# Multicriteria Evaluation in Computer Game-Playing, and its Relation to AI Planning

**Martin Müller**
Department of Computing Science, University of Alberta
Edmonton, Canada T6G 2E8
mmueller@cs.ualberta.ca

### Abstract

Games are a popular test bed for AI research. Many of the search techniques that are currently used in areas such as single-agent search and AI planning have been originally developed for games such as chess. Games share one fundamental problem with many other fields such as AI planning or operations research: how to evaluate and compare complex states? The classical approach is to 'boil down' state evaluation to a single scalar value. However, a single value is often not rich enough to allow meaningful comparisons between states, and to efficiently control a search.

In the context of games research, a number of search methods using multicriteria evaluation have been developed in recent years. This paper surveys these approaches, and outlines a possible joint research agenda for the fields of AI planning and game-playing in the domain of multicriteria evaluation.

## Introduction

The need for multicriteria evaluation techniques in game-playing programs is not immediately obvious. All popular games have final outcomes that are scalar, be it win-draw-loss as in chess, the number of points in games such as Go or Awari, or the amount of money in casino games. In games that are simple enough to allow a complete analysis, the exact value of a game position can be computed by the minimax evaluation rule (von Neumann 1928; von Neumann & Morgenstern 1947). However, complex games such as chess or Go rarely allow a complete analysis. Evaluation problems arise quickly when a player's knowledge about a game is less than perfect. In place of an intractable complete analysis, games are usually analyzed by a deep but far from exhaustive search using a scalar-valued heuristic evaluation function.

Another source of complexity are games where the complete game state is not known to a player because of hidden information such as the cards in other player's hands in most card games, or because of chance events such as dice throws or cards drawn from a deck during a game.

In this survey we will look at several techniques that use multicriteria evaluation in games. We start by summarizing some basic facts about the structures used in multicriteria evaluation: vector dominance, as used in multiobjective evaluations, and general partially ordered sets. Vector dominance defines a specific kind of partial order, and in turn each finite-dimensional partial order can be represented by a vector with the same dominance relation that is used in multiobjective evaluation.

The main part of the paper consists of an overview of two topics that the author has worked on: the search method of partial order bounding, and a class of games called combinatorial games which are based on a partial order of game values. We also briefly survey other related work on multicriteria techniques in games.

The final, mostly speculative part of the paper discusses possible relations between the two fields of game playing and AI planning. How can multicriteria planning methods be used in game programs? And how can techniques developed for game tree search be used in multicriteria planning?

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

## Background

In this section we describe the sum-of-features model for scalar evaluation, give definitions of multiobjective and partial order structures, and point out their close correspondence, at least in theory.

### The standard scalar approach: weighted sum of features

In the standard model of computer game-playing, position evaluation is a two step process. The first step maps a game position to an abstract representation. A number of relevant attributes are computed and collected in a high-dimensional feature vector $\mathbf{v}$. Within such a vector, single features are usually of a simple type such as 0-1 (boolean), integer, or real. Given a feature vector $\mathbf{v}$ and an integer- or real-valued weight vector $\mathbf{w}$, a scalar-valued evaluation is computed as the weighted sum $eval(\mathbf{v}) = \sum w_i v_i$.

The weighted sum approach to evaluation has been very successful in practice. It has proven to be a useful abstraction mechanism, with many desirable properties, such as simplicity, and ease of use in efficient minimax-based algorithms. Furthermore, in some games there is a natural mapping of positions to a numerical evaluation, for example the expected number of captured pieces in Awari or the balance of territory in Go. In games that end in a simpler outcome such as win, loss or draw, a scalar evaluation can be interpreted as a measure of the relative chance of winning.

Despite the great success of the weighted sum approach to evaluation, the method has quite a few weaknesses, and many of the alternative methods discussed in the survey (Junghanns 1998) were designed to address such weaknesses. The main drawback of using a single number for evaluation is that information is lost. All kinds of features are weighted, added and compared, even those for which addition and comparison do not really make sense. Problem topics include unstable positions, long term strategic features, and close-to-terminal positions. For a detailed discussion see (Müller 2001b). It is therefore natural to consider richer evaluation structures, such as partial orders.

### Partially Ordered Sets

Our definitions follow standard conventions. For more information see textbooks such as (Stanley 1997; Trotter 1992).

A *partially ordered set* or *poset* $P$ is a set (also called $P$) together with a reflexive, antisymmetric and transitive binary relation $\leq$. The dual relation $\geq$ is defined by $x \geq y \iff y \leq x$. Two elements $x$ and $y$ of $P$ are called *comparable* if $x \leq y$ or $y \leq x$, otherwise $x$ and $y$ are called *incomparable*. The relation $x < y$ is defined by $x < y \iff x \leq y \wedge x \neq y$, and $x > y$ is equivalent to $y < x$. A nonempty subset $A$ of $P$ is called an *antichain* if and only if any two distinct elements of $A$ are incomparable.

### Multiobjective Evaluation

The standard approach to evaluation in multiobjective search (Stewart & White 1991; Harikumar & Kumar 1996; Dasgupta, Chakrabarti, & DeSarkar 1996b; 1996a) uses a $m$-dimensional vector of scalar values from domains $Y_1 \ldots Y_m$. A partial order on such vectors is defined by the following vector dominance relation:

$$\mathbf{y} \leq \mathbf{y}' \;\Leftrightarrow\; y_i \leq y_i' \;\forall i \in 1, \ldots, m.$$

In partial order terminology, the poset defined by the vector dominance relation is the direct (or cartesian) product of the totally ordered domains, $Y_1 \otimes \ldots \otimes Y_m$.

While it is clear from the definition that each multiobjective evaluation is a partial order, the converse is also true, in the following sense: Any poset $P$ of finite *dimension* $dim(P)$ can be represented as the direct product of $dim(P)$ total orders (Ore 1962). However, it might be intractable to find such a representation for a given poset. Current algorithms for constructing a multiobjective representation for a given partial order are practical only for posets of "modest size" (Yanez & Montero 1999).

Any partial order technique can immediately be used in a multiobjective framework, whereas multiobjective techniques that rely on the vector dominance in their algorithm only work for the general case if a suitable vector representation can be found.

## Approaches to Multicriteria Evaluation in Computer Game-Playing

In this section we investigate problems of combining partial order evaluations with minimax search. We discuss how partial order evaluations arise in games, and some algorithms for dealing with them.

**Goals of Partial Order Evaluation** The main goal of partial order evaluation is to make comparisons between positions only when they are meaningful. In contrast, standard scalar evaluations are applied and used to compare positions regardless of whether the underlying positions are comparable. By refraining from judgment in doubtful cases, partial order evaluation aims at increasing the confidence in the validity of *better* and *worse* judgments derived by search.

### The Fundamental Problem of Using a Partial Order Evaluation in Minimax Tree Search

When using partially ordered evaluations, the result of a minimax search cannot be just a single value from the partially ordered set, because computing minima and maxima of such values is an ill-defined problem. A totally ordered set such as the integers or reals is closed under the application of the operators *min* and *max*: if $x_1, \ldots, x_n$ are values from a totally ordered set $T$, then both $\min(x_1, \ldots, x_n)$ and $\max(x_1, \ldots, x_n)$ are again elements of $T$, with the properties $\min(x_1, \ldots, x_n) \leq x_i$ and $\max(x_1, \ldots, x_n) \geq x_i$ for all $1 \leq i \leq n$. Furthermore, the minimum and the maximum coincide with one of these values. For values from a partially ordered set, it is no longer possible to define a *min* or *max* operator with these properties.

### Solutions for Special Cases

Several different approaches to overcome this fundamental problem have been tried. In some special cases, it is possible to define meaningful *min* and *max* operators with similar but more restricted properties. One possible approach in the case when the poset is a lattice is to define the least upper (greatest lower) bound of a set of incomparable values as the maximum (minimum) of these values. (Ginsberg 2001) develops such a search model and applies it to the imperfect information game of Bridge. For general lattice-valued evaluations, that do not possess the special semantics used in Ginsberg's model, this approach loses information, since propagating such bounds by a tree backup leads to computing bounds of bounds of bounds etc., which makes the approximation weaker and weaker.

Another approach (Dasgupta, Chakrabarti, & DeSarkar 1996b) keeps track of a set of nondominated sets of outcomes. This can lead to high complexity in the case when there are many incomparable values. A viable search procedure seems to require extra assumptions, such as totally ordered private preferences of the players, so this approach does not seem to be used in practice.

### Approximating Scalar Values

One natural way in which partial orders arise in game state evaluation is uncertainty about the true value of a scalar value. Intervals, "fat values" such as triples containing a lower bound, a realistic value and an upper bound (Beal 1999), and probability distributions (Baum & Smith 1997) are prominent examples. Different kinds of partial orders

can be defined over such structures. One natural interpretation of an interval is as a pair of upper and lower bounds on the unknown true value. The corresponding partial order is a vector dominance order. An example of a relatively strong ordering of probability distributions is *stochastic dominance* (Russell & Norvig 1995).

(Lincke 2002) studies the problem of building an opening book for a game, that can contain both exact and heuristic minimax scores. He defines a new type of min and max operators for "fat values" that keep value representations compact yet can preserve some information about choices between exact and heuristic plays. He further extends the model to deal with draw values arising from position repetition.

## Partial Order Bounding (POB)

Partial order evaluations are useful since they are more expressive than scalar evaluations. One practical problem is that many partial order search methods try to back up partially ordered values through the tree. Depending on the method used, this leads either to potentially huge sets of incomparable options, or to a loss of information, or both. In addition, some methods are applicable only to restricted types or specific representations of partial orders. *Partial order bounding (POB)* (Müller 2001b) avoids such problems by separating the comparison of partially ordered evaluations from the tree backup.

Partial order bounding is based on the idea of null window searches, which have already become very popular with scalar evaluation through search methods such as SCOUT (Pearl 1984) and MTD(f) (Plaat *et al.* 1996). Rather than directly computing minimax values, null window searches are used to efficiently compute bounds on the game value. In SCOUT, the purpose is to prove that other moves are not better than the best known move, while in MTD(f) the minimax value is discovered by a series of null window searches. The goal of a single null window search is to establish an inequality between a given fixed bound and the evaluation of a node in the search tree. POB extends this idea to the case of partial order evaluation.

In the case of a poset $P$, a bound $B \subseteq P$ can be given by an antichain in $P$ that describes the minimal acceptable outcomes a player wants to achieve.

The *success set* of $B$ in $P$ is defined by $S(B) = \{x \in P \mid \exists b \in B : x \geq b\}$, and the *failure set* of $B$ in $P$ is the complement of the success set, $F(B) = P - S(B)$.

The success set contains all values that are "good enough" with respect to the given bound $B$, while the failure set contains the remaining insufficient values. Minimax search is used to decide whether the first player can achieve a result $x \in S(B)$, or whether the opponent can prevent this from happening. In the example tree shown in Figure 1, leaves have been evaluated by pairs of integers. The usual vector dominance order is used. In the diagram, squares represent MAX nodes and circles MIN nodes. For illustration, we consider the following two out of the large number of possible bounds: $B_1 = \{(5,7),(10,3)\}$ and $B_2 = \{(5,8),(6,4)\}$. In this example, MAX can obtain the bound $B_1$ but fails to obtain the bound $B_2$. Leaf evaluations and backed-up values are shown in the figure, with a plus sign representing success and a minus sign representing failure for MAX. Note that MAX would not succeed by selecting one of the two single-element subsets of $B_1$ in this example.

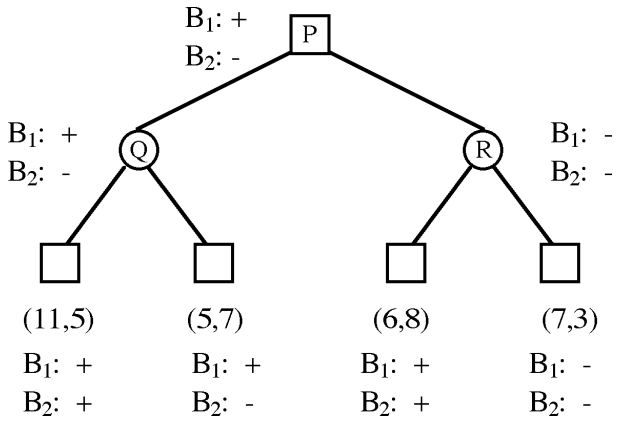


Figure 1: Example of search using POB

In POB, the comparison of partially ordered values is separated from the value backup procedure in the game graph. This simplifies the computation compared with previous approaches, since there are no sets of incomparable values that must be computed, stored, and backed up.

Partial order bounding can be combined with any minimax-based search method, such as alpha-beta or proof-number search (Allis 1994). A partial order evaluation can be added to a sophisticated state of the art search engine with minimal effort. The method has been successfully applied to solving capturing races in the game of Go (Müller 2001b).

The next topic, decomposition search, represents a very different approach to minimax game analysis, which leads to a partial order evaluation as well.

## Combinatorial Games and Decomposition Search

*Decomposition search* (Müller 1999) finds minimax solutions to games that can be partitioned into independent subgames. The method does not use traditional minimax search algorithms such as alpha-beta, but relies on concepts from combinatorial game theory to do locally restricted searches. The result of each local search is an element from the partially ordered domain of combinatorial games (Conway 1976; Berlekamp, Conway, & Guy 1982). In a last step, combinatorial games are combined to find a global solution. This divide-and-conquer approach allows the exact solution of much larger problems than is possible with alpha-beta.

## Combinatorial Game Theory

Combinatorial game theory (Conway 1976; Berlekamp, Conway, & Guy 1982) breaks up game positions into smaller pieces and analyzes the overall game in terms of these local subgames.

Each move in a game corresponds to a move in one subgame and leaves all other subgames unchanged. A game ends when all subgames have ended, and the final outcome

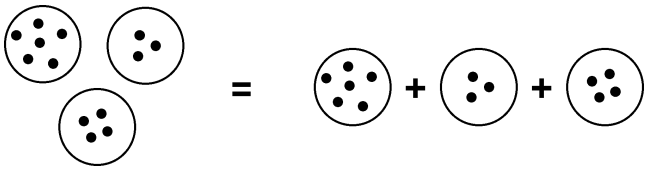

Figure 2: A three heap *Nim* position and its subgames

of the game can be determined from the subgame outcomes. A well-known example of a combinatorial game is *Nim*, shown in Figure 2, which is played with heaps of tokens. At each move, a player removes an arbitrary number of tokens from a single heap, and whoever runs out of moves first loses. Each Nim heap constitutes one subgame. While winning a single subgame is trivial, winning the *sum* of several heaps requires either exhaustive analysis, or, much more efficiently, a computation using the calculus of combinatorial games.

This theory can be seen as both a generalization and as a special case of classical minimax game theory. It is a generalization because *locally*, each player can move in each position, whereas in classical minimax games only one player has the move. On the other hand, from a global viewpoint combinatorial games are a special case, because only some games allow a decomposition into subgames.

## Decomposition Search

Decomposition search (Müller 1999) is a framework for solving games through decomposition, followed by a particular kind of local search named *local combinatorial game search (LCGS)* and the analysis of the resulting local game graphs through combinatorial game theory.

Let $G$ be a game that decomposes into a sum of subgames $G_1 + \ldots + G_n$. Let the combinatorial game evaluation of $G$ be $C(G)$. *Decomposition search* is defined as the following four step algorithm for determining optimal play of $G$:

1. Game decomposition and subgame identification: given $G$, find an equivalent sum of subgames $G_1 + \ldots + G_n$.
2. Local combinatorial game search (LCGS): for each $G_i$, perform a search to find its game graph $GG(G_i)$.
3. Evaluation: for each game graph $GG(G_i)$, evaluate all terminal positions, then find the combinatorial game evaluation of all interior nodes, leading to the computation of $C(G_i)$.
4. Sum game play: through combinatorial game analysis of the set of combinatorial games $C(G_i)$, select an optimal move in $G_1 + \ldots + G_n$.

We describe steps 2 and 4 further in the following paragraphs. For further details on the theory, the algorithm, and its applications see (Berlekamp, Conway, & Guy 1982; Müller 1995; 1999).

**Local Combinatorial Game Search** Local combinatorial game search (LCGS) is the main information gathering step of decomposition search. It is performed independently for each subgame. LCGS generates a game graph representing all relevant move sequences that might be played locally in the course of a game. LCGS works differently from minimax tree search in a number of ways, including move generation and recognition of terminal positions.

The game graph built by LCGS also differs from the tree generated by minimax search. In the case of minimax, players move alternately, so each position is analyzed with respect to the player on move. In contrast, there is no player-to-move-next in a subgame. All possible local move sequences must be included in the analysis, including sequences with several successive moves by the same player, because players can switch between subgames at every move.

Another difference is the treatment of cycles. Classical combinatorial game evaluation is defined only for games without cycles. However, similar methods based on a technique called *thermography* are being developed that can deal with cyclic subgames as well (Berlekamp 1996; Fraser 2002; Müller 2000).

## Sum Game Play

To find an optimal move in a sum game, the final step of decomposition search selects a move which most improves the position. This improvement is measured by a combinatorial game called the *incentive* of a move. The incentives of all moves in all subgames are computed locally. If one incentive dominates all others, an optimal move has been determined. This is the usual case for games with a relatively strongly ordered set of values such as Go.

Since incentives are combinatorial games and therefore only partially ordered, it can happen that more than one non-dominated candidate move remains. In this case, an optimal move is found by a more complex procedure involving the combinatorial summation of games (Conway 1976).

Since such a summation can be an expensive operation, there is no worst case guarantee that decomposition search is always more efficient than minimax search. In practice, it seems to work much better. The algorithm presents many opportunities for complexity reduction of intermediate expressions during local evaluation as well as during summation.

Even though all search and most analysis is local, decomposition search yields globally optimal play, which can switch back and forth between subgames in very subtle ways, as in the example of Figure 3.

An optimal 62 move solution sequence computed by decomposition search is shown in Figure 3. On a state of the art system 4 years ago, the complete solution took only 1.1 seconds. Full-board alpha-beta search, even with further enhancements that exploit locality, has no chance to solve this kind of problems. It requires time that is exponential in the size of the *whole problem*, whereas LCGS' worst case time is exponential in the size of the *biggest subproblem*. If the local combinatorial game evaluations generated during LCGS can be computed and compared without too much overhead, as usually seems to be the case in the Go endgames investigated, a dramatic speedup results (Müller 2001a).

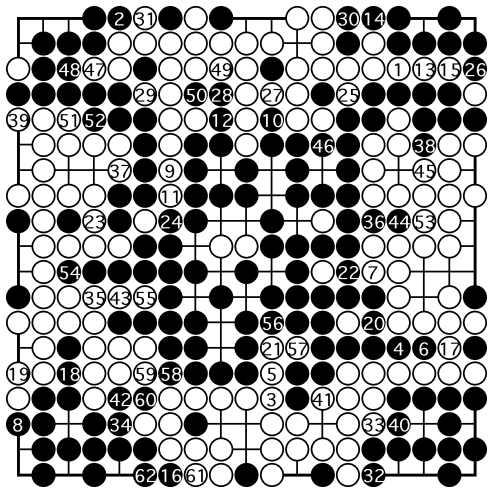

Figure 3: An optimal solution to problem C.11

## Game Search Techniques in Multicriteria Planning

This section presents some preliminary ideas for applying games methods in AI planning. The author hopes that they will become part of the emerging research agenda in multicriteria planning.

### Partial Order Bounding

The idea of using many simple yes/no questions to approach a complex problem is intuitively appealing. Can it be made to work in the domain of multicriteria AI planning? Many planning systems are slowed down by their manipulation of complex structures during the search. The challenge of developing a planner based on partial order bounding is to transform the planning problem into a series of decision problems that can be efficiently searched. One successful example of a similar approach are the existing methods for compiling planning problems into SAT instances.

### Decomposition Search

Combinatorial game theory uses one of the fundamental approaches for dealing with complexity: divide and conquer. The unique point of this approach is the rich partially ordered structure of combinatorial games that can be used on an intermediate level, to represent solutions to subproblems. Furthermore, a powerful mathematical apparatus can be used to combine partial solutions.

An analogous approach in AI planning could work as follows:

- Split a planning problem into subtasks.
- Use local search on each subtask, which is parameterized by the possible external contexts in which it might be applied.
- Find a partial order structure to represent the parameterized solutions to subtasks.
- Define a global combination operator, which might be based on a combination of: high-level search, and knowledge about the partial order structure of sub-solutions.

To give a more concrete example, the subtask of transporting some good $G$ from $A$ to $B$ might have a partially ordered solution space that is parameterized by the resources used, such as fuel, time and personnel. It can be further parameterized by the results achieved, such as the quantity of $G$ transported, the risk of failure and so on. The idea is a "plan library" with multiattribute annotations of subtasks and solutions.

## Multicriteria Planning in Games and Puzzles

Adversarial planning is more complex than single-agent planning, since normal planning usually assumes an unchanging environment under complete control of the agent, whereas in adversarial planning all possible hostile opponent actions have to be taken into account according to the minimax principle.

### Go

Go is an intricate game which requires a complex evaluation (Bouzy & Cazenave 2001; Müller 2002). Most successful Go programs utilize a complex hierarchy of objects to represent the state of a Go board, and very selectively generate moves that pursue goals related to these objects. The basic evaluation in Go is a scalar measuring the balance of *territory*, often obtained by summing up a value in the range between +1 (sure point for player) and -1 (sure point for the opponent) for all points on the board. Even so, in practice many other criteria are used to modify this value (Chen 2000).

In terms of planning, high-level plans for objects on the board are used. (Hu & Lehner 1997) propose several models for combining local plans in a Go framework, taking into account the overall minimax evaluation principle as well as the question of keeping the initiative while executing one plan, which allows a player to start on the next plan as well.

Multicriteria planning appears to be a natural framework for this kind of environment. A situation can be represented as a set of plans for each player. During a game, plans are in different stages of completion, and represent different degrees of local success or failure for each player. Each move played in a game of Go typically has effects on many levels and on many different plans. Some of these effects are good for the player, while others are detrimental. This naturally leads to a partial order evaluation structure.

### Planning in Sokoban

In ongoing work with Adi Botea and Jonathan Schaeffer (Botea 2002), we investigate an abstract model for planning in the puzzle of Sokoban (Junghanns 1999). This work roughly follows the decomposition search planning model outlined above. A Sokoban puzzle is split into subproblems called *rooms* and *tunnels* representing parts of the overall maze. Several static and search-based analyses are performed on the subproblems. This results in a compact intermediate representation of the possible local solutions to

each subproblem. The high-level global planner is now able to work on the much reduced abstracted planning problem instead of the original maze.

## References


Allis, L. 1994. *Searching for Solutions in Games and Artificial Intelligence*. Ph.D. Dissertation, University of Limburg, Maastricht.

Baum, E., and Smith, W. 1997. A bayesian approach to relevance in game playing. *Artificial Intelligence* 97(1-2):195–242.

Beal, D. 1999. *The Nature of Minimax Search*. Ph.D. Dissertation, Universiteit Maastricht.

Berlekamp, E.; Conway, J.; and Guy, R. 1982. *Winning Ways*. London: Academic Press. Revised version published 2001-2002 by AK Peters.

Berlekamp, E. 1996. The economist's view of combinatorial games. In Nowakowski, R., ed., *Games of No Chance: Combinatorial Games at MSRI*. Cambridge University Press. 365–405.

Botea, A. 2002. under submission.

Bouzy, B., and Cazenave, T. 2001. Computer Go: An AI-oriented survey. *Artificial Intelligence* 132(1):39–103.

Chen, K. 2000. Some practical techniques for global search in Go. *ICGA Journal* 23(2):67–74.

Conway, J. 1976. *On Numbers and Games*. London/New York: Academic Press.

Dasgupta, P.; Chakrabarti, P.; and DeSarkar, S. 1996a. Multiobjective heuristic search in AND/OR graphs. *Journal of Algorithms* 20(2):282–311.

Dasgupta, P.; Chakrabarti, P.; and DeSarkar, S. 1996b. Searching game trees under a partial order. *Artificial Intelligence* 82(1–2):237–257.

Fraser, B. 2002. *Computer-Assisted Thermographic Analysis of Go Endgames*. Ph.D. Dissertation, University of California at Berkeley. To appear.

Ginsberg, M. 2001. GIB: Imperfect information in a computationally challenging game. *JAIR* 14:303–358.

Harikumar, S., and Kumar, S. 1996. Iterative deepening multiobjective A*. *Information Processing Letters* 58(1):11–15.

Hu, S., and Lehner, P. 1997. Multipurpose strategic planning in the game of Go. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(9):1048–1051.

Junghanns, A. 1998. Are there practical alternatives to alpha-beta? *ICCA Journal* 21(1):14–32.

Junghanns, A. 1999. *Pushing the Limits: New Developments in Single-Agent Search*. Ph.D. Dissertation, University of Alberta.

Lincke, T. 2002. *?* Ph.D. Dissertation, ETH Zurich. To appear.

Müller, M. 1995. *Computer Go as a Sum of Local Games: An Application of Combinatorial Game Theory*. Ph.D. Dissertation, ETH Zürich. Diss. ETH Nr. 11.006.

Müller, M. 1999. Decomposition search: A combinatorial games approach to game tree search, with applications to solving Go endgames. In *IJCAI-99*, 578–583.

Müller, M. 2000. Generalized thermography: A new approach to evaluation in computer Go. In van den Herik, J., and Iida, H., eds., *Games in AI Research*, 203–219. Maastricht: Universiteit Maastricht.

Müller, M. 2001a. Global and local game tree search. *Information Sciences* 135(3–4):187–206.

Müller, M. 2001b. Partial order bounding: A new approach to evaluation in game tree search. *Artificial Intelligence* 129(1-2):279–311.

Müller, M. 2002. Computer Go. *Artificial Intelligence* 134(1–2):145–179.

Ore, O. 1962. *Theory of Graphs*, volume 38 of *Colloquium Publications*. Providence: American Mathematical Society.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison Wesley Publishing Company: Reading, Massachusetts. ISBN 0-201-05594-5.

Plaat, A.; Schaeffer, J.; Pijls, W.; and De Bruin, A. 1996. Best-first fixed-depth minimax algorithms. *Artificial Intelligence* 87:255–293.

Russell, S., and Norvig, P. 1995. *Artificial Intelligence: a Modern Approach*. Prentice Hall.

Stanley, R. 1997. *Enumerative Combinatorics Vol. 1*. Number 49 in Cambridge Studies in Advanced Mathematics. Cambridge University Press.

Stewart, B., and White, C. 1991. Multiobjective A*. *Journal of the ACM* 38(4):775–814.

Trotter, W. 1992. *Combinatorics and Partially Ordered Sets. Dimension Theory*. Baltimore: Johns Hopkins University Press.

von Neumann, J., and Morgenstern, O. 1947. *Theory of Games and Economic Behaviour*. Princeton University Press, Princeton, 2nd edition.

von Neumann, J. 1928. Zur Theorie der Gesellschaftsspiele. *Math. Ann.* 100:295–320.

Yanez, J., and Montero, J. 1999. A poset dimension algorithm. *Journal of Algorithms* 30:185–208.