

---

# The Game of Go and Multiagent Systems

---

**Daniel R. Kunkle**

Computer Science Dept.  
College of Computing and Information Sciences  
Rochester Institute of Technology  
Rochester, NY, 14623  
drk4633@rit.edu  
<http://www.rit.edu/~drk4633/>

February, 2002

## Abstract

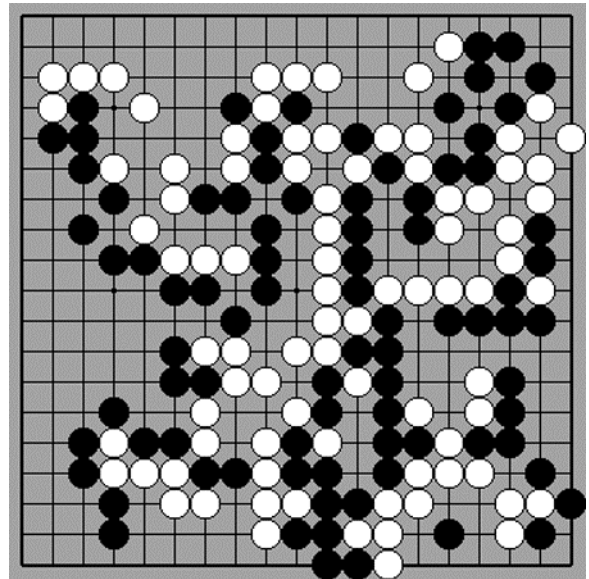
Go is two-player board game that poses one of the largest challenges to game programmers today. Multiagent systems are those composed of multiple autonomous, interacting computer systems, or agents. These two seemingly diverse areas have a number of similarities, which will be explored here.

## 1 INTRODUCTION

Go was originally developed in China about 3000 to 4000 years ago. It is played on a board with a 19 x 19 grid (sometimes smaller boards of 9 x 9 or 13 x 13 are used). Players take turn placing black and white stones on the intersections of the lines with the goal of building walls to surround the largest amount of territory. If a stone, or a group of connected stones of the same color, are ever completely surrounded by an opponents stones those stones are removed from the board. Players can pass their turn at any time and the game ends after two consecutive passes. See figure 1 for an example of a nearly finished game between two master Go players. The rules of Go are extremely simple, in fact the previous few sentences comprise most of the basic rules. Children can learn to play in minutes, yet some spend their whole lives mastering the game.

Like many other games, Go has recently become the focus of many game programmers who seek to develop an application that can play with success. So far, the results have been disappointing, as is illustrated by this quote from Müller:

“Of all games of skill, Go is second only to chess in terms of research and programming efforts spent. Yet in playing strength, Go programs lag far behind their counterparts in almost any other game. While Go programs have advanced considerably in the last 10-15 years, they can still be beaten easily by human players of moderate skill” (Müller, 2001).



**Figure 1:** A nearly finished game of Go illustrating how black and white stones are used to form groups and surround territory (reproduced from <http://gobase.org/>, Copyright © 1994-2002 Jan van der Steen).

Similarly, complex systems that contain a large number of interacting and autonomous agents pose many interesting problems and are beginning to receive a large amount of research and programming effort. Most basically, multiagent systems are those composed of multiple autonomous, interacting computer systems. Research in multiagent systems is spread over many areas, including the creation of intelligent agents, distributed problem solving and planning, search problems, decision making, and learning. These areas are the same that many Go programmers deal with. The Go board is a relatively large space with the distributed stones forming groups that can effect each other much in the same way agents do. Moves

in Go must be decided upon after planning and searching, and many programs have been constructed to learn over time from games played by experienced human players.

It is hoped that by comparing the difficulties and approaches of computer Go applications and multiagent systems that both disciplines can benefit.

## 2 COMPUTER GO ANALOGIES

In (Bouzy and Cazenave, 1996) a number of links between Go and natural complex systems of interacting agents are presented to illustrate concepts shared between the two. Some of these analogies are examined here to serve the same purpose.

### 2.1 Economy

In Go the player must choose to make an “investment” by playing a stone. This stone is invested to start a new group of stones in an empty area of the board, to increase the size of an already existing group of stones, or to connect two previously disconnected groups of stones. In an economic situation similar investments are made to create, grow, or join businesses. In a way, the Go player and the economy serve the same purpose, to allocate investments. Economic models have in fact been researched and applied as a mechanism for decision making in complex environments (Sandholm, 1999).

### 2.2 Social Sciences

Bouzy and Cazenave state “In Go, the value of a stone is mainly given by its relations to its environment and not by the stone itself (on the contrary of Chess where the value of the piece is mainly related to the piece). During the game, each stone has a different value depending on its environment, and on its link with surrounding friend and enemy stones.”

This dependence on the environment is a fact in a social system and a common occurrence in multiagent systems. Bouzy and Cazenave make the appealing analogy that the Go stones form groups just as agents form organizations. The communications and interactions of societies of agents also has a foundation of active research that can serve as support for this analogy (Huhns and Stephens, 1999).

### 2.3 Biology

Most biological structures are built on a hierarchy of nested systems. Micro-cellular components combine to form cells, which combine into tissues, then into organs, and finally organisms. These structures all have some form of dividing boundary that delineates “inside” from “outside”, whether it’s the cell membrane or our skin. These biological systems can also be considered “alive” or “dead” and often have defenses to protect their life.

Each of these biological qualities has an analogous quality in Go. The Go board is gradually filled with structures

that are built on hierarchies, with stones being the lowest structures, then groups of stones (being similar to cells), and finally larger structures of walls and groups. Also like biological structures, structures in Go can be either “alive” or “dead”. A Go structure that is alive can not be removed from the board by surrounding it with opponent stones. A Go structure with two “eyes” (or separate enclosed areas) is always alive, as well as any structure that can create two eyes if the opponent attempted to capture the group. Any structure that can not form two eyes before it is surrounded is said to be “dead”. Unlike in biological systems, however, it can often be very difficult to determine if a Go structure is actually “alive” or “dead”. This classification of structures as “alive” or “dead” is an extremely important task in playing Go as it has a substantial effect on subsequent strategy.

## 3 DIFFICULTIES IN COMPUTER GO

By exploring the obstacles raised against proficient Go playing programs, similarities can be found between multiagent systems and Go that may allow for a transfer of knowledge and techniques between the two disciplines.

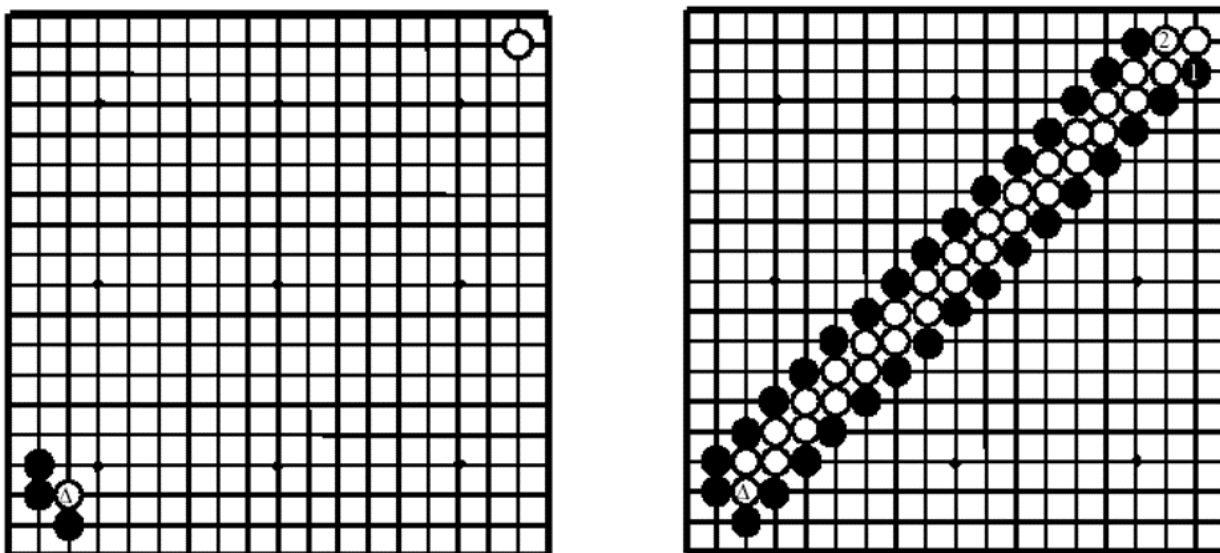
### 3.1 Complexity

One measure of the complexity of games is the number of states that are reachable from the start of the game. Checkers is estimated to have  $10^{17}$  reachable states. In the case of Chess it is estimated that there are  $10^{50}$  reachable states, whereas in Go the estimation is  $10^{160}$  reachable states (Bouzy and Cazenave, 1996). This complexity measure seems to correlate well with the ability levels of the best computer programs to play the games. The best Checkers playing program is clearly superior to the best human Checkers player, the best Chess program, Deep-Blue, is also better than the best human Chess player. The best Go programs on the other hand can be beaten by even moderately skilled human Go players.

However, state complexity can not be the only obstacle in achieving an advanced Go program. A variation of Go, played on a 9 x 9 board instead of the usual 19 x 19, is estimated to have  $10^{40}$  reachable states, fewer than chess. Despite the lower complexity, the strongest programs in this variation of Go are still much weaker than human players. This suggests that other forms of complexity and other difficulties exist in Go, differentiating them from other games that are solved by traditional game programming methods.

### 3.2 Interactivity

Because sufficiently complex environments can not be examined by simple state space search alone other complexity factors influence the ease of decision making. In Go, the stones on the board can affect and interact with each other in complex and often difficult to detect ways. Stones can have important influence on each other at great distances. For example, in figure 2, the white stone in the upper right corner has crucial ramifications for the



**Figure 2:** The white stone in the upper right corner has an important effect on those stones in the lower left corner, even though it is a great distance away. With this white stone present, black can not capture the white stone in the lower left corner using the ladder technique shown to the right, as it will break the ladder in the end (reproduced from Bouzy and Cazenave, 1996).

play occurring at the bottom left corner of the board. If that white stone were absent, black could capture the white stone it partially surrounds by using a technique known as the “ladder”. This would result in the string of white stones in the right half of figure 2 being captured once they reached the wall. However, with the white stone present the ladder can not be used because it would eventually be broken as the fleeing white stones met up with the already placed white stone. This would cause all of the black stones used in the ladder maneuver to be vulnerable to a white counter attack. Good go players must be able to see this eventuality and mark the white stone in the bottom left as “alive” or “dead” appropriately. Most of the time the effects of stones on each other are not as simple as this example.

### 3.3 Non-Linearity

A number of stones can be played successively that do little to change the overall state of the game. On the other hand, a single stone can have a major effect. Likewise, a player’s perception of the state of the game, or misperception, can have minor or major effect. This can occur when trying to decide if a particular group of stones is “alive” or “dead”. This non-linearity of the importance of stones and of the decisions made by the player further complicate the decision making process.

### 3.4 Combining Local and Global Knowledge

The traditional AI approach to Go programming is to collect local information and translate it to global information, at which point it can be used to make decisions (Burmeister et al., 1995). The weakness in this approach is that the global information itself has an effect

on the perception of the local state. One can not decide how important a local group is, or even if it is “alive” or “dead”, without knowledge from the rest of the board. To be successful at Go, both bottom-up and top-down processing must be integrated, interacting together to produce a coherent view of the game and eventually produce the best next move to make.

## 4 MULTIAGENT APPROACHES

The issues examined in the last section relating to Go, the complexity of the environment, the interactivity of stones and groups (agents and societies), the non-linearity of the effects of perceptions, decisions, and actions, and the need to combine local and global knowledge, all have a prominent role in multiagent systems. As such, these issues have received a good deal of research attention and have some well-founded methods to deal with them.

### 4.1 Search

Search has been an important part of AI since the beginning of the field. Although Go can not be played well simply by traditional search methods, those methods do produce good results when used in limited, local, tactical decision situations. Two common state space searches used for tactical decision making in Go are Alpha-Beta and Best-First (Bouzy and Cazenave, 1996). These two search methods, when used in a two-player game, seek to find the move that maximizes benefit for the player while minimizing opportunity for the opponent. They differ in the amount of data they must keep in memory and the size of the trees necessary to perform the search. Algorithms similar to these are also used to

perform searches in multiagent systems (Makoto and Ishida, 1999). It is possible that sophisticated parallel implementations of these search algorithms could be used to partially combat the overwhelming state space complexity of Go.

## 4.2 Societies of Agents

It was discussed in section 2.2 that an analogy can be made between the stones and groups of stones in a Go game and agents and societies of interacting agents in multiagent systems. In section 3.2 it is pointed out that the high degree of interactivity between these groups in Go is a source of difficulty in designing Go programs. It is natural then to examine existing mechanisms for handling agent interaction in multiagent systems.

In describing the importance of coordination in multiagent systems, Huhns and Stephens state:

“In an environment with limited resources, agents must coordinate their activities with each other to further their own interests or satisfy group goals. The actions of multiple agents need to be coordinated because there are dependencies between agents’ actions, there is a need to meet global constraints, and no one agent has sufficient competence, resources or information to achieve system goals” (Huhns and Stephens, 1999).

This quote can apply equally well to Go. The “limited resources” are the application of new stones to the board, the “interests of the agents” may be to create, grow or merge groups, the “global goal” (to have more total territory) may require that the sub-goals be sacrificed, and all of the information necessary to make decisions is distributed in a complex fashion and must be communicated efficiently.

The following are a few mechanisms (described in Huhns and Stephens, 1999) used to distribute tasks in a multiagent system that may have applications to Go programming.

### 4.2.1 Contract Net

The contract net is an interaction protocol modeled after contracting mechanisms used by businesses to govern the exchange of goods and services. The basic steps are:

- An announcement of a task is made
- Bids from potential contractors are made
- The contract is awarded to a suitable contractor
- Results are received and synthesized

The agents bidding on contracts in the case of Go could be positions on the board or groups of stones and the contracts could represent tactical and strategic maneuvers on the board, such as attacking an opponent group to capture it, expanding to gain territory, etc. The most suitable agents, those that could perform the task most “cheaply”, should be chosen by a contract manager to perform the task.

### 4.2.2 Blackboard System

The blackboard system is a metaphor for a group of specialists using a common workspace (a blackboard) to jointly solve a problem that none could solve alone. These specialists monitor the blackboard until they see an opportunity to contribute their specialized function. They then post their contribution to the public blackboard. Contributions are made in this way until the problem is solved.

The following benefits and properties are cited for the blackboard architecture:

- Independence of expertise – specialist knowledge is independent of the problem solving group.
- Diversity in problem-solving techniques – specialists can use whatever techniques they prefer as long as they use the common blackboard format.
- Flexible representation – the blackboard model itself does not place any restrictions on data formats.
- Event-based activation – new information can be placed on the blackboard anytime a salient event occurs.
- Need for control – the control of the interactions is a separate component that directs the problem solving and can be seen as a participating specialist.
- Incremental solution generation – the blackboard can be seen as a solution in progress at any time.

Specialized Go modules would need to handle tasks such as tactical state space search, determining whether groups are “alive” or “dead”, developing a strategic plan, etc.

### 4.2.3 Market Mechanisms

Computational economies are of particular use when there are a large number of agents that must coordinate and distribute tasks. In a system with many agents the previous two methods of task distribution would become too cumbersome as they may require a significant amount of agent-to-agent interaction. Market mechanisms on the other hand only require that agents communicate indirectly through the “prices” of “goods” in the market. To use a computation economy one must define the problem with the following properties:

- The goods to trade
- The consumer agents trading the goods
- The producer agents creating the goods
- The bidding and trading behaviors of the agents

A simple definition of Go in these terms might define the goods as processing power to be spent in examining a position or as new stones that could be used to create, grow, or expand stone groups. Agents could represent positions on the board or groups of stones. The producer is the CPU or the placer of the stones. The bidding and

trading behaviors of the agents would most likely need to be tuned over a period of testing for the system.

## 5 CONCLUSION

The short treatments of Go and multiagent systems given here are only very small fractions from their respective fields, but they hopefully create a point of contact between the two that was previously absent. The work being done in separate fields dealing with complex problems such as these often have insight to offer each other.

## 6 REFERENCES

- 1 Bouzy, B., Cazenave, T. (1996) "Shared concepts between complex domains and the game of Go". <http://citeseer.nj.nec.com/62244.html>
- 2 Bouzy, B., Cazenave, T. (2001) "Computer Go: An AI oriented survey". *Artificial Intelligence* Vol. 132, number 1, pp. 39-103, 2001.
- 3 Burmeister, J., Wiles, J. and Purchase, H. (1995) "On Relating Local and Global Factors: A Case Study from the Game of Go". In *Proceedings of*

the Third Australian and New Zealand Conference on Intelligent Information Systems, pages 187 - 192, Perth, November 1995. IEEE Western Australia Section.

- 4 Huhns, M., Stephens, L. (1999) "Multiagent Systems and Societies of Agents". In: Weiss, G. (ed.): *Multiagent Systems A Modern Approach to Distributed Artificial Intelligence*. MIT press 1999, pp. 201.
- 5 Müller, M. (2001) "Computer Go". Invited survey paper, special issue on games of *Artificial Intelligence Journal*, to appear.
- 6 Sandholm, T. (1999) "Distributed Rational Decision Making". In: Weiss, G. (ed.): *Multiagent Systems A Modern Approach to Distributed Artificial Intelligence*. MIT press 1999, pp. 201.
- 7 Makoto, Y. and Ishida, T. (1999) "Search Algorithms for Agents". In: Weiss, G. (ed.): *Multiagent Systems A Modern Approach to Distributed Artificial Intelligence*. MIT press 1999, pp. 201.