A PROGRAM FOR PLAYING TAROK

Mitja Luštrek[1]*, Matjaž Gams*[1] *and Ivan Bratko*[1]
Ljubljana, Slovenia

ABSTRACT

A program for playing the three-player tarok card game is presented and the complexity of the game is analyzed. To deal with the unknown distributions of other players' cards, the program uses a sampling method that accomplishes hierarchical clustering to select representative sets of cards from a host of randomly generated sets. A game tree is searched with the alpha-beta algorithm using several common enhancements and an *equivalence* transposition table, which groups the positions by strategic similarity instead of storing single positions.

1. INTRODUCTION

Game playing is a well-developed area of artificial intelligence and a number of programs for playing even relatively obscure games exist. However, card games, although otherwise quite popular, are not particularly well represented in computer game-playing. There are exceptions such as bridge (Ginsberg, 2001) and to a lesser extent poker (Billings et al., 2002). Programs for playing tarok (Tarok.net website; Triangle Productions website) are less common and little is known of how they work.

Tarok is a very popular game in Central Europe. Since it is a quite sophisticated game, some call it the "Rolls-Royce of card games" (Triangle Productions website). Its origins are in Italy (15th century). Nowadays, over a dozen of variants are played in many countries (Card Games website). They are known under different names such as tarock, taroky and königsrufen. Although the game is mostly played casually, competitions are also held: for example, the Tarok Association of Slovenia (see website) organizes an annual national championship.

Our team developed a three-player tarok program called SILICON TAROKIST (Luštrek and Gams, 2003). Moreover, we analyzed the complexity of the game. Apart from the standard techniques used in computer game-playing, the program incorporates a new sampling method and a novel equivalence transposition table.

The paper is organized as follows. Section 2 presents the game of tarok: subsection 2.1 explains the rules and subsection 2.2 analyzes the complexity. Section 3 deals with SILICON TAROKIST: subsection 3.1 presents our sampling method and subsection 3.2 describes game-tree search. Section 4 evaluates SILICON TAROKIST's play. Section 5 concludes the note and suggests some improvements of the program.

2. THE GAME

2.1 Rules

We chose the three-player version of the game, which is used in tournament play. A two-player version also exists, but is relatively unknown. A more complicated four-player version is probably the most popular one, but is played casually.

Figure 1: Some tarok cards.

[1] Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia. E-mail: mitja.lustrek@ijs.si.

Tarok is played with 54 cards. There are eight cards of each of the four suits and 22 trumps called taroks. The red suits supply (ranked from lowest to highest): 4, 3, 2, 1, Jack, Cavalier, Queen, King; the black suits supply (ranked from lowest to highest): 7, 8, 9, 10, Jack, Cavalier, Queen, and King. Each player is given 16 cards, which only he/she may see; the remaining six, called talon, are set aside unrevealed. Like in bridge, a game consists of two parts: bidding and playing.

During bidding, players name contracts of increasing difficulty, which bring correspondingly higher scores. The winner of the bidding becomes declarer. If nobody bids, the player who was to bid first may choose to play klop. This means that unlike under other contracts, all players have to strive for a score as low as possible. The talon is revealed and the declarer chooses between zero and three cards from it; they are exchanged for an equal number of cards from his/her hand. The number indicates what the contract[1] is. If, for example, two cards are to be exchanged, only the 1st and 2nd, the 3rd and 4th, or the 5th and 6th card from the talon can be taken. The cards removed from the hand are put on the declarer's won pile and may not include Kings. The declarer can make additional announcements, such as winning all four Kings, which add to the score if they are fulfilled and detract from it otherwise. The opponents may make counter-announcements, which double this additional score. The other two players oppose the declarer jointly.

During play, cards are played in tricks led by the winner of the previous trick. Suit must be followed if possible; a tarok should be played otherwise. The highest card of the leading suit wins the trick, unless it contains a tarok, in which case the highest tarok wins. The values of the highest four cards of each suit range from 2 to 5 points, the lowest tarok and the highest two taroks are worth 5 points each, and the other cards 1 point each. From these, one score is calculated for the declarer and another for the other two players. If the declarer's score is higher, he/she has won the game. In this case the value of the contract is added to his/her score, otherwise it is subtracted from the score. Normally several games are played in a session with the cumulative score deciding the overall winner. Detailed rules can be found on the Card Games website.

2.2 Complexity

In order to assess computational complexity of tarok, we calculated the number of leaf nodes in the full game tree for a single deal. In an average case, bidding and exchanging cards with talon contributes a factor of $\sim 10^3$, which was determined by constructing the tree of all bidding decisions and then pruning it at the branches that are rarely chosen; for each resulting contract, the complexity of exchanging cards with talon was calculated. Additional announcements and counter-announcements contribute a factor of ~10, which was calculated by combinatorial analysis of all the possible deals and reasonable announcements for each of them – the number is relatively small since for the majority of the deals, most announcements will almost certainly fail and the decision not to make them is automatic. The combination of both phases gives $\sim 10^4$, as can be seen in Table 1 under 'Bidding etc.' To evaluate the complexity of play in an average game, the game was simulated 10,000 times by SILICON TAROKIST. Legal moves for each turn were counted. Their numbers were multiplied for each game and averaged over all the simulation runs, giving $\sim 10^{28}$.

For the worst case, the same game phases were evaluated. For bidding and exchanging cards with talon, a simple program was used to enumerate all the 506 bidding options and calculate the complexity of exchanging cards with talon for each resulting contract, giving 7.5×10^3. Announcements and counter-announcements contribute further 5.0×10^5, resulting in the total of 3.7×10^9. For play, the worst case is where one player has only taroks and the other two have one tarok each (note that initial hand with no taroks is against the rules). In this situation, each player may play any card with the exception of the first tarok, which gives $16 \times (15!)^3 = 3.6 \times 10^{37}$ lines of play.

	3-Player Tarok			Bridge	2-Player Poker
	Bidding etc.	Play	Together		
Average	$\sim 10^4$	$\sim 10^{28}$	$\sim 10^{32}$	2.3×10^{24}	$\sim 10^{18}$
Worst Case	3.7×10^9	3.6×10^{37}	1.3×10^{47}	5.6×10^{44}	

Table 1: Complexity of tarok compared to bridge and poker.

[1] The possible contracts are zero, one, two, three, klop, and beggar. Zero through three differ only in the number of cards exchanged with talon, while klop (which is explained briefly) and beggar (very rare) have other properties. In contrast to bridge, the contract mainly indicates the starting position.

Compared to bridge (Smith et al., 1998), tarok is somewhat more complex, although fewer cards are used in actual play, i.e., 48 versus 52. This can be attributed to the fact that hands in tarok are larger, which gives each player more options. Also, there are 22 taroks, so when a tarok is played, there are particularly many options. Two-player Texas Hold'em poker (Billings et al., 2003) is, unsurprisingly, much less complex. Note that for poker, we do not differentiate between average and worst case.

3. THE PROGRAM

A tarok-playing program must perform three distinct tasks: bidding, exchanging cards with talon, and actual playing. Despite these tasks seemingly being quite different, SILICON TAROKIST performs them in basically the same way. It uses sampling to generate a number of representative distributions of other player's cards. For each distribution, the game is then treated as a perfect-information game. So a game tree can be searched to determine the best cards to play. This does not always capture all the nuances of the actual imperfect-information situation (Frank and Basin, 2001), but it is a reasonable heuristic.

Bidding. For each contract starting with the lowest, a number of trial games are played out with SILICON TAROKIST making choices for all three players using a series of shallow game-tree searches. The highest contract for which the trials have been won by a sufficient margin is considered viable. The program bids as high as this contract, unless no contract is viable, in which case it does not bid and chooses klop if possible.

Exchanging cards with talon. For each set of cards that can be taken from the talon, several combinations of cards are considered for removal from the hand. For each combination, a number of trials are performed and the best combination is chosen. The percentage of games where each possible announcement has been fulfilled is noted and if it is sufficiently high, the announcement is actually made.

Playing. When a card has to be played, a number of representative distributions of other players' cards are generated. For each distribution, a game tree is searched and the best card is selected. The best card over all the distributions is played.

The margin by which test games have to be won during bidding is adaptable. Initially it is set to 3 points above the minimum 35 required for victory – a value that proved suitable during testing. When the program wins the bidding and loses the game, it must have been too optimistic, so the margin for future games is increased by a point. When the program does not win the bidding and it wins the game anyway, it must have been too pessimistic, so the margin is decreased by a point. This margin is needed because the trials do not simulate the actual game perfectly and if they are won narrowly, the actual game may still be lost. Percentages of games where each possible announcement has been fulfilled are treated in a similar fashion.

3.1 Sampling

In tarok performing game-tree search for all possible distributions of cards is too time-consuming; therefore SILICON TAROKIST samples this large space. In such cases Monte-Carlo sampling (Frank and Basin, 2001), which samples the space randomly, is often used. Assuming uniform probability of the possible game states and given enough samples, it will generally choose the correct move. However, the assumption of uniform probability can often be improved upon. In SILICON TAROKIST, this is done to a certain extent by excluding distributions of cards that are known to be impossible. At the beginning of the game, the only distributions of such nature are those in which the declarer does not have the cards taken from the talon. Later, more information is available: if a player follows suit with a tarok, he/she does not have any cards of that suit, etc. The issue of sufficient number of samples is also addressed (note that a sample means one distribution of cards among the three players). Our method takes advantage of the fact that in Monte-Carlo sampling, the number of samples selected from each portion of the sample space is proportional to its size, i.e., that the number of card distributions belonging to a class of similar distributions is proportional to the likelihood of that class. But our method requires fewer samples than regular Monte-Carlo sampling to achieve comparable results. This is because for a given number of samples, it constructs a more representative set of samples than regular Monte-Carlo sampling would. The method is in part domain-independent and in part based on knowledge about tarok.

The sampling method uses hierarchical clustering (Jain et al., 1999), a common technique in data mining. To produce n representative samples, n_0 (which should be several times larger than n) random samples are

generated to ensure that various sets of hands are present in typical proportions. A distance measure for samples must be defined: in SILICON TAROKIST, it is the sum of distances between corresponding hands. The more different cards two hands contain, the more distant they are considered to be. The difference in higher cards is more important than in lower; specific suited cards have to be considered, while taroks, with the exception of 1, 21 and 22, are more or less interchangeable. This is shown in equation (1): $c^A_{i,j,k}$ equals 1 if k-th card of suit j is present in i-th player's hand in sample A, and is 0 otherwise; $score_{j,k}$ is the score brought by k-th card of j-th suit; T instead of j indicates tarok; parameter values 4 and 10 were determined experimentally, but the method does not seem to be very sensitive in this regard.

$$d = \sum_{i=1}^{3}\left(\left(\sum_{j=1}^{4}\sum_{k=1}^{8}(abs(c^A_{i,j,k} - c^B_{i,j,k}) \times score_{j,k}) + \sum_{k=1,21,22}(abs(c^A_{i,T,k} - c^B_{i,T,k}) \times score_{T,k})\right) \times 4 + \right.$$
$$\left. + abs\left(\sum_{k=1}^{22}((c^A_{i,T,k} - c^B_{i,T,k}) \times (k+10))\right)\right) \quad (1)$$

This distance measure is used to generate a number of clusters of similar samples. From each cluster i containing c_i samples, $n/n_0 \times c_i$ cards are selected to perform game-tree search on in practice.

We experimentally compared clustering-based sampling to regular Monte-Carlo sampling using the same number of samples n; n_0 used in clustering-based sampling was 300. An expert evaluated each move out of 500 test moves on which the methods disagreed. Grade +1 was given if the move proposed by clustering-based sampling was significantly better, -1 if it was significantly worse, +0.5 if it was slightly better, and -0.5 if it was slightly worse. When testing with 30 samples, the grade of clustering-based sampling compared to regular Monte-Carlo sampling was +3.8 per 100 moves. For 60 samples, the grade was +1 per 100 moves. This shows that it is worth implementing clustering-based sampling particularly when only a small number of samples can be used due to time constraints.

3.2 Game-Tree Search

SILICON TAROKIST uses alpha-beta search with the depth varying from 9 to 18. Unlike in most games, in tarok the sequence of *min* and *max* plies is not predetermined, because for each trick it depends on the winner of the previous one. Since two players play against one, there are twice as many *min* plies as there are *max* plies when the program is the declarer; the numbers are reversed when it is not. However, adaptation of alpha-beta in this regard is straightforward.

The alpha-beta algorithm is enhanced by:

- iterative deepening (Marsland, 1986) – searching in iterations with increasing depth;
- history heuristic (Schaeffer, 1989) – ordering moves based on their success in the past;
- selective move generation – discarding bad and redundant moves;
- transposition table (Breuker et al., 1997) – storing values of positions for reuse.

Minimal window (Marsland, 1986) was also tried, but with all the other enhancements in place, it turned out to be counterproductive.

Selective move generation is performed by discarding moves for which in a given situation, we are certain that they are either outright bad or at least no better than another move. We identified 36 situations regarding contract, number of cards played so far in the current trick, order of players etc., and nine categories of cards to be considered. So, which moves will be pruned is determined according to a matrix of 324 Boolean values chosen by an expert. For example, two taroks have been played, our partner has the higher one and we have some taroks as well. In this case, we will either play the lowest tarok in hand or the lowest that still wins the trick. A significant number of moves are pruned in this way, in particular for the second and the third card of each trick. Move generation could easily be expanded to include moves necessary for signaling, i.e., making specific moves meant to tell something to our partner. However, unlike in bridge, this is not a common practice in tarok.

A **transposition table** in SILICON TAROKIST stores values for sets of strategically equivalent game states, and is referred to as an *equivalence transposition table*. This is similar to partition search as used in the bridge program GIB (Ginsberg, 2001), except that in SILICON TAROKIST, game states are clustered heuristically and are not guaranteed to have the same value. On the one hand this allows misplays, but on the other hand it

makes it possible to sacrifice some reliability for speed and hopefully improve performance through an increased depth of search or more samples. Measurements do not reject this hypothesis. A comparison to a regular transposition table, 30 samples and depth 9 was made. With an equivalence transposition table, 30 samples and the depth increased to 12, the search was slower and the results were graded -1 per 100 moves (grading as described in Subsection 3.1). However, with the equivalence transposition table, depth 9 and the number of samples increased to 60, the search was three times faster and the results were graded +4 per 100 moves, showing that the equivalence transposition table is in fact beneficial. The qualitative comparison was made on 300 test moves. Now 60 samples are used during playing, while during bidding and exchanging cards with the talon, it holds that more than 30 samples make the program too slow.

Different definitions of strategic equivalence are used throughout the game. At the beginning there are usually some obviously good moves available and the game tree is wide, so reliability is not so important and speed is crucial. Towards the end it is often less clear which moves are good, but the tree is smaller and so the most reliable method should and can be used. At the beginning of the game, two hands are considered equivalent if in each suit, they either do not have any cards or they have the number of cards below Queen different for at most 1 and both do or do not have Queen or King; they must also have the number of taroks different for at most 1, their sum rounded to the nearest multiple of 20 must be equal, and they must both either have or not have taroks 1, 21 and 22. Towards the end of the game, two hands are considered equivalent if in each suit, they have the same number of low cards and the same specific high cards; they must also have an equal number of taroks, their sum rounded to the nearest multiple of 10 must be equal, and they must both either have or not have taroks 1, 21 and 22.

The **evaluation function** is designed in such a way that it takes advantage of the transposition table. Normally, the evaluation function is calculated for the entire game state, meaning that keys for the values stored in the transposition table must be full game states as well. In tarok, full game states incorporate information on current hands as well as on tricks that have been won since the beginning of the game. However, the strategy in tarok depends almost exclusively on current hands, so history can mostly be discarded. Each reduced game state thus incorporates a number of full game states, allowing more hits in the transposition table. Therefore the evaluation function in SILICON TAROKIST measures only the difference between the current state and either the state at the maximum depth of search or at the depth where a value can be retrieved from the transposition table. In the latter case, this value is combined with the measured difference, as can be seen in equation (2): t is the number of tricks played from the beginning of the current game tree search, v_i is the value of i-th trick, v_{TT} is the value retrieved from the transposition table (if there is one) and t_{TT} is t as it was when v_{TT} was stored in the transposition table.

$$V = (\sum_{i=1}^{t} v_i \cdot 0.9^i) + v_{TT} \cdot 0.9^{t-t_{TT}} \qquad (2)$$

The experimentally chosen factor 0.9 in equation (2) makes sure that tricks won deeper in the search are given less importance, because it is less certain whether they will indeed be played out as predicted. Without this factor, making an unprofitable but unavoidable move before a profitable one was sometimes evaluated equally as the other way around. After the unprofitable move was made, it has turned out it was not unavoidable after all. This corrective factor therefore gives slightly higher priority to immediate profit, which is a prudent strategy, because it prevents serious mistakes at the cost of playing somewhat suboptimally on occasions; it is also perfectly sensible in situations where there is not enough information available to form a better plan. For example, King of Hearts will likely win your partner's highest or second highest and your opponent's lowest Heart any time you play it, but playing it might reveal something about the other players' cards. The corrective factor by the second summand adjusts the value from the transposition table, because it might be too large if it was stored in the table after a shallower search than the current one.

Table 2 shows the benefits of the enhancements of alpha-beta search in different combinations. Iterative deepening alone is left out because without the history heuristic and the transposition table, it is not beneficial. Benefits of enhancements are measured in terms of nodes and time. 'Nodes' represent an average number of nodes visited during a single game-tree search

Enhancements	Nodes	Time (s)
All four	2,470	0.068
All except iterative deepening	3,799	0.119
All except history heuristic	3,325	0.091
All except selective move generation	9,936	0.212
All except equivalence transposition table	24,034	0.392
Only history heuristic	283,673	3.966
Only selective move generation	40,838	0.625
Only equivalence transposition table	14,457	0.293
None	1,598,924	21.266

Table 2: Enhancements of alpha-beta search.

for the first card to play in the game over several hundreds of starting positions. 'Time' represents the actual time used for the search on an Athlon XP P1800+ PC with 512 MB of RAM. Search depth was set to 9. Such a depth is in fact used in the early game and is later increased up to 18, thus enabling a response time of a couple of seconds, which is desirable in casual games.

It can be seen from Table 2 that the equivalence transposition table is most beneficial, followed by the selective move generation. The history heuristic does not contribute much because move ordering is mostly done by the transposition table. Iterative deepening is used because earlier iterations fill the transposition table with values that guide the search in later iterations. It is not particularly effective with an equivalence transposition table, although it is very helpful in conjunction with a regular transposition table, in which case it reduces the number of nodes visited by a factor of 3.1 instead of merely 1.5. This is because the overhead caused by iterative deepening is more pronounced when the transposition table itself is very effective.

4. EVALUATION OF PLAY

To illustrate the playing strength of SILICON TAROKIST, we have chosen an interesting example game. The program is making moves for players 1 and 3, while player 2 is human.

Bidding. T indicates tarok. These are the initial hands:

Player 1:	T1	T4	T6	T11	T14	T15	T21	♥K	♠K	♦4	♦3	♦1	♦J	♦K	♣8	♣J
Player 2:	T2	T8	T9	T13	T16	T17	T18	T19	♠8	♠Q	♦C	♦Q	♣7	♣10	♣C	♣K
Player 3:	T3	T10	T12	T22	♥4	♥3	♥2	♥J	♥Q	♠7	♠9	♠10	♠J	♦2	♣9	♣Q

Players 1 and 2 are willing to bid as high as two, so player 1, who has priority, ends up as the declarer.

Exchanging cards with talon

Talon: T7 ♥1 ♥C T5 T20 ♠C

Player 1 chooses T20 and ♠C from the talon, which is obviously the best pair. The cards ♣8 and ♣J are replaced by this pair, because if a player has no cards of a suit, he/she can win tricks in that suit with taroks. Player 1 then announces to win all four Kings, because he/she already has three and having no Clubs, he/she will likely be able to win ♣K with a tarok. This is the hand he/she starts playing with:

Player 1:	T1	T4	T6	T11	T14	T15	T20	T21	♥K	♠C	♠K	♦4	♦3	♦1	♦J	♦K

Playing. Tricks are enclosed in square brackets; the numbers before cards denote who the card was played by and the winner is underlined. Player 1 will certainly open with one of the Kings, because they bring a high score and there is little to be gained by saving them for later. Player 1 chooses ♦K, which is riskier than ♥K or ♠K, but promises a higher reward, because he/she has five low Diamonds in the hand, so one of the opponents will have to play either ♦C or ♦Q. Despite the risk, this is not an unreasonable move, although a bit premature. The first trick is thus [1: ♦K, 2: ♦C, 3: ♦2]. Player 1 continues with ♥K, but unfortunately player 2 has no Hearts: [1: ♥K, 2: T2, 3: ♥Q]. Player 2 then tries to lure out ♠K to win a future trick with ♠Q. However, player 1 decides that the promise of winning ♠Q outweighs the risk of playing ♠K when only five Diamonds are left and there is a greater chance that one of the opponents will have none, resulting in [2: ♠8, 3: ♠7, 1: ♠C], [1: ♠K, 2: ♠Q, 3: ♠9]. These are the hands at this point:

Player 1:	T1	T4	T6	T11	T14	T15	T20	T21	♦4	♦3	♦1	♦J
Player 2:	T8	T9	T13	T16	T17	T18	T19	♦Q	♣7	♣10	♣C	♣K
Player 3:	T3	T10	T12	T22	♥4	♥3	♥2	♥J	♠10	♠J	♣9	♣Q

Player 1 should now try to make the opponents play taroks without losing his/her own by playing Diamonds, but is reluctant to do so because this will enable the opponents to win ♦Q. Player 1 is unable to realize that this is inevitable, so the next trick is [1: T4, 2: T8, 3: T3]. Player 2, being human, does with Clubs what player 1 should have done with Diamonds, and player 3 plays an unimportant low card, resulting in [2: ♣7, 3: ♣9, 1: T1]. Player 1 continues to play taroks in [1: T6, 2: T9, 3: T10] and player 3 has no particularly exciting options: [3: ♠10, 1: T11, 2: T13]. Player 2's strategy of forcing player 1 to lose taroks has the consequence of losing player 3's ♣Q, but this probably could not have been avoided: [2: ♣10, 3: ♣Q, 1: T14]. Now player 1, seeing that he/she has only high taroks left, finally decides to start playing Diamonds: [1: ♦4, 2: ♦Q, 3: T12].

These are the hands at this point:

Player 1: T15 T20 T21 ♦3 ♦1 ♦J
Player 2: T16 T17 T18 T19 ♣C ♣K
Player 3: T22 ♥4 ♥3 ♥2 ♥J ♠J

Player 2 seems to have an advantage over player 1, but the program, now being able to search the whole game tree, plays flawlessly and gains the upper hand: [3: ♥4, 1: T20, 2: T16], [1: ♦3, 2: T17, 3: T22], [3: ♥3, 1: T21, 2: T18], [1: ♦1, 2: T19, 3: ♥J], [2: ♣C, 3: ♥2, 1: T15], [1: ♦J, 2: ♣K, 3: ♠J]. Despite winning by a significant margin, player 1 scores only 14 points because he/she failed to win all four Kings.

SILICON TAROKIST was compared to human players and to tarok-playing program TAROK.NET (Tarok.net website). Due to occasional mistakes it could not win against a good human player consistently, but its perfect memory and strong endgame were enough to defeat an average player as well as TAROK.NET. Table 3 shows the results in terms of the difference between SILICON TAROKIST's and its opponents' score per game averaged over 40 games.

Opponent	Score Difference / Game
Good player	-2.8
Average player	6.1
Tarok.net	3.5

Table 3: The quality of SILICON TAROKIST's play.

5. CONCLUSION

In this paper we analyzed the average and the worst-case complexity of tarok. We showed that it is greater than the complexity of bridge. Owing to this complexity in combination with the fact that tarok is an imperfect-information game, we may conclude that developing a good tarok-playing program is a truly challenging task.

Our program, SILICON TAROKIST, can deal with all aspects of three-player tarok and plays it reasonably well, according to human players. Sometimes it exhibits a behavior that would be considered 'cunning' in a human. However, in some cases it seems to be playing 'aimlessly', thus indicating that there is room for improvement. The program can be downloaded from the Internet (Silicon Tarokist website). We introduced two algorithmic improvements, viz. in sampling and game-tree search. Clustering-based sampling produces a more representative distribution of other players' cards, allowing a more reliable selection of the best move. The equivalence transposition table turned out to be the most effective enhancement to the alpha-beta algorithm. Both can be used in domains other than tarok.

Sampling in SILICON TAROKIST, while performing reasonably well, could be improved by biasing the selection of samples based on other players' past moves: samples consistent with moves actually made would more likely be selected. More importantly, the search algorithm is inadequate for truly high-level play. It is sufficient towards the end of the game, but at the beginning it cannot develop long-term strategies and does not play particularly well when no immediately beneficial moves are available. One way to remedy this would be to program explicitly such strategies, but we feel this might result in play that is too rigid and therefore predictable. Another way would be to perform game-tree search that would ignore all but a handful of representative plays and could therefore be sufficiently deep; it could then be refined by a shallower and more thorough search. This we leave for future work.

6. ACKNOWLEDGEMENT

The work reported in this paper was supported by the Slovenian Ministry of Education, Science and Sport.

7. REFERENCES

Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., and Szafron, D. (2003). Approximating Game-Theoretic Optimal Strategies for Full-scale Poker. *Proceedings of IJCAI-03*, pp. 661-668.

Billings, D., Davidson, A., Schaeffer, J., and Szafron, D. (2002). The Challenge of Poker. *Artificial Intelligence*, Vol. 134, No. 1-2, pp. 201-240. ISSN 0004-3702.

Breuker, D.M., Uiterwijk, J.W.H.M., and Herik, H.J. van den (1997). Information in Transposition Tables. *Advances in Computer Chess* 8 (eds. H. J. van den Herik and J. W. H. M. Uiterwijk), pp. 199-211. Universiteit Maastricht, Maastricht. ISBN 9062162347.

Card Games website. http://www.pagat.com [accessed 2003-05-14].

Frank, I. and Basin, D. (2001). A Theoretical and Empirical Investigation of Search in Imperfect Information Games. *Theoretical Computer Science*, Vol. 252, pp. 217-256. ISSN 0304-3975.

Ginsberg, M.L. (2001). GIB: Imperfect Information in Computationally Challenging Game. *Journal of Artificial Intelligence Research*, Vol. 14, pp. 303-358.

Jain, A.K., Murty, M.N., and Flynn, P.J. (1999). Data Clustering: A Review. *ACM Computing Surveys* (*CSUR*)**,** Vol. 31, No. 3, pp. 264-323. ISSN 0360-0300.

Luštrek, M. and Gams, M. (2003). Intelligent System for Playing Tarok. *Proceedings of ITI 2003*, pp. 391-396. SRCE University Computing Centre, University of Zagreb. ISBN 953-96769-6-7.

Marsland, T.A. (1986). A Review of Game-Tree Pruning. *ICCA Journal*, Vol. 9, No. 1, pp. 3-19. ISSN 0920-234X.

Schaeffer, J. (1989). The History Heuristic and Alpha-Beta Search Enhancements in Practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 11, pp. 1203-1212. ISSN 0162-8828.

SILICON TAROKIST website. http://tarok.bocosoft.com.

Smith, S.J.J., Nau, D., and Throop, T. (1998). Computer Bridge: A Big Win for AI Planning. *AI Magazine*, Vol. 19, No. 2, pp. 93-105. ISSN 0738-4602.

Tarok Association of Slovenia website. http://users.volja.net/tarokzveza [accessed 2003-05-14].

Tarok.net website. http://www.tarok.net [accessed 2003-06-12].

Triangle Productions website. http://www.gatecentral.com/triangle/produkte/index.html [accessed 2003-05-22].