Deep Improver of Two-move Chess Mate Problems

Yaakov HaCohen-Kerner 1,2, Fridel Fainshtein 2

1Department of Computer Sciences, Jerusalem College of Technology (Machon Lev)
21 Havaad Haleumi St., P.O.B. 16031, 91160 Jerusalem, Israel, kerner@jct.ac.il
2Department of Computer Sciences, Bar-Ilan University
52900 Ramat-Gan, Israel, fainshf@cs.biu.ac.il

Abstract. Computer composition of high-quality chess mate problems is relatively an uninvestigated research domain. A previous model, called an Improver of Chess Problems, which is based on a hill-climbing search, improved slightly the quality of 10 out of 36 known problems (about 28%). In this paper, we describe an improved model, called Deep Improver of Chess Problems. This model uses an improved version of Bounded Depth First Search. The experiment, we carry out on the same problems, shows that the quality of 32 problems (about 89%) has been improved. Moreover, some of the improvements are meaningful and impressive.

1. Introduction

More and more people are involved in developing chess playing programs. Computer chess programs attain the highest level at chess playing, the grandmaster rank. However, computer composition of high-quality chess mate problems is relatively an uninvestigated research domain. A quality of a chess problem can be measured according to various chess composition features included in the problem.

A previous model, called an Improver of Chess Problems (ICP) (HaCohen-Kerner et al. 1999), improved the quality of 10 out of 36 known two-move mate problems (about 28%). ICP tries to improve the quality of a given problem by a series of transformations. It uses a hill-climbing search satisfying several chess composition criteria at each step. The achieved improvements are considered as slight ones.
In this research, we describe an improved model, called Deep Improver of Chess Problems (DICP). This model is based partially on some components of ICP. Instead of using a hill-climbing search, DICP uses an improved version of bounded depth first search that prevents re-exploration of the same positions. The prevention is done by using a fixed ordered list of transformations. In addition, the definitions of several composition themes have been updated. In most cases, the new definitions were stricter.

The results of the experiment, we carried out on the same 36 problems, show that the quality of 32 out of 36 known problems (about 89%) has been improved. Moreover, some of the improvements are meaningful and impressive.

This paper is organized as follows. Section 2 gives background concerning the composition of chess problems. Section 3 presents what has been done concerning computer composition of chess-like and chess mate problems. Section 4 describes the DICP model. Section 5 illustrates two examples. Section 6 presents the results of the experiment and analyzes them. Section 7 summarizes the research and proposes future directions.

2. Composition of Chess Mate Problems

Orthodox chess mate problems are problems that rely absolutely on the rules of the chess game. In this type of problems, White has to mate Black in a limited number of moves against any defense of Black.

This kind of problems is the most investigated one by human composers and the most closely associated with the classical chess game. As in the previous model, ICP, we deal with the most composed kind of chess problems, the two-move mate problems.

Two-move mate problems have an advantage over longer problems from the aspect of the number of variants, because they usually show more variants leading to different and interesting mate moves by White. However, in modern two-move problems there is an
increasing tendency to compose problems with complex themes, which usually limits the variety of possible solutions.

3. Computer Composition of Chess-like and Chess Mate Problems

Noshita (1991), Hirose et al. (1997) and Watanabe et al. (2000) investigate the Tsume-Shogi domain, which means composition of new problems in a very popular Japanese chess-like game. The main difference between this kind of problems and chess problems is that in Tsume-Shogi each move of the attacker must be a check and the only theme is the number of moves in a problem. That is, the solution variants are relatively forced and therefore limited.

Noshita (1991) generates Tsume-Shogi problems using a random algorithm. The generated problems contain solution-variants with length of 13-19 plies. The main idea of his method was to apply several meaningful transformations, such as "deletion of a piece" and "replacement of a stronger piece by a weaker piece", on random positions and to filter them according to Tsume-Shogi's criteria.

Hirose et al. (1997) use a reverse/backward method (called also retrograde analysis) to generate Tsume-Shogi problems. His method is an extension to the Thompson's retrograde analysis (1986) and similar to Shlosser's method (1988, 1991). The main idea of the reverse method is to start from a known problem and to move backward by doing all possible legal moves in order to reach new problems with a higher number of moves to mate.

In the domain of chess endgames, Grandmaster Nunn (1993) published some endgame studies. He used computer-generated endgame databases for analyzing endgames and discovering new interesting endgames. Nunn describes different ways of using this software as follows: checking and correcting analyzed endgames, analyzing over the board endgames, exploring endgames to extend theory, discovering general rules which govern a certain type of ending, and forming key-positions which a human player can memorized. Thompson (1986,
1996) uses a retrograde analysis to build chess endgames database with no more than 5 pieces on the board. Nalimov (2000) builds more efficient endgames database. Thompson's and Nalimov's databases are in use by many strong chess programs.

Shlosser (1988, 1991) proposes a general method of using retrograde analysis for composition of chess problems, as follows: (1) constructing a complete database of new positions starting from given mate positions (potential new problems), (2) eliminating all incorrect positions according to chess composition rules, and (3) selecting high-quality chess problems based on evaluation values. Watanabe et al. (2000) summarize the ideas of the reverse methods and the algorithmic-generation methods combining them into a new computer composition approach.

The main disadvantage of the reverse method is that it does not investigate all possible potential problems. That is, positions that cannot be reached from the starting problems by forced legal variants will not be investigated. Thus, this method not necessarily leads to a global optimum and might miss many high-quality problems.

4. The Deep Improver of Chess Problems (DICP) Model

A major part of the knowledge needed for evaluating the quality of chess problems in general and of two-move problems in particular has been defined in ICP with the help of two international masters in composition of chess problems. This knowledge includes, among other things, definitions of terms, themes, bonuses, and penalties in the domain of chess composition for two-move problems (HaCohen-Kerner et al., 1999). The majority of this knowledge was collected from relevant composition books and correspondence (Haymann, 1988-1991; Howard, 1961; Howard, 1962; Howard, 1970; Harley, 1970).

In order to find the best improvement(s) for each problem, we need to investigate all possible close positions. Therefore, we use a brute-force search method to generate these
positions. All regular brute-force search methods have a time complexity of $O(b^d)$, where $b$ represents the branching factor and $d$ represents the depth of the generated positions tree. This depth is the number of the applied changes/transformations starting from an original problem.

In our model, practically, $b$ is on average 448. The brute-force search method runs for full 3 levels on average 40 minutes, while for full 4 levels it runs for some problems about 48 hours. Therefore, $d$ was chosen to be 3 as our fixed depth. That is, we traverse all positions in the first 3 levels to find the best improvement (if exist).

### 4.1. General Description

Our model uses a Bounded Depth First Search (BDFS) in order to find the best improvement(s) for a depth of 3 levels. Among all other brute-force search methods, BDFS is the faster and the cheaper way to do it, due to the following reasons (for more details see (Russell and Norvig, 2002) and (Korf, 1985)):

1. BDFS requires a storage of only $O(bd)$ nodes while Breadth First Search (BFS) requires a storage of $O(b^d)$ nodes.
2. For problems that might have most improvements at deeper levels (like in our application; see Table 2) DFS may actually be faster than BFS because it has a good chance finding many improvements after exploring a small portion of the whole space.
3. BDFS develops fewer nodes than Depth First Iterative Deepening (DFID) when both methods run to a fixed depth.
4. In general, DFID is the preferred brute-force search method when there is large search space and the depth of the solution is not known. This is not our situation, since we are limited to a bounded depth of 3 levels.

A general description of our model is presented below. Other components will be explained shortly afterwards.
BoundedDepthFirstSearch (OriginalProblem, MaxDepth)
1. PushIntoStack ({OriginalProblem, 0})  // 2nd parameter is for the node's level in the tree
2. While  NotEmptyStack ()
   2.1.  {CurrentPosition, Level} ← PopFromStack()
   2.2.  if  ImprovedMateProblem (CurrentPosition, OriginalProblem) then
          2.2.1. StoreImprovedMateProblem (CurrentPosition)
   2.3.  if (Level < MaxDepth) then
          2.3.1. For each Successor of CurrentPosition (from right to left)
                  2.3.1.1. CurrentPosition = ApplyNextTransformation (CurrentPosition)
                  2.3.1.2. PushIntoStack ({CurrentPosition, Level + 1})

ImprovedMateProblem (CurrentPosition, OriginalProblem)
if  ( LegalChessPosition (CurrentPosition) and
       LegalTwoMoveMateProblem (CurrentPosition) and
       ProblemEvaluator (CurrentPosition) > ProblemEvaluator (OriginalProblem))
then return true
else return false

Other components:

- LegalChessPosition: checks the legality of a position according to the chess rules.
- LegalTwoMoveMateProblem: tests whether a given position is a two-move mate problem
  (including the test for non-cooking, i.e., no more than one keymove). This component uses
  a suitable limited search engine and checks all legal chess moves including the two special
  moves (in contrast to ICP): castling and en passant capture.
- ApplyNextTransformation: applies the next transformation on a given position according
  to a fixed ordered list of transformations.
- Problem Evaluator: analyzes a position as a mate problem and computes its quality score.

The heuristic function that computes the quality square of a position is defined as follows:

\[
q_m = \begin{cases} 
0 & \text{Severe deficiency} \\
\sum_i V(T_i) + \sum_j V(B_j) - \sum_k V(P_k) & \text{otherwise} 
\end{cases}
\]
Where $q_m$ is the quality square, severe deficiency is considered as a cancel of a position as a proposed problem. It is defined when one of the following situations happens:

- Illegal chess position
- It is not a two-move mate problem
- There is more than one keymove (cooked)

$V$ is the value function, $T_i$ is the set of all themes included in the position, $B_j$ is the set of all bonuses granted to the position and $P_k$ is the set of all penalties granted to the position.

### 4.2. Transformations

DICP uses three kinds of transformations, while attempting to improve a problem: deletion of a specific piece, addition of a specific piece and transparency of all pieces:

a. Deletion of a specific piece from the board

The idea behind this transformation is a known chess motive that a problem with fewer pieces expressing the same ideas is of a higher quality.

b. Addition of a specific piece to the board

The idea is that a problem with more pieces can present more themes and bonuses and therefore it will have a higher quality.

c. Transparency of all pieces

This transformation transfers all pieces through two possible movements: (a) file-transparency: all pieces are transferred i files to the right-side or to the left-side and (b) rank-transparency: all pieces are transferred i ranks to the upper-side or to the lower-side.

Additional transformations (e.g.: moving a certain piece, exchanging a piece) are introduced by (Kerner, 1995 and HaCohen-Kerner et al., 1999). These transformations can be seen as complex transformations, based on the basic transformations: deletions and additions. Each such transformation can be presented by deletion(s) and addition(s). For instance, moving a piece from $d4$ to $c4$ can be achieved by deletion the piece on $d4$ and addition of the
same piece on c4.

DICP, in contrast to ICP, uses only three kinds of transformations. Therefore: (1) the branching factor of DICP is smaller and (2) the depth of the tree (the number of the applied transformations) developed by DICP should be higher.

ICP uses a hill-climbing search based on a heuristic function to find local maxima in the near problem space. Hill-climbing search is an informed search. That is, it uses information of the newly developed nodes (Russell et al. 2002) in order to prune some of the positions.

In contrast, DICP uses BDFS. This is an uninformed search. That is, it does not depend on any information contained in the newly developed nodes. This approach overcomes the main weakness of ICP, which is not developing the sub-trees whose roots are:

1. not legal according to the rules of chess composition.
2. not a two-move mate problem with only one keymove.
3. of the lower quality score than the original problem.

DICP develops all possible transformations on all levels. That is, it traverses all nodes to a fixed depth in order to find the best improvement (global maximum). To prevent repetition of the same positions by different order of transformations, DICP uses a fixed ordered list of transformations. That is, each transformation is applied according to a fixed order on (1) a specific transformation (addition, deletion and transparency), (2) a specific piece (pawn, knight, bishop, rook, queen and king) or pieces, (3) a specific color (Black and/or White), and (4) a specific square or ranks or columns.

The main differences between DICP and other previous models related to computer composition of chess-like and chess mate problems (mentioned in Section 3) are:

1. DICP uses a move-forward technique from the given problem instead of using either move-backward or random techniques.
2. DICP tries all possible transformations for every examined position, not only forced
legal transformations. Therefore, its branching factor is much higher.

3. DICP starts with 2-movers and ends with them. There is no change in the number of the moves.

4. DICP deals with a relatively high number of composition themes.

4.3. Complexity of DICP

Shannon (1950) estimated the number of different legal chess positions to be about $10^{43}$. Shannon reached this estimation by the following calculation:

$$\frac{64!}{(32! \times (8!)^2 \times (2!)^6)}$$

which is more exactly about $4.63 \times 10^{42}$. The explanation for this combinatorial calculation is as follows: there are 64 squares on the chessboard, 32 different pieces, 8 White pawns, 8 Black pawns, 6 different groups of two identical pieces (two rooks, two bishops and 2 knights for both sides) and 4 groups of one piece each (king and queen for both sides) which do not imply on the denominator.

This calculation takes into consideration only possible arrangements of the pieces on the board. However, there are positions which are not legal (e.g. the two kings are in near squares; both kings are checked, etc.). Therefore, chess problemists and mathematicians (Nievergelt, 1977) estimate the number of different legal chess positions to be $10^{40}$.

To overcome this combinatorial explosion, ICP uses a straight-forward hill-climbing search. However, this AI technique has a well-known disadvantage, which means that it not necessarily leads to a global optimum and might miss many high-quality problems. Therefore, only a small part of the problems (about 28%) have been improved. Moreover, the improved problems contain only rather slight and unimpressive changes.

In contrast, DICP uses an improved version of BDFS. To reduce the complexity of the algorithm, DICP prevents repetition of the same positions, which can be reached by different order of transformations. The prevention is fulfilled by using a fixed ordered list of
transformations, as explained before.

The minimal number of pieces for a specific problem (for the discussed database) is 7 pieces. For such a position, there are $64 - 7 = 57$ empty squares. A maximum of ten kinds of chess pieces (queen, rook, bishop, knight, and pawn for both sides) can be added to this position. Thus, the maximal number of additions to the given position is $57 \times 10 = 570$. In addition, there are 7 possible deletions, 7 possible file-transparencies and 7 possible rank-transparencies. Therefore, the maximal number of possible transformations on the first level of the transformations tree (developed from a given problem) is 591.

Chess composition rules do not allow a chess problem to contain more than one king, one queen, two rooks, two bishops, two knights, and eight pawns for each color (Harley 1970, Howard 1961). Therefore, practically, the number of transformations on the first level depends on the given problems. For our database the maximal number of transformations on the first level is 527 and the minimal number is 305. On average, we have 448 transformations on the first level, as shown in Table 1. On the second level, for most of the nodes (all nodes that contain an addition transformation), the number of maximal potential son-nodes decreased since one square has been occupied and one kind of piece has been used.

BDFS has a time complexity of $O(b^d)$, where $b$ represents the branching factor (448 on average) and $d$ (3 in our application) represents the depth of the tree, which is the number of the applied transformations. Many positions are repeated by permutations of the transformations. DICP prunes these permutations by using a fixed ordered list of transformations. Each list containing the same $d$ transformations leads to the same position. That is, the same position can be reached in $d!$ different permutations. Thus, actually, on the level $d$ we need to explore only $O(b^d/d!)$ nodes. The results of table 1 support this theory. For instance for $d=1$ the number of generated positions on average is 448 ($= 448^1/1!$); for $d=2$ the number of generated positions on average is 96,066, upper bounded by $448^2/2!$ ($= 100,352$)
and for \(d=3\) the number of generated positions on average is 12,499,298, upper bounded by \(448^3/3! = 14,985,898\). The running-time needed for investigating all generated nodes in the first 3 levels for an average problem is approximately 40 minutes.

Table 1. Average number of generated positions and running-time by DICP for all 36 original problems

<table>
<thead>
<tr>
<th># of problems</th>
<th># of generated positions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-level</td>
</tr>
<tr>
<td><strong>Average # of nodes</strong></td>
<td>448</td>
</tr>
<tr>
<td><strong>Deviation (nodes)</strong></td>
<td>38</td>
</tr>
<tr>
<td><strong>Average time (sec.)</strong></td>
<td>0.13</td>
</tr>
<tr>
<td><strong>Deviation (sec.)</strong></td>
<td>0.06</td>
</tr>
</tbody>
</table>

5. Examples

In this section, we annotate two mate-problems that have been improved by DICP. The first one was also improved by ICP. The second problem has been improved by DICP only.

5.1. Example no. 1

The first example is the original miniature presented in Position 1. The composition theme expressed in this problem is “self-blocking”, which means that Black closes one of his king-flights by at least one of his moves, enabling White to mate him.
The solution to Position 1 is the key-move Bishop a5 - d8. Then there are four variants, which can be derived from the four possible answers of the Black. Variant (d) expresses the self-blocking theme:

**Black’s move**  | **White’s mate-move**  | **Themes included in the variant**
--- | --- | ---
(a) 1... King  a4 - b5.  2. Bishop  e6 - d7.
(b) 1... King  a4 - a3.  2. Rook  b5 - a5.
(c) 1... Pawn  a6 - a5.  2. Rook  b5 - a5.
(d) 1... Pawn  a6 : b5.  2. Rook  c8 - a8.  self-blocking

ICP, while trying to improve Position 1, applies the following transformation “taking off the White bishop on a5”. As a result, we reach Position 2. The solution to Position 2 is: the key-move Bishop e6 - d7. There are three possible variants then:

**Black’s move**  | **White’s mate-move**  | **Themes included in the variant**
--- | --- | ---
(a) 1... King  a4 - a3.  2. Rook  b5 - a5.
(b) 1... Pawn  a6 - a5.  2. Rook  b5 - b3.  self-blocking, direct-battery fired
(c) 1... Pawn  a6 : b5.  2. Rook  c8 - a8.  self-pinning

The deletion of the White bishop on a5 leads to omission of the first variant for the problem in Position 1. Therefore, Position 2 has two main disadvantages in comparison to
**Position 1:**

1. Loss of the nice mate achieved in the mentioned variant.

2. Loss of the Black’s king-flight 1... King a4 - b5 in the same variant.

However, **Position 2** is considered to be better than **Position 1** for four main reasons:

1. The same composition theme (self-blocking) is expressed by a smaller problem (with one less bishop to White, the strongest side!).

2. In contrast to the solution-variants in **Position 1**, all solution-variants in **Position 2** include different mate-moves. That is, all variants are rather different and each of them contributes a novelty by its mate-move.

3. In addition, in **Position 2** we achieved two new composition themes: (1) “self-pinning” (i.e., The Black makes a move and pins the Black king) in the third variant when the Black makes a move 1... Pawn a6 : b5 and pins his king; and (2) "direct-battery fired" (i.e., A battery with a White piece in the middle of the battery where a battery is defined as follows: a piece is standing between a long-range piece (queen, rook or bishop) and a king) in the second variant when the Black makes a move 1... Pawn a6 - a5 and the White mate with Bishop on d7 because of the move 2. Rook b5 - b3.

4. Moreover, after making the key-move for **Position 1**, White has a mate-threat 2. Rook b5 - a5. In **Position 2** White has no mate-threat yet succeeds in mating Black in two moves. This feature is called “tempo”, which is also considered to be an additional composition theme.

**DICP** finds a better improvement to **Position 1**. It applies the 3 following transformations: “taking off the White bishop on a5”, “addition of a White bishop on f8”, and “addition of a Black pawn on d6”. As a result, we reach **Position 3**. The solution to **Position 3** is the same as in **Position 2**: the key-move Bishop e6 - d7. There are four possible variants:
<table>
<thead>
<tr>
<th>Black’s move</th>
<th>White’s mate-move</th>
<th>Themes included in the variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 1... Pawn d6 - d5. 2. Rook b5 - d5.</td>
<td>direct battery fired</td>
<td></td>
</tr>
<tr>
<td>(b) 1... Pawn a6 - a5. 2. Rook b5 - b3.</td>
<td>self-blocking, direct battery fired</td>
<td></td>
</tr>
<tr>
<td>(c) 1... Pawn a6 : b5. 2. Rook c8 - a8.</td>
<td>self-pinning</td>
<td></td>
</tr>
<tr>
<td>(d) 1... King a4 - a3. 2. Rook b5 - a5.</td>
<td>self-pinning</td>
<td></td>
</tr>
</tbody>
</table>

Position 3 is considered to be much better than Positions 1 and 2. Since above we have compared Positions 1 and 2, now we shall compare Positions 2 and 3. Position 3 has many advantages in comparison to Position 2, as follows:

1. Position 3 has 4 different variants in contrast to Position 2 that has only 3 variants.
2. While in Position 3 all variants are thematic (express themes), in Position 2 one variant is non-thematic.
3. In Position 3, there are 2 variants that include the theme “self-pinning” and 2 variants that include the theme “direct battery fired”, while in Position 2 there is only one variant for each theme.
4. While Position 2 is achieved after only one transformation, Position 3 is achieved after 3 transformations (can be regard as 2 because the deletion of the White Bishop and its addition on another square can be regarded as one transformation).

The disadvantages of Position 3 in comparison with Position 2 are:

Position 3 includes an additional White Bishop (which returns us to the original position) than Position 2, but also an additional black pawn. Therefore Position 3 is not considered a miniature (a problem which contains at most 7 pieces).

5.2. Example no. 2

The second example is the miniature presented in Position 4. The composition themes expressed in this problem are: “Tempo/Waiting move”, “self-blocking” and “king flights”.
The solution to Position 4 is the key-move: kNight f2 – e4. Then, there are 4 possible variants, only 3 of which express “self-blocking”.

<table>
<thead>
<tr>
<th>Black’s move</th>
<th>White’s mate-move</th>
<th>Themes included in the variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 1... Pawn e7-e6. 2. Bishop d7 – c6.</td>
<td>self-blocking</td>
<td></td>
</tr>
<tr>
<td>(b) 1... Pawn e7-e5. 2. Queen c3 – d3.</td>
<td>self-blocking</td>
<td></td>
</tr>
<tr>
<td>(c) 1... Pawn e7-d6. 2. nKnight e4 – f6.</td>
<td>self-blocking</td>
<td></td>
</tr>
<tr>
<td>(d) 1... King d5- e4. 2. Bishop d7 – c6.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The problem presented in Position 4 has not been improved by ICP. However, it has been improved significantly by DICP to Position 5. The solution to Position 5 is the same key-move: kNight f2 – e4. Then, there are 5 possible variants, 4 of them an additional theme “pickaniny”, which is considered as a rather complex theme. The “pickaniny” theme means that there are 4 different variants for the same Black pawn and for each one of them there is another mate-move by White. The 5 possible variants are:

<table>
<thead>
<tr>
<th>Black’s move</th>
<th>White’s mate-move</th>
<th>Themes included in the variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 1... Pawn e7-e6. 2. Bishop d7 – c6.</td>
<td>self-blocking; pickaniny</td>
<td></td>
</tr>
<tr>
<td>(b) 1... Pawn e7-e5. 2. Queen c3 – d3.</td>
<td>self-blocking; pickaniny</td>
<td></td>
</tr>
</tbody>
</table>
(c) 1... Pawn e7-d6.  2. King g5 – f4.  direct battery fired; pickaniny
(d) 1... Pawn e7-f6.  2. Knight g8 – f6.  pickaniny
(e) 1... King d5- e4.  2. Bishop d7 – c6.

Position 5 is considered to be better than Position 4 for the following main reasons:

1. Position 5 includes two new composition themes: “pickaniny” and “direct battery fired”.

2. In addition, Position 5 contains one variant that includes these two new themes and the mate move is done by the White King!

3. Position 5 contains one additional thematic variant. That is, Position 5 contains 4 variants out of 5 which are thematic (80%), while Position 4 contains only 3 variants out of 4 which are thematic (75%).

There are two slight disadvantages of Position 5 comparing to Position 4:

1. Position 4 includes only 7 pieces (i.e. it is regarded as a miniature), while Position 5 is not a miniature any more.

2. Position 4 includes 3 variants that express the theme “self-blocking”, while Position 5 includes only 2 variants that express this theme.

6. Results

DICP has been tested on the same 36 problems as in ICP. Most of the problems were collected from relevant composition books and correspondences (Haymann, 1988-1991; Howard, 1961; Howard, 1962; Howard, 1970; Harley, 1970). Each problem included at least one theme from the themes that have been defined in (HaCohen-Kerner et al., 1999).

Table 2 presents the number of generated improvements by DICP for all 36 original problems. On average, for each problem 6, 429 and 29422 improvements were found on the 1, 2 and 3 levels respectively. Most of the improvements were found on the 3 level.
Therefore, it is naturally that the best improvement for each problem (if exist) will be found on the 3 level, as shown in Table 3.

**Table 2.** Number of all generated improvements by DICP for all 36 original problems

<table>
<thead>
<tr>
<th># of problems</th>
<th># of improvements</th>
<th>Total # of improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-level</td>
<td>2-level</td>
</tr>
<tr>
<td>Average # of nodes</td>
<td>6</td>
<td>429</td>
</tr>
<tr>
<td>Deviation (nodes)</td>
<td>9</td>
<td>701</td>
</tr>
</tbody>
</table>

**Table 3.** Rate of improved problems in ICP and DICP (best improvement for each problem)

<table>
<thead>
<tr>
<th># of improved problems</th>
<th>Best improvement after exactly</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 transformation</td>
</tr>
<tr>
<td>ICP</td>
<td>#</td>
</tr>
<tr>
<td></td>
<td>%</td>
</tr>
<tr>
<td>DICP</td>
<td>#</td>
</tr>
<tr>
<td></td>
<td>%</td>
</tr>
</tbody>
</table>

Table 3 presents the Rate of improved problems in ICP and DICP (for each problem only the best improvement if found). While ICP improves only 10 out of 36 problems (about 28%), DICP improves 32 out of 36 problems (about 89%). Most of the best improvements achieved by ICP occurred after only one transformation and no problem has been improved after three transformations. In contrast, 87.5% of DICP's best improvements were achieved after three transformations.

Table 4 presents other interesting statistics concerning the performance of DICP. The clearest finding is that an addition of pieces to a certain extent adds quality. There is an
average addition of 1.66 pieces (increases from 7.31 to 8.97, 22.7%) in every improved problem. This causes to: (1) the average number of themes increases from 2.25 to 3.09 (37.4% of improvement), (2) the average number of unique thematic variants increases from 1.69 to 2.22 (31.5% of improvement), and (3) the average proportional rate of the thematic variants increases from 0.43 to 0.52 (21.6% of improvement).

Table 4. Statistics concerning the performance of DICP (all improved problems)

<table>
<thead>
<tr>
<th></th>
<th># of pieces</th>
<th># of themes</th>
<th># of unique thematic variants</th>
<th>proportional rate of thematic variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) average number for original problems</td>
<td>7.31</td>
<td>2.25</td>
<td>1.69</td>
<td>0.43</td>
</tr>
<tr>
<td>Deviation for A</td>
<td>0.96</td>
<td>0.69</td>
<td>1.01</td>
<td>0.31</td>
</tr>
<tr>
<td>(B) average number for original problems</td>
<td>8.97</td>
<td>3.09</td>
<td>2.22</td>
<td>0.52</td>
</tr>
<tr>
<td>Deviation for B</td>
<td>1.41</td>
<td>0.87</td>
<td>1.13</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Table 5. Distribution of the improving transformations.

<table>
<thead>
<tr>
<th></th>
<th>add</th>
<th>del</th>
<th>add-add-</th>
<th>del-add-</th>
<th>add-add-add</th>
<th>trans-add-</th>
<th>trans-del-</th>
</tr>
</thead>
<tbody>
<tr>
<td># of problems</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>16</td>
<td>9</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5 presents the distribution of the improving transformations taken for the 32 problems that have been improved by DICP. One clear finding is that the "addition" transformation is the most contributing transformation. "Deletion" is next. "Transparency" is the less contributing. 25 out of the 32 best improvements occurred with applying either one deletion with two additions or three additions.
Figure 1. Improved problems according to the change in pieces’ number

Figure 2. Improved problems according to the change in themes’ number

Figure 1 shows that most improved problems contain more pieces. This finding supports the hypothesis that adding more material improves the quality of the problems. Figure 2 shows that most new problems contain more themes. This finding supports the hypothesis that adding more material to a certain extent increases the number of the themes included in the problems. We can also conclude from Table 4 that it also increases the average number of unique thematic variants and the average proportional rate of the thematic variants.

7. Summary and Future Work

DICP presents an ability of improving the quality of many existing chess problems, composed by human composers. Some of the improvements are rather impressive, considering that most of the tested problems were composed by very experienced composers.

The "addition" transformation was found to be the most frequent successful transformation. On average, successful addition of pieces to a given problem to a certain extent improves an average problem by increasing the average number of themes, the average number of unique thematic variants, and the average proportional rate of the thematic variants. However, it must be pointed that "addition" as a single transformation is not enough. It usually appears with one or two additional transformations, e.g., deletion and addition.
In computer chess, high playing level has been proven to be inefficient without deep searching. We believe that this is also true in order to achieve a high level in composing chess problems. Therefore, the use of a more complex searching technique to a deeper depth (probably with some pruning) will further enhance the model.

Another idea is to evaluate the potential of DICP as an intelligent support system for weak and intermediate composers. Application of this idea may reinforce DICP’s strength in the long run whilst simultaneously improving these composers’ performance.

The current model can be extended to k-move (k>2) mate problems. This extension will demand: 1) use of faster search algorithms, 2) definition of various special new themes dedicated to k-move problems as it follows from Horward (1970).

**Acknowledgments**

Thanks to Josef Retter and Jean Haymann, both international masters of the FIDE for chess composition. These two international masters supplied us with relevant background concerning composition of chess problems.

**References**


