# A.I. for Risk Game

Wah Loon Keng[*] Benjamin H. Draves[†]

April 16, 2015

**Abstract**

We implement an A.I. to play the board game *Risk*. The game is formalized as an optimization problem in graph theory, where countries are represented as nodes in an undirected graph, and decisions are considered based on the graph properties. We introduce solution algorithms, with variable parametrizations, which are then implement as A.I.s with different personalities in `JavaScript`. We set up games among A.I.s to measure their performances, and discover unusual game strategies.

## 1 Introduction

The board game *World Domination RISK* ® is a game of military strategy, where players of different factions try to conquer all 42 countries on the map, by deploying armies to attack and defend. We use the classic *Hasbro* version of the game; the rules are well known and can easily be found online, but they will be included as we walk through our algorithms.

In this paper, we first set up our formalization of the game as a graph optimization problem. Next we introduce graph algorithms to carry out each of the game moves, and explain our reasoning behind them. Then, we combine these to implement an A.I., with variable personalities based on the parametrization of its internal algorithms. The different A.I.s then play multiple games against one another, with their results recorded. Finally, we analyze their performances, and find several unusual, interesting strategies discovered by the A.I.s that are never observed from human players.

This entire project is public on GitHub: `https://github.com/kengz/Risk-game`.

[*]Lafayette College, Easton, PA 18042, USA. kengw@lafayette.edu.
[†]Lafayette College, Easton, PA 18042, USA. dravesb@lafayette.edu.

## 2 Formalization

The game is inherently dependent on the board, which is a world map of 42 countries, interconnected in specific ways. Decisions to attack or defend are based on the distribution of the armies, the surroundings of a location, and connectivity of countries. These motivate our formalizing the game board and algorithms based on an undirected graph. From now we shall refer to the graph representation as *map*:

**Definition 1.** *A map is a connected, undirected planar graph, with 42 nodes, each representing a country. The nodes are connected the same way as are countries on the game board, by undirected edge of weight 1.*

We assign data fields to each node, namely its country name, the continent it is in, its player owner, the number of armies of the owner in it, its worth and pressure as determined by some metric described below.

**Definition 2.** *A region is a connected subgraph consisting of nodes all owned by the same player. Each player can own many regions, which together partition the map.*

**Definition 3.** *Radius is the measure of shortest distance from an origin node. We identify the neighbors of node $\mathcal{O}$ at radius $k$ to be the nodes whose shortest distance from $\mathcal{O}$ is $k$.*

Draft:

1. Formalization of problem

2. Algorithms and decisions

3. AI Implementation and variations

4. results, performance, analysis

5. AI behaviors, surprises

**Definition 4.** *A block code is a rectangular array of $n$-nary letters (entries), with non-repeating columns and rows. Notate any block code of $n$-nary, $p$-columns and $k$-rows as*

$$BC(n, p, k)$$

*where the letters are elements of the set $\mathcal{M} = \{1, 2, \ldots, n\}$. It is easy to see that $1 \leq k \leq n^p$ due to the non-repeating columns and rows.*

Equivalently, a block code is a collection of $n$-nary codewords (the rows) of length $p$ (number of columns). Below is an example of block code with $n = 2, p = 3$, with the maximum number of $2^3 = 8$ rows. It is a listing of $\{0, 1, 2, \ldots, 7\}$ in binary.[1]

$$BC(2,3,2^3) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Block codes are equipped with the following set of three operations, under which a block code is still considered to be equivalent to the original:

$$\textit{Column-swapping} \tag{1}$$
$$\textit{Row-swapping} \tag{2}$$
$$\textit{Column-wise letter-permutation}^2 \tag{3}$$

To illustrate the operations, take a ternary block code with the operations:

$$BC(3,2,3) = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 2 & 2 \end{bmatrix}$$

$$\text{permutation to column 1:} \;\; 1 \mapsto 3,\; 2 \mapsto 2,\; 3 \mapsto 1$$
$$\text{permutation to column 2:} \;\; 1 \mapsto 2,\; 2 \mapsto 3,\; 3 \mapsto 1$$
$$\text{swap the columns}$$

The resultant block code is then

$$\begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 2 & 2 \end{bmatrix} \mapsto \begin{bmatrix} 3 & 2 \\ 3 & 3 \\ 2 & 2 \end{bmatrix} \mapsto \begin{bmatrix} 3 & 3 \\ 3 & 1 \\ 2 & 3 \end{bmatrix} \mapsto \begin{bmatrix} 3 & 3 \\ 1 & 3 \\ 3 & 2 \end{bmatrix}$$

Any block code generated by these operations is considered equivalent to the original. Thus, the operations allow us to define class on the block codes.

**Definition 5.** *A class is a collection of a block code and all the possible block codes obtained by applying these operations to it. Thus, any block codes are said to be equivalent(in the same class) if and only if they can be made identical using these operations.*

[1]For readability, block codes come with brackets in this paper; we do not consider matrix operations.

**Comment.** This problem is originally motivated by a research in Quantum Foundations: the classification of Hardy-type paradoxes. Consider an experiment setup of $p$-parties, $n$-nary party-outcome, with $k$ possible outcomes. The setup is still physically equivalent under the relabeling of parties (column swapping), reordering of the occurence of outcomes (row-swapping), and relabeling of party-outcome (column-wise letter-permutation). Thus it is sufficient to study only a representative of these equivalent setups.

## 3 Problem Statement

The problem is to identify the class of a block code, or to determine whether two block codes are equivalent (whether they belong to the same class). This is equivalent to the generalization of the problem of canonicalizing matrices:

**Special case.** Given two matrices of the same size, determine whether or not they can be identical under row and column swapping.

**General case.** The same as above, but with an additional operation of column-wise letter-permutation (or row-wise letter-permutation when a matrix is transposed).

*Fripertinger '98* computed the number of classes for block codes $BC(n, p, k)$ for up to $n = 7$, and later produced the representatives of these classes by rewriting them using vectors of $n$-adic numbers. However, the number of block codes, and the number of distinct classes, increases quickly due to combinatorial explosion even when the parameters $n, p, k$ are small.

This makes it unfeasible to identify the classes or determine the equivalence between block codes by exhaustive generation and comparison of all the class members.

## 4 The Bundled Form and Algorithm

We now present a non-exhaustive algorithm that solves the problem. The main idea is to transform a given block code using the allowed operations into a unique, canonical form of the class, called the *Bundled Form*. The problem is solved by directly comparing the *Bundled Forms* of the block codes.

### 4.1 Notations and Definitions

$BC(n, p, k)$: *The generic block code* specified by three parameters: $n$-nary entries, $p$-columns, $k$-rows, where $k \leq n^p$. Block codes obey the three operations of column-swapping, row-swapping, and column-wise letter-permutation. To distinguish a specific instance of the generic block code, index it with subscript $b$, like $BC(n,p,k)_b$.

$\mathcal{S}^c_{i\ldots j}$: *Bundle.* It is a sub-column containing only identical letters. Obviously, one can swap the rows of a block code to result in a column having nicely bundled entries, i.e. identical letters are grouped together in the column, and the bundles form the column.

The superscript $c$ specifies the column of the block code the *bundle* resides. The subscript $i \ldots j$ is a number sequence of length $c$, $j$ indexes the bundles down the $c$-column of the block code.

$||\mathcal{S}^c_{i\ldots j}||$: *The length* of *bundle*: the number of identical letters in it.

$\mathcal{B}(\mathcal{S}^c_{i\ldots j})$: *The sub-block code* $BC(n, p-c, ||\mathcal{S}^c_{i\ldots j}||)$ on the right of the *bundle* $\mathcal{S}^c_{i\ldots j}$, spanning the columns $c+1, \ldots, p$ and the same rows as the bundle. For generality, call the original block code a sub-block code of its super-bundle $\mathcal{S}^0$, so $BC(n,p,k) = \mathcal{B}(\mathcal{S}^0)$.

$\mathcal{S}^{c+1}_{i\ldots jh}$: *Sub-bundle* of the bundle $\mathcal{S}^c_{i\ldots j}$ immediately right to the bundle, i.e. it is a bundle of the sub-block code $\mathcal{B}(\mathcal{S}^c_{i\ldots j})$.

Now we can write the sub-block code as:

$$\mathcal{B}(\mathcal{S}^c_{i\ldots j}) = \{\mathcal{S}^{c+1}_{i\ldots jh}, \mathcal{S}^{c+2}_{i\ldots jhg}, \mathcal{S}^{c+3}_{i\ldots jhgf}, \ldots, \mathcal{S}^{p}_{i\ldots jhgf\ldots e}\}$$

where the subscript variables range on separate valid index sets.

For a fixed bundle $\mathcal{S}^c_{i\ldots j}$ with a fixed $j$-value, we have a fixed sub-block code $\mathcal{B}(\mathcal{S}^c_{i\ldots j})$ on its right. Since we can perform swapping on the rows spanned by the sub-block code, we can get nicely-bundled entries on the $(c+1)$-column. $h$ indexes the sub-bundles $\{\mathcal{S}^{c+1}_{i\ldots jh} : 1 \leq h \leq n\}$ that form the column. Since we require that the same entries be bundled together, and there can be at most $n$-different $n$-nary letters, we get $j \leq n$.

Note that the definitions of sub-block code and sub-bundles are recursive. Symmetrically we can define super-bundles of the sub-bundles. The recursive process of partitioning the block code into smaller bundles "refines" it as we proceed from column 1 to $p$. Also note that no sub-bundle can belong to different super-bundles.

Furthermore, since there can be no repeating rows in a block code, the "finest refinement" must be reached at column $p$, i.e. $||\mathcal{S}^p_{i\ldots q}|| = 1$, or else there will be more than one rows that share the same super-bundles all the way from column $p$ to 1 - a contradiction.

We give an example to illustrate the concept of sub-bundles and sub-block codes.

$$BC(3,3,6)_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 3 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \\ 2 & 3 & 1 \end{bmatrix} \mapsto \left[\begin{array}{ccc} 1 & 2 & 1 \\ \cline{3-3} 1 & 2 & 2 \\ \cline{2-3} 1 & 1 & 1 \\ \hline 2 & 3 & 1 \\ \cline{3-3} 2 & 3 & 2 \\ \hline 3 & 2 & 1 \\ \hline \end{array}\right] = \left[\begin{array}{ccc} \mathcal{S}^1_1 & \mathcal{S}^2_{1,1} & \mathcal{S}^3_{1,1,1} \\ \cline{3-3} \vdots & \vdots & \mathcal{S}^3_{1,1,2} \\ \cline{2-3} \vdots & \mathcal{S}^2_{1,2} & \mathcal{S}^3_{1,2,1} \\ \hline \mathcal{S}^1_2 & \mathcal{S}^2_{2,1} & \mathcal{S}^3_{2,1,1} \\ \cline{3-3} \vdots & \vdots & \mathcal{S}^3_{2,1,2} \\ \hline \mathcal{S}^1_3 & \mathcal{S}^2_{3,1} & \mathcal{S}^3_{3,1,1} \\ \hline \end{array}\right]$$

Above, we pick a member of the generic block code $BC(3,3,6)$. No column-swapping is performed. We swap the rows to result in bundles $\mathcal{S}^1_1, \mathcal{S}^1_2, \mathcal{S}^1_3$ on the first column. These bundles and their sub-block codes are partitioned from each other by horizontal lines. Then, for each bundle $\mathcal{S}^1_i$, we swap the rows to obtain sub-bundles in the sub-block codes, and repeat the process recursively, refining the original block code down to the last column.

For example, look at the first bundle $\mathcal{S}^1_1$. Because the letter '1' occurs three times, $||\mathcal{S}^1_1|| = 3$. Recursive bundle-refinement on the sub-block codes (and sub-sub-block codes) give finer sub-bundles $\mathcal{S}^2_{1,1}, \mathcal{S}^2_{1,2}$, and $\mathcal{S}^3_{1,1,1}, \mathcal{S}^3_{1,1,2}, \mathcal{S}^3_{1,2,1}$. In terms of sub-block codes,

$$\mathcal{B}(\mathcal{S}^1_1) = \{\mathcal{S}^2_{1,1}, \mathcal{S}^2_{1,2}, \mathcal{S}^3_{1,1,1}, \mathcal{S}^3_{1,1,2}, \mathcal{S}^3_{1,2,1}\} = \{\mathcal{S}^2_{1,1}, \mathcal{S}^2_{1,2}, \mathcal{B}(\mathcal{S}^2_{1,1}), \mathcal{B}(\mathcal{S}^2_{1,2})\}$$

$$\mathcal{B}(\mathcal{S}^2_{1,1}) = \{\mathcal{S}^3_{1,1,1}, \mathcal{S}^3_{1,1,2}\}, \qquad \mathcal{B}(\mathcal{S}^2_{1,2}) = \{\mathcal{S}^3_{1,2,1}\}$$

Now that we have the notation of bundles and sub-block codes, we can proceed to define the unique, canonical *Bundled Form.* This is done by using the *Bundled Form Algorithm*, which transforms a block code using the allowable operations.

## 4.2 Characteristics of The Bundled Form

With the notations and concepts, we can characterize the *Bundled Form.* It is basically a reordering of the sub-bundles and sub-block codes via row and column swapping such that for all $c \in \{1, 2, \ldots, p\}$ and for all $i, j, h, e \in \{\textit{some valid index set}\}$:

$$||\mathcal{S}^c_{i\ldots j}|| \geq ||\mathcal{S}^c_{i\ldots j+1}|| \tag{4}$$

$$||\mathcal{S}^{c+1}_{i\ldots j,h}|| \geq ||\mathcal{S}^c_{i\ldots j}|| \tag{5}$$

$$||\mathcal{S}^{c+1}_{i\ldots j,h}|| \geq ||\mathcal{S}^{c+1}_{i\ldots j+1,h}|| \tag{6}$$

$$||\mathcal{S}^p_{i\ldots j,h,\ldots,e}|| = 1 \tag{7}$$

In addition to these, there's a final characteristic which gives uniqueness to the Bundled Form. The description is part of the algorithm (refer to Lemma 1).

## 4.3 The Bundled Form Algorithm

The algorithm essentially ranks all the possible bundling of a block code and pick one with the higest rank (see Lemma 1). Starting from a given block code $BC(n, p, k)_b = \mathcal{B}(\mathcal{S}^0)$, apply the algorithm recursively to it and the sub-block codes $\mathcal{B}(\mathcal{S}^c_{i\ldots j})$ starting from $c = 0$ until it terminates at column $c = p$.

Note that whenever columns and rows are swapped, and letters are permuted on columns, even when mentioned in the context of sub-block codes, it is understood that they are always performed on the entire block code $\mathcal{B}(\mathcal{S}^0)$, so that it obeys the operations and stay in the same class.

1. Focus on the block code $\mathcal{B}(\mathcal{S}^c_{i\ldots j})$. Scan each of its columns, and look for the highest number of letter-repetition. Note that there may be more than one such column. Call this *multiplicity.* For later comparison, index them with $t \in \mathcal{T}$, where $\mathcal{T}$ is some valid index set. For each of these columns:

   1.1. Move this $t$-column to position $c + 1$, i.e. the first column of $\mathcal{B}(\mathcal{S}^c_{i\ldots j})$.

   1.2. Swap the rows such that all identical letters are bundled together, and the bundles are arranged down the column with decreasing sizes, i.e.

$$\begin{bmatrix} \mathcal{S}^{c+1}_{i\ldots j,1} \\ \mathcal{S}^{c+1}_{i\ldots j,2} \\ \vdots \\ \mathcal{S}^{c+1}_{i\ldots j,H} \end{bmatrix}_t \quad \textit{such that} \quad ||\mathcal{S}^{c+1}_{i\ldots j,1}||_t \geq ||\mathcal{S}^{c+1}_{i\ldots j,2}||_t \geq \cdots \geq ||\mathcal{S}^{c+1}_{i\ldots j,H}||_t \tag{8}$$

2. Since the Bundled Form is unique, we need to select some of the many $t$-columns before proceeding. We do so by fixing $h \in \{1, 2, \ldots, H\} = \mathcal{H}$ and checking all $t \in \mathcal{T}$:

   2.1. Starting from $h = 1$, find $Max\{||\mathcal{S}^{c+1}_{i\ldots j,h}||_t\}_{t\in\mathcal{T}}$, and keep only the indices $t \in \mathcal{T}$ that yield $||\mathcal{S}^{c+1}_{i\ldots j,h}||_t = Max\{||\mathcal{S}^{c+1}_{i\ldots j,h}||_t\}_{t\in\mathcal{T}}$.

   2.2. If the index set $\mathcal{T}$ still contains more than one element, i.e. $||\mathcal{T}|| > 1$, repeat 2.1 for $h = h + 1$ with $Max\{||\mathcal{S}^{c+1}_{i\ldots j,h+1}||_t\}_{t\in\mathcal{T}}$.

   2.3. If the process terminates when:

      2.3.1. $||\mathcal{T}|| = 1$.
      Then there is a unique column $c+1$ with the bundles $\{\mathcal{S}^{c+1}_{i\ldots j,1}, \mathcal{S}^{c+1}_{i\ldots j,2}, \ldots, \mathcal{S}^{c+1}_{i\ldots j,H}\}$. Repeat the algorithm from step 1 for each of the sub-block codes

$$\mathcal{B}(\mathcal{S}^{c+1}_{i\ldots j,1}), \mathcal{B}(\mathcal{S}^{c+1}_{i\ldots j,2}), \ldots, \mathcal{B}(\mathcal{S}^{c+1}_{i\ldots j,H})$$

      2.3.2. $||\mathcal{T}|| > 1$ at $h = H$.

Then there are several $t$-columns like e.q.(8). For each $t \in \mathcal{T}$, repeat the algorithm from step 1 for each of the sub-block codes

$$\mathcal{B}(\mathcal{S}^{c+1}_{i\dots j,1})_t, \mathcal{B}(\mathcal{S}^{c+1}_{i\dots j,2})_t, \dots, \mathcal{B}(\mathcal{S}^{c+1}_{i\dots j,H})_t$$

Note that the index set $\mathcal{T}$ can expand due to *multiplicity* when step 1 is repeated. For example, $t = 1$ under multiplicity is expanded from an element to a set of sub-indices $t = 1 \mapsto \{11, 12, 13, \dots, 1u\}$ for some $u$. Update the index set $\mathcal{T}$ so that it is a set of sets:

$$\mathcal{T} = \{\{11, 12, 13, \dots, 1u\}, \{21, 22, 23, \dots, 2u_2\}, \dots, \{T1, T2, T3, \dots, Tu_T\}\}$$

Now, the recursive version of e.q. (8) is

$$\begin{bmatrix} \mathcal{S}^{c+2}_{i\dots j,h,1} \\ \mathcal{S}^{c+2}_{i\dots j,h,2} \\ \vdots \\ \mathcal{S}^{c+2}_{i\dots j,h,G} \end{bmatrix}_t \quad \textit{such that} \quad ||\mathcal{S}^{c+2}_{i\dots j,h,1}||_t \geq ||\mathcal{S}^{c+2}_{i\dots j,h,2}||_t \geq \dots \geq ||\mathcal{S}^{c+2}_{i\dots j,h,G}||_t \tag{9}$$

Step 2 is repeated with a modification for recursion: fix $h \in \mathcal{H}$, fix $g \in \{1, 2, \dots, G\} = \mathcal{G}$, and check all $t \in \mathcal{T}$.
For each value for $h \in \mathcal{H}$, run through index $g \in \mathcal{G}$. Whenever a sub-index $tv \in \mathcal{T}$ is deleted, delete also from $\mathcal{T}$ the entire set containing the sub-index

$$\{t1, t2, \dots, tv, \dots, tu_t\}$$

3. The algorithm will eventually terminate and produce a unique Bundled Form. This gives the solution to the **special case** of the problem, i.e. without letter-permutation.

   Alternatively, we can apply letter-permutation so that the sub-bundles belonging to the same super-bundle have increasing letter-value down the column. This gives the unique *Bundled Form Class Representation of a block code.*

**Lemma 1.** *The termination of the algorithm and the uniqueness of the Bundled Form are guaranteed.*

*Proof.* This is because the algorithm ranks the potential Bundled Forms by comparing the sizes of the bundles down each column. When the ranking is indeterminate, it then repeats the comparison on the next column. The process ends at column $p$ when the "finest refinement" is obtained and $||\mathcal{S}^p_{i\dots,e}|| = 1$, due to the non-repeating rows.

Alternatively, one can imagine rewriting each column of the potential Bundled Forms with a vertical string of number representing the bundle sizes.The algorithm essentially ranks all potential bundled forms by comparing the digits down the string, and then the digits down the substrings.

Due to the *Well-Ordering Principle*, there must be a potential Bundled Form with the highest rank, or several Bundled Forms with the same highest rank. For the latter, simply perform letter-permutation to yield the unique *Bundled Form Class Representation of a block code*, and we are done. □

**Theorem 1.** *Block codes of the same class have the same Bundled Form.*

*Proof.* This is straighforward. Since the algorithm obeys all the operations that define the block code classes, it does not change the class of a block code. Therefore, all the block codes in a class can be transformed into the same unique *Bundled Form Class Representation.* □

Also, block codes of different classes have different Bundled Forms, or else this would contradict Theorem 1.

## 5 Conclusion

This paper sets out to solve the problem of the identification of the class of a block code. We do so by introducing a new *Canonical Bundled Form* as a unique class representation of the block code.

The Bundled Form and its algorithm too solves the special problem of determining the equivalence between matrices under column/row swapping, and the general problem which allows column-wise letter-permutation to the sub-problem. Row-permutation can be done by transposing the matrices.

## 6 Citations

H. Fripertinger. Enumeration, construction and random generation of block codes. *Designs, Codes and Cryptography,* Volume 14 Issue 3: 213-219, 1998.