Soft Computing (2024) 28:3321–3340
https://doi.org/10.1007/s00500-023-09437-6

SOFT COMPUTING IN DECISION MAKING AND IN MODELING IN ECONOMICS

A hierarchical branch and bound algorithm for Mahjong deficiency

Xueqing Yan[1] · **Yongming Li**[2]

Accepted: 1 November 2023 / Published online: 7 December 2023

Abstract
As a testbed for the development of artificial intelligence (AI) techniques, Mahjong has become a hot spot in computer game research due its characteristics of imperfect information. An important aspect of the game is the evaluation of the quality of a hand, which significantly influences the decision-making process for players, including discarding tiles, pong, kong, and so on. In this paper, an effective and efficient algorithm is proposed to measure the quality (i.e., deficiency) of a Mahjong hand by introducing a hierarchical branch and bound method. In the proposed approach, an updating knowledge-based octree search method is first developed to explore all possible quasi-decompositions for one hand to ensure the correctness of the result. Meanwhile, an updated knowledge-based evaluation method is designed to calculate the cost of each quasi-decomposition, and a hierarchical branch and bound method is presented to accelerate the efficiency of getting its deficiency by layering the quasi-decompositions according to the number of their tiles. Moreover, the block decomposition is further adopted in the process of finding all quasi-decompositions to enhance its efficiency. Compared to existing methods, the proposed algorithm not only demonstrates computational efficiency but also provides the exact deficiency in all cases. Experimental results on different types of hands show its effectiveness and efficiency.

Keywords Mahjong · Deficiency · Branch and bound · Knowledge base · Decomposition

1 Introduction

As we all know, games are always regarded as important platforms to develop and enhance artificial intelligence (AI) techniques, and many games and their AI systems have been studied and presented over the last decades (Berliner 1977; Schaeffer et al. 2007; Wiering 2010; Shannon 1950; Silver et al. 2017; Samuel 1959; Bowling et al. 2015; Schmid et al. 2017; Brown and Sandholm 2018; Mizukami and Tsuruoka 2015; Brown and Sandholm 2019; Holcomb et al. 2018; Heinrich and Silver 2016; Yoshimura et al. 2016; Brown and Sandholm 2017; Sandholm 2018; Brown and Sandholm 2019; Zhao et al. 2022; Rong et al. 2019; Jiang et al. 2019), including Go, Chess, Backgammon, Checkers, Poker, Mahjong, and so on. Specifically, games can be classified simply as those with perfect information and those with imperfect information, and it is usually more difficult to study games with imperfect information than those with perfect information. In fact, for games with perfect information, such as go (Silver et al. 2017), chess (Shannon 1950) and checkers (Samuel 1959), the players (agents) can always have full knowledge of both their own states and the states of their opponents throughout the entire game process. On the other hand, for games with imperfect information, such as poker (Bowling et al. 2015) and Mahjong (Mizukami and Tsuruoka 2015), the agents cannot accurately obtain complete knowledge of the game states at all times. Therefore, the search algorithms used for games with perfect information to determine the response action cannot be directly applied to games with imperfect information. Thus, it is vital and challenging to develop AI systems for games with imperfect information.

Among the games with imperfect information, Mahjong contains a huge amount of invisible information during the game, and its rules are more complex. Thus, it can be regarded

✉ Yongming Li
liyongm@snnu.edu.cn

Xueqing Yan
xueqingyan@snnu.edu.cn

[1] School of Computer Science, Shaanxi Normal University, No.620, West Chang'an Avenue, Xi'an 710119, Shaanxi, China

[2] School of Mathematics and Statistics, Shaanxi Normal University, No.620, West Chang'an Avenue, Xi'an 710119, Shaanxi, China

as one of the typical representatives of games with imperfect information. Currently, Mahjong has been widely studied by AI researchers and has achieved many remarkable results (Li et al. 2020; Kurita and Hoki 2021; Mizukami and Tsuruoka 2015; Sato et al. 2017; Li and Yan 2019; Wang et al. 2020; Yan et al. 2021; Atsushi et al. 2014; Tang 2014; Silver 2017; Gao et al. 2019; Yoshimura et al. 2016; Wang et al. 2019; Ueno et al. 2019; Cheng et al. 2020; Zheng and Li 2020; Wang et al. 2022; Zhang et al. 2022; Gao and Li 2022). For example, by designing and introducing global reward prediction, oracle guiding and run-time policy adaption, Li et al. (2020) proposed a Mahjong AI system called suphx based on reinforcement learning, and the test results on the "Tenhou" platform showed its effectiveness. Kurita and Hoki (2021) abstracted the Mahjong process by defining multiple Markov decision processes, and then constructed an effective search tree for optimal decision-making. The proposed program showed excellent performance especially when the player's hand was close to winning. Moreover, to effectively improve the accuracy of the algorithm, Mizukami and Tsuruoka (2015) presented a novel method for building a Mahjong program by modeling opponent players and performing Monte Carlo simulation with the models. The test results on "Tenhou" platform showed that the proposed approach is effective. In addition, to improve the accuracy of the AI system for games with imperfect information, based on the fact that the players' strategy for matching opponent players is usually better than their own best strategy in the game process, Sato et al. (2017) proposed a new method to classify the opponent players' strategy by analyzing Mahjong playing records, and the experimental results verified the effectiveness of this idea and method. Specifically, more works of AI Mahjong can be further found in Zheng and Li (2020), where the advantages and disadvantages of each method are analyzed and future development directions are also provided. While these works have made significant contributions to the field of Mahjong AI, they often require a large number of game records obtained from human professional players for training, which can be difficult to obtain. Additionally, their underlying rationales are often difficult to explain and demonstrate. Therefore, it is necessary and desirable to develop a new Mahjong AI system that is more implementable and explainable. Clearly, efficiently making a reasonable response for the player is a crucial aspect of the Mahjong AI program, and this is always influenced significantly by the state of the game. Specifically, for a given hand, the number of the tiles needed to change to form a winning hand (i.e., deficiency) might be helpful to evaluate the quality of the hand, and then effectively guide the player to make a more rational decision about his/her response action. Therefore, it is vital to design a promising approach for calculating the deficiency of a hand.

To effectively calculate the deficiency of one hand, many search algorithms have also been developed in recent years (Li and Yan 2019; Wang et al. 2020, 2022; Zhang et al. 2022; Yan et al. 2021). Specifically, Li and Yan (2019) presented a quadtree-based search algorithm (QSA) by finding and evaluating all the pseudo-decompositions to compute the deficiency of a Mahjong hand. Wang et al. (2020) developed a new search approach (WRA) by constructing a weighted restarting automaton over the tropical semiring to determine the deficiency. Furthermore, based on the numbers of pseudomelds and isolated tiles, Wang et al. (2022) proposed a specific algorithm to compute the deficiency number. Additionally, Zhang et al. presented a method for calculating the deficiency number by taking into account the numbers of melds, jokers and pairs and their relationships in Zhang et al. (2022). Although these methods could evaluate the quality of a hand to a certain extent, the information of all visible tiles on the table is not considered in them, which might lead to inaccuracy on the deficiency of the hand in many cases. Moreover, to alleviate the shortcoming above, Yan et al. (2021) further developed an enhanced block deficiency algorithm (BDA) by taking into consideration the information of all visible tiles on the table and using the type of each quasi-decomposition to compute its cost. Even though the proposed block method further improves the correctness and search efficiency of the algorithm, the updated knowledge base is still not fully considered and the use of types would omit some different quasi-decompositions, which could easily occur over pessimistic or optimistic estimation of the deficiency in some cases. Furthermore, during the calculation of deficiency, all quasi-decompositions or types of one hand are always required to be evaluated in Li and Yan (2019); Yan et al. (2021), which is very exhaustive. Thus, it is necessary to further develop a more promising search algorithm for the deficiency of one hand.

Based on the above considerations, this paper presents a novel effective and efficient algorithm (HBBA) to calculate the deficiency of one Mahjong hand by introducing a hierarchical branch and bound method. In particular, the main contributions of this paper are as follows.

1) To ensure the correctness of the deficiency for a hand, an updating knowledge-based octree search (UKOS) method is developed to explore all possible quasi-decompositions, and an updating knowledge-based evaluation (UKE) method is designed to calculate the cost of each quasi-decomposition by fully utilizing the information of both the tiles used in the hand and its knowledge base. Unlike QSA (Li and Yan 2019) and BDA (Yan et al. 2021), the UKOS method makes full use of the knowledge base and always considers all possible cases to create the quasi-decompositions during the search process, while compared with the methods in Li and

Yan (2019); Wang et al. (2020); Yan et al. (2021), the UKE method dynamically updates the knowledge base for each quasi-decomposition and employs four special approaches to evaluate its cost by making full use of its corresponding knowledge base. Then these proposed methods can effectively guarantee the effectiveness and correctness of the calculation.

2) To accelerate the calculation efficiency of the deficiency for a hand, a hierarchical branch and bound (HBB) method is presented by layering all quasi-decompositions according to the number of their tiles and gradually evaluating their costs with logical reasoning. Specifically, the computation process will be terminated once the minimal or sub-minimal cost is obtained in the current layer. Differing from QSA (Li and Yan 2019) and BDA (Yan et al. 2021), the HBB method only requires evaluating a few layers when the deficiency is obtained. Then the proposed method can effectively enhance the calculation efficiency for the deficiency of a hand.
3) Considering the fact that the block decomposition (BD) technique can speed up the process of finding all quasi-decompositions for a hand (Yan et al. 2021), this method is further adopted in the proposed algorithm. Then the proposed algorithm has a more promising performance.

Therefore, the proposed algorithm can not only obtain a correct deficiency for each hand but also has promising computation efficiency. Finally, numerical experiments are carried out to demonstrate the performance of the proposed algorithm by comparing it with three related methods on three classes of datasets. The experimental results show that the proposed algorithm can consistently obtain the correct deficiency number and has more efficient performance.

The remainder of this paper is organized as follows. Section 2 gives the related works and preliminaries. The proposed algorithm is presented in Sect. 3. Section 4 provides and discusses the experimental results. Finally, conclusions are drawn in Sect. 5.

2 Backgrounds

In this section, preliminaries and two foundational deficiency algorithms shall be described.

2.1 Preliminaries

Herein, some common terms and related concepts of this paper, including deficiency, knowledge base and decomposition, will be introduced, respectively.

2.1.1 Common terms

This paper mainly considers the version of the Mahjong game where only the colors of *bamboo* (B), *character* (C), and *dot* (D) are included. Specifically, this includes one bamboo to nine bamboo ($B1 - B9$), one character to nine character ($C1 - C9$), and one dot to nine dot ($D1 - D9$). The set of these 27 different tiles above is denoted by MJ, with each tile having four identical ones and a total of 108 tiles. For convenience, this subsection provides the relevant terms and meanings (concepts) in the Mahjong game, which are listed in Table 1.

From Table 1,[1] one can see that a meld is a chow, kong or a pong. A chow or pong can be formed by other players' discarded tile, while a kong can be formed by drawing four identical tiles (concealed kong), using the fourth identical tile with the exposed pong (exposed kong from pong), or using three identical tiles with the one identical tile discarded from other players (exposed kong). Additionally, a winning hand refers to a hand that conforms to the winning rules. It is worth noting that, except for drawing a tile from the wall, a player can also win using other players' discarded tiles or by robbing a kong.

In particular, for a winning hand T, a decomposition π of T is a sequence of five subsequences of T such that $\pi[i]$[2] $(1 \le i \le 4)$ is a meld and $\pi[5]$ is a pair (if the case, $\pi[5]$ is also called an eye. Moreover, the combination of two tiles that can form a chow or a pong is called premeld (abbr. pmeld), such as $(B3B4)$ or $(C5C5)$, where the former is also called a prechow (abbr. pchow).

2.1.2 Concepts

Here, the related concepts of the Mahjong game will be given, including deficiency, knowledge base and decomposition.

Definition 1 (deficiency Li and Yan (2019)) The *deficiency number* (or simply *deficiency*) of a 14-tile T is defined recursively:

- T has deficiency 0 if it is a winning (complete) hand;
- In general, for $\ell \ge 0$, T has deficiency $\ell + 1$ if it has no deficiency smaller than or equal to ℓ and there exists a tile t in T and another tile t' s.t. $T[t/t']$ (the 14-tile obtained from T by replacing t with t') has deficiency ℓ.

[1] In this paper, any sequence of tiles is regarded as a multiset, which allows multiple instances for each of its elements. The usual set operations $\cup, \cap, \setminus$ are also defined for multisets. Here, a multiset, e.g., $\{B3, B4, B9, D7, D9\}$ is also written as ordered tuples like $(B3B4B9)(D7D9)$.

[2] Suppose S is a sequence which includes k elements. We write $S[i]$ for the $(i + 1)$-th value of S for $0 \le i \le k - 1$, and $S[0]$ is the first element of S, as represented in some programming languages.

Table 1 Related terms and concepts in Mahjong

Common terms	Explanation
hand	A sequence of 13 or 14 tiles the player has drawn
chow	A sequence of three consecutive tiles of the same color, such as $(B3B4B5)$
kong	A sequence of four identical tiles, such as $(B9B9B9B9)$
pong	A sequence of three identical tiles, such as $(C5C5C5)$
pair	A pair of identical tiles, such as $(D7D7)$
meld	A chow, kong, or a pong
winning (complete) hand	A hand that can be decomposed into four melds and one eye, such as T=$(B1B2B3B5B5B7B7B7)(C1C2C3C7C7C7)$
exposed kong	The player declares a kong by a pong in his hand and a discarded tile from other players
concealed kong	The player declares a kong by identical four tiles in his hand
exposed kong from pong	The player declares a kong by an exposed pong and the fourth identical tile in his hand
robbing a kong	The player achieves a winning hand by one tile which another player is currently trying to use to form an exposed kong from pong.

If the deficiency of T is ℓ, we write $\mathsf{dfncy}(T) = \ell$. Particularly, if T is a winning (complete) hand, *i.e.*, $\mathsf{dfncy}(T) = 0$, we also say T is complete for simplicity.

To be precise, for a 14-tile T, the deficiency of T is the minimum number of tile changes needed for T to form a winning hand. Here, the tile change operations can be achieved by drawing a tile from the wall, forming a pong or chow or winning hand from other players' discarded tiles, or robbing a kong from other players' tile that was previously used to form an exposed kong from an exposed pong.

For example, for a 14-tile $T = (B1B2B3B5B5B7B7B9)(C1C2C3C7C7C7)$, since a winning hand can be obtained when $B9$ is replaced with $B5$ or $B7$, or $B7$ is replaced with $B8$, then the deficiency of T is 1. However, it is worth noting that during the replacement process above, there must be at least one among the tiles $B5$, $B7$ and $B8$ in the set of invisible tiles. If not, we can see that the deficiency of T will not be 1. This means that the number of some invisible tiles has an important impact on the calculation result of deficiency. For clarity, the concept of the knowledge base is introduced and given below.

Definition 2 (knowledge base) The *knowledge base* of the agent is a 27-tuple KB. For each tile $t \in MJ$, $KB(t)$ denotes the remaining number of t the agent *believes* to be available, which can be expressed as follows:

$$KB = \{(x_0, x_1, x_2, \cdots, x_{26}) \mid 0 \leq x_i \leq 4, 0 \leq i \leq 26\} \quad (1)$$

where $x_i (0 \leq i \leq 8)$ represents the remaining number of $B1 - B9$, $x_i (9 \leq i \leq 17)$ represents the remaining number of $C1 - C9$, and $x_i (18 \leq i \leq 26)$ represents the remaining number of $D1 - D9$ respectively.

In what follows, the update rule of $KB(t)$ for each tile t is given (initially, $KB(t) = 4$). When the agent has drawn its 13-tile (14-tile) hand at the beginning of the game, we will let $KB(t)$ = 4 - the number of t in the agent's hand. After this, the value of $KB(t)$ will be updated as follows:

- if t has been drawn by the agent, $KB(t) = KB(t) - 1$.
- if t has been discarded or used to form an exposed kong from an exposed pong by the other player, $KB(t) = KB(t) - 1$.
- if t has been ponged by the other player, then $KB(t) = KB(t) - 2$.
- if t has been used to form a concealed kong by the other player, then $KB(t) = KB(t) - 4$.
- if t has been drawn by the other player and then declared to form a winning hand, then $KB(t) = KB(t) - 1$.
- if t is chowed by the other player, after the three tiles are sorted alphabetically, consider the following three cases:
  a. when t is the left one, then $KB(t^+)$[3] $= KB(t^+) - 1$ and $KB(t^{++}) = KB(t^{++}) - 1$.
  b. when t is the middle one, then $KB(t^-) = KB(t^-) - 1$ and $KB(t^+) = KB(t^+) - 1$.
  c. when t is the right one, then $KB(t^{--}) = KB(t^{--}) - 1$ and $KB(t^-) = KB(t^-) - 1$.

For clarity, the last example is also used here with $T = (B1B2B3B5B5B7B7B9)(C1C2C3C7C7C7)$. If the player's knowledge base satisfies $KB(B5) > 0$, $KB(B7) > 0$ or $KB(B8) > 0$, then the deficiency of T is 1; Otherwise, if $KB(B3) > 0$ and $KB(B4) > 0$ hold, we can use $B5$ or $B7$ to form a meld by changing $B9$ to $B3$ and changing $B5$ to $B4$, then the deficiency of T is 2. Based on this fact, the

[3] Given a tile t, let t^+ and t^{++} denote the tiles which the number of t increasing one and two respectively, t^- and t^{--} denote the tiles which the number of t decreasing one and two respectively.

concept of deficiency which can reflect the knowledge base is further given, i.e., knowledge-aware deficiency.

Definition 3 (knowledge-aware deficiency Yan et al. (2021)) The *deficiency number* (or simply *deficiency*) of a 14-tile T w.r.t. a knowledge base KB is defined recursively:

- T has deficiency 0 if it is complete;
- In general, for $\ell \geq 0$, T has deficiency $\ell + 1$ if it has no deficiency smaller than or equal to ℓ and there exists a tile t in T and another tile t' that is available in KB (i.e., $KB(t') > 0$) s.t. $T[t/t']$ has deficiency ℓ w.r.t. the updated knowledge base KB' in which $KB'(t') = KB(t') - 1$.

If the deficiency of T is ℓ, we write $\mathsf{dfncy}(T, KB) = \ell$. We say T is incompletable if it has no finite deficiency.

In the following, we will further introduce the concept of quasi-decomposition in Yan et al. (2021), where the knowledge base is respected.

Definition 4 (quasi-decomposition Yan et al. (2021)) Let T be a k-tile and KB a knowledge base. A quasi-decomposition (qDCMP) $\pi = (\pi[0], \pi[1], \cdots, \pi[k])$ of T w.r.t. KB is a set of (possibly identical) subsequences of T s.t.

- $k \leq 4$ and each $\pi[i]$ is a meld, a pair, or a pchow.
- If $k = 4$, π contains at least one pair.
- Except at most one pair, all pmelds in π are completable under KB.
- $\bigcup_{i=0}^{k} \pi[i]$ is contained in T.

The *remainder* of π in T is the sequence of tiles in T that are not in $\bigcup_{i=0}^{k} \pi[i]$.

2.2 Two deficiency algorithms

In the following, the main contributions of the quadtree algorithm (QSA) Li and Yan (2019) and the block deficiency algorithm (BDA) Yan et al. (2021) shall be described, respectively.

2.2.1 The quadtree algorithm

As mentioned in Li and Yan (2019), the quadtree method determines the deficiency of one hand T by constructing and evaluating all its possible pseudo-decompositions exhaustively. Here, a pseudo-decomposition (pDCMP) is a sequence π of five sequences, $\pi[1]$, ..., $\pi[5]$, s.t. $\pi[5]$ is a pair, a single tile, or empty, and for $1 \leq i \leq 4$, each $\pi[i]$ is a meld, a pmeld, a single tile, or empty. In addition, $\pi[0] = T \setminus \cup_{i=1}^{5} \pi[i]$ is the sequence of remaining tiles of T. In detail, to get all the possible pDCMPs of one hand T, the following four actions (A_1, A_2, A_3, and A_4) are employed to construct a quadtree. We note here that each node α is a string over $\Sigma = \{1, 2, 3, 4\}$ and is attached with both a pDCMP π_α and a subsequence S_α, S_α denotes the set of tiles remaining to be processed, and $t = S_\alpha[0]$ is the first tile in S_α.

A_1 (delete t). Define $S_{\alpha 1} = S_\alpha \setminus (t)$ and $\pi_{\alpha 1} = \pi_\alpha$.

A_2 (make chow). If t^+ or t^{++} is in S_α, define $S_{\alpha 2} = S_\alpha \setminus (tt^+t^{++})$, $\pi_{\alpha 2}[i] = (tt^+t^{++}) \cap S_\alpha$ and $\pi_{\alpha 2}[j] = \pi_\alpha[j]$, where i is the first index in $\{1, 2, 3, 4\}$ s.t. $\pi_\alpha[i] = \varnothing$, and $j \neq i$ (the same below).

A_3 (make eye). If $(tt) \subseteq S_\alpha$ and $\pi_\alpha[5]$ is empty, define $S_{\alpha 3} = S_\alpha \setminus (tt)$ and $\pi_{\alpha 3}[5] = (tt)$ and $\pi_{\alpha 3}[j] = \pi_\alpha[j]$ for $1 \leq j \leq 4$.

A_4 (make pong). If $(tt) \subseteq S_\alpha$, define $S_{\alpha 4} = S_\alpha \setminus (ttt)$, $\pi_{\alpha 4}[i] = (ttt) \cap S_\alpha$ and $\pi_{\alpha 4}[j] = \pi_\alpha[j]$.

In particular, the concrete construction process of the quadtree can be described as follows. First, an empty set S_{pDCMP} and an empty queue Q are respectively generated to store the found quasi-decompositions and the nodes that need to be further explored, and put the root node $\alpha = (S_\alpha, \pi_\alpha)$ in the queue Q, where $S_\alpha = T$ and π_α is an empty set. Then, the first node $\alpha = (S_\alpha, \pi_\alpha)$ is popped out from Q, and the four actions mentioned above are used to create its corresponding child nodes. Next, for each child node, if its S_α is empty, it will be added into S_{pDCMP}; otherwise, it will be added into Q. The procedures above will be continuously executed until Q is empty.

Moreover, the evaluation process of pDCMP in the quadtree method can be shown as follows. First, the cost of a pDCMP ($cost_T(\mathsf{pDCMP})$) is predefined by the number of necessary tile changes to complete it. Besides, the following cases are further considered to adjust the value of cost. (1) When some $\pi_\alpha[i]$ for $1 \leq i \leq 5$ are empty in pDCMP, its cost $cost_T(\mathsf{pDCMP})$ will be further updated as $cost_T(\mathsf{pDCMP}) - n_e$, where n_e denotes the number of empty $\pi_\alpha[i]$ for $1 \leq i \leq 4$. (2) When $\pi_\alpha[5]$ is empty and the number of all tiles belonging to $\pi_\alpha[0]$ in T is smaller than 3, its cost $cost_T(\mathsf{pDCMP})$ will be further updated by $cost_T(\mathsf{pDCMP}) = cost_T(\mathsf{pDCMP}) - 1$.

Finally, the deficiency of T is obtained by finding the smallest one among all $cost_T(\mathsf{pDCMP})$s of its pDCMPs.

In summary, the framework and the idea diagram of the quadtree algorithm can be found in Algorithm 1 and Fig. 1, respectively.

2.2.2 The block deficiency algorithm

The block deficiency algorithm (BDA) is an enhanced version of the quadtree algorithm (Yan et al. 2021). Differing from the quadtree algorithm (Li and Yan 2019), all possible types of qDCMPs for T are designed and employed in it to

Algorithm 1 The framework of the quadtree algorithm

Input: A Mahjong hand T.
Initialize the deficiency of T $dfncy = 100$;
Set $S_{\mathsf{pDCMP}} = \varnothing$ and $Q = \varnothing$;
Create the root node $\alpha = (S_\alpha, \pi_\alpha)$, where $S_\alpha = T$ and $\pi_\alpha = \varnothing$;
Add the root node α into Q;
while $Q \neq \varnothing$ **do**
  Pop out the first element from Q denoted as α;
  for $\ell = 1 : 4$ **do**
    Generate the corresponding child node α by the ℓ-th action A_ℓ;
    if S_α is empty **then**
      add the child node α into S_{pDCMP};
    else
      add the child node α into Q;
    end if
  end for
end while
for each pDCMP in S_{pDCMP} **do**
  calculate its cost $cost_T$(pDCMP);
  if $cost_T$(pDCMP) $< dfncy$ **then**
    Let $dfncy = cost_T$(pDCMP);
  end if
end for
Output: $dfncy$.

calculate the deficiency, and the knowledge base is also considered in it. Specifically, the detailed operations of BDA can be described as follows.

First, the hand T is divided into different disconnected blocks (the set of blocks denoted as S_B), and for each block B, all its possible qDCMPs (the set of qDCMPs denoted as S^l_{qDCMP}) are obtained by constructing a 6-ary tree, whose detailed procedures are similar to that in Li and Yan (2019), where the node α is further attached with seven attributes (m,n,p,e,re,rm,em). Herein, m, n and p are the number of melds, pmelds and pairs in π respectively, e is the number of pairs in π which are incompletable, re and rm are the Boolean indexes of whether there is a tile in the remainder tiles, which can be formed as a pair and meld with the current KB respectively, and em is the Boolean index of whether there is the case that when e $= 0$, re $=$ rm $= 1$, and we need to make both a meld and the eye, but we cannot make a meld from a tile in the remainder tiles after making the eye starting from a tile in them. Then, for each block, all different types of its local qDCMPs (the set of different types denoted as S^l_{types}) are decided by comparing their seven attributes, and the various types of each block are fully combined to get all possible types of T, where the set of the all possible types of T is denoted as S^T_{types}. After the generation of S^T_{types}, each type $type_T$ of S^T_{types} is evaluated by its corresponding seven attributes (m,n,p,e,re,rm,em) and another two indicators ke and km, where ke and km are the Boolean indexes of whether there is a pair and a meld in KB, respectively. Finally, according to the cost of each type $type_T$ in S^T_{types}, the smallest one is chosen as the deficiency of T.

In summary, the framework and the idea diagram of the block deficiency algorithm can be found in Algorithm 2 and Fig. 2, respectively.

Algorithm 2 The framework of the block deficiency algorithm

Input: A Mahjong hand T and its knowledge base KB.
Initialize the deficiency of T $dfncy = 100$;
Divide T into different blocks, and their set is denoted by S_B;
for each block B in S_B **do**
  Generate all possible qDCMPs, and their set is denoted by S^l_{qDCMP};
  Find the various types of S^l_{qDCMP}, and their set is denoted by S^l_{types};
end for
Join each type in different S^l_{types} fully to get all possible types of T, and their set is denoted by S^T_{types};
Set the values of ke and km based on KB;
for each type $type_T$ in S^T_{types} **do**
  Calculate its cost $cost_T(type)$ according to its seven attributes (m, n, p, e, re, rm, em) and ke and km;
  if $cost_T(type) < dfncy$ **then**
    Let $dfncy = cost_T(type)$;
  end if
end for
Output: $dfncy$.

Clearly, based on the descriptions of QSA and BDA above, it is evident that the evaluation of pDCMPs in both methods does not fully take into account the knowledge base. As a result, an accurate deficiency number cannot

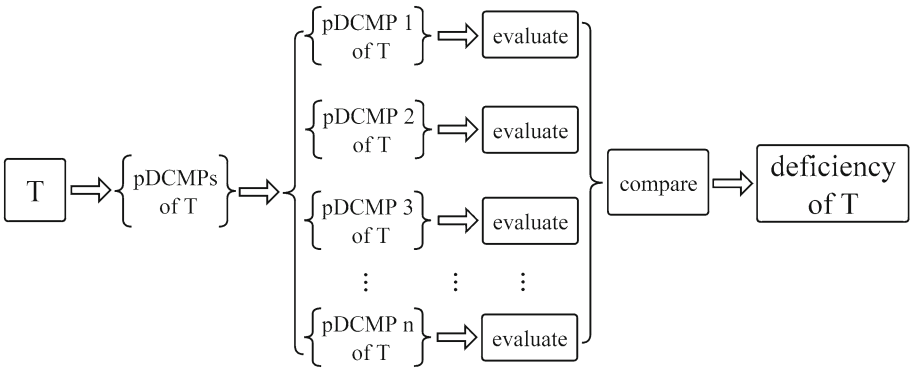

Fig. 1 The idea diagram of the quadtree algorithm

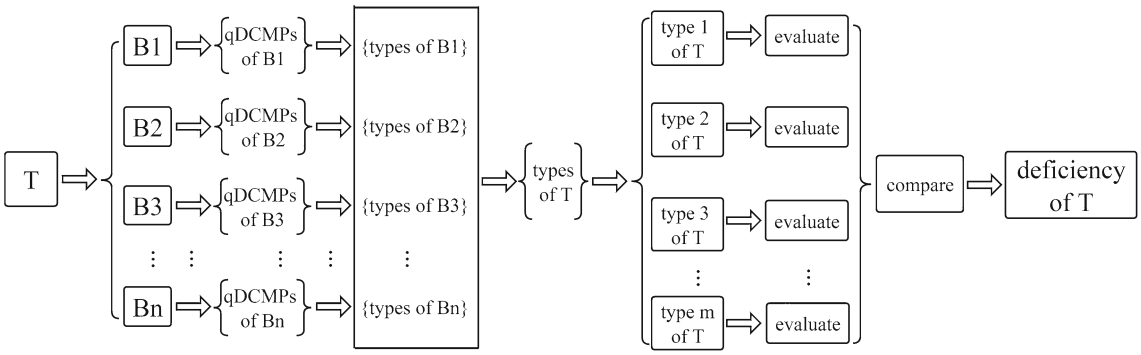

Fig. 2 The idea diagram of the block deficiency algorithm

be obtained. For example, with respect to the hand $T1 = (B1B2B5B6B7)(C1C1C1C4C5C6C7)(D3D3)$, since $T1$ can be completed by replacing the tile $C4$ or $C7$ with $B3$, the deficiency of $T1$ can be calculated as 1 by the rule of QSA. Noticeably, when four tiles of $B3$ have already been revealed on the table, the real deficiency of $T1$ will not be 1, but the one obtained by QSA still is 1. On the other hand, with respect to the hand $T2 = (B1B5B7B8B9)(C2C3C4C8C8C8C9)(D2D5)$ with knowledge base $KB = (012000011)(110001000)(100100110)$ and its one qDCMP $\pi = (B7B8B9)(C2C3C4C8C8C8)$, since $(B7B8B9)$, $(C2C3C4)$ and $(C8C8C8)$ are three melds, and the tile $B1$ in the remainder tiles just can form one meld $(B1B2B3)$ with the tiles $B2$ and $B3$ in its knowledge base KB, one can get $\mathsf{m} = 3, \mathsf{n} = 0, \mathsf{p} = 0, \mathsf{e} = 0, \mathsf{re} = 0, \mathsf{rm} = 1$ and $\mathsf{em} = 0$. Meanwhile, since there is just one pair $(B3B3)$ in its knowledge base KB, one have $\mathsf{ke} = 1$ and $\mathsf{km} = 0$. Then, the cost of this π can be calculated as 4 by the rule of BDA. However, after forming the meld $(B1B2B3)$ based on $T2$ and its knowledge base KB, it is not possible to form the eye $(B3B3)$ in fact, and vice versa. Thus, the real cost of this π should be infinite, but not 4. Moreover, their final steps are often time-consuming to calculate the deficiency, and although the full tree are all used in them to search all possible pDCMPs for a hand, some cases are still neglected, thus leading to incorrect deficiency. Based on these considerations, a novel deficiency algorithm is proposed and described in the following section.

3 The proposed algorithm

As described in the above sections, many methods have been developed to calculate the deficiency of a hand (Li and Yan 2019; Wang et al. 2020, 2022; Yan et al. 2021). However, there are still several shortcomings in them. For example, during the process of finding all possible quasi-decompositions, the knowledge base is always not considered and some quasi-decompositions are often easily omitted. For instance, when a hand owns a pmeld $B3B4$, and it can be expanded to both $B2B3B4$ and $B3B4B5$ according to its corresponding knowledge base KB. However, the method proposed in Yan et al. (2021) only updates its KB in one way, while there are actually two different ways. Moreover, for the evaluation of quasi-decomposition, the information of the knowledge base is not always fully utilized, while all quasi-decompositions are always required to be evaluated during the deficiency calculation, which is exhaustive. To alleviate these drawbacks, we propose a novel search algorithm to calculate the deficiency of a Mahjong hand in this section. Specifically, an updating knowledge-based octree search (UKOS) method and an updating knowledge-based evaluation (UKE) method are proposed to find all possible quasi-decompositions and calculate the cost of each quasi-decomposition, respectively. Meanwhile, a hierarchical branch and bound (HBB) method is presented to calculate the deficiency of the hand by layering all quasi-decompositions according to the number of their tiles, and the block decomposition (BD) technique is further adopted in the proposed algorithm to enhance search efficiency.

For the convenience of the later discussions, let T denote a 14-tile hand, KB denote a knowledge base, π denote a qDCMP of T, R_π and KB_π denote the remainder of T and the modified KB under π respectively. Same as the symbol in BDA, m and n denote the number of melds and pmelds in π respectively, p denotes the number of pairs in all pmelds, e denotes the Boolean index of whether there is a pair acting as an eye. Obviously, one can always have $3 \times \mathsf{m} + 2 \times \mathsf{n} \leq 14$, $\mathsf{n} > \mathsf{p} > \mathsf{e}$ and $\mathsf{e} = 0$ or $\mathsf{e} = 1$.

3.1 UKOS method

As pointed out in Refs. Li and Yan (2019); Yan et al. (2021), it is very vital to find all suitable quasi-decompositions for one hand to get its real deficiency. However, this is still not achieved because the knowledge base is not fully considered and some quasi-decompositions are always omitted in this process. Thus, to do this, we propose the following UKOS

method to find all possible quasi-decompositions for one hand.

Similar to the quadtree search method (Li and Yan 2019), in the UKOS method, each node α is represented as a string over $\Sigma = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and is attached with $(\pi_\alpha, KB_\alpha, F_\alpha, S_\alpha)$, which is created by a possible action except for the root node. Here, π_α represents a qDCMP of T, KB_α represents a modified KB under π_α, F_α represents the value of a four-tuple (m,n,p,e), and S_α represents the set of tiles in T that are yet to be processed. In the UKOS method, there are eight actions, denoted as $ACTION = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\}$, which are used to create the child nodes. Specifically, the concrete procedure of the UKOS method can be described as follows.

First, an empty set S_{qDCMP} and an empty queue Q are generated, which will be used to store the found quasi-decompositions and the nodes to be further explored respectively, the root node $\varepsilon = (\pi_\varepsilon, KB_\varepsilon, F_\varepsilon, S_\varepsilon)$ is created with $S_\varepsilon = T$, $KB_\varepsilon = KB$, $F_\varepsilon = (0, 0, 0, 0)$ and $\pi_\varepsilon = \{\pi_\varepsilon[i] = \varnothing \mid i = 0, 1, 2, 3, 4\}$, and we put the root node ε in Q.

Then, the first node $\alpha = (\pi_\alpha, KB_\alpha, F_\alpha, S_\alpha)$ in the queue Q is popped out, and the eight actions mentioned above will be respectively used for it to create its corresponding eight child nodes $\alpha\ell = (\pi_{\alpha\ell}, KB_{\alpha\ell}, F_{\alpha\ell}, S_{\alpha\ell})$, where $\ell \in \{1, 2, 3, 4, 5, 6, 7, 8\}$. Specially, the detail process of each action A_ℓ to generate its corresponding child node $\alpha\ell = (\pi_{\alpha\ell}, KB_{\alpha\ell}, F_{\alpha\ell}, S_{\alpha\ell})$ shall be described in the following, where let $t = S_\alpha[0]$ be the first tile in S_α, and $S_{\alpha\ell}, \pi_{\alpha\ell}, KB_{\alpha\ell}$ and $F_{\alpha\ell}$ are initially set to $S_\alpha, \pi_\alpha, KB_\alpha$ and F_α, respectively.

A_1 (delete t). Set $S_{\alpha 1} = S_\alpha \setminus (t)$.

A_2 (make chow). If t^+ and t^{++} is in S_α, set $S_{\alpha 2} = S_\alpha \setminus (tt^+t^{++})$, $F_{\alpha 2}(\mathsf{m}) = F_\alpha(\mathsf{m}) + 1$ and $\pi_{\alpha 2}[i] = (tt^+t^{++})$ and $\pi_{\alpha 2}[j] = \pi_\alpha[j]$, where i is the first index in $\{0, 1, 2, 3, 4\}$ with $\pi_\alpha[i] = \varnothing$ and $j \neq i$ (the same as below).

A_3 (make pchow-type1). If t^+ is in S_α and $KB(t^-) > 0$, set $S_{\alpha 3} = S_\alpha \setminus (tt^+)$, $\pi_{\alpha 3}[i] = (tt^+)$, $\pi_{\alpha 3}[j] = \pi_\alpha[j]$, $KB_{\alpha 3}(t^-) = KB_\alpha(t^-) - 1$, and $F_{\alpha 3}(\mathsf{n}) = F_\alpha(\mathsf{n}) + 1$.

A_4 (make pchow-type2). If t^+ is in S_α and $KB(t^{++}) > 0$, set $S_{\alpha 4} = S_\alpha \setminus (tt^+)$, $\pi_{\alpha 4}[i] = (tt^+)$, $\pi_{\alpha 4}[j] = \pi_\alpha[j]$, $KB_{\alpha 4}(t^{++}) = KB_\alpha(t^{++}) - 1$, and $F_{\alpha 4}(\mathsf{n}) = F_\alpha(\mathsf{n}) + 1$.

A_5 (make pchow-type3). If t^{++} is in S_α and $KB(t^+) > 0$, set $S_{\alpha 5} = S_\alpha \setminus (tt^{++})$, $\pi_{\alpha 5}[i] = (tt^{++})$, $\pi_{\alpha 5}[j] = \pi_\alpha[j]$, $KB_{\alpha 5}(t^+) = KB_\alpha(t^+) - 1$, and $F_{\alpha 5}(\mathsf{n}) = F_\alpha(\mathsf{n}) + 1$.

A_6 (make pair). If $(tt) \subseteq S_\alpha$ and $KB(t) > 0$, set $S_{\alpha 6} = S_\alpha \setminus (tt)$, $\pi_{\alpha 6}[i] = (tt)$, $\pi_{\alpha 6}[j] = \pi_\alpha[j]$, $KB_{\alpha 6}(t) = KB_\alpha(t) - 1$, and $F_{\alpha 6}(\mathsf{n}) = F_\alpha(\mathsf{n}) + 1$, $F_{\alpha 6}(\mathsf{p}) = F_\alpha(\mathsf{p}) + 1$.

A_7 (make eye). If $e = 0$ and $(tt) \subseteq S_\alpha$, set $S_{\alpha 7} = S_\alpha \setminus (tt)$, $\pi_{\alpha 7}[i] = (tt)$, $\pi_{\alpha 7}[j] = \pi_\alpha[j]$, and $F_{\alpha 7}(\mathsf{n}) = F_\alpha(\mathsf{n}) + 1$, $F_{\alpha 7}(\mathsf{p}) = F_\alpha(\mathsf{p}) + 1$, $F_{\alpha 7}(\mathsf{e}) = 1$.

A_8 (make pong). If $(ttt) \subseteq S_\alpha$, set $S_{\alpha 8} = S_\alpha \setminus (ttt)$, $\pi_{\alpha 8}[i] = (ttt)$, $\pi_{\alpha 8}[j] = \pi_\alpha[j]$, and $F_{\alpha 8}(\mathsf{m}) = F_\alpha(\mathsf{m}) + 1$.

Next, for each child node $\alpha\ell$, if its $S_{\alpha\ell}$ is empty, it will be added into S_{qDCMP}; otherwise, it will be added into Q.

Finally, the first node in Q will be continuously popped out and executed with the above operations until Q is empty.

Based on the above descriptions, one can find that the proposed UKOS method employs eight actions to search all possible quasi-decompositions for one hand, and the knowledge base is always dynamically updated during the whole search process. Unlike the methods (Li and Yan 2019; Yan et al. 2021) that do not consider the changing of the knowledge base even for the knowledge base and always miss some quasi-decompositions during the search process, the UKOS method makes full use of the knowledge base and always considers all possible cases to create the quasi-decomposition during the search process. Thus, it could ensure the effectiveness and completeness of the found quasi-decompositions. For clarity, the framework of the UKOS method is also provided in Algorithm 3.

Algorithm 3 The framework of the UKOS method

```
Input: A Mahjong hand T and its knowledge base KB.
Set S_qDCMP = ∅ and Q = ∅;
Create the root node ε = (π_ε, KB_ε, F_ε, S_ε), where S_ε = T, KB_ε = KB, F_ε = (0,0,0,0) and π_ε = {π_ε[i] = ∅ | i = 0,1,2,3,4};
Add the root node ε into Q;
while Q ≠ ∅ do
    Pop out the first element from Q denoted as α;
    for ℓ = 1 : 8 do
        Generate the corresponding child node αℓ = (π_αℓ, KB_αℓ, F_αℓ, S_αℓ) by the ℓ-th action A_ℓ in ACTION;
        if S_αℓ is empty then
            add the child node αℓ into S_qDCMP;
        else
            add the child node αℓ into Q;
        end if
    end for
end while
Output: The set S_qDCMP.
```

3.2 UKE method

In Li and Yan (2019); Wang et al. (2020), the cost of each qDCMP π is just calculated by its own information, where the knowledge base is not considered, which might lead to a severally wrong evaluation for its quality in some cases. Moreover, although the BDA method (Yan et al. 2021) has taken into consideration the fact that the knowledge base shall be dynamically updated during the process of generating the remainder eye and/or melds, the knowledge base is still not fully considered. For example, suppose we need to make both an eye and one meld based on KB_π, where a pair and

a meld have just existed and there is a co-used tile in them, so the indicators ke and km will still be set to 1, and the cost might be wrong. For clarity, the following example is further provided to show this shortcoming.

Example 1 Let

$$T = (B1B1B1B2B4B6B6B8B9)(C1C1C1C6C7)$$
$$KB = (\underline{0,1,0,0,0,0,0,0,1},$$
$$\underline{0,1,0,1,0,0,1,1,1},\underline{0,0,0,1,0,1,0,0,1})$$

It is easy to obtain one of qDCMPs $\pi = (B1B1B1)(C1C1C1)(C6C7)(B6B6)$, where the pmeld $C6C7$ can form a meld using the tile $C8$ in the knowledge base. Obviously, a meld is still needed to complete π, as $\mathsf{rm} = 0$ and $\mathsf{km} = 1$, then the deficiency number will be computed as 4. In fact, there are no melds in the knowledge base as the tile $C8$ has been used to form meld for the pmeld $C6C7$. Thus, the real deficiency is infinite.

Based on the above consideration, the knowledge base should be further comprehensively considered during the process of calculating the cost of each qDCMP. To alleviate this shortcoming, this paper shall present a more promising approach to compute the cost of each π. Specifically, based on the structure of the qDCMP mentioned in the last subsection, the concept of the cost of qDCMP is first given as follows.

Definition 5 (cost) Let π be a qDCMP of a Mahjong hand T under the knowledge base KB, F_π be a feature of π, R_π and KB_π be the remainder of T under π and the modified KB of π, respectively. The *cost* of π, written $cost(\pi, F_\pi, R_\pi, KB_\pi)$, is the number of the missing tiles required to complete four melds and one pair under R_π and KB_π. If a qDCMP is incompletable, then we say it has an infinite cost.

From the definition of cost above, one can intuitively find that the cost of π could be simply calculated by counting the number of the missing tiles required to complete it based on KB_π. So, the calculation formula for the cost of π can be described by

$$cost = 14 - 3 \times \mathsf{m} - 2 \times (\mathsf{n} - \mathsf{e}) - 2 \times \mathsf{e} - \mathsf{ue} - \mathsf{um}, \quad (2)$$

where ue and um denote the maximal number of the tiles in R_π used to complete the eye and the remainder $4-\mathsf{m}-\mathsf{n}+\mathsf{e}$ melds, respectively. Obviously, we has $\mathsf{ue} = 0$ if $\mathsf{e} = 1$, and $\mathsf{um} = 0$ if $\mathsf{m} + \mathsf{n} - \mathsf{e} = 4$. Specially, the value of ue can be given by

$$ue = \begin{cases} 1, & \text{if } \mathsf{e} = 0 \text{ and } f_{rp} = 1, \\ 0, & \text{if } \mathsf{e} = 1 \text{ or if } \mathsf{e} = 0 \text{ and } f_{rp} = 0 \text{ and } f_{kp} = 1, \\ -100, & \text{otherwise,} \end{cases} \quad (3)$$

where f_{rp} and f_{kp} represent the Boolean index of whether there is a tile in R_π which can form an eye based on KB_π and whether there is a pair in KB_π respectively, and $\mathsf{ue} = -100$ means that there is no tile in R_π which can form an eye based on KB_π and there is no pair in KB_π. Meanwhile, the value of um can be given by

$$um = \begin{cases} 0, & \text{if } \mathsf{m} + \mathsf{n} - \mathsf{e} = 4, \\ 4 - \mathsf{m} - \mathsf{n} + \mathsf{e}, & \text{if } \mathsf{m} + \mathsf{n} - \mathsf{e} < 4 \text{ and } f_{rm} \\ & \quad \geq 4 - \mathsf{m} - \mathsf{n} + \mathsf{e}, \\ f_{rm}, & \text{if } 4 - \mathsf{m} - \mathsf{n} + \mathsf{e} < 4, \quad f_{rm} \\ & \quad < 4 - \mathsf{m} - \mathsf{n} + \mathsf{e} \text{ and} \\ & \quad f_{km} \geq 4 - \mathsf{m} - \mathsf{n} + \mathsf{e} - f_{rm}, \\ -100, & \text{otherwise.} \end{cases} \quad (4)$$

where f_{rm} and f_{km} represent the maximal number of the tiles in R_π which can be simultaneously used to form the melds based on KB_π and that of the melds which can be simultaneously created in KB_π respectively, and $\mathsf{um} = -100$ means that there are no enough tiles in R_π to form $4 - \mathsf{m} - \mathsf{n} + \mathsf{e}$ melds based on KB_π and there is no enough melds in KB_π to complete π.

Furthermore, to save computational resources, a more simple and achievable approach for the cost of π is further presented with the following propositions.

Proposition 1 *If $\mathsf{e} = 1$ and $\mathsf{m} + \mathsf{n} - \mathsf{e} = 4$, then the cost of π is $4 - \mathsf{m}$.*

Proof In this case, we already have one eye, m melds and $\mathsf{n} - \mathsf{e}$ pmelds, and have $\mathsf{n} - \mathsf{e} = 4 - \mathsf{m}$. So, the remaining number of the tiles needed to form the winning hand is $14 - 3 \times \mathsf{m} - 2 \times (4 - \mathsf{m}) - 2 \times \mathsf{e}$, which is equal to $4 - \mathsf{m}$. □

Proposition 2 *If $\mathsf{e} = 0$ and $\mathsf{m} + \mathsf{n} = 4$, then the cost of π is $6 - \mathsf{m} - \mathsf{ue}$.*

Proof In this case, we already have m melds and n pmelds, and have $\mathsf{n} = 4 - \mathsf{m}$. Moreover, due to the definition of ue, then the remaining number of the tiles needed to form the winning hand is $14 - 3 \times \mathsf{m} - 2 \times (4 - \mathsf{m}) - \mathsf{ue}$, which is equal to $6 - \mathsf{m} - \mathsf{ue}$. □

Proposition 3 *If $\mathsf{e} = 1$ and $\mathsf{m} + \mathsf{n} - \mathsf{e} < 4$, then the cost of π is $14 - 3\mathsf{m} - 2\mathsf{n} - \mathsf{um}$.*

Proof In this case, we already have one eye, m melds and $\mathsf{n} - \mathsf{e}$ pmelds, and have $\mathsf{n} - \mathsf{e} < 4 - \mathsf{m}$. Then $4 - \mathsf{m} - \mathsf{n} + \mathsf{e}$ melds are needed to form to complete this π. Moreover, due to the definition of um, then the remaining number of the tiles needed to form the winning hand is $14 - 3 \times \mathsf{m} - 2 \times (\mathsf{n} - \mathsf{e}) - 2 \times \mathsf{e} - \mathsf{um}$, which is equal to $14 - 3\mathsf{m} - 2\mathsf{n} - \mathsf{um}$. □

Proposition 4 *If $\mathsf{e} = 0$ and $\mathsf{m} + \mathsf{n} < 4$, then the cost of π is $14 - 3\mathsf{m} - 2\mathsf{n} - \mathsf{u}$, where u denotes the maximal number of*

the tiles in R_π which can be used to simultaneously form an eye and $4-m-n$ melds based on KB_π.

Proof In this case, we already have m melds and n pmelds, and have $n < 4-m$. Then both one eye and $4-m-n$ melds are needed to form to complete this π. Moreover, according to the definition of u, the remaining number of the tiles needed to form the winning hand is $14-3\times m-2\times n-u$. □

In fact, the value of u is very closely related to that of ue, and if $ue = -100$, u will also be set to -100, which means there is no need to further calculate the cost of π and the cost of π is infinite. Moreover, when $ue = 0$ or $ue = 1$, the following two cases shall be considered to decide the value of u.

(i) When $ue = 1$. In this case, one tile in R_π must be used to form an eye with KB_π, and the remainder melds will be formed based on the corresponding modified R_π (denoted as $R_\pi^{'}$) and modified KB_π (denoted as $KB_\pi^{'}$), where $R_\pi^{'} = R_\pi \setminus (t)$, $KB_\pi^{'}(t) = KB_\pi(t) - 1$, and t is a tile belonging to the set $R_t^{'} = \{t \mid t \in R_\pi$ and t can form an eye based on $KB_\pi\}$. Moreover, since there might be more tiles in R_π which can be used to form an eye with KB_π, the maximal number of the tiles in R_π which can be used simultaneously to form an eye and melds based on KB_π would be $1 + um_{max1}$, where um_{max1} represents the maximal value of um for all kinds of $R_\pi^{'}$ and $KB_\pi^{'}$.

(ii) When $ue = 0$. In this case, there is no tile in R_π which can be used to form an eye with KB_π. So, the eye must be formed based on KB_π, and the remainder melds will be formed based on R_π and the corresponding modified KB_π (denoted as $KB_\pi^{''}$), where $KB_\pi^{''}(t) = KB_\pi(t) - 2$, and t is a tile belonging to the set $R_t^{''} = \{t \mid t \in R_\pi$ and $KB_\pi(t) > 1\}$. Moreover, since there might be more pairs in KB_π, the maximal number of the tiles in R_π which can be used simultaneously to form an eye and melds based on KB_π would be um_{max2}, where um_{max2} represents the maximal value of um for all kinds of $KB_\pi^{''}$ based on R_π.

Thus, based on the above discussions and analyses, the value of u will be given by

$$u = max(1 + um_{max1}, um_{max2}). \tag{5}$$

In summary, from the above descriptions and analyses, one can see that the proposed UKE method makes full use of the characteristic of each qDCMP and fully considers the changing of the knowledge base during the process of evaluating its cost. Differing from the QSA (Li and Yan 2019) and WRA (Wang et al. 2020), where the cost of each qDCMP π is just calculated by its own information, the UKE method employs the information of both qDCMP and its corresponding knowledge base during the evaluation process. Meanwhile, unlike the BDA (Yan et al. 2021), where seven attributes and two indicators ke and km are utilized to evaluate the cost of each type, the UKE method updates the knowledge base dynamically for each qDCMP and employs four special approaches to evaluate its cost by making full use of its corresponding knowledge base. Therefore, it could not only effectively save the computational resource, but also correctly calculate the cost of each qDCMP. For clarity, the framework of the UKE method is further provided in Algorithm 4. Herein, we let the cost of a qDCMP be 100 when it cannot be completed.

Algorithm 4 The framework of the UKE method

Input: A Majhong hand T, a qDCMP π, the feature of π F_π = (m,n,p,e), and its corresponding knowledge base KB_π.
Initialize the cost of π $cost = 100$;
if $e = 1$ and $m + n - e = 4$ **then**
  Let $cost = 4 - m$.
end if
if $e = 0$ and $m + n - e = 4$ **then**
  Calculate the remainder R_π of π under T, and the value of ue by Eq.(3);
  Let $cost = 6 - m - ue$.
end if
if $e = 1$ and $m + n - e < 4$ **then**
  Calculate the remainder R_π of π under T, and the value of um by Eq.(4);
  Let $cost = 14 - 3m - 2n - um$.
end if
if $e = 0$ and $m + n - e < 4$ **then**
  Calculate the remainder R_π of π under T, and the value of u by Eq.(5);
  Let $cost = 14 - 3m - 2n - u$.
end if
if $cost > 100$ **then**
  Let $cost = 100$
end if
Output: $cost$.

3.3 HBB method

As mentioned above, the deficiency of one hand is always calculated by evaluating all its quasi-decompositions in Li and Yan (2019); Yan et al. (2021), which might be more exhaustive. Moreover, it should be noted that the branch and bound method is often able to improve the search efficiency for the integer programming problem. Thus, by making full use of the branch and bound method, an HBB method is designed to compute the deficiency of one hand here. In detail, the rationality and concrete operation of the HBB method will be analyzed and described as follows.

In particular, from the definition of the deficiency and the cost for qDCMP, one can easily find that for a given hand T, the qDCMP with more tiles always owns a lower cost in

most cases. So, their costs would be more likely to reach the real deficiency, and the qDCMP with more tiles should be first checked during the process of calculating the deficiency. Moreover, it should be noted that for the qDCMPs which have the same number of tiles, there are at most two combinations of the value of m and n. In fact, for each qDCMP with m melds and n pmelds, there are at most two sets of integer solutions for m and n to meet the constraints $0 \leq \mathsf{m} \leq 4, 0 \leq \mathsf{n} \leq 5$ and $3\mathsf{m} + 2\mathsf{n} \leq 14$. Specifically, to clearly find the relationship between the number of the tiles in one qDCMP and its cost, Table 2 lists all possible cases about them, where all qDCMPs are divided into 15 layers according to the number of their tiles.

From Table 2, one can easily get that as the number of the tiles decreases in qDCMP, the minimal cost is gradually increased, and the qDCMPs with larger m always own a smaller minimal cost in the same layer. So, for one hand T, if the corresponding minimal cost is found in the layer whose qDCMPs owning the most tiles, it will be not necessary to further evaluate the costs of other qDCMPs. Moreover, according to Table 2, one can also find that for a given qDCMP layer, although its corresponding minimal cost is not found, it will be also not necessary to further evaluate the costs of the qDCMPs with fewer tiles when the minimal cost of the next layer can be found in the current layer. Specially, we define the minimal cost of the next layer as the sub-minimal cost for the current layer. Thus, to calculate the deficiency of one hand T, we can first divide its all qDCMPs by the number of their tiles, and then gradually check them as the number of their tiles decreases. When the minimal cost is found by one qDCMP or the sub-minimal cost is found among all qDCMPs in the current layer, the deficiency of T will be directly get by the minimal cost or the sub-minimal cost. For clarity, Table 3 further lists the minimal cost and the sub-minimal cost for all different layers. Furthermore, when both the minimal cost and the sub-minimal cost are not obtained in the current layer, one can also get from Table 2 that there is only a fewer number of layers but not all the remaining layers that need to further evaluate. This is due to the fact that among the remaining layers, the minimal cost of some layers will be larger than the obtained best cost in the current layer. For example, when we get the best cost is 6 in the 7-th layer, it is just necessary to further evaluate the 8-th and 9-th layers at most. So, based on the above analyses and discussions, the proposed HBB method could effectively enhance the calculation efficiency for the deficiency of one hand. Specifically, the maximal layer needed to further evaluate for different costs and the overall framework of the HBB method is further provided and described in Table 4 and Algorithm 5, respectively.

In summary, from the above descriptions and analyses, one can see that the proposed HBB method divides all qDCMPs into different layers according to the number of their tiles and calculates the costs of qDCMPs layer by layer to evaluate the deficiency. Unlike both QSA Li and Yan (2019) and BDA Yan et al. (2021), where the deficiency of one hand is just got when its all qDCMPs have been wholly evaluated, the HBB method only needs to evaluate a few numbers of layers when the deficiency is obtained. Thus, the proposed method could effectively enhance the calculation efficiency for the deficiency of one hand.

Algorithm 5 The framework of the HBB method

Input: A Majhong hand T, all its qDCMPs.
Initialize the deficiency of T $dfncy = 100$;
Divide all qDCMPs of T into different layers by the number of their tiles, and sort them as the number of tiles decreases;
Set the index of the current layer $layer_{index} = 1$, the maximal number of the layers needed to further evaluate $num_{max} = 15$, and the estimate deficiency of T $dfncy_T = 100$;
while $layer_{index} \leq num_{max}$ **do**
    Divide all qDCMPs in the i-th layer based on the value of m into two distinct groups, labeled as Group 1 and Group 2 respectively, as shown in Table 2;
    Set the current obtained best cost $cost_{best} = 100$;
    for each qDCMP π_i in Group 1 **do**
        Calculate the cost $cost_i$ of π_i by **Algorithm 4**;
        if $cost_i$=the minimal cost in the current layer **then**
            Let $dfncy = cost_i$;
            Break;
        end if
        if $cost_i < cost_{best}$ **then**
            Let $cost_{best} = cost_i$;
        end if
    end for
    if $cost_{best}$=the sub-minimal cost in the current layer **then**
        Let $dfncy = cost_{best}$;
        Break;
    end if
    for each qDCMP π_i in Group 2 **do**
        Calculate the cost $cost_i$ of π_i by **Algorithm 4**;
        if $cost_i < cost_{best}$ **then**
            Let $cost_{best} = cost_i$;
        end if
    end for
    if $cost_{best}$=the sub-minimal cost in the current layer **then**
        Let $dfncy = cost_{best}$;
        Break;
    else
        if $cost_{best} < dfncy_T$ **then**
            Let $dfncy_T = cost_{best}$;
            Reset the value of num_{max} by $cost_{best}$ according to Table 4;
        end if
    end if
    Let $layer_{index} = layer_{index} + 1$;
end while
Let $dfncy = dfncy_T$;
Output: $dfncy$.

Table 2 The relationship between the number of the tiles in qDCMP and its possible costs

Layer	The number of tiles in qDCMP	Possible costs			
		Group 1	*cost*	Group 2	*cost*
1	14	$m = 4, n = 1$	0	–	–
2	13	$m = 3, n = 2$	1	–	–
3	12	$m = 4, n = 0$	1,2,100	$m = 2, n = 3$	2
4	11	$m = 3, n = 1$	2,3,100	$m = 1, n = 4$	3
5	10	$m = 2, n = 2$	3,4,100	$m = 0, n = 5$	4
6	9	$m = 3, n = 0$	3,4,5,100	$m = 1, n = 3$	4,5,100
7	8	$m = 2, n = 1$	4,5,6,100	$m = 0, n = 4$	5,6,100
8	7	$m = 1, n = 2$	5,6,7,100	–	–
9	6	$m = 2, n = 0$	5,6,7,8,100	$m = 0, n = 3$	6,7,8,100
10	5	$m = 1, n = 1$	6,7,8,9,100	–	–
11	4	$m = 0, n = 2$	7,8,9,10,100	–	–
12	3	$m = 1, n = 0$	7,8,9,10,11,100	–	–
13	2	$m = 0, n = 1$	8,9,10,11,12,100	–	–
14	-	-	-	–	–
15	0	$m = 0, n = 0$	9,10,11,12,13,14,100	–	–

Table 3 The minimal and sub-minimal cost in different layers

Layer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
The minimal cost	0	1	1	2	3	3	4	5	5	6	7	7	8	-	9
The sub-minimal cost	0	1	2	3	3	4	5	5	6	7	7	8	9	-	9

Table 4 The maximal layer needed to further evaluate for different costs

The obtained cost	0	1	2	3	4	5	6	7	8	9	≥ 10
The maximal layer needed to further evaluate	1	2	3	4	6	7	9	10	12	14	15

3.4 The framework of the HBBA

By designing an updating knowledge-based octree search (UKOS) method, an updating knowledge-based evaluation (UKE) method and a hierarchical branch and bound (HBB) method above, the overall detailed procedure of the proposed algorithm shall be described here. Moreover, by considering that the BD technique (Yan et al. 2021) can effectively enhance the search efficiency of finding all possible qDCMPs for one hand, it is further introduced in the proposed algorithm by combining with the UKOS method. Specifically, an important definition is first provided below, and then the enhanced UKOS method will be described.

Definition 6 (knowledge-aware block) Given a Mahjong hand T and its knowledge base KB, a subsequence B of T is a KB-block if

- $B \neq \varnothing$ and all tiles in B have the same color;
- If t is a tile in B, then any tile in T that is KB-connected to t is also in B,

where two tiles $t = (c, n)$ and $t' = (c', n')$ are KB-connected if (i) $t = t'$, or (ii) either T or KB has a tile t'' s.t. $(tt't'')$ is a chow.

In detail, the concrete operation of the enhanced UKOS method can be described as follows. For one hand T, the first operation in the enhanced UKOS method is to divide the tiles in T into different blocks according to the above definition. Then the UKOS method will be used for each block to search all its possible local qDCMPs. Finally, for different blocks, their local qDCMPs will be suitably integrated to create all possible qDCMPs for T. Clearly, the most important part is the final step in the enhanced UKOS method. So, its specific rules are only provided here. (1) For the generation of qDCMPs for T, it can be easily obtained by just combining all possible cases from each block's qDCMPs except for the qDCMPs from different blocks which have more than one $F(e) = 1$. (2) For the generation of the feature F_π of each qDCMP for T, the value of F_π can be just obtained by respectively adding the components of the corresponding local features. (3) For the generation of the knowledge base

KB_π of each qDCMP for T, the value of KB_π can be easily got by setting its component to the minimal component among the corresponding knowledge bases, respectively. For clarity, the following example is further provided to illustrate the above rules.

Example 2 Suppose π_{B_1} and π_{B_2} are one of qDCMPs of the block B_1 and B_2 respectively, where $\pi_{B_1} = (B1B2B3)(B5B5)$, $F_{\pi_{B_1}} = (1, 1, 1, 1)$,

$$KB_{\pi_{B_1}} = (\underline{1, 2, 0, 2, 0, 1, 2, 1, 1}, \underline{1, 2, 1, 1, 3, 4, 1, 2, 0}, \underline{2, 2, 1, 3, 0, 1, 2, 3, 4})$$

and $\pi_{B_2} = (C3C4)$, $F_{\pi_{B_2}} = (0, 1, 0, 0)$,

$$KB_{\pi_{B_2}} = (\underline{1, 2, 0, 2, 0, 1, 2, 1, 1}, \underline{1, 1, 1, 1, 3, 4, 1, 2, 0}, \underline{2, 2, 1, 3, 0, 1, 2, 3, 4}).$$

Then the resulted qDCMP based on them is $\pi^{'} = (B1B2B3)(B5B5)(C3C4)$, the resulted feature based on them is $F^{'} = F_{\pi_{B_1}} + F_{\pi_{B_2}} = (1, 2, 1, 1)$, and the resulted knowledge base based on them is $KB^{'}(t) = min(KB_{\pi_{B_1}}, KB_{\pi_{B_2}}) = (\underline{1, 2, 0, 2, 0, 1, 2, 1, 1}, \underline{1, 1, 1, 1, 3, 4, 1, 2, 0}, \underline{2, 2, 1, 3, 0, 1, 2, 3, 4})$.

In summary, according to the descriptions of the enhanced UKOS method, UKE method and HBB method, the framework and the idea diagram of HBBA are presented in Algorithm 6 and Fig. 3 respectively, where the main works have been marked by underline.

Algorithm 6 The framework of the HBBA

1: **Input:** A Mahjong hand T and its knowledge base KB.
2: Divide the hand T into different blocks based on the definition of the knowledge-aware blocks, and store them in the set S_{Block};
3: **for** each block B_i in S_{Block} **do**
4: Create all its possible qDCMPs and their corresponding features $F_\pi^{B_i}$
5: and $KB_\pi^{B_i}$ under KB by **Algorithm 3**;
6: **end for**
7: Generate all possible qDCMPs and their corresponding features F_π and KB_π for T based on the updating rules in subsection 3.4.
8: Calculate the deficiency of T by **Algorithm 5**;
9: **Output:** The deficiency of T.

From Algorithm 6, one can find that for one hand T, all its possible qDCMPs and their corresponding features and updated knowledge bases are first created by the enhanced UKOS method, and then the UKE method and HBB method are employed to calculate its deficiency. In particular, for the enhanced UKOS method, an octree structure is constructed to search all possible quasi-decompositions and the knowledge base is dynamically updated during the search process, such that the rationality and effectiveness of the search results can be guaranteed. For the UKE method, all qDCMPs are suitably classified into four cases, and different equations are designed to calculate their costs based on the updating knowledge bases and their characteristics, respectively. So, the computational resource could be effectively saved, and the cost of qDCMP could be got correctly. For the HBB method, the idea of the branch and bound method is introduced and integrated to find the deficiency, where all possible qDCMPs are first divided into different layers based on the number of tiles in them, and the costs of the qDCMPs are computed layer by layer until the minimal or sub-minimal cost is found in the current layer or the specific number of layers have been searched. Then this HBB method could effectively save a larger number of evaluations on qDCMPs during the calculation process of deficiency. Thus, the proposed algorithm could effectively and efficiently calculate the deficiency of one hand.

Moreover, from Fig. 3, one can easily see that differing from QSA in Li and Yan (2019), HBBA further incorporates the BD technique, employs eight actions to search all possible qDCMPs for one hand, dynamically updates the knowledge base of each qDCMP, and introduces the HBB method to calculate the deficiency. Meanwhile, unlike BDA in Li and Yan (2019), HBBA takes fully into consideration the knowledge base during the whole calculation process of deficiency, and logical reasoning is also added to save the computational budget. Therefore, HBBA has a more promising performance to evaluate the quality of one hand.

3.5 Complexity analysis

Furthermore, the complexity of the proposed algorithm is also discussed here. In particular, according to the descriptions of HBBA above, its main operations are the enhanced UKOS method and HBB method. So, their complexities are just discussed below.

First, for the enhanced UKOS method, one can see that its main steps are to divide the hand T into different blocks, search each block to find all its possible qDCMPs, and combine the local qDCMPs of each block to obtain the all qDCMPs of T. In fact, the dividing process of T only requires sorting the tiles in T and visiting each tile one time, so its complexity is $O(N \cdot \log_2 N + N)$, where N denotes the number of tiles in T. Meanwhile, all qDCMPs of each block are generated by constructing an octree tree and visiting each node, so its complexity is $O(8^{N-1})$. Moreover, all qDCMPs of T can be got by finding all possible combinations of the local qDCMPs in each block, so its complexity is $O(8^{N\cdot(N-1)})$. Thus, the complexity of the enhanced UKOS method is $O(8^{N\cdot(N-1)} + 8^{N-1} + N \cdot \log_2 N + N) = O(8^{N\cdot(N-1)})$.

On the other hand, for the HBB method, one can get that its main steps are to classify all qDCMPs of T into different layers, evaluate the qDCMPs in each layer and compare the

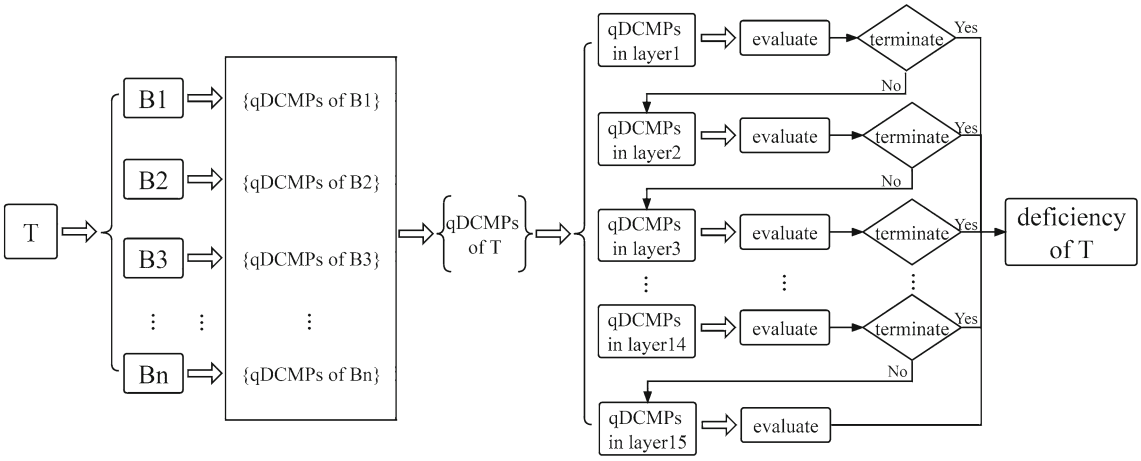

Fig. 3 The idea diagram of HBBA

cost of each qDCMP to obtain the deficiency of T. Actually, the classification of qDCMPs can be achieved just by counting the number of the tiles in each qDCMP, so its complexity is $O(N \cdot 8^{N-1})$. Meanwhile, the cost of each qDCMP can be obtained by recording its corresponding feature and calculating the parameters ue, um or u based on its remainder tiles and knowledge base, so the complexity of the evaluation for each layer is $O(max\{K_1, K_2\} \cdot 8^{N-1})$, where K_1 and K_2 denote the number of tiles in the remainder tile and the knowledge base, respectively. Moreover, the deficiency of T can be easily obtained by comparing the cost of each qDCMP, so its complexity is $O(8^{N-1})$. Thus, the complexity of the HBB method is $O(N \cdot 8^{N-1} + max\{K_1, K_2\} \cdot 8^{N-1} + 8^{N-1}) = O((max\{K_1, K_2\} + N + 1) \cdot 8^{N-1})$. Overall, the complexity of HBBA is $O(8^{N\cdot(N-1)} + (max\{K_1, K_2\} + N + 1) \cdot 8^{N-1}) = O(8^{N\cdot(N-1)})$.

It should be pointed out that the number of the blocks (N_b) and that of the local qDCMPs (N_d) in each block are usually much smaller than the theoretical ones, *i.e.*, $N_b \ll N$ and $N_d \ll 8^{N-1}$, and a few numbers of qDCMPs of T would be evaluated in fact during the calculation of the deficiency. Therefore, the proposed algorithm would not cause severe computational burdens.

4 Experiments

In this section, the performance of the HBBA will be evaluated by experimental tests on three different types of hands, including the hands with one color, two colors and three colors. Specifically, to ensure the rationality of the following evaluations, 1000 randomly generated hands are employed for each type of hands, 50, 100 and 100 knowledge bases are randomly generated based on a normal distribution for the above hands with one color, two colors and three colors respectively, and the true deficiencies of the tested hands are obtained in advance. Besides, it should be noted that for each tested hand T, its knowledge base KB is always legal, meaning that the total number of any tile in T and KB does not exceed four. Then the performance of the algorithm can be effectively tested and evaluated. In detail, the specific information of the tested hands and their knowledge bases can be found in Table 5.

Moreover, in these experiments, the effectiveness of the proposed methods is also analyzed, and three typical deficiency algorithms are compared. To appropriately evaluate the performance of the algorithm, the accuracy rate and running time are used to measure their performance. All experiments are implemented in Python 3.8 and executed on a PC (Intel i7-6700 CPU and 16 GB RAM).

4.1 The effectiveness of the proposed methods

In this subsection, we shall illustrate the effectiveness of the proposed methods in HBBA, including the UKOS method, the UKE method, the HBB method and the BD method.

4.1.1 The effectiveness of the UKOS method

To show the effectiveness of the UKOS method, we develop two HBBA variants, HBBA_{1-1} and HBBA_{1-2}, and compare them with HBBA on the above three types of hands. The variants are HBBA with the quadtree tree search method in Li and Yan (2019) and the 6-ary tree search method in Yan et al. (2021) to find all possible qDCMPs for each block, respectively. Obviously, HBBA_{1-1} and HBBA_{1-2} can demonstrate the benefits of the eight actions and the updated knowledge base during the process of finding qDCMPs for one hand. Thus, these two variants can effectively show the advantage of the UKOS method.

In this experiment, the other operations in the two variants are consistent with that in HBBA. Table 6 reports their

Table 5 The detailed information of the used hands and their knowledge bases

	1-color hands	2-color hands	3-color hands
The number of used tiles	36	72	108
The maximal number of tiles in KB	22	58	94
The number of tested hands	1000	1000	1000
The number of test knowledge bases for each hand	50	100	100

Table 6 Experimental results of HBBA and $HBBA_{1-1}$ and $HBBA_{1-2}$ on different types of hands

Types	Number of tiles in KB	Number of hands	$HBBA_{1-1}$ Accuracy Rate	$HBBA_{1-2}$ Accuracy Rate	HBBA Accuracy Rate
1-color hands	[0, 5)	7978	70.46%	70.56%	100%
	[5, 10)	14997	98.94%	98.95%	100%
	[10, 15)	15811	99.87%	99.87%	100%
	[15, 22]	11214	99.98%	99.98%	100%
	[0,22]	50000	94.92%	94.94%	100%
2-color hands	[0, 10)	9917	60.65%	60.60%	100%
	[10, 20)	18779	93.62%	94.47%	100%
	[20, 30)	25117	98.57%	99.09%	100%
	[30, 40)	23690	99.39%	99.74%	100%
	[40, 50)	15849	99.83%	99.96%	100%
	[50, 58]	6648	100%	100%	100%
	[0,58]	100000	94.37%	94.76%	100%
3-color hands	[0, 10)	4972	69.05%	68.04%	100%
	[10, 20)	8091	79.37%	79.58%	100%
	[20, 30)	11519	94.70%	95.03%	100%
	[30, 40)	14446	97.89%	98.15%	100%
	[40, 50)	15859	99.01%	99.18%	100%
	[50, 60)	15150	99.50%	99.60%	100%
	[60, 70)	12757	99.78%	99.82%	100%
	[70, 80)	9388	99.90%	99.88%	100%
	[80, 94]	7818	99.93%	99.93%	100%
	[0,94]	100000	93.91%	94.00%	100%

experimental results on each type of hand, where for each type of hand, the tested hands are further classified into various groups according to the number of tiles in their KBs, and the last row of each type provides the overall results on these groups(the same below).

From Table 6, we see that the UKOS method can effectively ensure the correctness of deficiency for a hand. Specifically, for the hands with one color, $HBBA_{1-1}$ and $HBBA_{1-2}$ always do not get all right deficiency in all groups, and HBBA, $HBBA_{1-1}$ and $HBBA_{1-2}$ obtain 100%, 94.92% and 94.94% in term of the overall accuracy rate, respectively. Meanwhile, for the hands with two colors, $HBBA_{1-1}$ and $HBBA_{1-2}$ do not get all right deficiency in the first five groups, and have all right deficiency in the last group. HBBA, $HBBA_{1-1}$ and $HBBA_{1-2}$ obtain 100%, 94.37% and 94.76% in terms of the overall accuracy rate in this case, respectively. Moreover, for the hands with three colors, $HBBA_{1-1}$ and $HBBA_{1-2}$ still do not get all right deficiency in all groups, and HBBA, $HBBA_{1-1}$ and $HBBA_{1-2}$ obtain 100%, 93.91% and 94.00% in term of the overall accuracy rate, respectively. This could be attributed to the fact that the UKOS method takes into account all possible cases during the generation of qDCMPs by employing eight actions to construct the tree and fully considering the knowledge base. Thus, the UKOS method could be more effective to improve the correctness of the algorithm.

4.1.2 The effectiveness of the UKE method

Moreover, to show the effectiveness of the UKE method, two HBBA variants, $HBBA_{2-1}$ and $HBBA_{2-2}$ are designed and compared with HBBA on the above three types of hands. The

Table 7 Experimental results of HBBA and $HBBA_{2-1}$ and $HBBA_{2-2}$ on different types of hands

Types	Number of tiles in KB	Number of hands	$HBBA_{2-1}$ Accuracy Rate	$HBBA_{2-2}$ Accuracy Rate	HBBA Accuracy Rate
1-color hands	[0, 5)	7978	64.78%	96.45%	100%
	[5, 10)	14997	84.06%	99.99%	100%
	[10, 15)	15811	81.46%	100%	100%
	[15, 22]	11214	80.23%	100%	100%
	[0,22]	50000	79.30%	99.43%	100%
2-color hands	[0, 10)	9917	60.73%	77.85%	100%
	[10, 20)	18779	93.68%	99.35%	100%
	[20, 30)	25117	98.57%	100%	100%
	[30, 40)	23690	99.43%	100%	100%
	[40, 50)	15849	99.85%	100%	100%
	[50, 58]	6648	100%	100%	100%
	[0,58]	100000	94.40%	97.68%	100%
3-color hands	[0, 10)	4972	69.05%	62.03%	100%
	[10, 20)	8091	79.46%	86.21%	100%
	[20, 30)	11519	94.71%	99.67%	100%
	[30, 40)	14446	97.95%	100%	100%
	[40, 50)	15859	99.02%	100%	100%
	[50, 60)	15150	99.52%	100%	100%
	[60, 70)	12757	99.77%	100%	100%
	[70, 80)	9388	99.84%	100%	100%
	[80, 94]	7818	99.93%	100%	100%
	[0,94]	100000	93.92%	95.26%	100%

variants are HBBA with the evaluation method of qDCMP in Li and Yan (2019) and the evaluation method of type in Yan et al. (2021), respectively. Obviously, $HBBA_{2-1}$ and $HBBA_{2-2}$ can effectively illustrate the benefit of making full use of the knowledge base. Thus, these two variants can effectively show the performance of the UKOS method. Specifically, the other operations in the two variants are consistent with that in HBBA, and Table 7 reports their experimental results on each type of hand.

From Table 7, we see that the UKE method can rightly calculate the cost of each qDCMP. Specifically, for the hands with one color, HBBA can correctly get the true deficiency for all hands, while $HBBA_{2-1}$ always does not get the true deficiency in all groups and $HBBA_{2-2}$ just can obtain all right deficiency in the last two groups. Moreover, HBBA, $HBBA_{2-1}$ and $HBBA_{2-2}$ obtain 100%, 79.30% and 99.43% in terms of the overall accuracy rate for this case, respectively. For the hands with two colors, HBBA and $HBBA_{2-2}$ always get all right deficiency in all cases except for $HBBA_{2-2}$ on the first two groups, and $HBBA_{2-1}$ obtains all right deficiency on the last group. Moreover, HBBA, $HBBA_{2-1}$ and $HBBA_{2-2}$ obtain 100%, 94.40% and 97.68% in terms of the overall accuracy rate for this kind of hands, respectively. For the hands with three colors, HBBA and $HBBA_{2-2}$ always get all right deficiency in all cases except for $HBBA_{2-2}$ on the first three groups, and $HBBA_{2-1}$ does not obtain all right deficiency in all groups. Moreover, HBBA, $HBBA_{2-1}$ and $HBBA_{2-2}$ obtain 100%, 93.92% and 95.26% in terms of the overall accuracy rate for this kind of hands, respectively. The reason for this might be that the UKE method fully considers the information of both the hand and its corresponding knowledge base to evaluate each qDCMP, which could precisely characterize the state of one hand. Thus, the UKE method could effectively enhance the correctness of the algorithm.

4.1.3 The effectiveness of the HBB and BD method

Furthermore, to show the effectiveness of HBB and BD methods, we develop three HBBA variants, $HBBA_{3-1}$, $HBBA_{3-2}$ and $HBBA_{3-3}$, and compare them with HBBA on the above three types of hands. The variants are HBBA without the BD method, HBB method and both of them, respectively. Clearly, $HBBA_{3-1}$ and $HBBA_{3-2}$ can effectively show the effects of the BD method and HBB method on the performance of HBBA respectively, and $HBBA_{3-3}$ can illustrate the merits of HBB method and BD method. Thus, this experiment can effectively show the advantage of both the HBB method and the BD method. In particular, the other opera-

Table 8 Experimental results of HBBA and $HBBA_{3-1}$, $HBBA_{3-2}$ and $HBBA_{3-3}$ on different types of hands

Types	Number of tiles in KB	Number of hands	$HBBA_{3-1}$ Running time(s)	$HBBA_{3-2}$ Running time(s)	$HBBA_{3-3}$ Running time(s)	HBBA Running time(s)
1-color hands	[0, 5)	7978	2.20×10^{-2}	4.17×10^{-2}	4.23×10^{-2}	2.11×10^{-2}
	[5, 10)	14997	1.00×10^{-1}	2.22×10^{-1}	2.25×10^{-1}	9.62×10^{-2}
	[10, 15)	15811	2.34×10^{-1}	3.52×10^{-1}	3.56×10^{-1}	2.24×10^{-1}
	[15, 22]	11214	3.81×10^{-1}	4.93×10^{-1}	4.99×10^{-1}	3.66×10^{-1}
	[0,22]	50000	1.93×10^{-1}	2.95×10^{-1}	2.99×10^{-1}	9.26×10^{-2}
2-color hands	[0, 10)	9917	6.72×10^{-3}	1.22×10^{-2}	1.52×10^{-2}	3.43×10^{-3}
	[10, 20)	18779	2.36×10^{-2}	3.61×10^{-2}	4.88×10^{-2}	1.01×10^{-2}
	[20, 30)	25117	5.25×10^{-2}	6.13×10^{-2}	8.83×10^{-2}	2.35×10^{-2}
	[30, 40)	23690	8.15×10^{-2}	9.18×10^{-2}	1.33×10^{-1}	3.71×10^{-2}
	[40, 50)	15849	1.07×10^{-1}	1.19×10^{-1}	1.72×10^{-1}	4.94×10^{-2}
	[50, 58]	6648	1.20×10^{-1}	1.33×10^{-1}	1.92×10^{-1}	5.54×10^{-2}
	[0,58]	100000	6.26×10^{-2}	7.28×10^{-2}	1.04×10^{-1}	2.84×10^{-2}
3-color hands	[0, 10)	4972	2.46×10^{-3}	2.52×10^{-3}	3.31×10^{-3}	1.61×10^{-3}
	[10, 20)	8091	4.81×10^{-3}	1.13×10^{-2}	1.38×10^{-2}	2.14×10^{-3}
	[20, 30)	11519	8.37×10^{-3}	1.22×10^{-2}	1.70×10^{-2}	3.30×10^{-3}
	[30, 40)	14446	1.36×10^{-2}	1.79×10^{-2}	2.54×10^{-2}	5.55×10^{-3}
	[40, 50)	15859	1.84×10^{-2}	2.44×10^{-2}	3.45×10^{-2}	7.75×10^{-3}
	[50, 60)	15150	2.34×10^{-2}	3.10×10^{-2}	4.35×10^{-2}	1.00×10^{-2}
	[60, 70)	12757	2.62×10^{-2}	3.53×10^{-2}	4.93×10^{-2}	1.14×10^{-2}
	[70, 80)	9388	2.95×10^{-2}	4.01×10^{-2}	5.56×10^{-2}	1.30×10^{-2}
	[80, 94]	7818	3.09×10^{-2}	4.27×10^{-2}	5.90×10^{-2}	1.36×10^{-2}
	[0,94]	100000	1.84×10^{-2}	2.52×10^{-2}	3.51×10^{-2}	7.92×10^{-3}

tions in the three variants are consistent with that in HBBA, and Table 8 reports their average running times on each type of hand.

From Table 8, we see that both the HBB method and BD method can effectively accelerate the search efficiency of HBBA. Specially, for the hands with one color, HBBA and $HBBA_{3-3}$ always have the smallest and largest running time in all groups respectively, and HBBA, $HBBA_{3-1}$, $HBBA_{3-2}$ and $HBBA_{3-3}$ obtain $9.26 \times 10^{-2}s$, $1.93 \times 10^{-1}s$, $2.95 \times 10^{-1}s$ and $2.99 \times 10^{-1}s$ in term of the average running time for this kind of hands, respectively. Meanwhile, for the hands with two colors, HBBA and $HBBA_{3-3}$ also have the smallest and largest running time in all groups respectively, and HBBA has a better overall performance than its three variants on all groups, especially for the last two cases. Moreover, for the hands with three colors, HBBA still owns the best computation efficiency in each group, and HBBA, $HBBA_{3-1}$, $HBBA_{3-2}$ and $HBBA_{3-3}$ obtain $7.92 \times 10^{-3}s$, $1.84 \times 10^{-2}s$, $2.52 \times 10^{-2}s$ and $3.51 \times 10^{-2}s$ in term of the average running time for this kind of hands, respectively. This might be attributed to the fact that the BD method can effectively reduce the search scale during the construction of the tree, and the HBB method can significantly degrade the number of calculating the cost of qDCMPs during the process of getting the deficiency for one hand. Thus, the HBB method and BD method could be more effective to improve the search efficiency of HBBA.

4.2 Comparisons and discussions

To illustrate the performance of HBBA, a comparison of HBBA with three typical deficiency algorithms, including QSA (Li and Yan 2019), WRA (Wang et al. 2020) and BDA (Yan et al. 2021), is further conducted on the above three types of hands. In particular, QSA (Li and Yan 2019) is an original algorithm based on the quadtree tree search to calculate the deficiency of one hand, where the knowledge base is not considered to evaluate the qDCMPs. Meanwhile, WRA (Wang et al. 2020) is a new approach to evaluate the quality of one hand by constructing a weighted restarting automaton over the tropical semiring. Moreover, BDA (Yan et al. 2021) is an enhanced version of QSA, where a block decomposition technique is developed to reduce its search scale, and the knowledge base is employed to calculate the cost of each qDCMP's type. Obviously, these three algorithms above are the special methods to calculate the deficiency for a hand,

Table 9 Experimental results of HBBA and three typical deficiency algorithms on different types of hands

Types	Number of tiles in KB	Number of hands	QSA Accuracy Rate	BDA Accuracy Rate	WRA Accuracy Rate	HBBA Accuracy Rate
1-color hands	[0, 5)	7978	70.58%	96.90%	43.07%	100%
	[5, 10)	14997	99.50%	99.97%	49.67%	100%
	[10, 15)	15811	99.97%	100%	45.41%	100%
	[15, 22]	11214	99.98%	100%	43.45%	100%
	[0,22]	50000	99.98%	99.50%	45.87%	100%
2-color hands	[0, 10)	9917	32.80%	77.73%	24.47%	100%
	[10, 20)	18779	80.22%	99.36%	43.14%	100%
	[20, 30)	25117	92.85%	100%	34.20%	100%
	[30, 40)	23690	97.34%	100%	29.47%	100%
	[40, 50)	15849	99.24%	100%	27.04%	100%
	[50, 58]	6648	99.89%	100%	25.77%	100%
	[0,58]	100000	87.07%	97.67%	32.10%	100%
3-color hands	[0, 10)	4972	7.12%	62.19%	6.48%	100%
	[10, 20)	8091	41.23%	86.52%	39.76%	100%
	[20, 30)	11519	62.74%	99.68%	48.39%	100%
	[30, 40)	14446	75.85%	100%	46.71%	100%
	[40, 50)	15859	83.20%	100%	43.99%	100%
	[50, 60)	15150	87.88%	100%	41.70%	100%
	[60, 70)	12757	89.64%	100%	40.29%	100%
	[70, 80)	9388	91.28%	100%	38.51%	100%
	[80, 94]	7818	92.32%	100%	38.84%	100%
	[0,94]	100000	74.04%	96.99%	40.29%	100%

and they have the typical roles in the development of deficiency calculation. So, they are all chosen as the compared ones here. In detail, their experimental results on each type of hand are listed in Table 9.

From Table 9, we can see that HBBA has a more promising performance than the compared ones. In particular, for the hands with one color, HBBA always gets the true deficiency in all groups, QSA and WRA do not get the true deficiency in all cases, and BDA just obtains the true deficiency in the last two groups. Meanwhile, for the hands with two colors, HBBA and BDA always get the true deficiency in all groups except for BDA in the first two groups, and QSA and WRA do not get the true deficiency in all cases. Moreover, for the hands with three colors, HBBA and BDA get the true deficiency in all groups except for BDA in the first three groups, and QSA and WRA still do not get the true deficiency in all cases. Furthermore, one can also find that HBBA, QSA, BDA and WRA obtain 100%, 99.98%, 99.50% and 45.87% for the one-color hands, 100%, 87.07%, 97.67% and 32.10% for the two-color hands, and 100%, 74.04%, 96.99% and 40.29% for the three-color hands in term of the overall accuracy rate, respectively. The reasons for these might be that HBBA makes full use of and dynamically updates the knowledge base for each qDCMP during the processes of both searching the qDCMPs and evaluating their costs, while QSA and WRA do not employ the knowledge base to evaluate the quality of each qDCMP, and the type of qDCMP is just used in BDA to calculate the deficiency of one hand, and some possible cases of generating the qDCMPs for one block can be forgotten. Thus, HBBA is a more promising algorithm for the deficiency of one hand.

5 Conclusion

Due to its nature of imperfect state information, the Mahjong game has become one of the important experimental platforms for researchers in the field of artificial intelligence (AI) to explore new techniques, and has been widely considered and studied. In the Mahjong AI system, the evaluation of the quality for one hand (i.e. deficiency) always plays an important role in the decision-making of the game player, such as discarding tiles, pong, kong and so on. To do this, this paper proposes an effective and efficient algorithm (HBBA) to calculate the deficiency of one hand. In the proposed algorithm, to ensure the correctness of the deficiency for a hand, a UKOS method is developed to explore all possible quasi-decompositions, and a UKE method is designed

to calculate the cost of each quasi-decomposition by making full use of both the tiles used in the hand and its knowledge base. Meanwhile, to accelerate the efficiency of getting the deficiency, an HBB method is presented by layering all its quasi-decompositions according to the number of their tiles. Moreover, the block decomposition is further adopted in the UKOS method to enhance its search efficiency. Compared with the existing methods, HBBA considers all possible generation cases of quasi-decompositions, dynamically updates and makes full use of the knowledge base for each quasi-decomposition to evaluate its cost, and adopts a faster method to get the deficiency for one hand. Thus, HBBA could not only obtain a correct deficiency for each hand, but also own a promising computation efficiency. Finally, the performance of HBBA is evaluated and compared with three typical methods on three different kinds of hands. The experimental results show its effectiveness and efficiency.

In our future work, we will utilize the proposed deficiency algorithm to develop a more advanced Mahjong AI. Furthermore, by combining machine learning or deep learning techniques, a more simplified and promising method will be designed for evaluating the quality of the Mahjong hand.

Acknowledgements This work is supported by the National Natural Science Foundation of China No. 11671244 and 12071271. The authors thank Professor Sanjiang Li for numerous discussions on the notions and results presented in the current paper.

Funding This article was funded by National Natural Science Foundation of China (12071271, 11671244)

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Berliner HJ (1977) Experiences in evaluation with BKG: a program that plays backgammon. Proceedings of the 5th international joint conference on Artificial intelligence - Volume 1

Schaeffer J, Burch N, Bjoernsson Y, Kishimoto A, Mueller M, Lake R, Lu P, Sutphen S (2007) Checkers is solved. Science 317(5844):1518–1522

Wiering MA (2010) Self-play and using an expert to learn to play backgammon with temporal difference learning. J Intell Learn Syst Appl 2(2):57–68

Shannon Claude E (1950) Programming a computer for playing chess. Philosophical Mag 41(314):256–275

Silver David, Schrittwieser Julian, Simonyan Karen, Antonoglou Ioannis, Huang Guez (2017) Mastering the game of go without human knowledge. Nature 550(7676), 354–359

Samuel AL (1959) Some studies in machine learning using the game of checkers. IBM J Res Dev 3(3):211–229

Bowling M, Burch N, Johanson M, Tammelin O (2015) Heads-up limit hold'em poker is solved. Science 347(6218):145–149

Schmid Martin, Lisy Viliam, Burch Neil, Bowling Michael, Moravcik Matej (2017) Deepstack: expert-level artificial intelligence in heads-up no-limit poker. Science 356(6337), 508–513

Brown N, Sandholm T (2018) Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. Science 359(6374):418–424

Mizukami N, Tsuruoka Y (2015) Building a computer Mahjong player based on monte carlo simulation and opponent models. IEEE Conference on Computational Intelligence & Games

Brown N, Sandholm T (2019) Solving imperfect-information games via discounted regret minimization. National Conference on Artificial Intelligence

Holcomb SD, Porter WK, Ault SV, Mao G, Wang J (2018) Overview on deepmind and its AlphaGo zero AI. Proceedings of the 2018 International Conference on Big Data and Education

Heinrich J, Silver D (2016) Deep reinforcement learning from self-play in imperfect-information games. Preprint at https://doi.org/10.48550/arXiv.1603.01121

Yoshimura K, Hochin T, Nomiya H (2016) Searching optimal movements in multi-player games with imperfect information. 2016 IEEE/ACIS 15th international conference on computer and information science (ICIS)

Brown N, Sandholm T (2017) Safe and nested subgame solving for imperfect-information games. Preprint at https://doi.org/10.48550/arXiv.1705.02955

Sandholm T (2018) Depth-limited solving for imperfect-information games. Science 347(6218):122–3

Brown N, Sandholm T (2019) Superhuman ai for multiplayer poker. Science 365(6456):885–890

Zhao E, Yan R, Li J, Li K, Xing J (2022) Alphaholdem: high-performance artificial intelligence for heads-up no-limit texas hold'em from end-to-end reinforcement learning. Proceedings of the AAAI Conference on Artificial Intelligence

Rong J, Qin T, An B (2019) Competitive bridge bidding with deep neural networks. Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems

Jiang Q, Li K, Du B, Chen H, Fang H (2019) Deltadou: expert-level doudizhu AI through self-play. Proceedings of the 28th International Joint Conference on Artificial Intelligence

Li J, Koyamada S, Ye Q, Liu G, Hon HW (2020) Suphx: Mastering Mahjong with deep reinforcement learning. Preprint at https://doi.org/10.48550/arXiv.2003.13590

Kurita M, Hoki K (2021) Method for constructing artificial intelligence player with abstractions to markov decision processes in multiplayer game of mahjong. IEEE Trans Games 13(1):99–110

Sato H, Shirakawa T, Hagihara A, Maeda K (2017) An analysis of play style of advanced mahjong players toward the implementation of strong ai player. Int J Parallel Emergent Distributed Syst 32(2):195–205

Li S, Yan X (2019) Let's Play Mahjong. Preprint at https://doi.org/10.48550/arXiv.1903.03294

Wang Q, Li Y, Chen X (2020) A Mahjong-strategy based on weighted restarting automata. MLMI: 2020 the 3rd international conference on machine learning and machine intelligence

Yan X, Li Y, Li S (2021) A fast algorithm for computing the deficiency number of a Mahjong hand. Preprint at https://doi.org/10.48550/arXiv.2108.06832

Atsushi W, Masaki H, Hajime M, Kanako K, Yoshiyuki K (2014) Estimating risk of discarded tiles in mahjong using svr. The Special Interest Group Technical Reports of IPSJ, 1–3

Tang Jenn (2014) Designing an anti-swindle mahjong leisure prototype system using rfid and ontology theory. J Netw Comput Appl 39:292–301

Silver D (2017) Technical perspective solving imperfect information games. Commun ACM 60(11):80

Gao S, Okuya F, Kawahara Y, Tsuruoka Y (2019) Building a computer Mahjong player via deep convolutional neural networks. Preprint at https://doi.org/10.48550/arXiv.1906.02146

Wang M, Yan T, Luo M, Huang W (2019) A novel deep residual network-based incomplete information competition strategy for four-players mahjong games. Multimedia Tools Appl 78(16):23443–23467

Ueno M, Hayakawa D, Isahara H (2019) Estimating the purpose of discard in Mahjong to support learning for beginners. Distributed Computing and Artificial Intelligence, 15th International Conference

Cheng Y, Li CK, Li SH (2020) Mathematical aspect of the combinatorial game "Mahjong". Preprint at https://doi.org/10.48550/arXiv.1707.07345

Zheng Y, Li S (2020) A review of Mahjong AI research. RICAI: 2020 2nd International Conference on Robotics, Intelligent Control and Artificial Intelligence

Wang Q, Zhou Y, Zhu D, Li Y (2022) A new approach to compute deficiency number of mahjong configurations. Entertainment Comput 43:100509

Zhang X, Liu L, Gan C, Yang X (2022) Research on Mahjong game strategy combining hand tiles optimization and situation search. 2022 34th Chinese control and decision conference (CCDC)

Gao S, Li S (2022) Bloody mahjong playing strategy based on the integration of deep learning and xgboost. CAAI Trans Intell Technol 7(1):95–106