

CSIS0270 Artificial Intelligence, 2002–2003

Assignment 2

Deadline Mar 24, 2003, 5:00pm.

This assignment contains both written parts (Questions 1 and 4b) and programming parts (Question 2 and 3 and 4a). For the written parts, hand-in your answers to the assignment box (R2). For programming parts, hand-in the programs that you write via the hand-in system of the department.

1. CSPs (25%)

- a. (Textbook question 5.6): Solve the crypt-arithmetic problem $TWO+TWO=FOUR$ by hand, using backtracking, forward checking, and the MRV and least-constraining-value heuristics. You may supplement the methods with any other methods we know from the lecture (arc consistency, all-diff special checker, etc.).
- b. (Textbook question 5.11): Show that any CSP (with high degree constraints) can be transformed into a CSP containing only binary constraint:
 1. Show how a single ternary constraint such as " $A + B = C$ " can be turned into three binary constraints by using an auxiliary variable. You may assume finite domains. (*Hint*: consider a new variable that takes on values which are pairs of other values, and consider constraints such as " X is the first element of the pair Y ".)
 2. Show how constraints with more than 3 variables can be treated similarly.
 3. Show how unary constraints can be eliminated by altering the domains of variables.

2. Local Search (25%) Write a program to allow the user to input a number N , find and output a sequence with the following property using hill climbing with random restart and random tie-breaking.

- The sequence contains $2N$ numbers, where each of the integers between 1 and n appears twice in the sequence.
- The two numbers k in the sequence is separated by exactly k other numbers.

For example, for $N = 3$, your program may output the following sequence:

3 1 2 1 3 2

In your answer (either as program comment or together with other questions), explain how you formulate the question as a local search problem. In particular, explain what is a state, what is the successor function, what is the evaluation function, and what value of the evaluation function means the problem is solved.

You may assume that the problem is solvable only when N is a multiple of 4, or one less than a multiple of 4. I.e., for other N , your program can simply report that there is no solution without searching.

Note: In the first assignment of another course last year, students are asked to solve the same problem using backtracking technique. Their answers are considered exceptionally good if they can solve all problems of size smaller than 30. You should find that solving problems

of size more than 300 is quite possible with local search (by employing a few tricks to reduce time cost and dealing with plateaus).

3. **Othello (30%)** Write a program that plays Othello using Alpha-Beta. Here is a short review of the game:

- The game is played on a board of 8×8 cells, with pieces which is one side white and the other side black for putting into cells. We count rows from top to bottom, columns from left to right. Column numbers are from 1 to 8, row numbers are from A to H.
- Initially, cells D4 and E5 have a black piece, and cells D5 and E4 have a white piece.
- One side plays for white, the other plays for black. The game proceeds by each side putting a piece of his own color. The winner is the one who get more pieces of his color at the end of the game.
- When a piece of color black (resp. white) piece is put onto the board, and on the same horizontal, vertical or diagonal line there is another black (resp. white) piece that, between it and the piece put onto the board, each cell has a white (resp. black) piece, than all these white (resp. black) pieces are flipped to black (resp. white).
- A move can only be made if causes flipping of pieces. A player can pass a move if, and only if, there is no move that causes flipping. The game ends if neither side can make a move.

Focus on playing a better game rather than the interface. (The web page contains an Othello program with no AI, which may be used as a reference implementation). You should make your program reasonably responsive by limiting the number of steps searched and by using a reasonable evaluation function. A simple evaluation function can be to give a certain credit to each location of the cell. A more sophisticated game can be played by bounding the number of steps that is playable by the opponent. Document the method you used, either in your program as comments or separately with the written parts of your assignment.

4. **Probabilistic Games (20%)**

- a. Some time ago, in a TV programme there is a simple game played in the following way: there are two (teams of) players in the game. At the beginning, a number is randomly chosen between 1 to 100. This number is kept secret by the game official, so that the players know only that the number must be between 1 and 100. Alternatively, one of the player choose a number (say, 27) within the current range. If the number is the secret number, the player **loses**. Otherwise, the game official will tell whether the number is between the lower range (e.g., 1–26) or the upper range (e.g., 28–100), and the sequence repeats by the other player choosing a number.

In the middle of one dramatic game, one of the player is left with the range 25–27, and must choose a number. He wrongly choose the number 27, effectively reducing the chance of winning from $2/3$ to $1/3$. He eventually lose the game (the number turned out to be 25, while the correct play is 26). Write a program which has the input being the current range, and report the best move and the probability of winning. (Note: it is easy to write the program so that it works quickly even if the input is the whole range 1–100.)

- b. Extend the algorithm *AlphaBetaDecision* so that it allows chance nodes. The algorithm

should take two numbers *MaxUtil* and *MinUtil* as parameters, specifying the maximum and minimum utility possible respectively. Your algorithm should consider these values when performing pruning. Write all the functions that are used by the extended *AlphaBetaDecision*.