Einführung in die Informatik

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

Sommersemester 2014

Minimax-Algorithmus

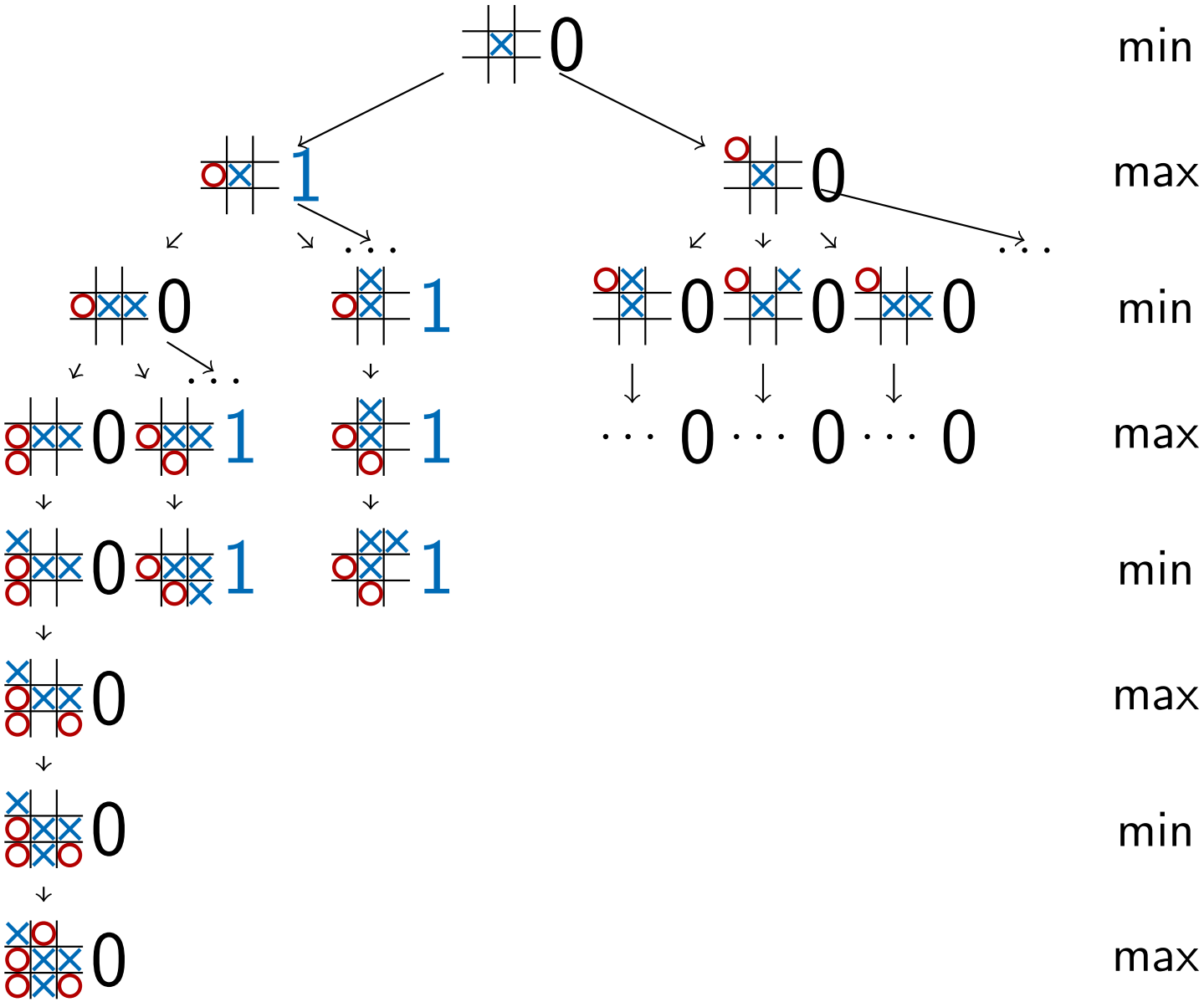
Der Minimax-Algorithmus

Der Minimax-Algorithmus eignet sich für Zwei-Personen-Nullsummenspiele mit perfekter Information (kein Zufall, keine geheimen Züge). Annahmen:

- Zwei Personen ziehen abwechselnd.
- Ein guter Zug für mich ist schlecht für den Gegner (Nullsumme)
- Der Gegner macht immer den für sich besten Zug.

Die Idee ist rekursiv die Züge zu verfolgen.

Beispiel: Tic-Tac-Toe



Pseudo-Code: Minimize (für Kreiszüge)

```
int minimize(Board board) {
  if (board.won(Player.X))
    return 1;
  if (board.isRemis())
    return 0;
  int minimum = 1;
  for (int i = 0; i < 9; i++) {
    Board newB = board.makeMove(Player.O, i));
    if (newB == null)
      continue;
    int value = maximize(newB);
    if (value < minimum)
      minimum = value;
  }
  return minimum;
}
```

Pseudo-Code: Maximize (für Kreuzzüge)

```
int maximize(Board board) {
  if (board.won(Player.O))
    return -1;
  if (board.isRemis())
    return 0;
  int maximum = -1;
  for (int i = 0; i < 9; i++) {
    Board newB = board.makeMove(Player.X, i));
    if (newB == null)
      continue;
    int value = minimize(newB)
    if (value > maximum)
      maximum = value;
  }
  return maximum;
}
```

Minimax mit Negation

Die beiden Funktionen sind sehr ähnlich.

- Negiere die Werte nach jedem Zug: Plus bedeutet aktiver Spieler steht gut, minus bedeutet Gegner steht gut.
- Dann genügt es nur noch zu maximieren.

Pseudo-Code: minimax

```
int minimax(Board board, Player me, Player him) {
  if (board.won(him))
    return -1;
  int maximum = -1;
  for (int i = 0; i < 9; i++) {
    Board newBoard = board.makeMove(me, i));
    if (newBoard == null)
      continue;
    int value = - minimax(newBoard, him, me);
    if (value > maximum)
      maximum = value;
  }
  return maximum;
}
```

Zugtiefe einschränken

- Bisher berechnen wir die Partie bis zum bitteren Ende.
- Für Alapo nicht möglich (jedenfalls nicht in 10 Minuten).
- Wir müssen die Suchtiefe beschränken.

```
int minimaxDepth(Board board, int depth,
                 Player me, Player him) {
  if (depth > MAX_DEPTH)
    return 0; // oder bessere Stellungsbewertung
  ...
  for (int i = 0; i < 9; i++) {
    ...
    int value = - minimax(..., depth + 1, ...);
    ...
```

Optimierungen

- Alpha-Beta-Pruning (siehe Wikipedia): Nicht alle Stellungen sind relevant.
- Zugvorsortierung (gute Züge zuerst probieren).
- Speichere berechnete Stellungen.
- Schrittweises Erhöhen der Zugtiefe.
- Zero-Window-Techniken
- Ruhesuche