Varying Complexity in CHINESE DARK CHESS Stochastic Game

Nicolas Jouandeau
LIASD
Paris8 University
Email: n@ai.univ-paris8.fr

***Abstract*— CHINESE DARK CHESS is an interesting stochastic game that combines revealing, positioning and capturing moves. As many variants are possible, we propose a general protocol that allows to play this game as puzzle, competitive or cooperative game, by 1 to multiple players. Varying complexity can be addressed by this variations. A short visualisation of 4 different games are presented.**

I. INTRODUCTION

CHINESE DARK CHESS is a popular stochastic two players game in Asia that is often played on 4x8 rectangular board where players do not know flipping moves' payoff. One player is black and the other player is red. The first flipping move defines the first player color. Each player starts with the same set of pieces. Relationship between pieces defines capture possibilities. Pieces evolve on squares and can move vertically and horizontally from one square to an adjacent free square (*i.e.* up, down, left and right). All pieces can move on adjacent squares except cannons that can jump. Jumping are conditioned by a jumping piece and a target piece. Free spaces can stands in the path, before and after the jumping piece.

Chen *et al.* [1] used *alpha-beta* algorithm with different revealing policies combined with a *initial-depth flipping* method to reduce the branching factor. They distinguished opening, middle and endgame to apply different policies.

Chen *et al.* [2] built an endgame databases with retrograd analysis. Created databases are done for each first move color, up to 8 revealed pieces. They used 2TB of memory to represent 10^{12} positions. Positions status are stored as win, lost or draw.

Yen *et al.* [3] presented a non-deterministic Monte Carlo tree search model by combining chance nodes [4] and Monte Carlo Tree Search (MCTS). They create shorter simulation by moderating the three policies named *Capture First*, *Capture Stronger Piece First* and *Capture and Escape Stronger Piece First*. As draw rate decreases, win rate increases and simulations are more meaningful for MCTS.

Chang and Hsu [5] solved the 2x4 variant. They created a *Oracle* variant where every pieces are known. Comparing the *Oracle* variant and the classical variant shows that the first move is crucial on 2x4 board.

Safidine *et al.* [6] exploit pieces combinations to reduce endgame databases. By combining material symmetry identified by relations between pieces and endgames building with retrograd analysis, winning positions are recorded in databases. This general method has been applied to SKAT, DOMINOES and CHINESE DARK CHESS. Even if relationship between pieces in CHINESE DARK CHESS creates intricate symmetries, they reduced the size of 4 elements endgame tables by 9.92 and the size of 8 elements endgame tables by 3.68.

As many variants are possible in CHINESE DARK CHESS, we propose an new protocol that allows to play different variants of this stochastic game. Varying games should help to understand new tactics and new strategies, from single to multi-players.

The paper is organized as follows. Section II describes a general protocol to play with multiple variants. Section III presents a visualisation of the resulting complexity of 4 different variants before the conclusion.

II. DESCRIPTION OF A GENERAL PROTOCOL

To vary complexity in the CHINESE DARK CHESS stochastic games, we propose a simple modular protocol that defines format messages between programs (*i.e.* between a server and a client during a game). Multiple clients are allowed and clients do not communicate between themselves. Communications are over TCP. Clients connect to the server using the ports printed out by the server. Messages use symbolic expressions (*i.e.* S-expressions) as message structure. S-expressions come from Lisp programming language, that allows to easily represent tree structures. S-expressions are strings between parenthesis. They are simple to parse and to extend. Even if they are long, they stay easily readable by humans. To initiate the protocol, each client has to send a first version string `'((version 1.0))'` (see Tab. III).

TABLE I
SERVER TO CLIENT COMMUNICATIONS.

`(id PLAYER_ID)`	player's id.
`(players NB_PLAYER)`	number of players.
`(team TEAM_ID)`	player's team.
`(teammate ID)`	teammates in the team.
`(size SIZE)`	board's size.
`(moves (POS_I POS_F)...)`	list all possible moves.
`(jumps (ORI TG_I TG_F)...)`	list all possible jumps.
`(revs (POS_0)(POS_1)...)`	list all possible reveals.
`(game TYPE)`	game's type.
`(pieces XXXXXXXX)`	pieces at start.
`(board XXXXXXXX)`	current board.
`(hand P)`	player's hand.
`(puts (POS_0)(POS_1)...)`	list all possible puts.
`(turn TURN ID))`	current turn and player's id to play.
`(info TURN ID))`	current turn and player information.
`(win PLAYER_ID TEAM_ID))`	game's result.

Expressions presented in Tab. I show server to clients communications. Inside `id` expression, the value `PLAYER_ID` is varying from 1 to N. In the same time, the server should send the expression `(players` N`)`. `team` expression allows to team-play scoring. `teammate` expression allows to consider team member and multi-player games. `size` moves and `jumps` expressions allow to consider different board's configuration, even non rectangular once. To reduce combinatorial of jumping positions, `jump` expression contains 3 values that defines origin, first and last target (considering that all positions between first and last target are possible targets). Even if `jump` expression list possible jump, jumping condition has to be checked while applying a jump on a board. Enumerating all moves and jumps allow to build asymmetric boards and topologies. We believe that it should be interesting to fit game situations to real situations, where topologies are commonly not symmetric. This should also produce games that are closer to classical real-time strategic games. The main variants of CHINESE DARK CHESS are played with 4x8 and 8x16 boards with 2 players. On such symmetric boards with L lines and C columns (with $L \geq 2$ and $C \geq 2$), the number of moves is equal to $4LC - 2L - 2C$ and the number of jumps is equal to $L(C-2)(C-1) + C(L-2)(L-1)$. It makes 104 moves and 216 jumps on 4x8 board, and respectively 464 and 2352 on 8x16 board. By using 3 values inside `jumps` expression, we reduced the number of `jumps` expressions needed. As a board should be partially revealed at start, `revs` list all positions that can be revealed. `game` expression defines game's type (*i.e.* traditional or strategic positional for example). `pieces` expression defines involved pieces. According to CHINESE DARK CHESS games, pieces can be identified by letters as presented in Tab. II. Thus a 2x4 board set should be `KPPPkppp`. For unrevealed pieces on the board, we introduce a '`X`' character. For empty positions, we introduce a '`.`' character. Thus a 2x4 board expression with 2 unrevealed pieces and 1 empty position should be `KPPP.kpXX`. Transmitting the board at each step should allow players to request timeout to check their current player. To allow players to put pieces in a empty position, `hand` expression gives a piece to a player's hand. `puts` expression lists all possible positions where players can put a piece. `turn` expression should be sent to the current player whereas `info` expression should be sent to other players. In a multi-player game, these expressions should be practical to inform other players status. When a player wins, the server should send the `win` expression to all players. Inside the `win` expression, 0 values are admitted to express individual and team victory. For example, the expression `(win 1 0)` indicates player-1 victory whereas the expression `(win 0 2)` indicates team-2 victory.

Expressions presented in Tab. III show clients to server communications. Inside `init` expression, each client declares its name and contributes to the server seed. According to the `board` expression ever presented in Tab. I, clients reminds to the server the initial situation enhanced with one of the five next expressions: `rev` expression is used to reveal `POS` position; `mov` expression is used to move a piece from `POS_I` to `POS_F`; `resign` allows client to resign the current game; `bag` and `put` are useful for a variant where unknown pieces are out of the board and where their placement depends on client's choice. In this variant, the initial board is empty. All pieces are unknown and a player chooses one unknown piece, reveals its status and places it on the board. Then until some unknown pieces exist, each player can move one of their pieces on the board according to traditional rules or pick up a new piece for unknown set, reveal and place it on the board. When all pieces are places on the board, each player must move a piece on the board. The game ends when one player is not able to move one of its pieces.

TABLE II

TRADITIONAL CHINESE DARK CHESS PIECES.

Pieces names	Red Player Num.	Red Player Icon	Red Player Char	Black Player Num	Black Player Icon	Black Player Char
King	⑦	將	`K`	❼	帥	`k`
Guard	⑥	士	`G`	❻	仕	`g`
Bishop	⑤	象	`B`	❺	相	`b`
Knight	④	馬	`N`	❹	馬	`n`
Rook	③	車	`R`	❸	車	`r`
Cannon	②	炮	`C`	❷	炮	`c`
Pawn	①	卒	`P`	❶	兵	`p`

TABLE III

CLIENT TO SERVER COMMUNICATIONS.

`(version 1.0)`	initializes comm with server.
`(init NAME SEED)`	declares its name and contributes to seed.
`(board XXXXXXXX)`	current board.
`(rev POS)`	reveal order.
`(mov POS_I POS_F)`	move order.
`(resign)`	resign order.
`(bag)`	select one piece from unknown set.
`(put PIECE POS)`	put hand's piece `PIECE` at `POS` place.

According to Tab. I and Tab. III expressions, we can defines game types :

- traditional `1 VS 1` game
- handicap game where players have different sets according to their level
- strategic positional variant where players pick unknown pieces from a bag, reveal them and choose their first position on the board
- single player game where the goal is to remove all piece of one color as fast as possible
- new multi-player games (`1 VS N` or `N VS 1` or `N VS N`) where teams of player can be settled
- specific game that start form fixed board
- varying board's size
- varying game complexity by reducing initial unknown

TABLE IV

TRADITIONAL 2X4 GAME.

S → C1	`((version 1.0))`
S ← C1	`((init rand_player 666))`
S → C2	`((version 1.0))`
S ← C2	`((init uct_player 1))`

S → C1	`((id 1)(players 2)(game traditional)`
	`(size 8)(pieces KPPPkppp)`
	`(moves (0 1)(0 4)(1 0)(1 2)(1 5)(2 1)(2 3)`
	`(2 6)(3 2)(3 7)(4 0)(4 5)(5 4)(5 6)(5 1)`
	`(6 5)(6 2)(6 7)(7 6)(7 3))`
	`(jumps (0 2 3)(1 3 3)(2 0 0)(3 0 1)(4 6 7)`
	`(5 7 7)(6 4 4)(7 4 5))`
	`(revs (0)(1)(2)(3)(4)(5)(6)(7))`

S → C2	`((id 2)(players 2)(game traditional)`
	`(size 8)(pieces KPPPkppp)`
	`(moves (0 1)(0 4)(1 0)(1 2)(1 5)(2 1)(2 3)`
	`(2 6)(3 2)(3 7)(4 0)(4 5)(5 4)(5 6)(5 1)`
	`(6 5)(6 2)(6 7)(7 6)(7 3))`
	`(jumps (0 2 3)(1 3 3)(2 0 0)(3 0 1)(4 6 7)`
	`(5 7 7)(6 4 4)(7 4 5))`
	`(revs (0)(1)(2)(3)(4)(5)(6)(7))`

S → C1	`((board XXXXXXXX)(turn 0 1))`
S → C2	`((board XXXXXXXX)(info 0 1))`
S ← C1	`((board XXXXXXXX)(rev 0))`
S → C1	`((board KXXXXXXX)(info 0 2))`
S → C2	`((board KXXXXXXX)(turn 0 2))`
S ← C2	`((board KXXXXXXX)(rev 1))`
S → C1	`((board KPXXXXXX)(turn 1 1))`
S → C2	`((board KPXXXXXX)(info 1 1))`
S ← C1	`((board KPXXXXXX)(rev 4))`
S → C1	`((board KPXXpXXX)(info 1 2))`
S → C2	`((board KPXXpXXX)(turn 1 2))`
S ← C2	`((board KPXXpXXX)(mov 4 0))`
S → C1	`((board pPXX.XXX)(turn 2 1))`
S → C2	`((board pPXX.XXX)(info 2 1))`
...	`...`
S → C1	`((board .P.P.XPX)(turn 5 2))`
S → C2	`((board .P.P.XPX)(info 5 2))`
S ← C2	`((board .P.P.XPX)(resign))`
S → C1	`((win 1 0))`
S → C2	`((win 1 0))`

pieces number

In the single player strategic positional variant with classical sets, the optimal policy is to put pieces on the board and wisely sort them on the board according to future capturing moves. When all pieces of one color are reveal, `N` pieces are captured within `N` moves if `N` paths exists for `N` master pieces of `N` subsets. For the single player traditional game, the optimal policy is also to reveal pieces while one color is completely revealed. Then the goal is the identify short independent paths, combination of blue or red moves, minimizing turn back or maximizing exploitation of cannons' abilities.

As example, Tab. IV and Tab. V present respectively one traditional `1 VS 1` game on 2x4 board and one strategic positional variant `1 VS 1` game on 2x4 board. All other variants are then easy to settle.

III. PRACTICAL VISUALISATION

In this section, we present a visualisation of a single player game and 3 two players games. As games finish when a player has no more move, games are visualized according the number of moves' evolution during the game.

TABLE V

STRATEGIC VARIANT 2X4 GAME.

S → C1	`((version 1.0))`
S ← C1	`((init rand_player 666))`
S → C2	`((version 1.0))`
S ← C2	`((init uct_player 1))`

S → C1	`((id 1)(players 2)(game strategic)`
	`(size 8)(pieces KPPPkppp)`
	`(moves (0 1)(0 4)(1 0)(1 2)(1 5)(2 1)(2 3)`
	`(2 6)(3 2)(3 7)(4 0)(4 5)(5 4)(5 6)(5 1)`
	`(6 5)(6 2)(6 7)(7 6)(7 3))`
	`(jumps (0 2 3)(1 3 3)(2 0 0)(3 0 1)(4 6 7)`
	`(5 7 7)(6 4 4)(7 4 5))`
	`(puts (0)(1)(2)(3)(4)(5)(6)(7))`

S → C2	`((id 2)(players 2)(game strategic)`
	`(size 8)(pieces KPPPkppp)`
	`(moves (0 1)(0 4)(1 0)(1 2)(1 5)(2 1)(2 3)`
	`(2 6)(3 2)(3 7)(4 0)(4 5)(5 4)(5 6)(5 1)`
	`(6 5)(6 2)(6 7)(7 6)(7 3))`
	`(jumps (0 2 3)(1 3 3)(2 0 0)(3 0 1)(4 6 7)`
	`(5 7 7)(6 4 4)(7 4 5))`
	`(puts (0)(1)(2)(3)(4)(5)(6)(7))`

S → C1	`((board XXXXXXXX)(turn 0 1))`
S → C2	`((board XXXXXXXX)(info 0 1))`
S ← C1	`((board XXXXXXXX)(bag))`
S → C1	`((board XXXXXXXX)(hand K)(turn 0 1))`
S → C2	`((board XXXXXXXX)(hand K)(info 0 1))`
S ← C1	`((board XXXXXXXX)(put K 0))`
S → C1	`((board KXXXXXXX)(info 0 2))`
S → C2	`((board KXXXXXXX)(turn 0 2))`
S ← C2	`((board KXXXXXXX)(bag))`
S → C1	`((board KXXXXXXX)(hand P)(info 0 2))`
S → C2	`((board KXXXXXXX)(hand P)(turn 0 2))`
S ← C1	`((board KXXXXXXX)(put P 7))`
S → C1	`((board KXXXXXXP)(turn 1 1))`
S → C2	`((board KXXXXXXP)(info 1 1))`
S ← C1	`((board KXXXXXXP)(bag))`
S → C1	`((board KXXXXXXP)(hand p)(turn 1 1))`
S → C2	`((board KXXXXXXP)(hand p)(info 1 1))`
S ← C1	`((board KXXXXXXP)(put p 1))`
S → C1	`((board KpXXXXXP)(info 1 2))`
S → C2	`((board KpXXXXXP)(turn 1 2))`
...	`...`
S → C1	`((board p.kXXPPP)(turn 3 2))`
S → C2	`((board p.kXXPPP)(info 3 2))`
S ← C2	`((board p.kXXPPP)(resign))`
S → C1	`((win 1 0))`
S → C2	`((win 1 0))`

Fig. 1 presents a single player game on 8x4 board. All possible moves (*i.e.* all pieces of all colors) are represented in blue and free spaces (*i.e.* captured pieces) in red. The goal is then to remove all pieces of one color as fast as possible while revealing pieces can lead to move both color to make shorter paths.

Fig. 2 presents a very short traditional games on 8x4 board that ends in 31 turns (*i.e.* 62 plies). Fig. 3 presents a more competitive traditional games on 8x4 board that respectively finish in 53 turns (*i.e.* 105 plies). Fig. 4 presents a strategic variant on 8x4 board that ends in 62 turns(*i.e.* 124 plies). In two players figures, curves represent the number of possible moves (including moves, jumps and reveals) for each player over turns. Turns are represented on horizontal axis and number of possible moves on vertical axis. Black player is shown in blue and red player in red. In Fig. 4, the green curve shows the number of free positions on the board.

In the first game shown in Fig. 1, the player tries to

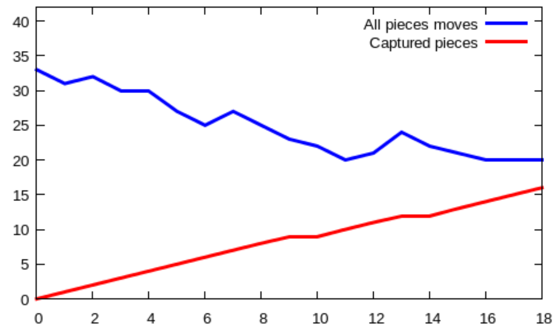

Fig. 1. Single player 4x8 game.

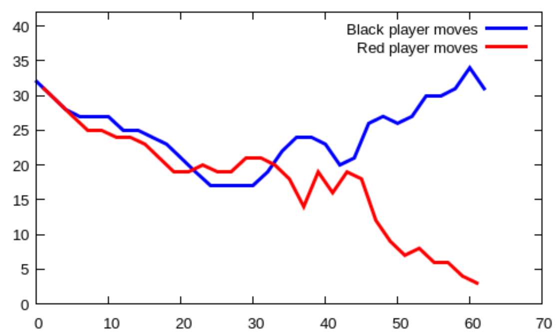

Fig. 2. Short 4x8 traditional game.

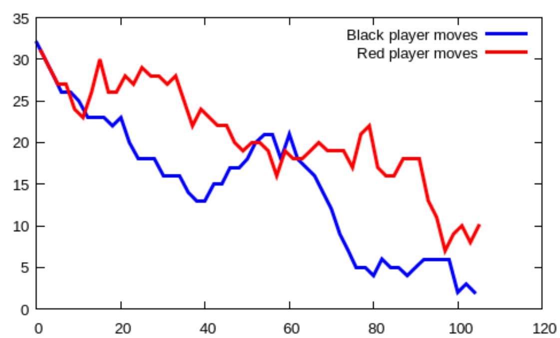

Fig. 3. Competitive 4x8 traditional game.

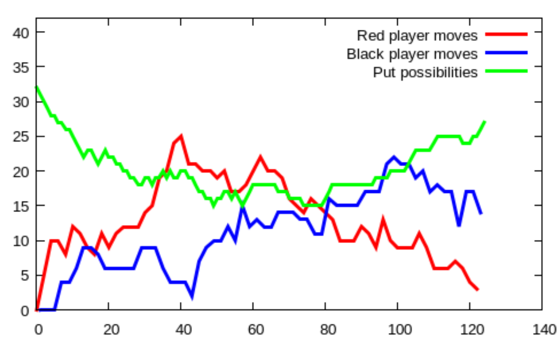

Fig. 4. 4x8 strategic game.

maximize the coverage of one set of pieces regarding to the other set until one color is fully localized. In a second phase, the goal is to reduce the path to capture all pieces (considered as opponent pieces).

In the second game shown in Fig. 2, a strong first player is facing a weak second player. At the end, the strong player wins. It shows that in traditional variant, the complexity stays the same during most of the time. If it decreases drastically for one player while increasing for the other player, then it raises one player win.

In the third game shown in Fig. 3, players' level are equivalent. It makes a game that is more competitive than the first. During the first 40 plies, the red player takes advantage and keep it until the end by reducing black player's moves thus constraining him to reveal moves. It shows that for the winning player, the number of moves is at least as important as for the other player in most cases. For the winner, the state space remains important until 5 to 10 plies before the end.

In the fourth game shown in Fig. 4, players' level are equivalent. Each player stacks units and try to constraint opponents pieces. If a player gains a material advantage, then the game will probably evolve in a victory for this player. As it appears in the figure, when puts possibilities (in green) increase with one player (blue) moves while the other player's (red) moves decrease, it means that the first player is taking benefit of its position.

IV. Conclusion

Varying complexity in games is a way to develop more general algorithms. Chinese Dark Chess is an interesting stochastic game that allows to simply vary many parameters, from pieces inside sets to rules to put pieces on the board. We have presented a general protocol that allows to play this game with different degrees of complexity. This protocol allows to play with other players, even with a cooperative mode. We believe that providing variants based on Chinese Dark Chess will promote the game and helps researchers to enhance their knowledge in stochastic games.

References

[1] B-N. Chen and B-J. Shen and T-S. Hsu. *Chinese Dark Chess*. ICGA Journal, 2010, vol. 33, num. 2, pp. 93.
[2] Jr-C. Chen and T-Y. Lin and S-C. Hsu and T-S. Hsu, *Design and Implementation of Computer Chinese Dark Chess Endgame Database*. TCGA Computer Game Workshop (TCGA-2012).
[3] S-J. Yen and C-W. Chou and Jr-C. Chen and I-C. Wu and K-Y. Kao, *The Art of the Chinese Dark Chess Program DIABLE*. Proc. of the Int. Computer Symposium (ICS-2012).
[4] M. Lanctot and A. Saffidine and J. Veness and C. Archibald and M. Winands, *Monte Carlo *-Minimax Search*, 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI-2013).
[5] H-J. Chang and T-S. Hsu. *A Quantitative Study of 24 Chinese Dark Chess*, Proc. of the 8th Int. Conf. on Computers and Games (CG-2013).
[6] A. Saffidine and N. Jouandeau and C. Buron and T. Cazenave, *Material Symmetry to Partition Endgame Tables*, Proc. of the 8th Int. Conf. on Computers and Games (CG-2013).