

| **Title:** Implementation of Constraint Satisfaction concepts |
| --- |

**Objective:** Implementation of Local search algorithm

**Expected Outcome of Experiment:**

| **Course Outcome** | **After successful completion of the course students should be able to** |
| --- | --- |
| **CO2** | Analyse and solve problems for goal based agent architecture (searching and planning algorithms). |

**Books/ Journals/ Websites referred:**

1. **“Artificial Intelligence: a Modern Approach” by Russel and Norving, Pearson education Publications**
2. **“Artificial Intelligence” By Rich and knight, Tata Mcgraw Hill Publications**
3. **https://www.geeksforgeeks.org/constraint-satisfaction-problems-csp-in-artificial-intelligence/**
4. **https://www.scaler.com/topics/artificial-intelligence-tutorial/constraint-satisfaction-problem-in-ai/**
5. **https://intellipaat.com/blog/what-is-constraint-satisfaction-problem-in-artificial-intelligence/#algorithms\_in\_csp**
6. **https://www.almabetter.com/bytes/tutorials/artificial-intelligence/constraint-satisfaction-problem-in-ai**

**Pre Lab/ Prior Concepts:** Informed, uninformed search,Local search

**Historical Profile:**

The goal of AI is to create intelligent machines that can perform tasks that usually require human intelligence, such as reasoning, learning, and problem-solving. One of the key approaches in AI is the use of constraint satisfaction techniques to solve complex problems.

CSP is a specific type of problem-solving approach that involves identifying constraints that must be satisfied and finding a solution that satisfies all the constraints. CSP has been used in a variety of applications, including scheduling, planning, resource allocation, and automated reasoning.

**New Concepts to be learned:** Constraint Satisfaction, CSP with backtracking

**Definition:**

A **Constraint Satisfaction Problem** in artificial intelligence involves a set of variables, each of which has a domain of possible values, and a set of constraints that define the allowable combinations of values for the variables. The goal is to find a value for each variable such that all the constraints are satisfied.

More formally, a CSP is defined as a triple (*X*,*D*,*C*), where:

* X is a set of variables { *x*1​, *x*2​, ..., *xn*​}.
* D is a set of domains {*D*1​, *D*2​, ..., *Dn*​}, where each *Di*​ is the set of possible values for *xi*​.
* C is a set of constraints {*C*1​, *C*2​, ..., *Cm*​}, where each *Ci*​ is a constraint that restricts the values that can be assigned to the corresponding variables.

The goal of a CSP is to find an assignment of values to the variables that satisfies all the constraints. This assignment is called a solution to the CSP.

Constraint Satisfaction Problems (CSPs) consist of several basic components that define the problem and its solution space. These components help structure and solve problems where variables need to be assigned values within certain constraints to find a valid solution. The basic components of CSPs are:

* **Variable:** Variables are the items that need to be determined. The objects in a CSP that must have values given to them in order to meet a specific set of constraints are known as variables. Boolean, integer, and category variables are just a few examples of a variety of variables. The set of variables is denoted as {X1,X2,…..,Xn} and often takes values from columns that are predefined and represent possible values they can assume.
* **Domain:**Domains describe the variety of possible values that a variable might have. A domain may be finite or limitless, depending on the problem. For example, in Sudoku, a variable that represents a puzzle cell can have as its domain a range of values from 1 to 9. It is denoted by **“D”**. Domains can be finite, like {1, 2, 3}, or continuous, such as real numbers between 0 and 1.
* **Constraints:**Constraints are the rules that control how variables interact with one another. The ranges of acceptable values for variables are determined by constraints in a CSP. The different types of constraints include unary constraints, binary constraints, and higher-order constraints, to mention a few. For example, in a sudoku puzzle, the limitations might be that only one of each number from 1 to 9 can appear in each row, column, and 3\*3 boxe.

Constraints can be expressed in various ways, such as equations, inequalities, or logical expressions.

**Domain Categories in CSP**

* **Finite Domain:** Variables in many CSPs have finite domains that are made up of discrete values. Examples comprise:
  + **Binary Domains**: Domains that only have two values (for binary CSPs, this would be 0 and 1).
  + **Integer Domains:** Domains made up of a limited number of integer values, such as 1, 2, 3, and 4, are known as integer domains.
  + **Enumeration Domains:** Domains containing a limited number of distinct values, such as “red, green, and blue” in an issue involving color assignment.
* **Continuous Domains:** Some CSPs contain variables whose domains are continuous, i.e., they can accept any real number falling within a given range. Examples comprise:
  + **Real-valued Domains:** Variables may accept any real number that falls within a given range (for example, X [0, 1]).
  + **Interval Domains**: Variables are limited to a specific range of real values in the interval domain. e.g., X ∈ [−π, π])

**Types of Constraints in CSP**

* **Unary Constraints:** Unary constraints limit the possible values of a single variable without considering the values of other variables. It is the easiest constraint to find, as it has only one parameter. Example: The expression X1 ≠ 7 says that the variable X1 cannot have the value 7.
* **Binary Constraints:** Binary constraints describe the relationship between two variables and consist of only two variables. Example: X1< X2 indicates that X1 must be less than X2 in order to be true.
* **Global Constraints:** In contrast to unary or binary constraints, global constraints involve multiple variables and impose a more complex relationship or restriction between them. Global constraints are often used in CSP problems to capture higher-level patterns, structures, or rules. These restrictions can apply to any number of variables at once and are not limited to pairwise interactions.

It is further divided into two main categories:

* **Alldifferent Constraint:**The Alldifferent constraint (AllDiff) requires that each variable in a set of variables has a unique value. You commonly apply alldifferent constraints, when you want to be sure that no two variables in a set can take the same value. Example: The expression alldifferent(X1, X2, X3) ensures that the values of X1, X2, and X3 must be unique.
* **Sum Constraint:**The Sum Constraint requires that the sum of the values assigned to a group of variables meet a particular requirement. It is useful for expressing restrictions like “the sum of these variables should equal a certain value.” Example: The expression Sum(X1, X2, X3) = 15 demands that the sum of the values for X1, X2, and X3 be 15.

**Algorithms in CSP**

* **The Backtracking Algorithm:** The backtracking algorithm is a popular method for resolving CSPs. It looks for the search space by picking a variable, setting a value for it, and then recursively scanning through the other variables. In the event of a conflict, it goes back and tries a different value for the preceding variable. The backtracking algorithm’s essential elements are:
  + **Variable Ordering:** The order in which variables are chosen is known as variable ordering.
  + **Value Ordering:** The sequence in which values are assigned to variables is known as value ordering.
  + **Constraint Propagation**: Reducing the domain of variables based on constraint compliance is known as constraint propagation.
* **Forward Checking:** The backtracking technique has been improved using forward checking. It tracks the remaining accurate values of the unassigned variables after each assignment and reduces the domains of variables whose values don’t match the assigned ones. As a result, the search space is smaller, and constraint propagation is more effectively accomplished.
* **Constraint Propagation:** Constraint propagation techniques reduce the search space by removing values inconsistent with current assignments through local consistency checks. To do this, techniques like generalized arc consistency and path consistency are applied.

**The sequence in which variable-constraint assignments are considered by CSP algorithms to improve the backtracking efficiency:**

1. Variable Selection: Choose an unassigned variable. The selection can be done in various ways, such as choosing the variable with the fewest legal values left (minimum remaining values), or the variable that is involved in the largest number of constraints on other unassigned variables (degree heuristic).
2. Value Ordering: For the selected variable, consider its possible values one by one. The order can be determined by heuristics, such as least constraining value, which prefers the value that rules out the fewest choices for the neighboring variables in the constraint graph.
3. Assignment: Assign a value to the selected variable. The assignment should not violate any constraints.
4. Forward Checking: After each assignment, look ahead and check the remaining variables to see if they have any valid values left. If a variable has no valid values left, there is no need to proceed further, and the algorithm can backtrack immediately.
5. Backtracking: If an assignment leads to a future failure, backtrack to the previous variable and try the next value. This is done until a solution is found or all values have been tried.
6. Inference (Arc Consistency): This is an optional step that can further improve efficiency. Arc consistency ensures that for every value in the domain of each variable, there is some consistent value of each neighboring variable.

This sequence is followed until a solution is found or all possibilities have been exhausted.

**Problem chosen:**

E A T + T H A T = A P P L E

**Step by step solution to the problem:**

Step 1: Define the problem. Here, the Criptarithmetic Puzzle Problem is being solved with Constraint Satisfaction Problem (CSP) techniques.

a = 'EAT'

b = 'THAT'

c = 'APPLE'

Step 2: Define variables. It is a set of unique characters from strings 'a', 'b' and 'c'.

vars = "".join(set(a + b + c))

# vars = ETAHPL

Step 3: Define domains. Each variable can take value from 0 to 9.

combinations = [combination for combination in permutations(range(10), len(vars))]

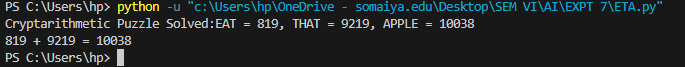
*In the above code,* range(10) *specifies the domain of the variables.*

Step 4: Define constraint. The leading variable of each word cannot be zero, i.e., 'E', 'T' and 'A' cannot be zero as they are leading characters of 'EAT', 'THAT' and 'APPLE' respectively. Further, each variable must have unique value, i.e, E != T != A != H != P != L

combinations = [combination for combination in permutations(range(10), len(vars))]

*In the above code, the* permutations(range(10), len(vars)) *ensures that tuples are generated such that each tuple has as many integers as the number of variables, specified by* len(vars) *, and that each variable lies in the range specified by* range(10)*. Further, the working of* permutations() *function is such that for each tuple, each integer within that tuple is unique.*

Step 5: Find solution



**Code:**

from itertools import permutations

a = 'EAT'

b = 'THAT'

c = 'APPLE'

vars = "".join(set(a + b + c))

combinations = [combination for combination in permutations(range(10), len(vars))]

for combination in combinations:

    A = a

    B = b

    C = c

    for i in range(len(vars)):

        char = vars[i]

        num = str(combination[i])

        A = A.replace(char, num)

        B = B.replace(char, num)

        C = C.replace(char, num)

    A = int(A)

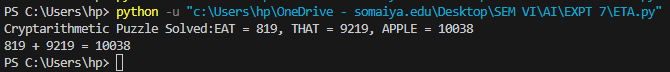
    B = int(B)

    C = int(C)

    if(A + B == C):

        print(f'Cryptarithmetic Puzzle Solved:{a} = {A}, {b} = {B}, {c} = {C}\n{A} + {B} = {C}')

**Output:**

****

**Post Lab objective Questions:**

**To need to backtrack in constraint satisfaction problem 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**

**Ans. b) Constraint Propagation**

The need to backtrack in a constraint satisfaction problem can be eliminated by **Constraint Propagation**.

Constraint propagation is a technique used in constraint satisfaction problems (CSPs) where the constraints are used to significantly reduce the search space. It works by inferring that certain values are not part of the solution and eliminating them from the domains of the variables, thus reducing the need for backtracking.

While constraint propagation can significantly reduce the need for backtracking, it does not guarantee the elimination of backtracking in all cases. The effectiveness of constraint propagation can depend on the specific characteristics of the CSP.

a) **Forward Searching**: This is a strategy where the solver moves forward from the start state and uses the constraints to reduce the search space. However, it doesn’t eliminate the need for backtracking. If a dead-end is reached or a constraint is violated, the solver will have to backtrack.

c) **Backtrack after a forward search**: This is essentially the standard backtracking algorithm. The solver performs a forward search and if it hits a dead-end (i.e., it can’t find a solution that satisfies all constraints), it backtracks to a previous state and tries a different path. So, by definition, this option involves backtracking.

d) **Omitting the constraints and focusing only on goals**: This would not be a viable strategy for solving constraint satisfaction problems. The constraints are a critical part of these problems - they define the conditions that the solution must satisfy. Ignoring the constraints might lead to finding solutions that don’t actually satisfy the problem statement, and it doesn’t eliminate the need for backtracking.

**Consider a problem of preparing a schedule for a class of student. What type of problem is this?**

**a) Search Problem**

**b) Backtrack Problem**

**c) CSP**

**d) Planning Problem**

**Ans. c) CSP**

The problem of preparing a schedule for a class of students is a type of **Constraint Satisfaction Problem (CSP)**.

In this problem, we have to assign time slots to classes in such a way that no two classes are at the same time and the same place, among other constraints. Each class, time, and place can be seen as a variable, and the restrictions as constraints. The goal is to find an assignment of values to variables that satisfies all constraints.

a) **Search Problem**: This is a more general category. In a search problem, one is looking for a path from a start state to a goal state. While scheduling could be seen as a search problem, it’s more specifically a constraint satisfaction problem because of the need to satisfy multiple constraints simultaneously.

b) **Backtrack Problem**: Backtracking is a strategy used for solving problems, including CSPs, where one tries out different solutions and when a solution doesn’t work, one goes back and tries a different one. It’s not a type of problem itself, but rather a method for solving certain types of problems, including CSPs.

d) **Planning Problem**: Planning problems involve deciding on a sequence of actions to achieve a specific goal, given certain conditions or states. While there might be elements of planning in a scheduling problem, the key aspect of a scheduling problem is satisfying multiple constraints, which makes it a CSP.

**Q1. How do you solve a CSP Problem?**

**Ans.** Constraint Satisfaction Problems (CSPs) can be challenging to solve due to their combinatorial nature. However, several techniques, such as backtracking and constraint propagation, can be employed to find valid solutions efficiently.

**1. Backtracking Search for CSP in Artificial Intelligence:**

Backtracking is a widely used technique for solving CSPs. It is a systematic search algorithm that explores possible assignments for variables, backtracking when it encounters constraints that cannot be satisfied. The algorithm follows these steps:

* Choose an unassigned variable.
* Select a value from its domain.
* Check if the assignment violates any constraints.
* If a constraint is violated, backtrack to the previous variable and try another value.
* Continue this process until all variables are assigned values, or a valid solution is found.

**2. Constraint Propagation:**

Constraint propagation is a powerful technique that enforces constraints throughout the CSP solving process. It narrows down the domains of variables by iteratively applying constraints. It's often used in conjunction with backtracking to improve efficiency. The concept of constraint propagation can be illustrated as follows:

* **Step 1:** Start with an initial CSP problem in ai with variables, domains, and constraints.
* **Step 2:** Apply constraints that have been specified in the problem to narrow down the domains of variables. For instance, if two variables have a binary constraint that one must be double the other, this constraint will eliminate many inconsistent assignments.
* **Step 3:** After constraint propagation, some variables may have their domains reduced to only a few possibilities, making it easier to find valid assignments.
* **Step 4:** If a variable's domain becomes empty during propagation, it indicates that the current assignment is inconsistent, and backtracking is needed.

**Q2. Explain CSP with an example.**

**Ans.**

Let's consider a simplified Sudoku puzzle to illustrate the problem-solving process step by step:

Variables: 9x9 grid cells

Domains: Numbers from 1 to 9

Constraints: No number can repeat in the same row, column, or 3x3 subgrid.

Step 1: Start with an empty Sudoku grid.

Step 2: Apply the initial constraints for the given numbers, reducing the domains of variables based on the puzzle's clues.

Step 3: Use constraint propagation to narrow down the domains further. For example, if a row has two cells with domains {2, 5}, and the constraint specifies that these two cells cannot have the same number, we can eliminate the possibility of 5 for one of them.

Step 4: Continue applying constraints and propagating until the domains of variables are either empty or filled with single values. If they are all filled, you have a valid solution. If any variable's domain is empty, you backtrack to the previous step and try an alternative assignment.

This simple example demonstrates how backtracking and constraint propagation work together to efficiently find a solution to a CSP. The combination of systematic search and constraint enforcement allows for solving complex problems in various domains.