Using Multi-Agent System Technologies in Risk Bots

Stefan J. Johansson
School of Engineering, Blekinge Institute of Technology
Ronneby, Sweden
sja@bth.se

Fredrik Olsson
Progressive Media ApS
Aalborg, Denmark
FredrikO@progressivemedia.dk

Abstract

We present a multi-agent architecture for playing the game of Risk which is a multi-player board game in which the players control armies and try to conquer the world through attacking each others territories. Our solution puts an agent in every territory, and let them negotiate about what actions to prioritize in the phases of placing armies, attack, and fortify. The results of a tournament of 13 participating bots, shows outstanding results for our solution, MARS which ends first or second (out of six participants) in 507 out of 792 matches.

Introduction

The creation of computational players for two-player board games is a well known domain of computer science and mathematics. For games with manageable search spaces, these methods include extensive searching for possible moves, and weighting of found positions to find the best move from the current position. Such methods can then be improved by using heuristic methods, such as *Iterative deepening* (Slate & Atkin 1977) and *Alpha-Beta pruning* (Hart & Edwards 1961; Knuth & Moore 1975). However, such solutions are not applicable to all kinds of games, e.g. when the branching factor of the search tree is huge.

Risk is a turn-based board game in which each player (of up to six) control a number of armies placed in the territories owned by the player. The world is divided into 42 territories in six continents and the outcomes of attacks between the players are decided through throwing dice. The players move in sequence, get extra armies before their turn and are allowed to attack for as long as they can in each turn. Typically a player has about 23 different attack possibilities in the first move of the initial round, and each of these may use up to three attacking armies (assuming that the player has seven territories, each bordering an averaged 3.3 opponent territories and that the player is given three new armies in the beginning of its turn). The rest of the armies may be used to attack elsewhere, or be used to defend one of the initial seven territories. In this example, each new army may be used a total of $23^3 + 7 = 12174$ possible ways. This means that the number of possible *openings* for the six players is huge ($\approx 3.3 \cdot 10^{24}$), compared to e.g. Chess (400), Go (144780) or even Backgammon (≈ 549000).

An *agent* is a computer system that is situated in some environment, and that is capable of autonomous action in its environment in order to meet its design objectives (Wooldridge 2002). A *Multi-agent System (MAS)* can therefore be defined as a collection of possibly heterogeneous, computational entities, having their own problem solving capabilities and which are able to interact among them in order to reach an overall goal (Ferber 1995).

Attempts to create Multi-Agent System (MAS) based players for board games include AntChess (Drogoul 1995) playing Chess by putting agents in the pieces and then letting them negotiate about threats and attacks. The Israeli Diplomat (Kraus & Lehmann 1995) used a heterogeneous MAS to make successful negotiations with the opponents in Diplomacy, and HAAI is a homogeneous MAS outperforming other available bots in *no-press* Diplomacy (Johansson & Håård 2005). In the game of Risk, a TD-learning solution was trained (without significant success) to play the game (Keppler & Choi 2000).

The rest of the presentation is structured as follows. We will first explain the domain of Risk and go through the structure of our proposed bot (MARS). We then describe the experimental set-up, and present the results. The final sections cover a discussion of MARS and its implication on the results, some conclusions drawn and future work.

The domain — Risk

Risk is a board game which is a registered trademark by Hasbro Inc. The game is played on a somewhat simplified version of a world map shown in Figure 1 and the goal is to conquer the world. A Risk game includes: *a board, a deck of cards, five dice,* and *pieces representing armies.*

In the game there can be up to six players, where the worlds' 42 territories are partitioned by the players. At least one army (from the same player) is in each territory. A game of Risk starts out by randomly dividing the territories among the players. The players then move in turns and *Trade in cards, Place armies, Attack* and *Fortify.*

Trade in cards

During the game the player will receive cards with different symbols. If you have three cards of the same kind or one of each, you can trade them in for five new armies.

Figure 1: World map used in Risk.

Place armies

At the beginning of a placement phase the player is given a number of armies, a, based on the number of territories n, possibly card bonus b, and owned continents $C \subseteq U$:

$$a = \max(3, \lceil \frac{n}{3} \rceil) + b + \sum_{c \in C} d_c \quad (1)$$

The continent bonus, d_c, is 7 for Asia, 5 for both Europe and North America, 3 for Africa, and 2 for both South America and Australia. The armies must now all be placed in at least one of the territories owned by the player.

Attack

Once a player has placed all armies the attack phase is entered. In this phase the player may, from any territory owned by the player which has more than one army, attack a neighbouring territory owned by an opponent. The outcome of an attack is determined by the dice. The attacker chooses the number of armies to attack with and rolls that number of dice (up to three). The defender rolls a die for each of the armies in the defending territory, (up to two). The highest attacking die is matched with the highest defending die and the one with the lowest value looses an army. In case of a tie the attacker looses. If the defender rolled two dice, the process is repeated with the second highest dice.

If the defender runs out of armies the attacker takes over that territory and must decide on the number of armies to bring in into the new territory. This number must be at least the number of dice rolled by the attacker in the last attack. If a defender lose its last territory, the player has lost the game.

Fortify

When a player has finished its attack phase, it may move armies within its territory clusters. Out of all variants of this rule, we follow the *common house rule*, listed in the Risk FAQ (Lyne *et al.* 2005). It states that during the fortification phase any army may move to any neighbouring territory, but it may only move a total of one step during a turn. As before, no territory may have zero armies at the end of the phase.

Limitations of this study

Players may not cooperate e.g. by signal their intentions in such a way that it affects the actions of the other players.

To avoid that a single match of Risk takes too long we have set a limit of 100 rounds. If the game has not terminated by then, it is considered to have reached a *stalemate* and the game is terminated by the game server.

Multi-Agent Risk System

When designing a system like this, there are three main foci: *Agent distribution*: How many agents and what do they control? *Agent communication*: How will the agents communicate? *Agent negotiation*: How will the agents negotiate?

Agent Distribution

There are two main ways in which a system may be structured. A *task-oriented* agent system has one agent for each relevant task, e.g. defence, card handling, or conquering North America. These systems are heterogeneous in the sense that all the agents are basically different.

An *entity-oriented* agent system has one agent per relevant entity in the domain. Here the agents are homogeneous. They are all built in the same way and may represent continents, territories, or even individual armies.

We have chosen to let each territory (own and opponents') be represented by an agent. This gives us a good balance between the complexities of the agents and the architecture. In addition we have one agent that takes care of the cards and an additional agent that communicates with the game server and facilitates the auctions.

Agent Communication

Since standard Risk bots are monolithic, it is safe to assume that any game server found will need to communicate through a single point. This can be solved by introducing a *Mediator agent* (Hayden, Carrick, & Yang 1999; Shu & Norrie 1999). The Mediator will create the territory agents and handle all communication between them. This means that the territory agents need not know about the other territory agents and have one single point for communication. The Mediator is also a layer of abstraction for the game server, letting the game server see MARS as a single bot instead of a whole system of agents.

Agent Negotiation

The typical case of negotiation that will take place in MARS will be to settle disputes such as: *Who needs this army the most?* and *What territories should we attack?*

We have solved these problems through using a first price sealed bid auction mechanism (Wooldridge 2002). This will work in such a way that the mediator agent will put out a request for bids on the available armies. Upon receiving such a request every territory agent will reply by sending a bid of how useful the extra armies will be for the agent. If the bid wins the auction, the agent will receive the new armies.

Architecture

MARS has three different types of agents, as can be seen in Figure 2 (a closer description of the general architecture used can be found in (Johansson 2006)).

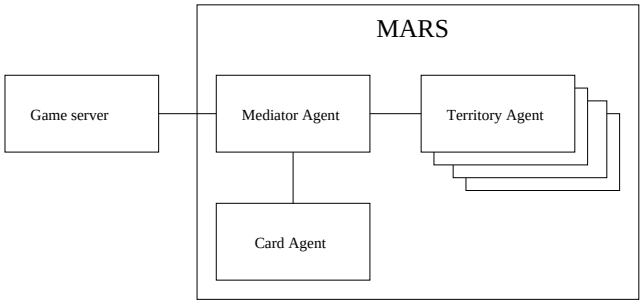

Figure 2: MARS Architecture.

The Mediator agent receives the calls from the game server about what phase the game is in and coordinates the other agents' activities in the phases. It requests bids from the agents and then selects the winner, thereafter the actions of the winning agent are sent to the game server to be performed in the actual game.

In every territory on the board there is *a Territory agent.* On request for a bid, it will calculate a bid based on its current neighbourhood and send it back to the mediator.

The Card agent is quite simple. Its actions are based on a cost and reward system optimizing the use of the cards.

Parameters

MARS is a fairly parametrized system. In Tables 1–2 you see the main parameters used, where W_p, P_{db} and P_{ob} are the most important ones. Since P_{db} and P_{ob} are not independent, given a value of one of them, we may search for an optimal value of the other. In our experiment, we searched the 2D space of W_p and P_{ob} for an optimal combination using narrowing techniques. The less significant parameters, (e.g. in Table 2) were set in more or less *ad hoc* ways.

Param.	Value	Explanation
P_{db}	3.5	Bonus multiplier, higher value makes the bot more defensive.
P_{ob}	170	Bonus multiplier, higher value makes the bot more offensive.
W_p	0.7375	The probability treshold for a win in order to attack.
W_{p1}	0.25	The value of W_p before the first win.
G_l	5	Maximum path length.

Table 1: General MARS parameters.

Territory Evaluation

In Risk, not all territories are the same. Some territories might be very valuable to have, while others may not and this varies over the game. The value of a territory consists of two parts, — a static part and a dynamic part.

The static part is the same for all territories in the same continent and is based on the continents bonus value V_b, its size V_s and the number of borders V_{nb}. The dynamic part is based on what the neighbourhood looks like. Factors affecting the territory value are presented in Table 3.

Param.	Value	Explanation
C_{sv}	70	Bonus multiplier, used to adjust the static territory value
C_{fn}	1.2	Value of having a friendly neighbour
C_{en}	-0.3	Value of having an enemy neighbour
C_{fnu}	0.05	Value of each own neighbouring unit
C_{enu}	-0.03	Value of each neighbouring enemy unit
C_{cb}	0.5	Bonus for bordering a continent
C_{oc}	20	Bonus for owning a whole continent
C_{eoc}	4	Enemy owns the whole continent

Table 2: Territory specific MARS parameters.

Variable	Explanation
$V_b \in \mathbb{N}$	The continent bonus value
$V_s \in \mathbb{N}$	The size of the continent
$V_{nb} \in \mathbb{N}$	The number of borders to the continent
$V_{sv} \in \mathbb{Q}$	$V_b/(V_s * V_{nb})$ The static territory value
$V_{cp} \in [0, 1]$	How much of this continent do we own
$V_{fn} \in \mathbb{N}$	How many friendly neighbours
$V_{fnu} \in \mathbb{N}$	How many armies do the friendly neighbours have
$V_{en} \in \mathbb{N}$	How many enemy neighbours
$V_{enu} \in \mathbb{N}$	How many armies do the enemy neighbours have
$V_{cb} \in \mathbb{N}$	How many continents does this territory border
$V_{oc} \in \{0, 1\}$	Do we own the whole continent except this territory
$V_{eoc} \in \{0, 1\}$	Does an enemy own the continent

Table 3: The MARS evaluation variables.

Using the values of Table 3, and the parameters from Table 2, the value of a territory V_c is defined as follows:

$$V_c = P_{sv} + P_{fn} + P_{fnu} + P_{en} + P_{enu} + P_{cb} + V_b * (V_{cp} + P_{oc} + P_{eoc}), \text{ where} \quad (2)$$

$$P_x = V_x * C_x \quad (3)$$

Dice Odds

A large part of the odds calculations in Risk is based on dice. With a battle is meant a single roll of one to three dice by the attacker and one to two by the defender. This process is often repeated when the defender and the attacker have multiple armies.

By calculating the outcome of every possible dice combination, the odds in Table 4 are given.

Number of attacking dice:			1	2	3
Number of defending dice	1	Attacker wins:	42%	58%	66%
		Defender wins:	58%	42%	34%
	2	Attacker wins:	25%	23%	37%
		Defender wins:	75%	45%	29%
		One each:	—	32%	34%

Table 4: Dice odds for battles

Actions

During each turn there are three phases which are described here (we omit the more or less trivial card trading phase). The mediator agent dispatches the phase information to the relevant agents who in turn act on it.

Place armies The Mediator first sends out a message to all agents in territories *not* owned by MARS. These agents will then tell all their neighbours about the value of owning that specific territory and how many armies are defending it. When an agent receives such a message from a neighbour it will append its values and send the message on. The message will be discarded if the chain of values exceeds G_l.

When a territory that is owned by MARS receives such a list, it will store it in its goal list, containing all possible goals achievable by a territory. In the second step, all own agents calculate and submit both offensive and defensive bids for zero to n armies. A bid for zero armies is included to get bids for goals that require no extra armies since only the goals that win a bidding will be pursued. Given that p_i is the probability to reach goal i,[1] w_i is the value, d_i is the expected probability of defending i (given the current neighbourhood), and v being the value of the current territory the equations of the offensive and defensive bids are:

$$b_o = \max_{i=0}^{n} \begin{cases} p_i w_i d_i & i = 0, \\ \frac{p_i w_i d_i}{i} & i \geq 1 \end{cases} \quad (4)$$

$$b_d = \max_{i=0}^{n} \begin{cases} v * d_{(i+m)} & i = 0 \\ \frac{v * d_{(i+m)}}{i} & i \geq 1 \end{cases} \quad (5)$$

Once all this is done, the agent will take the maximum value of b_o and b_d and submit it to the Mediator. The Mediator will go through all bids received and select the maximum bid as the winner. If this number is less than n the process will be repeated with the remaining armies. For more details about the dice probabilities of Risk, we refer to Lemke (Lemke 1999).

Attack Once the attack phase has started the Mediator will ask all eligible agents to submit bids for attacking. To be eligible an agent needs to be in a territory owned by MARS, have an enemy neighbour and have more than one army. The agent with the highest success odds, (must be $\geq W_p$), will get allowed to perform its attack. If the attacker wins the conflict it will send the goal path to the conquered agent, which will accept it as its own goal and the process is repeated until the odds are below W_p.

Fortify The fortification phase starts out by the Mediator retrieving a list of all owned clusters of connected territories and their distribution of armies. The number of armies a territory can spare for movements elsewhere depends on its environment, how strong the enemies are, or if the territory is a cluster, or a continent border. When this is done the Mediator has a list of all possible sellers in the cluster.

[1]The value of P_i can be calculated through the probabilities of reaching the goal territory through its shortest path, given the number of armies MARS and the opponent(s) have.

Name	1st	2nd	3rd	4th	5th	6th	Sm
MARS	338	169	60	62	69	84	10
Bort	206	47	130	156	149	100	4
EvilP.	205	56	105	153	138	133	2
Boscoe	184	52	142	174	157	76	7
Yakool	135	57	118	162	167	146	7
Quo	177	103	102	132	134	142	2
Pixie	118	153	102	123	139	154	3
Shaft	117	370	157	61	36	38	13
Cluster	100	137	165	161	118	105	6
Nefar.	84	115	128	141	168	156	0
Comm.	16	278	199	130	85	74	10
Stinky	7	139	204	141	147	149	5
Angry	6	17	89	113	207	358	2

Table 5: The outcome of the tournament. Sm is the number of stalemates.

For each seller in the list there will be an intra-cluster auction where the available armies are sold to the territories that need them the most (within the cluster). Since each army can move one step only, a transportation cost is included and max-flow calculations (Edmonds & Karp 1972) are performed so that the move constraints are not violated.

Experimental Set-up

We evaluate our solution by letting it play against a number of other bots.[2] There are several platforms available, e.g. *JavaRisk* (Kirsch *et al.* 2005), *JRisk* (Kinlan 2005), and *Lux* (SillySoft 2005). These have three, two and twelve bots available respectively. Thus the choice fell upon Lux featuring the following bots: *Angry, Communist, Stinky, Cluster, Pixie, Shaft, Yakool, Boscoe, Bort, EvilPixie, Quo* which are implemented by the developers of Lux (SillySoft 2005) and *Nefarious* which is an independent implementation. The bots are described in more detail by Olsson in (Olsson 2005).

The evaluation was performed through running a round robin tournament with the thirteen available bots. We ran one match with each six element subset of the thirteen bots. The order of the bots was random, leaving $\binom{13}{6}$=1716 games, where each bot played 792 of them. The territories were distributed randomly in the beginning of the game.

Results

The results are presented in terms of *Wins* and *Runtime*. The experiment involved playing 1716 matches of which all but 23 ended with a winner before the 100:th turn.

MARS won more matches (338) than any other bot, as can be seen in Table 5 and Figure 3. When looking at both first and second placements, MARS finished top two in 507 out of the total 792 played matches. This is almost as much as the sum of the second and third placed bots *Bort* and *EvilPixie* (514). Note that *Shaft* is the most runner-up (370).

[2]Evaluating the solution against human players would lead to problems of both slow evaluation and unfair competition (a common agreement among many human competitors in tournaments open to bots is to terminate the bots first).

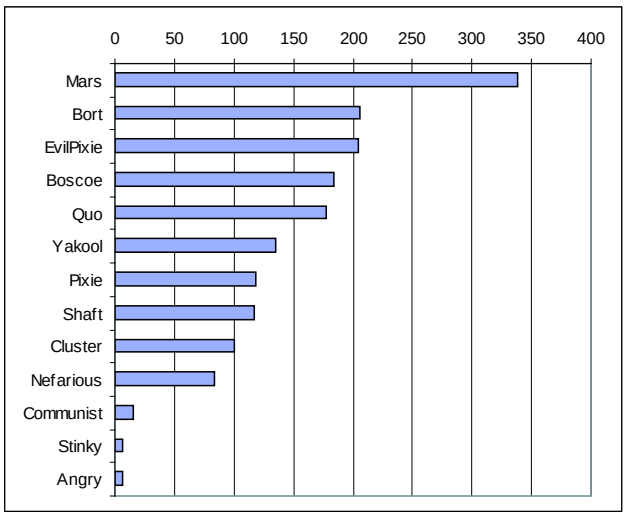

Figure 3: Wins in the tournament.

Name	$Win\%$	$\overline{t_{match}}$	$\frac{turns}{match}$	$\overline{t_{turn}}$	$\frac{Win\%}{\overline{t_{turn}}}$
MARS	42.7	79.72	32.0	2.49	17.1
Boscoe	23.2	40.04	27.1	1.48	15.7
EvilP.	25.9	44.10	25.1	1.76	14.7
Bort	26.0	45.75	25.5	1.79	14.5
Quo	22.3	33.87	21.6	1.57	14.2
Pixie	14.9	40.04	29.9	1.34	11.1
Yakool	17.0	37.62	22.7	1.66	10.3
Shaft	14.8	74.47	34.5	2.16	6.86
Cluster	12.6	54.32	25.4	2.14	5.90
Nefar.	10.6	46.82	18.7	2.51	4.22
Comm.	2.0	49.37	29.1	1.70	1.18
Stinky	1.1	48.88	24.0	2.04	0.54
Angry	1.0	41.33	14.7	2.82	0.36

Table 6: The win percentage, average runtimes (in seconds) per match and per turn, and number of wins per hour of processing time.

As shown in Table 6 and Figure 4, MARS spend more time (t_{match}) than any other bot per match. Shaft is second and has the highest averaged turns per match. *Angry* spends the most time per turn (t_{turn}), whereas *Pixie* is the fastest. MARS, despite being among the slowest bots, reaches the first place when it comes to the number of wins per t_{turn}.

Discussion

We focus our discussion on the validity of the results, the reasons for them, and their relevance, especially to MAS.

Validity There are several platforms which host other bots. The main reason for choosing Lux was the supply of them. The platform itself did not give any specific advantages to MARS. The quality of the implementation of the opponent bots is of course an important aspect. From what we have seen neither the platforms, nor other maps did offer any evidently better bot than the best ones in our experiment.

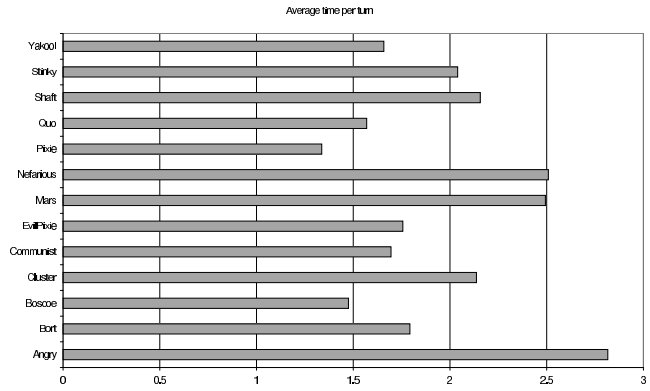

Figure 4: The average runtime t_{turn} (s).

We ran a total of 1732 matches in the experiments equally distributed over the circumstances. MARS is doubtless the most winning bot. When it comes to average placing, it is still the better than Shaft with a probability of 71.7 per cent.

Qualitative Analysis We believe that there are several factors contributing to the good performance of MARS, e.g.: In each phase, all relevant territory agents cooperate to find a good solution. The handling of uncertainties is essential. MARS uses the dice odds to calculate the expected outcome of invading a territory several steps away (including the expected losses on the way). The use of the max-flow algorithm makes MARS propagate as many armies as possible through its territories in each fortification phase.

However, when calculating the mean placement of the bots in the games, MARS is not alone in the top (average placed at 2.49). Shaft placed at an averaged 2.54 is close. Shaft will try to take one of the smaller continents early but if it is unlucky in the draw of the territories it will be ultra defensive, placing all armies in one territory and not attacking. Saving up for a explosive expansion halfway into a game. Even if it succeds in capturing that first continent it will be extremely defensive and only expand once it can take a large amount of its surrounding territories. If it is successful in both capturing a continent early as well as in the crucial explosive expansion mid-game, it has a good chance of winning. In most of the matches however it will either not be able or even try to capture a continent and never find it advantageus to perform an explosive expansion, or it will expand too late/early. These scenarious most likely result in second and third place finishes.

When it comes to MARS vs. *Shaft*, MARS wins 155 (Shaft 37) of the 302 matches where they met. In the 132 matches where they were the only bots left, MARS won 118.

Relevance Risk holds a number of properties that it shares with domains that are considered to be hard (such as robot football), e.g. the number of possible future states (even a few steps ahead) is too large to facilitate exhaustive search. Moreover the environment is hostile, in the sense that several actors try to gain from your loss and coordination of actions is needed in order to succeed in performing certain tasks.

We argue that the domain has passed the level of toy prob-

lems (even though there are of course more complex environments available) and that our multi-agent based solution is novel in the domain of Risk.

Even though Risk shares several properties such as being competitive, demanding supportive actions at piece level, etc. with Chess and Diplomacy, there are great differences too, e.g. the moves and the types of pieces used.

These differences make us to approach the problems from different directions when designing agent systems able to play the games. For instance, Chess allow (and force) us to plan the support several steps ahead, something that is very hard to do in both Risk and Diplomacy. Risk is a game involving rolling dice. When calculating the possibility of reaching a far goal within the turn of the bot, it has to keep multiple models representing the possible outcomes of conquering intermediate territories on its way to the final goal. This is in order to be able to cancel the final goal if the losses on the way are too high.

Still there are parts of the systems that are common:

- A MAS is easily modularized in pieces or territories.
- The agents can, by negotiating with other agents within reach of it, make common efforts to achieve (possibly partly) common goals.
- The agent metaphor provides a good way of modelling utilities at the individual agent level.

Is then MAS techniques the only way of implementing MARS? Although the functionality of MARS could be implemented by means of traditional centralistic approaches, we have found that agents are well suited to model the territories of Risk, both from the perspective of the current implementation, as well as of the possible extentions, e.g. by letting the territories learn their own valuation functions.

Conclusions and Future Work

We have presented a novel distributed solution for a Risk bot based on a multi-agent system. The agents of the system represent the territories in the game. By using auctions for negotiating about how to allocate the resources, our solution showed good offensive performance reached at an acceptable amount of time in competition with available state-of-the-art solutions. Future work will include:

- Calibrating all the parameters of the model in order to find a near optimal settings for the Risk domain.
- Try our approach in domains, e.g. Settlers of Catan.
- Improve the theoretical understanding of the complexity of MAS based bots.
- Adding a module of strategic analysis. At the moment, Risk performs well without paying any attention to strategic matters. We believe that some strategic consideration would improve MARS even further.

References

Drogoul, A. 1995. When ants play chess (or can strategies emerge from tactical behviours?). In Castelfranchi, C., and Müller, J.-P., eds., *From Reaction to Cognition — Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-93 (LNAI Volume 957)*, 13–27. Springer-Verlag: Heidelberg, Germany.

Edmonds, J., and Karp, R. M. 1972. Theoretical improvements in the algorithmic efficiency for network flow problems. *Journal of the ACM* 19:248–264.

Ferber, J. 1995. *Multi-Agent Systems, - An Introduction to Distributed Artificial Intelligence*. Addison W.

Hart, T., and Edwards, D. 1961. The Tree Prune (TP) algorithm. Technical report, Massachusetts Institute of Technology, Cambridge, Massachusetts. Artificial Intelligence Project Memo 30.

Hayden, S.; Carrick, C.; and Yang, Q. 1999. Architectural design patterns for multiagent coordination. In *Proceedings of the International Conference on Agent Systems '99*.

Johansson, S., and Håård, F. 2005. Tactical coordination in no-press diplomacy. In *Proceedings of Autonomous Agents and Multi-agent Systems (*AAMAS*)*.

Johansson, S. 2006. On using multi-agent systems in playing board games. In *Proceedings Autonomous Agents and Multi-agent Systems (*AAMAS*)*.

Keppler, D., and Choi, E. 2000. An intelligent agent for risk. Technical Report CS 473, Computer Science Department, Cornell University.

Kinlan, P. 2005. Jrisk v1.0.8.4. http://sourceforge.net/projects/jrisk/ URL last visited on 2006-01-24.

Kirsch, S.; Domsch, C.; Engberg, D.; and Krug, S. 2005. Javarisk v2.0.93. http://sourceforge.net/projects/javarisk/ URL last visited on 2006-01-24.

Knuth, D., and Moore, R. 1975. An analysis of alpha-beta-pruning. *Artificial Intelligence* 6(4):293–326.

Kraus, S., and Lehmann, D. 1995. Designing and building a negotiating automated agent. *Computational Intelligence* 11(1):132–171.

Lemke, K. 1999. Risk board game dice odds. http://www.plainsboro.com/∼lemke/risk/ URL last visited on 2006-01-24.

Lyne, O.; Atkinson, L.; George, P.; and Woods, D. 2005. Risk - frequently asked questions v5.61. http://www.kent.ac.uk/IMS/personal/odl/riskfaq.htm URL last visited on 2006-01-24.

Olsson, F. 2005. A multi-agent system for playing the board game risk. Master's thesis, Blekinge Institute of Technology.

Shu, S., and Norrie, D. 1999. Patterns for adaptive multi-agent systems in intelligent manufacturing. In *Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems*, 67–74.

SillySoft. 2005. Lux v4.3. http://sillysoft.net/ URL last visited on 2006-01-24.

Slate, D., and Atkin, L. 1977. Chess 4.5 — the northwestern university Chess program. In Frey, P., ed., *Chess skill in Man and Machine*. Berlin: Springer Verlag. 82–118.

Wooldridge, M. 2002. *An introduction to Multi-Agent Systems*. Wiley.