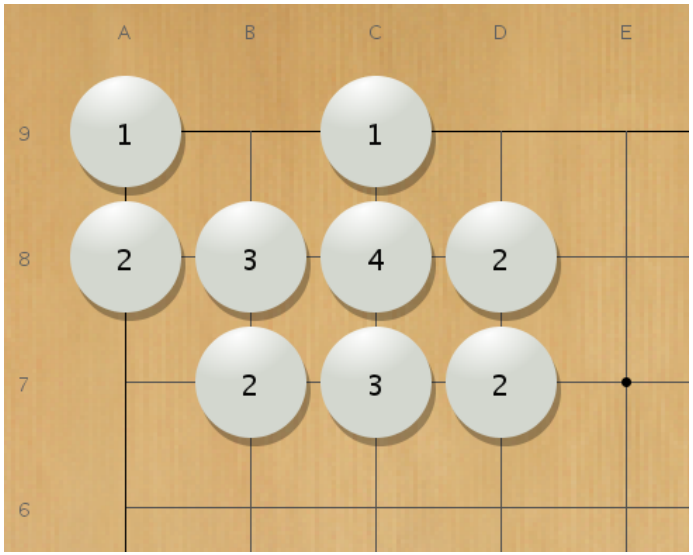
Graph abstraction: group graph

- *Group*: set of blocks of the same colour that share at least one surround
- Construct the *group graph* G_g by collapsing groups of G_b



Feature engineering: nodes

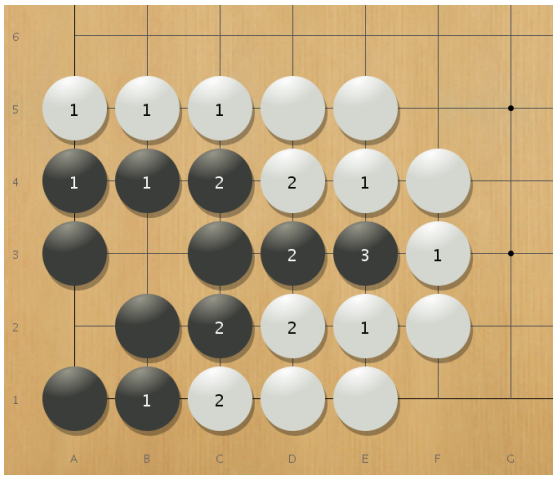
- Given a node $v \in G_b$, for each point $i \in v$ compute the number of adjacent points A_i that are also in v
- Node's feature is a vector F, where $F_k = |\{i : A_i = k\}|$
- Provides a powerful summary of the region's shape



$$F = \{2, 4, 2, 1\}$$

Feature engineering: edges

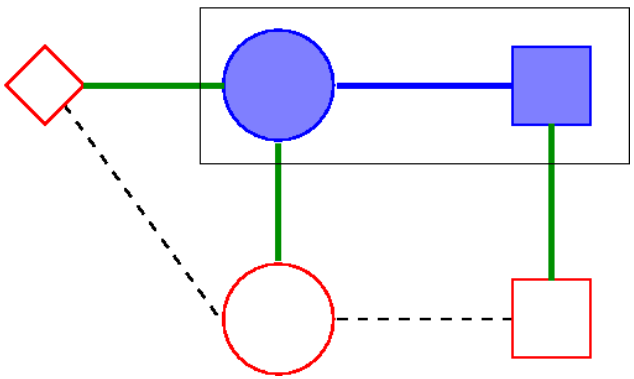
- For two nodes $v_1, v_2 \in G_b$, A_i^1 is the number of points in v_2 that are adjacent to $i \in v_1$ and vice-versa for A_i^2
- Edge's features are two vectors F^1 and F^2 that are constructed using A^1 and A^2 respectively
- Provide extra information such as the boundary shape



$$F^1 = \{3, 3, 1\}, F^2 = \{6, 3, 0\}$$

Parameter sharing

Parameter sharing takes into account all relevant symmetries



	Current Edge		Neighbour Edges	
	Param.	Feat.	Param.	Feat.
Nodes	$\vec{\theta}_{\bigcirc}$	●	$\vec{\theta}^{n}_{\diamond}$	◇
	$\vec{\theta}_{\square}$	■	$\vec{\theta}^{n}_{\bigcirc}$	○
			$\vec{\theta}^{n}_{\square}$	□
Edges	$\vec{\theta}_{\bigcirc\square}$	● → ■	$\vec{\theta}^{n}_{\diamond\bigcirc}$	◇ → ●
	$\vec{\theta}_{\square\bigcirc}$	■ → ●	$\vec{\theta}^{n}_{\bigcirc\diamond}$	● → ◇
			$\vec{\theta}^{n}_{\bigcirc\bigcirc}$	○ → ● ● → ○
			$\vec{\theta}^{n}_{\square\square}$	□ → ■ ■ → □

Experiments

- 9×9 endgame positions of van der Werf et al., 2005
- Learning
  - Use the block graph G_b
  - Optimization with LBFGS
- Prediction
  - Use the group graph G_g
  - Use CRFs MAP assignment as predicted label

Results

	Error (%)			
Algorithm	Vertex	Block	Winner	Game
Naive	6.79	17.57	30.79	75.70
Stern et al., 2004	4.77	7.36	13.80	38.30
Block graph	2.36	3.56	4.53	13.02
Block graph + neighbour features	1.87	2.76	3.42	9.60
Block graph + other enhancements	1.54	2.20	2.09	7.90
* GnuGo	-	-	-	1.32
* van der Werf et al., 2005	0.19	$\leq$ 1.00	0.50	1.10

*: employs Go-specific features and was used to label data

Conclusion

- First real application of Globerson and Jaakkola algorithm
- 2-stage graph reduction of the Go positions. The first used for learning, the later for prediction
- Generic node and edge features. Parameter sharing between equivalent node and edge types

Future work

- Find better ways to classify empty regions
- Add more domain-specific knowledge
- Extend to 19×19 games
- Incorporate into a Monte-Carlo based program

Questions?

"Some cause happiness wherever they GO, others whenever they GO" - Oscar Wilde