**Chapter 7**

**Constraint Satisfaction Problem**

**The chapter consist of Short type Questions & Answers , Descriptive Question & Answer and MCQs & answers**

Contents

[Short type Questions & Answers 2](#_Toc14887636)

[Define constraint satisfaction problem. 2](#_Toc14887637)

[Define linear constraints. 2](#_Toc14887638)

[Define Compound Label in constraint satisfaction problems. Define Solution using the concept of compound label. 2](#_Toc14887639)

[What are the types of Constraints? 3](#_Toc14887640)

[Define Triangle Inequality. 3](#_Toc14887641)

[Descriptive Question & Answer 3](#_Toc14887642)

[What are Popular Problems of CSP 3](#_Toc14887643)

[Q Explain constraint satisfaction problem  and solve SEND + MORE = MONEY 4](#_Toc14887644)

[The MAP-COLORING problem is defined as follows: Given a map of countries, and a fixed set of colors, assign a color to each region in the map in such a way that no two adjacent regions have the same color. For example, the map of New England + New York can be colored from the set {Red, White, Blue} by the assignment: Maine is Red, New Hampshire is Blue, Vermont is Red, Massachussetts is White, Rhode Island is Blue, Connecticut is Red, New York is Blue. (The adjacency relations in this map are as shown below.) 6](#_Toc14887645)

[VT--NH---ME 6](#_Toc14887646)

[|\ | / 6](#_Toc14887647)

[| \ | / 6](#_Toc14887648)

[| \| / 6](#_Toc14887649)

[NY--MA 6](#_Toc14887650)

[\ | 6](#_Toc14887651)

[\ | \ 6](#_Toc14887652)

[\| \ 6](#_Toc14887653)

[CT---RI 6](#_Toc14887654)

[A. Write a non-deterministic algorithm to solve the MAP-COLORING problem given a map and a coloring. You can assume that the map is given in the form of a collection of statements ``Region R borders on Region Q.'' 6](#_Toc14887655)

[B. Give bounds on the branching factor, the depth of the search space, and the size of the search space as functions of the number of regions and the number of colors. 6](#_Toc14887656)

[C. What is the size of the state space? What is the branching factor? What is the diameter of the state space (i.e. the maximum distance from one state to another.) 7](#_Toc14887657)

[D. Consider a starting state S in which all countries are labelled "White"? What is the minimum cost neighbor of this state? What is the cost of these two states? 7](#_Toc14887658)

[E Find a state S that is a local minimum of the cost function but is not a solution; that is, does not have cost 0. ("Local minimum" in the non-strict sense; the cost of S is less than or equal to the cost of any of its neighbors.) 7](#_Toc14887659)

[Solve crossword puzzle problem (CSP) using backtracking 7](#_Toc14887660)

# Short type Questions & Answers

## Define constraint satisfaction problem.

Constraint Satisfaction problem is defined by a set of variables, X1,

X2,…..Xn and a set of constraints, c1,c2,…..,cm. Each variable xi has a nonempty domain

Di of possible values. Each constraints Ci involves some subset of the variables and specifies the allowable combinations of values for that subset.

## Define linear constraints.

Linear constraints are the constraints in which each variable appears only in

linear form.

## 

## • What is meant by the horizon in a planning problem?

Answer: The number of time steps for which the problem is “rolled out.” •

## What are initial state constraints in a CSP problem?

Answer: They constrain the state variables at time 0, i.e. before any action has occurred.

## • What are goal constraints?

Answer: They constrain the state variables at some time k, where k is the horizon.

## • What are precondition constraints?

Answer: They are constraints between state variables at time t and actions at time t. In other words, they specify what must hold for an action to take place.

## • What are effect constraints?

Answer: They are constraints between state variables at time t, actions at time t and state variables at time t + 1. In other words, the state variable at time t + 1 is affected by the actions at time t and its own previous value at time t.

## Define Compound Label in constraint satisfaction problems. Define Solution using the concept of compound label.

Compound label: the finite set of labels representing simultaneous assignments.

Solution: a compound label for all variables which satisfy all the constraints.

## 

## What are the types of Constraints?

Unary Constraints:

Unary constraints are one which restricts the value of a single

variable.

Binary Constraints:

Binary constraints are one with only binary constraints. It can be represented as a constraint graph.

## Define Triangle Inequality.

A heuristic h(n) is consistent if, for every node n and every successor n’ pf n generated by any action a, the estimated cost of reaching the goal form n is no greater than the step cost of getting to n’ plus the estimated cost of reaching the goal form n’:

H(n)                     <= c(n, a, n’)     +  h(n’)

This is a form of the general triangle equality.

# Descriptive Question & Answer

## What are Popular Problems of CSP

The following problems are some of the popular problems that can be solved using CSP:

1. CryptArithmetic (Coding alphabets to numbers.)
2. n-Queen (In an n-queen problem, n queens should be placed in a nXn matrix such that no queen shares the same row, column or diagonal.)
3. Map Coloring (Coloring different regions of map ensuring no adjacent regions have the same color.)
4. Crossword (Everyday puzzles appearing in newspapers.)
5. Sudoku (A number grid.)
6. Latin Square Problem

## Q Explain constraint satisfaction problem  and solve SEND + MORE = MONEY

A constraint satisfaction problem (CSP)[consists](https://www.blogger.com/null) of 

* a set of variables,
* a domain for each variable, and
* a set of constraints.

The aim is to choose a value for each variable so that the resulting possible world satisfies the constraints; we want a model of the constraints.   
A finite CSP has a finite set of variables and a finite domain for each variable. Many of the methods considered in this chapter only work for finite CSPs, although some are designed for infinite, even continuous, domains.   
The multidimensional aspect of these problems, where each variable can be seen as a separate dimension, makes them difficult to solve but also provides structure that can be exploited.   
Given a CSP, there are a number of tasks that can be performed: 

* Determine whether or not there is a model.
* Find a model.
* Find all of the models or enumerate the models.
* Count the number of models.
* Find the best model, given a measure of how good models are
* Determine whether some statement holds in all models.

This chapter mostly considers the problem of finding a model. Some of the methods can also determine if there is no solution. What may be more surprising is that some of the methods can find a model if one exists, but they cannot tell us that there is no model if none exists.   
CSPs are very common, so it is worth trying to find relatively efficient ways to solve them. Determining whether there is a model for a CSP with finite domains is NP-hard (see box) and no known algorithms exist to solve such problems that do not use exponential time in the worst case. However, just because a problem is NP-hard does not mean that all instances are difficult to solve. Many instances have structure that can be exploited.  
  
Consider the following cryptarithmetic problem as an example,

1) SEND + MORE = MONEY   
  
    5   4    3    2    1  
         S   E   N   D  
+      M  O   R   E  
        c3  c2  c1  
----------------------  
  M  O   N   E   Y

1. From Column 5, M=1, since it is only carry-over possible from sum of 2

    single digit number in column 4.

2. To produce a carry from column 4 to column 5 'S + M' is atleast 9 so

    'S=8or9' so 'S+M=9or10' & so 'O = 0 or 1'. But 'M=1', so 'O = 0'.

3. If there is carry from Column 3 to 4 then 'E=9' & so 'N=0'. But

    'O = 0' so there is no carry & 'S=9' & 'c3=0'.

4. If there is no carry from column 2 to 3 then 'E=N' which is

    impossible, therefore there is carry & 'N=E+1' & 'c2=1'.

5. If there is carry from column 1 to 2 then 'N+R=E mod 10' & 'N=E+1'

    so 'E+1+R=E mod 10', so 'R=9' but 'S=9', so there must be carry

    from column 1 to 2. Therefore 'c1=1' & 'R=8'.

6. To produce carry 'c1=1' from column 1 to 2, we must have 'D+E=10+Y'

    as Y cannot be 0/1 so D+E is atleast 12. As D is atmost 7 & E is

    atleast 5 (D cannot be 8 or 9 as it is already assigned). N is atmost 7

   & 'N=E+1' so 'E=5or6'.

7. If E were 6 & D+E atleast 12 then D would be 7, but 'N=E+1' & N would

    also be 7 which is impossible. Therefore 'E=5' & 'N=6'.

8. D+E is atleast 12 for that we get 'D=7' & 'Y=2'.

SOLUTION:  
  
     9   5   6   7  
+  1   0   8   5  
-----------------  
1  0   6   5   2  
  
VALUES:  
S=9  
E=5  
N=6  
D=7  
M=1  
O=0  
R=8  
Y=2

## The MAP-COLORING problem is defined as follows: Given a map of countries, and a fixed set of colors, assign a color to each region in the map in such a way that no two adjacent regions have the same color. For example, the map of New England + New York can be colored from the set {Red, White, Blue} by the assignment: Maine is Red, New Hampshire is Blue, Vermont is Red, Massachussetts is White, Rhode Island is Blue, Connecticut is Red, New York is Blue. (The adjacency relations in this map are as shown below.)

## VT--NH---ME

## |\ | /

## | \ | /

## | \| /

## NY--MA

## \ | \

## \ | \

## \| \

## CT---RI

## A. Write a non-deterministic algorithm to solve the MAP-COLORING problem given a map and a coloring. You can assume that the map is given in the form of a collection of statements ``Region R borders on Region Q.''

**Answer**

MAP-COLORING(in M : map; CS : set of colors; out CC : coloring)

begin CC := empty coloring;

for R in M do

choose a color C in CS such that

there is no region R1 in M such that

R1 borders R in M and R1 is assigned color C in CC;

add the assignment R -> C to CC

endfor

return CC

end.

## B. Give bounds on the branching factor, the depth of the search space, and the size of the search space as functions of the number of regions and the number of colors.

**Answer:**The branching factor is at most the number of colors, |CS|. The depth of the search space is the number of regions |M|. The search space therefore has size at most |CS| \*\* |M|.

## C. What is the size of the state space? What is the branching factor? What is the diameter of the state space (i.e. the maximum distance from one state to another.)

**Answer:**Size of the state space: 37 = 2187. Branching factor: 14 (there are always 7 regions whose color can be changed to either of 2 alternative values). Diameter: 7. (To turn state S into state T, change one region at a time from its value in S to its value in T.)

## D. Consider a starting state S in which all countries are labelled "White"? What is the minimum cost neighbor of this state? What is the cost of these two states?

**Answer:** The minimum cost neighbor has Massachussetts labelled "Red" or "Blue". The cost of the all white map is 10. The cost of the second map is 5.

## E Find a state S that is a local minimum of the cost function but is not a solution; that is, does not have cost 0. ("Local minimum" in the non-strict sense; the cost of S is less than or equal to the cost of any of its neighbors.)

**Answer:**

MA: Red

ME: White

NH: Blue

VT: White

NY: White

CT: Blue

RI: White

## Solve crossword puzzle problem (CSP) using backtracking

1 2 3 4 5

+---+---+---+---+---+ Given the list of words:

1 | 1 | | 2 | | 3 | AFT LASER

+---+---+---+---+---+ ALE LEE

2 | # | # | | # | | EEL LINE

+---+---+---+---+---+ HEEL SAILS

3 | # | 4 | | 5 | | HIKE SHEET

+---+---+---+---+---+ HOSES STEER

4 | 6 | # | 7 | | | KEEL TIE

+---+---+---+---+---+ KNOT

5 | 8 | | | | |

+---+---+---+---+---+

6 | | # | # | | # | The numbers 1,2,3,4,5,6,7,8 in the crossword

+---+---+---+---+---+ puzzle correspond to the words

that will start at those locations.

We introduce a variable to represent each word in the puzzle. So we have the variables:

VARIABLE | STARTING CELL | DOMAIN

================================================

1ACROSS | 1 | {HOSES, LASER, SAILS, SHEET, STEER}

4ACROSS | 4 | {HEEL, HIKE, KEEL, KNOT, LINE}

7ACROSS | 7 | {AFT, ALE, EEL, LEE, TIE}

8ACROSS | 8 | {HOSES, LASER, SAILS, SHEET, STEER}

2DOWN | 2 | {HOSES, LASER, SAILS, SHEET, STEER}

3DOWN | 3 | {HOSES, LASER, SAILS, SHEET, STEER}

5DOWN | 5 | {HEEL, HIKE, KEEL, KNOT, LINE}

6DOWN | 6 | {AFT, ALE, EEL, LEE, TIE}

The domain of each variable is the list of words that may be the value of that variable. So, variable 1ACROSS requires words with five letters, 2DOWN requires words with five letters, 3DOWN requires words with four letters, etc. Note that since each domain has 5 elements and there are 8 variables, the total number of states to consider in a naive approach is 5\*\*8 = 390,625.

The constraints are all binary constraints:

1ACROSS[3] = 2DOWN[1] i.e. the third letter of 1ACROSS must be equal

to the first letter of 2DOWN

1ACROSS[5] = 3DOWN[1]

4ACROSS[2] = 2DOWN[3]

4ACROSS[3] = 5DOWN[1]

4ACROSS[4] = 3DOWN[3]

7ACROSS[1] = 2DOWN[4]

7ACROSS[2] = 5DOWN[2]

7ACROSS[3] = 3DOWN[4]

8ACROSS[1] = 6DOWN[2]

8ACROSS[3] = 2DOWN[5]

8ACROSS[4] = 5DOWN[3]

8ACROSS[5] = 3DOWN[5]

We consider three solution methods for constraint satisfaction problems, Generate-and-Test, Backtracking [possibly Dependency Directed], and Consistency Driven.

**Backtracking**

We order the variables in some fashion, trying to place first the variables that are more highly constrained or with smaller ranges. This order has a great impact on the efficiency of solution algorithms and is examined elsewhere. We start assigning values to variables. We check constraint satisfaction at the earliest possible time and extend an assignment if the constraints involving the currently bound variables are satisfied.

**Example:** In our crossword puzzle we may order the variables as follows: 1ACROSS, 2DOWN, 3DOWN, 4ACROSS, 7ACROSS, 5DOWN, 8ACROSS, 6DOWN. Then we start the assignments:

1ACROSS <- HOSES

2DOWN <- HOSES => failure, 1ACROSS[3] not equal to 2DOWN[1]

<- LASER => failure

<- SAILS

3DOWN <- HOSES => failure

<- LASER => failure

<- SAILS

4ACROSS <- HEEL => failure

<- HIKE => failure

<- KEEL => failure

<- KNOT => failure

<- LINE => failure, backtrack

3DOWN <- SHEET

4ACROSS <- HEEL

7ACROSS <- AFT => failure

................................

What we have shown is called *Chronological Backtracking*, whereby variables are unbound in the inverse order to the the order used when they were bound. *Dependency Directed Backtracking* instead recognizes the cause of failure and backtracks to one of the causes of failure and skips over the intermediate variables that did not cause the failure.

The following is an easy way to do dependency directed backtracking. We keep track at each variable of the variables that precede it in the backtracking order and to which it is connected directly in the constraint graph. Then, when instantiation fails at a variable, backtracking goes in order to these variables skipping over all other intermediate variables.

Notice then that we will backtrack at a variable up to as many times as there are preceding neighbors. [This number is called the *width* of the variable.] The time complexity of the backtracking algorithm grows when it has to backtrack often. Consequently there is a real gain when the variables are ordered so as to minimize their largest width.

## There’s a big football game tonight, and you can’t miss it. You’re trying to decide whether to watch it in person or on TV. Watching it in person requires having some money for a ticket. Watching it on TV is only possible if you have a TV and there isn’t a local television blackout on the game. If you need money for a ticket, you can always sell your TV. Figure 1 shows a CSP representation for this planning problem where the goal is to watch the game.

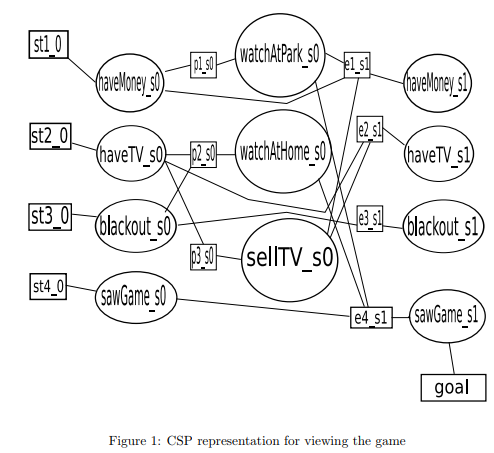
## A. What are the actions?

## B. What are the state variables?

## C. What is the horizon shown in Figure 1?

## D. Give the truth tables for the precondition constraint for action watchAtPark (labelled p1 s0 in the figure) and the effect constraint between blackout at step 0 and blackout at step 1 (labelled e3 s1).

## E. What is the minimum horizon needed to achieve the goal, if the start constraints specify that you have no money and that there is a TV blackout?

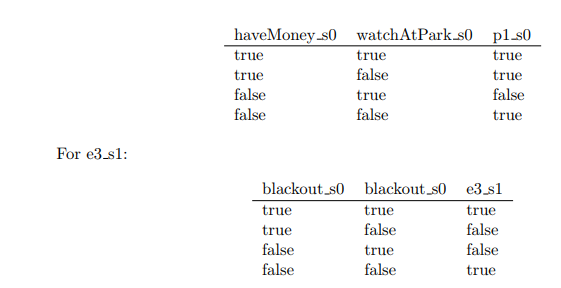
****

• What are the actions? Answer: watchAtPark, watchAtHome, sellTV

• What are the state variables? Answer: haveMoney, haveTV, blackout, sawGame

• What is the horizon shown in Figure 1? Answer: The horizon is 1.

• Give the truth tables for the precondition constraint for action watchAtPark (labelled p1 s0 in the figure) and the effect constraint between blackout at step 0 and blackout at step 1 (labelled e3 s1). Answer: For p1 s0:

****

• What is the minimum horizon needed to achieve the goal, if the start constraints specify that you have no money and that there is a TV blackout?

Answer: A horizon of 2. At step 1 you sell the TV and at step 2 you watch the game in person.

**MCQs & answers**

**1.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations.

(a) Constraints satisfaction problems

(b) Uninformed search problems

(c) Local search problems

(d) All of the above

**2.** USA + USSR = PEACE find P + E + A + C + E.

(a) 10

(b) 14

(c) 25

(d) 12

**3.** What among the following constitutes to the incremental formulation of CSP?

(a) Path cost

(b) Goal cost

(c) Successor function

(d) All of the above

**4.** tee + let = All, where E = 5 find A + L + L.

(a) 17

(b) 9

(c) 10

(d) 12

**5.** If, Ever + Since = Darwin, then D + a + r + w + i + n is \_\_\_\_\_\_\_\_\_\_\_?

(a) 23

(b) 41

(c) 34

(d) 16

**6.** To overcome the need to backtrack in CSP can be eliminated by \_\_\_\_\_\_\_\_\_\_\_

(a) Forward searching

(b) Constraint propagation

(c) Backtrack after a forward search

(d) Omitting the constraints and focusing only on goals

**7.** If (HE)^H=SHE, where the alphabets take the values from 0 to 9 and all the alphabets are single digit then find the value of (S+H+E)?

(a) 14

(b) 19

(c) 13

(d) 15

**8.** Constraint satisfaction problems on finite domains are typically solved using a form of \_\_\_\_\_\_\_\_\_\_\_.

(a) Search algorithms

(b) Heuristic search algorithms

(c) Greedy search algorithms

(d) All of the above

**Answers**

**1. (a) 2. (a) 3. (d) 4. (c) 5. (a) 6. (a) 7. (c) 8. (d)**