Upgrading Checkers Compositions

Yaakov HaCohen-Kerner, Daniel David Levy, Amnon Segall
Department of Computer Sciences, Jerusalem College of Technology (Machon Lev)
21 Havaad Haleumi St., P.O.B. 16031, 91160 Jerusalem, Israel

Abstract: Computer checkers programs achieve outstanding results at playing checkers. However, no existing program can either compose or improve adequate checkers compositions. In this paper, we present a model that is capable of improving the quality of a part of the existing checkers compositions. In this model we attempt to improve a given composition by a series of meaningful checkers transformations using hill-climbing search, while satisfying several criteria at each step. This model has been implemented in a working system. The results of the experiment we carried out show that about half of the compositions have been improved.

Key-Words: Artificial Intelligence, Checkers, Complexity, Composition of Problems, Computer Checkers Programs, Computer Game Playing, Evaluation Function, Pruning, Transformations

1 Introduction

Over the years, games in general and checkers in particular have proven to be very interesting and attractive domains, whilst also proving fertile ground for techniques and ideas that later have been used in various domains of computer sciences in general and artificial intelligence in particular. For example, research concerning game theoretic approaches to multi-robot planning [5] and to robot tracking problem [7] have been presented.

However, to the best of our knowledge, there is no program able to either compose or improve adequate checkers compositions.

In this paper, we propose a model that is capable of improving a part of the existing checkers compositions. We formalize a major part of the knowledge needed for judging the quality of checkers compositions (8 x 8 draughts). The basic idea is to attempt to improve a given composition with respect to the difficulty to solve the composition. The improvement is performed by a series of meaningful checkers transformations using a hill-climbing search while satisfying various criteria, step by step, until no further adequate transformations are available.

This model has been implemented in a working system. It has been tested on 139 known checkers compositions taken from checkers web-sites [e.g.: 6 and 8]. The results show that about half of the compositions have been improved.

This paper is organized as follows: Section 2 gives background concerning history of checkers programs and composition of checkers compositions. Section 3 describes the proposed model. Section 4 presents the results of the experiment and analyzes them. Section 5 summarizes the research and proposes future directions.

2 Checkers Compositions

2.1 Checkers Programs History

The history of checkers starts about 3400 years ago, when it was known as "Alquerque" in ancient Egypt, played on a 5 x 5 board. During the centuries, checkers has been developed under different names, sizes of boards and rules for various countries.

The history of checkers programs starts on 1959. The first checkers program was programmed by Samuel [10 and 11]. His program was also a milestone in machine

learning. His method relies in part on the use of a polynomial evaluation function comprising a subset of weighted features chosen from a larger list of possibilities. The learning technique relies on an innovative self-learning procedure whereby one version that learns competed against another version. The loser was replaced with a deterministic variant of the winner by altering the weights on the features that were used.

The current computer world champion checkers program is *Chinook* [3, 12 and 13], developed at the University of Alberta by a team of researchers led by Schaeffer. It was the first computer checkers program to win the world man-machine checkers championship. The Chinook team produces databases which give the computer perfect knowledge for all positions with 10 pieces or less on the board of the form win/loss/draw. Chinook relies on a linear evaluation function that considers several features of the game board including: 1) piece count, 2) kings count, 3) trapped kings, 4) turn, 5) runaway checkers (unimpeded path to king). No machine learning methods have been employed successfully in the development of Chinook.

Chellapilla and Fogel [1 and 2] develop another interesting checkers program. In contrast to the two previous programs, they did not rely on look-up tables, perfect end game databases, grand master openings, or even features about checker positions that are believed to be important. Instead, they applied a coevolutionary learning process on generations of neural networks, where each network serves as an evaluation function to describe the quality of the current board position. Their program did not use any human expertise in the form of features that are believed to be important to playing well. After 840 generations, their process has generated a neural network that can play checkers at the expert level.

2.2 Checkers Compositions

Some of the checkers compositions are composed by human beings, and others are taken from actual play or analysis. In most of the compositions, White begins and wins. In the rest, White begins and draws.

However, to the best of our knowledge, there is no program able to either compose or improve adequate checkers compositions. In a related domain, chess, HaCohen-Kerner et. al. [4] have developed a model that has been implemented in a working system for improving two-move chess compositions.

In this research, we have developed a similar model for checkers. Checkers and chess resemble each other in many ways. Nevertheless, there are many differences between the two games. For example: checkers is played only on half of the squares (32) while chess is played on the whole board (64). Checkers has 2 kinds of pieces while chess has 6. In checkers a typical position without any captures has 8 legal moves, while in chess the average is about 35-40 moves. Additional differences can be found in [12 and 13].

There are also differences between checkers compositions and chess compositions. While in chess the most frequent compositions are mate in either two or three moves, in checkers most compositions are solved by a series of moves (usually much more three moves).

3 The model

The improvement process attempts to improve a given composition through by a series of meaningful checkers transformations, while satisfying several criteria at each step.

The transformations that we use were:

- Move of a piece from one square to another square
- Deletion of a piece from the board
- Addition of a piece to the board

The input composition is tested according to a tree search. Then, it is analyzed automatically by a special composition evaluator in order to find its quality-score.

After each transformation, we apply the following three checks: (1) It is a legal checkers position, (2) It is a winning composition and (3) The new composition

is of a higher quality than the original composition. In such a case, the new position is stored as an improvement and used as the current position.

In Fig. 1 we describe the main flow of the composition algorithm.

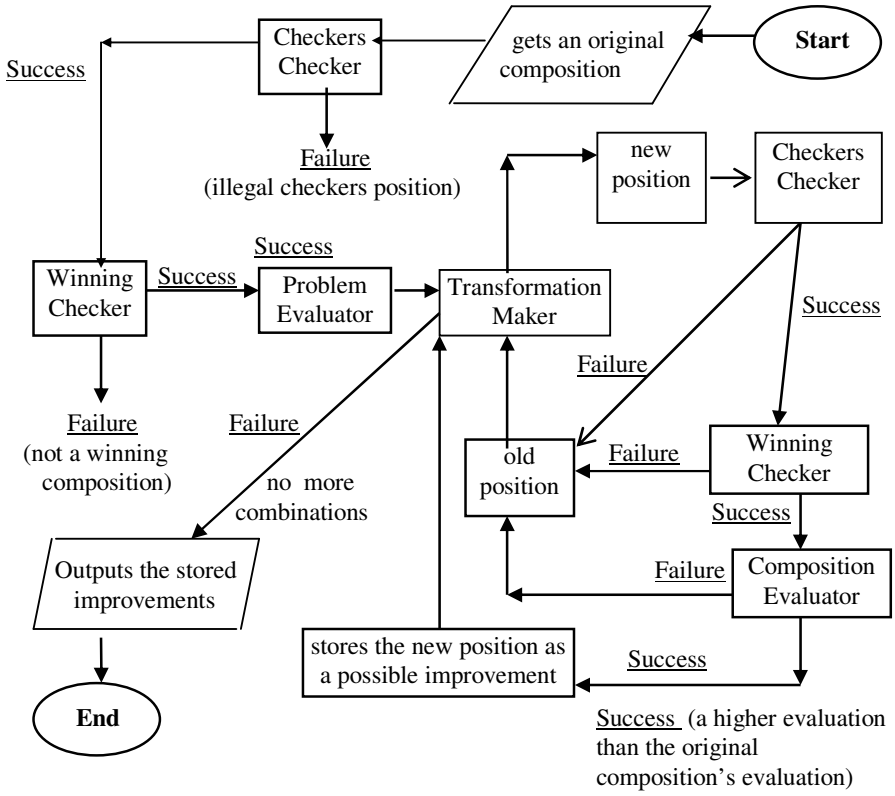


Fig. 1. Main flow of the composition algorithm

3.1 Complexity and Pruning

The number of possible checkers configurations is estimated as $5*10^{20}$ [12 and 13], while in chess the number of different legal positions is estimated at about 10^{40} [9]. However, not all are positions legally reachable from the starting position. Schaeffer et. al. [12 and 13] estimate that there are $O(10^{18})$ legal checkers positions. Theoretically, given a certain checkers composition, it is possible to check all these positions as candidates for improvements of the given composition. All these positions can be reached, for example, by using various deletion and addition transformations.

In practice, however, our model has overcome this combinatorial explosion by using pruning. The pruning is performed by the checks mentioned before. In this way, the number of positions the model has examined in any given composition has been reasonable to deal with.

The function for evaluating the quality mark of a composition is defined in Fig. 2. The concepts referenced in this function are defined later.

$$\begin{aligned}&Problem\_value = \\ &\sum_{pieces\}} \begin{pmatrix} (position * piece\_value) + Freedom\_level \\ + (Blocking\_Move * 3) + (Forcing * 9) \\ + Move\_Value + Piece\_Status \\ + \sum_{0}^{all-glasses} (glasses\_value) \end{pmatrix}\end{aligned}$$

$$where: Freedom\_level = Position + Move\_value + piece\_status$$

$$glasses\_value = \sum \begin{pmatrix} \sum(Player\_Friendly*2) + \sum(Player\_Half*3) \\ + \sum(Player\_Full*5) - \sum(Opponent\_Friendly*2) \\ - \sum(Opponent\_Half*3) - \sum(Opponent\_Full*5) \end{pmatrix}$$

Fig. 2. The evaluation function

Position 1 presents the common notation for the squares of a 8 x 8 draughts.

	32		31		30		29
28		27		26		25	
	24		23		22		21
20		19		18		17	
	16		15		14		13
12		11		10		9	
	8		7		6		5
4		3		2		1	

Position 1

For instance, squares 4 and 29 in position 2 get 6 points each because they are placed on the main diagonal.

Piece_value is defined as follows: A king has a value of three points, and a soldier has a value of one point.

Freedom_level is a parameter of "how free to move" is a piece before making the current move. This value is calculated by the future position of the piece, the value of the move, and the status of the piece after the move is made.

Blocking_move gets 3 points if the location of the piece is blocking the opponent from capturing a player's piece, and 0 if not.

Forcing occurs when we leave the opponent no choice other then making a specific move. Forcing gets 4 points.

Move_value checks the move we want to make, and gives it a value of 9 points to a move leading to forcing and 3 points to a move leading to a blocking_move.

Piece_status is defined in Table 1. The worst piece status is where the piece is threatened and not threatening. The best piece status is where the piece is threatening and not threatened.

Table 1. The piece status

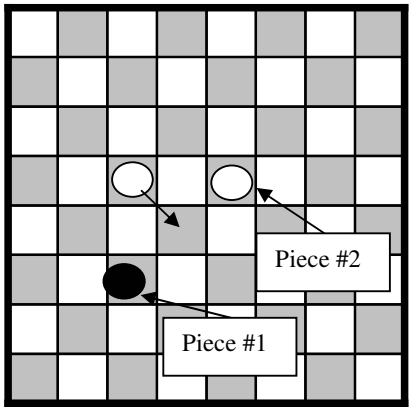
King	Soldier	Threatened	Threatening
- 3	- 1	V	X
+ 1	+ 1	V	V
+ 3	+ 2	X	V
0	0	X	X

Position 2 presents the values of the "position" feature for these squares.

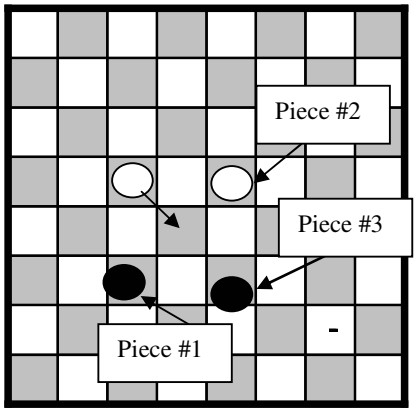
	4		4		4		6
4		3		3		3	
	2		2		2		3
3		1		1		1	
	1		1		1		3
3		2		2		2	
	3		3		3		4
6		4		4		4	

Position 2

Glasses_value: This value indicates the value of all the "Glasses" on the board. "Glasses" is a situation where a piece can enter a square that is between two other pieces of any type (White/Black, King/Soldier) as shown in position 3 (White's turn to play).



Position 3



Position 4

"Full Glasses" is a situation when there is Glasses and the third piece is an opponent piece that can capture or be captured, as shown in position 4.

The following flags indicate the kind of Glasses that exists:

- **Player_Friendly**: A player's piece is between 2 player's pieces.
- **Player_Half**: A player's piece is between 2 mixed pieces (one is a player's piece and the other is an opponent's piece).
- **Player_Full**: A player's piece is between 2 opponent's pieces, and can capture or be captured by a third opponent's piece.
- **Opponent_Friendly**: Same as Player_Friendly, but for the opponent side.
- **Opponent_Half**: Same as Player_Half, but for the opponent side.
- **Opponent_Full**: Same as Player_Full, but for the opponent side.

4 Results

We have tested our model on one hundred thirty nine real compositions. Most of the compositions were collected from relevant checkers web-sites [e.g.: 6 and 8].

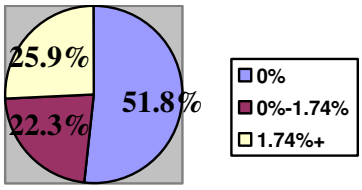


Fig. 3. The results

Fig. 3 describes the general results of our system: unimproved compositions, improved compositions in a rate less than the standard deviation of the improvements and improved compositions in a rate greater than the standard deviation. The results in detail are as follows:

- Total compositions:139
- Average improvement: 1.347%
- The Variance of the improvements is 3.03%
- The standard deviation of the improvements is 1.74%
- 72 compositions have not been improved (51.8%)
- 31 compositions have been improved by 0%-1.74% (22.3%)
- 36 compositions have been improved by more than 1.74% (25.9%)
- Total 67 compositions that have been improved (48.2%)
- Range of improvements is 0%-15% of the quality-scores of the original compositions
- Total changes made: 321
- Average number of changes for each composition: 2.31

Table 2 presents the distribution of the successful transformations leading to improvements.

Table 2. Distribution of the transformations leading to improvements

Transformation	# of improvements
<1>	129
<2>	162
<3>	1
<4>	2
<5>	27

A legend for the transformations included in Table 2 is given below:

<1> Move to Min.: Moves a piece to a square where the composition's value would minimally increase.

<2> Move to capture without being captured: Moves a piece in order to capture an opponent piece, without being threatened.

<3> Move to max.: Moves a piece to a square where the composition's value would maximally increase.

<4> Add to capture, can be captured: Add a piece to a square where it can capture an opponent piece, but also can be threatened.

<5> Add to capture without being captured: Add a piece to a square where it threats an opponent piece, without being threatened.

As shown in Table 2, the most

effective changes were: '<2> Move to capture without being captured.' and '<1> Move to Min.'. Both changes are made on White pieces in order to lower the composition's value to the White player. Apparently, it is more effective to lower the composition's value for White, than to raise the composition's value for Black.

5 Summary and Future Work

We have developed a model that is capable of improving the quality of a part of the existing checkers compositions. The results of the experiment we have made show that about half of the examined compositions have been improved.

In computer game-playing, high levels of playing have proven inefficient without deep searching. We believe that this is also true in order to achieve a high level in composing checkers compositions. Therefore, adding a more complex searching technique rather than using a hill-climbing search would further enhance our model. That is, in order to find the best improvements, we would need to allow the application of transformations even when the last tested transformations do not satisfy the necessary criteria. In this way, after making a set of several transformations, we may reach better and more complex improvements.

Another idea is to evaluate the potential of our system as an intelligent support system for weak and intermediate human composers. Application of this idea over time, on the one hand, can improve our system's strength and on the other hand, can improve these composers' performance.

References:

[1] Chellapilla K. and Fogel D. B.: *Anaconda Defeats Hoyle 6-0: A Case Study Competing an Evolved Checkers Program against Commercially Available Software*. Proc. of CEC, 2000, pp. 857-863.

[2] Chellapilla K. and Fogel D. B.: Evolving an Expert Checkers Playing Program without Using Human Expertise. *IEEE Trans. Evolutionary Computation*, Volume 5, Number 4, 2001, pp. 422-428.

[3] Chinook: http://www.cs.ualberta.ca/~chinook, 2004.

[4] HaCohen-Kerner Y., Cohen N., and Shasha E.: An Improver of Chess Compositions. *Cybernetics and Systems*. 30, 5, 1999, pp. 441-465.

[5] Galuszka A. and Swierniak A.: Game Theoretic Approach to Multi-Robot Planning. *WSEAS Transactions on Computers*, Issue 3, Volume 3, July 2004, pp. 537-542.

[6] Jetten H.: http://www.xs4all.nl/~hjetten/damnl.html#favlinks, 2004.

[7] Lucatero C. R., De Albornoz Bueno A. and Lozano Espinosa R.: A Game Theory Approach to th Robot Tracking Problem. *WSEAS Transactions on Computers*, Issue 4, Volume 3, Oct 2004, pp. 862-868.

[8] Lyman A.: http://www.acfcheckers.com/origin.html, 2004.

[9] Nievergelt, J.: Information Content of Chess Positions. *ACM SIGART Newsletter* 62, 1977, pp. 13-14.

[10] Samuel, A. L.: Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3, 3, 1959, pp. 211-229.

[11] Samuel, A. L.: Some Studies in Machine Learning Using the Game of Checkers II- Recent Progress. *IBM Journal of Research and Development*, 11, 6, 1967, pp. 601-617.

[12] Schaeffer J., *One Jump Ahead: Challenging Human Supremacy in Checkers*, Springer, Berlin, 1997.

[13] Schaeffer J., Culberson J., Treloar N., Knight N. B., Lu P. and Szafron D.: A World Championship Caliber Checkers Program, *Artificial Intelligence*, Volume 53, Number 2-3, 1992, pp. 273-290.

Acknowledgment:

Thanks to Reuven Gallant for valuable comments.