

$$\begin{array}{rcccccc}
& & 1 & & & & 1 \\
& & 5 & O & N & A & L & 5 \\
+ & G & 9 & 7 & A & L & 5 \\
\hline
& 7 & O & B & 9 & 7 & 0
\end{array}$$

Observing the third column, we know that there must be a carry in since 9 is odd. Moreover A equal either 9 or 4. Since E equal 9, A equal 4.

$$\begin{array}{rcccccc}
& & 1 & & & 1 & 1 \\
& & 5 & O & N & 4 & L & 5 \\
+ & G & 9 & 7 & 4 & L & 5 \\
\hline
& 7 & O & B & 9 & 7 & 0
\end{array}$$

L must equal 8. G must equal 1

$$\begin{array}{rcccccc}
& & 1 & & & 1 & 1 \\
& & 5 & O & N & 4 & 8 & 5 \\
+ & 1 & 9 & 7 & 4 & 8 & 5 \\
\hline
& 7 & O & B & 9 & 7 & 0
\end{array}$$

There are three digits left not assigned to a letter: 2, 3, 6. And there are three letters not assigned a value: B, N, O. Try all possible combinations of assignment to N and B. Only one works: N=6, B=3. Therefore, O = 2. We have a solution.

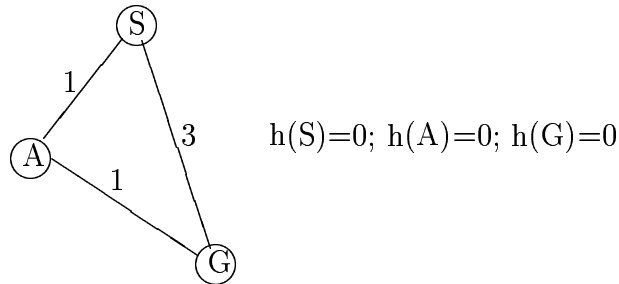
$$\begin{array}{rcccccc}
& & 5 & 2 & 6 & 4 & 8 & 5 \\
+ & 1 & 9 & 7 & 4 & 8 & 5 \\
\hline
& 7 & 2 & 3 & 9 & 7 & 0
\end{array}$$

- b) The solution is unique. All assignments are those that must be made.
- c) D=5 and there are 9 other letters. Given that each letter must stand for a different letter, there are $9! = 362880$ possible full digit assignments.
- c) By years of experience of algebra, human can easily discover sophisticated strategies that can efficiently reduce the size of search tree. It is difficult to design a computer program that can automatically obtain such high-level knowledge.

Human try to narrow down the choices by conducting “constraint propagation”, a CSP approach.

Problem 2 (10pt)

A* is designed to guarantee the optimality of the solution. If it terminates as soon as a goal node is found, the solution might be non-optimal. See the following example.



Problem 3 (25pt)

1. Breadth-First Search

From the given figure it can be deduced for the following cases

- (a) No Avoidance - S A B C D D G
- (b) No Self-Loop - S A B C D D G
- (c) No Cycles - S A B C D D G
- (d) No Repeats - S A B C D G

2. Uniform Cost Search

In this we have the following queue of the states and they have the associated costs with them

S - 0
 S A - 2
 S B - 7
 S A C - 5

S A D - 4
 S B D - 9
 S B G - 11
 S A C C - 7
 S A D A - 5
 S A D B - 6
 S A D A D - 7
 S A D B D - 8
 S A C C C - 9
 S A C C C C - 11
 S A D A D A - 8
 S A D B D B - 10

From the above maintained queue, we can easily form the list for the following cases

- (a) No Avoidance - S A D A C B B C D A D C
- (b) No Self-Loop - S A D A C B B D A D D B
- (c) No Cycles - S A D C B B D A B G
- (d) No Repeats - S A D C B G

3. Depth-First Search

In this case we get the following list traversing through the deepest nodes first

- (a) No Avoidance - S A C C C C C C C C C C
- (b) No Self-Loop - S A C D A C D A C D A C
- (c) No Cycles - S A C D B G
- (d) No Repeats - S A C D B G

4. Iterative Deepening

In this case, we traverse the states by depth-first strategy at each level of depth and construct the following list

- (a) No Avoidance - S S A B S A C D B D G
- (b) No Self-Loop - S S A B S A C D B D G
- (c) No Cycles - S S A B S A C D B D G
- (d) No Repeats - S S A B S A C D B G

Problem 4 (15pt)

For each method the list of expanded nodes was worth 3 points, and the final path to the goal node was worth 2 points. The majority of students did not write the assumptions they used in the search strategies (i.e. how ties were broken, what avoidance strategies were used) and thus had incomplete solutions. The solutions below use the avoidance strategies from problem 3 with ties broken alphabetically by the state label.

- a) Uniform-cost
 - (a) S,B,A,C,D,E,F,G,Z
 - (b) S,A,D,G,Z
- b) Greedy search
 - (a) S,C,F,E,G,Z
 - (b) S,C,F,E,G,Z
- c) A* search
 - (a) S,B,A,C,D,E,F,G,Z
 - (b) S,A,D,G,Z

Problem 5 (15pt)

Solution Using the hint from the problem formulation, let X_{ij} be a boolean variable that is *true* if we have a queen at position i, j and *false* otherwise. We have $1 \leq i \leq N$ and $1 \leq j \leq N$. A solution must have some properties. I will enumerate them and give the CNF encoding for each of them together with the exact number of clauses produced:

1. There is at most one queen on any row. This is equivalent with saying that on any two squares on the same line in at least one of them we don't have a queen. We get the clauses:

$$(\neg X_{ij} \vee \neg X_{ik}), \quad 1 \leq i \leq N, \quad 1 \leq j \leq N, \quad j < k \leq N \quad (1)$$

The number of clauses generated is $(N^3 - N^2)/2$.

2. There is at most one queen on any column. We get $(N^3 - N^2)/2$ clauses of the form:

$$(\neg X_{ij} \vee \neg X_{kj}), \quad 1 \leq i \leq N, \quad 1 \leq j \leq N, \quad i < k \leq N \quad (2)$$

3. There is at most one queen on any left-right diagonal. This is equivalent with saying that on any two squares on the same diagonal, at least one of the squares doesn't contain a queen. We get the clauses:

$$(\neg X_{ij} \vee \neg X_{i+k,j+k}), \quad 1 \leq i \leq N, \quad 1 \leq j \leq N, \quad 1 \leq k \leq \min(N - i, N - j) \quad (3)$$

To compute the number of clauses produced in this case it is helpful to think about what happens when you add 1 to $n - 1$, the current dimension of the board. We add $2n - 1$ new squares. The number of clauses is $T(n - 1)$, the number of clauses we had before, plus $n - i$ clauses for the new square at position $(1, n - i)$ or $(n - i, 1)$ (since this is the number of squares we had before on the left-right diagonal on which $(1, n - i)$ or $(n - i, 1)$ square is). We have the recursion:

$$\begin{aligned} T(n) &= T(n - 1) + (n - 1) + 2 \sum_{i=2}^n (n - i) \\ &= T(n - 1) + (n - 1) + 2 \frac{(n - 2)(n - 1)}{2} \\ &= T(n - 1) + (n - 1)^2 \end{aligned} \quad (4)$$

We have $T(1) = 0$ which means the solution of the recursion is:

$$T(N) = \sum_{i=1}^{N-1} (N - i)^2 = \sum_{n=1}^{N-1} n^2 = \frac{(N - 1)N(2N - 1)}{6} \quad (5)$$

4. There is at most one queen on any right-left diagonal. In a similar way as before we get the clauses:

$$(\neg X_{ij} \vee \neg X_{i+k,j-k}), \quad 1 \leq i \leq N, \quad 1 \leq j \leq N, \quad 1 \leq k \leq \min(N - i, j + 1) \quad (6)$$

The number of clauses generated in this manner is for similar reasons as before $\frac{(N-1)N(2N-1)}{6}$.

5. There are at least N queens on the board. Since the goal of this rule is to force the number of queens to be exactly N (we had the constraints of at most type before), we can use a stronger but easier to encode condition: at least one queen per row:

$$(X_{i1} \vee \cdots \vee X_{iN}), \quad 1 \leq i \leq N \quad (7)$$

We have N such clauses (one for each row).

The total number of clauses is:

$$\begin{aligned} T_c &= 2 \frac{(N^3 - N^2)}{2} + 2 \frac{(N-1)N(2N-1)}{6} + N \\ &= \frac{5N^3 + 4N}{3} \end{aligned} \quad (8)$$

Grading and common mistakes From the 15 points for this problem, I gave 10 points for the boolean encoding (2+2 for rows and columns, 4 for diagonals, 2 for enforcing exactly N queens on the board) and 5 points for the number of clauses (3 for the $O(N^3)$ solution, 4 for some mistakes in computing the exact number, 5 for the correct solution; if only for some cases the number of clauses was correct I subtracted points proportional with the importance of the case).

Some of you made a boolean encodings not in CNF. If you covered correctly all the cases you got 10pt for the encoding but 0pt for the *how many clauses* part since the term *clause* has a very clear meaning (disjunction in the CNF form) and even if you counted something usually it was not clauses.

The most common mistake for the non CNF encoding is the omission of the *at most N queens on the board* constraint for which I took 2 points off. An other mistake made by some of you that encoded facts like exactly one queen per row and column was that you encoded also facts like exactly one queen per diagonal, facts which are false since you have N queens and $4N - 2$ diagonals (you probably got 4 points off).

Some of you just gave an English description of how to solve the problem. Since you were asked to produce an *encoding* not how you would approach the encoding, you got somewhere between 1 and 5 points depending on how elaborate your discussion was.

For just repeating the hint or writhing nothing you got 0 points.

Problem 6 (20pt)

Solution As mentioned in the class, CSP encodings are easier to solve (you can prune big parts of the search tree) when compared with *State space – Operators* encodings, so a good CSP encoding can reduce the time required to find a solution dramatically. This problem tested your ability to write CSP encodings for problems formulated in natural language (English). It is possible to produce many different correct CSP encodings of the given problem and any of them (if they were CSP encodings) were considered correct solutions. I will insist on a particular solution but mention other choices that could have been made.

A CSP encoding consists of two parts: a set of variables with the corresponding (discrete) domains and a set of constraints on the possible values the variables can take. A solution of the problem is an assignment of values to the variables so that the constraints are satisfied. Note that the CSP encodings behave like systems of equations, they don't have dynamics (time dependency). The State Space–Operators encoding has dynamics (the notion of current state, what can you do at the next step, etc.) and actually the original problem has dynamics (so the State Space–Operators encoding is more *natural*). A solution to the problem is a path from the initial situation to the final situation, not only what is the final situation. To capture the dynamic of the problem in a CSP encoding, we need to index the variable by the time (equivalent with introducing a set of variables for each moment of time). Since the CSP encodings have finite number of variables but the time is unbounded, we have to bound the time by taking into consideration solutions of length at most, say, 20.

Producing an encoding is easier if we try to capture the state with variables and the operators in the form of constraints on variables. Note though that just mentioning the states and the operators is not a solution of the problem, the translation into variables and constraints only is a solution (CSP encoding). The CSP encoding must be self sufficient, in the sense that solving the CSP instance is based only on the encoding, not on supplementary informations from the original problem.

Let X_k ($0 \leq k \leq 20$) be a tuple that encodes the number of cannibals and missionaries on the left and right side of the river ($X_{10} = (1, 3, 1, 0)$ means that at moment 10 there are 1 cannibal and 3 missionaries on the left side of the river an 1 cannibal an 0 missionaries at the right side of the river). The domain of the variables is $0, 1, 2, 3 \times 0, 1, 2, 3 \times 0, 1, 2, 3 \times 0, 1, 2, 3$. For k odd

the boat is on the left side of the river, for k even on the right side. Capturing the succession of states gives a complete solution of the problem (we don't necessarily have to *memorize* into variables the operators at every step since we can deduce them from the succession of states). Other correct choices for the variables include (but are not limited to): keeping different variables for cannibals and missionaries, introducing a distinct variable for the boat, having variables for states and operators also, using boolean variables to encode the state, using different variables for left-right and right-left movement of the boat, etc.

The constraints on the variable can have two different forms (both acceptable): what combinations are acceptable, or what combinations are not acceptable. Note that the constraint that the number of cannibals or missionaries on the left or right side is between 0 and 3 is already captured in the domain of the variables.

The actual constraints are obtained by plugging in values for k from 0 to 20. Constraints:

- Conservation condition for the cannibals and missionaries: for every k , $X_k = (C_k^l, M_k^l, C_k^r, M_k^r)$ we have $C_k^l + C_k^r \leq 3$, $M_k^l + M_k^r \leq 3$, since some of the cannibals and/or the missionaries might be in the boat.
- At least one person is in the boat at every time or we transported everybody on the right side and we are doing nothing from now on: for every k $|C_k^l - C_k^r| + |M_k^l - M_k^r| \geq 1$ or $C_k^r = M_k^r = 3$ for all $l \geq k$
- At most two persons in the boat at any time: $|C_k^l - C_k^r| + |M_k^l - M_k^r| \leq 2$
- The number of missionaries is at least equal with the number of cannibals on both sides, at any time: $C_k^l \leq M_k^l$, $C_k^r \leq M_k^r$
- Boundary conditions: at the beginning $X_0 = (3, 3, 0, 0)$, at moment 20 $X_{20} = (0, 0, 3, 3)$

Grading and common mistakes A pure State Space – Operators solution without mentioning any variable got between 1 and 5 points. Solutions where there are variables but are not indexed by time have somewhere between 3-7 points. If you picked reasonable variable variables indexed by time but haven't formulated clear constraints (mentioning states and operators do not count for clear constraints) you got between 8-10 points. Getting

variables and a small number of constraints 10-15 points. Getting variables and constraints 16-20 points.

Grading between this limits was definitely subjective and depended mostly on how convinceable you were in the solution. All the solutions where it was very clear what the variables and constraints are without any confusion got 20 points.