Druga domaća zadaća

Igra Connect4

1. Implementacija

```
int minimax(Board b, int depth, bool isSerial) {    // funkcija za pronalazak najboljeg poteza
    double best_value;
    int best_move = -1;

    for (int c = 0; c < COLUMNS; c++) {
        if (b.legalMove(c)) {   // ovdje 'ne postavljamo' novo stanje igre, nego probavamo sve poteze
            b.nextMove(c, 1);
            double value = evaluate(b, 1, c, depth, !isSerial);
            b.undoMove(c);
            if (best_move == -1 || value > best_value) {
                best_move = c;
                best_value = value;
            }
        }
    }

    return best_move;
}
```

Funkcija koja pronalazi najbolji potez. Ovdje se isprobava svaki legitiman potez koji se može napraviti te se na temelju dobivene vrijednosti radi najefikasniji potez.

```
else if (isPar && depth == 0) {     // raspodijeli posao procesima
    bool valid[COLUMNS];
    for (int c = 0; c < COLUMNS; c++) {
        if (b.legalMove(c)) {
            valid[c] = true;
            num_moves++;
            b.nextMove(c, nextPlayer);
            sendHelp pom;
            pom.board = Board::copyBoard(b);
            pom.prev_move = c;
            MPI_Send(&pom, sizeof(pom), MPI_BYTE, (c % (size - 1) + 1), 0, MPI_COMM_WORLD); // koristi sve slave (bez master) i tako ciklicno
            b.undoMove(c);
        }
        else {
            valid[c] = false;
        }
    }
```

Master raspoređuje zadatke slave procesima ako je dostignuta odgovarajuća dubina i koristi se paralelizacija.

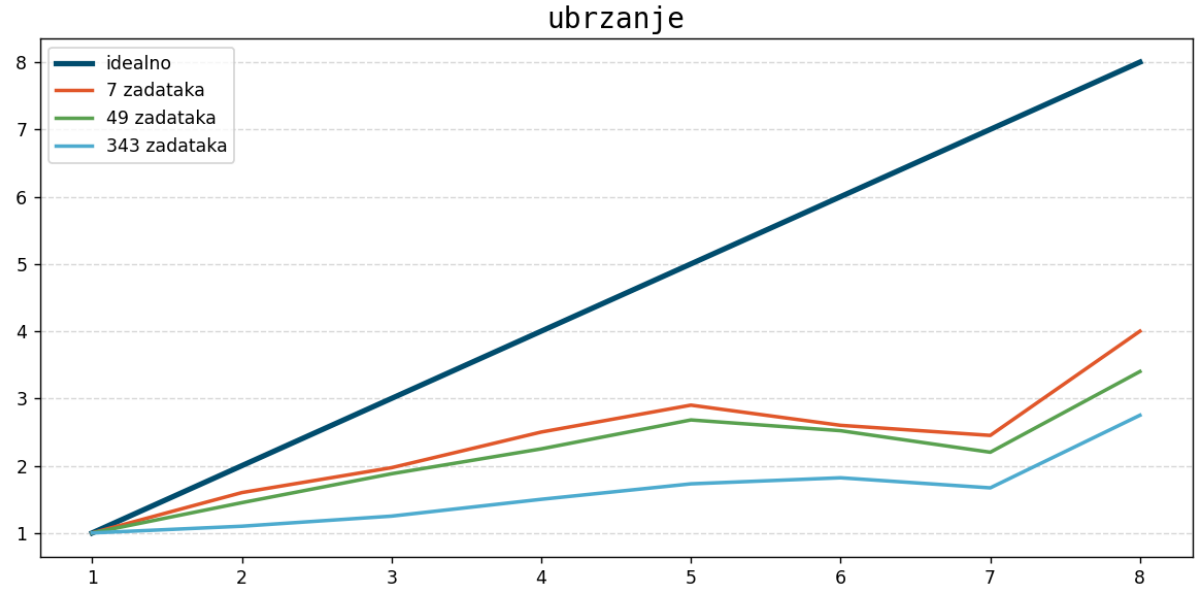
```
void slave_function(int depth) {    // slave-ovi cekaju podatke od mastera i vracaju vrijednosti
    while (true) {
        sendHelp recive;
        MPI_Recv(&recive, sizeof(sendHelp), MPI_BYTE, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

        if (recive.prev_move == -1) return;
        double val = evaluate(recive.board, recive.board.last_move, recive.prev_move, depth, false);
        MPI_Send(&val, sizeof(val), MPI_BYTE, 0, 1, MPI_COMM_WORLD);
    }
}
```

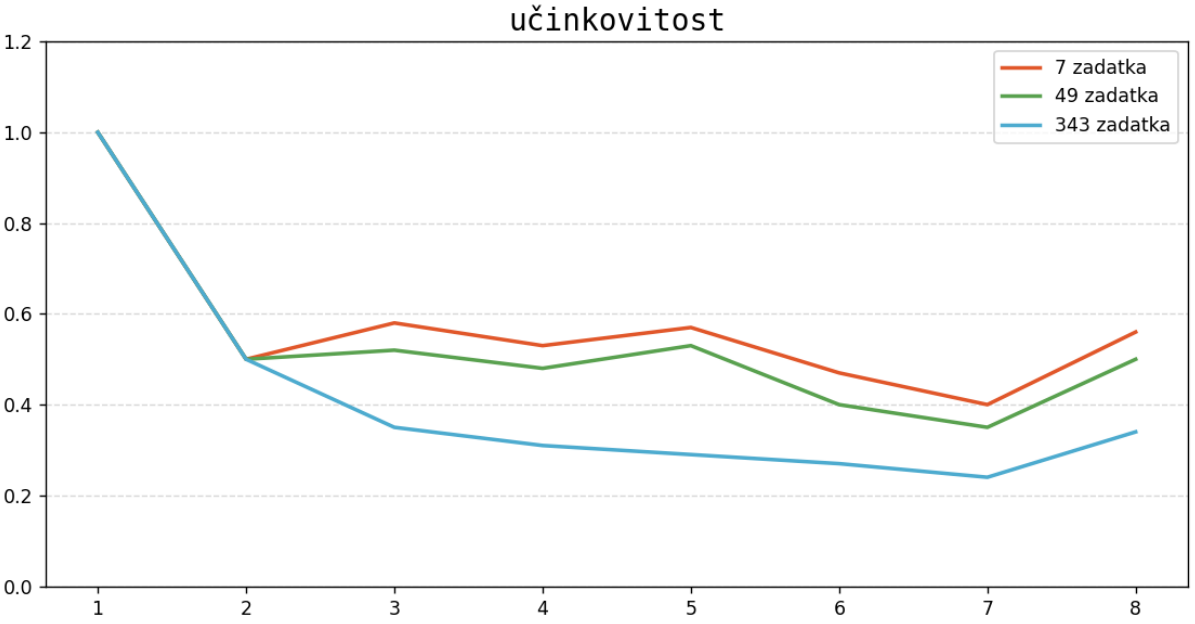
Slave prima zadatke, obrađuje i vraća rezultat.

```
Choose your move (0 - 6): 3
 -  -  -  X  -  -  -
 -  -  -  0  -  -  -
 -  -  -  0  -  -  -
 -  -  -  0  X  -  -
 -  X  X  X  0  -  0
 -  X  0  X  0  -  X
 0  1  2  3  4  5  6
**********

Time needed to calculate the best move: 1.679000 s
Computer played 5
 -  -  -  X  -  -  -
 -  -  -  0  -  -  -
 -  -  -  0  -  -  -
 -  -  -  0  X  -  -
 -  X  X  X  0  -  0
 -  X  0  X  0  0  X
 0  1  2  3  4  5  6
**********

Choose your move (0 - 6): 2
 -  -  -  X  -  -  -
 -  -  -  0  -  -  -
 -  -  -  0  -  -  -
 -  -  X  0  X  -  -
 -  X  X  X  0  -  0
 -  X  0  X  0  0  X
 0  1  2  3  4  5  6
**********

Time needed to calculate the best move: 1.754000 s
Computer played 2
 -  -  -  X  -  -  -
 -  -  -  0  -  -  -
 -  -  0  0  -  -  -
 -  -  X  0  X  -  -
 -  X  X  X  0  -  0
 -  X  0  X  0  0  X
 0  1  2  3  4  5  6
**********

Better luck next time
```

Posljednja dva koraka (računala i igrača) igre u kojoj računalo pobjeđuje.

2. Kvantitativna analiza





Očekivano paralelni program najbolje je najefikasniji za 7 zadataka. Za 49 zadataka efikasnost i ubrzanje padaju, ali je program također dovoljno efikasan. Kod 343 su oba segmenta značajno pala.