Juan Barrera
Professor Mark Voortman
Introduction to AI
Module 3 Assignment (Part B)
October 6 2025

Questions:

Think of a problem not identified in the book and explain in a detailed fashion how you would apply CSP to solve it.

The problem I will be taking into account is course scheduling, will be modeled as a CSP to be solved on a much larger scale. Here the courses are variables, with the domain listing the Acceptable (TimeSlot, Room) assignments: for example, Course_101 --> (Monday 9 am, Room A) or (Tuesday 11 am, Room B). Constraints are placed to guarantee that professors and students are not in conflicting schedules; constraints such as Time(Course_101) ≠ Time(Course_202) if both courses are assigned to the same professor or are taken by the same student. Then constraints are room-wise: Capacity(Room) ≥ Enrollment(Course), Equipment(Room) ⊇ Requirements(Course). Preferences like avoiding early or late hours can be modeled by excluding certain time slots from a professor's domain. To solve this, we can apply backtracking search enhanced with forward checking and arc consistency (AC-3), using heuristics like Minimum Remaining Values (MRV) and Least Constraining Value to guide variable and value selection. Forward checking would exclude that slot from courses sharing students or instructors, should it be assigned for Course_101 at (Mon 9am, Room A). In Python, one can define variables and constraints using the python-constraint library as shown in problem.addVariable("Course_101", [("Mon 9am", "Room A"), ("Tue 11am", "Room B")]) and problem.addConstraint(lambda c1, c2: c1[0] != c2[0], ("Course_101", "Course_202")). At the end of this CSP formulation, a conflict-free schedule-with the consideration of resources, meets institutional and individual constraints.

How many solutions are there for the map-coloring problem in Figure 6.1Links to an external site.? How many solutions if four colors are allowed? Two colors?

The map-coloring problem shown in Figure 6.1 is a basic constraint satisfaction problem (CSP) whose objective is to color every one of the regions in Australia such that no two adjacent regions have the same color. This is not merely a combinatorial problem it displays the union of graph theory, algorithmic reasoning, and computational complexity. The constraint graph in part (b) of the figure abstracts the geographical map into a set of nodes (regions) and edges (adjacency relations), allowing us to analyze the problem using formal methods. When restricted to three colors, which is the minimum number required to satisfy the constraints for this particular graph, there are 18 distinct valid solutions. These solutions are found through systematic enumeration or backtracking algorithms respecting the binary constraints deduced from each edge. The graphical

structure. Particularly the central role of South Australia (SA), which borders five other areas creates a dense set of constraints that significantly limits the number of possible assignments.

As the number of colors available increases to four, the constraint space opens to encompass 94 valid solutions. This increase is due to exponential expansion of the solution space under a relaxation of constraints, as widely documented in CSP research. Adding an additional color introduces increased degrees of freedom, reducing the likelihood of conflicts and enabling more assignments that hold even under the adjacency constraints. It is different with two colors, and the problem is then unsolvable. This is due to the fact that the constraint graph has odd-length cycles, which violate the conditions necessary for a graph to be bipartite, a necessary condition for two-colorability. The SA, NT, Q, NSW, V subgraph induces cycles that cannot be colored in two sets so that no two neighboring nodes share the same color.

This analysis not only highlights the computational impact of constraint density and graph topology but also suggests the importance of domain-specific heuristics for effectively solving CSPs. The Australia map-coloring problem is an educational case study of artificial intelligence and operations research, demonstrating how real-world problems can be expressed formally and solved using algorithmic approaches. It also illustrates the overall principle that the complexity and tractability of CSPs are extremely sensitive to the interaction between domain size and constraint structure.

Explain why it is a good heuristic to choose the variable that is most constrained but the value that is least constraining in a CSP search.

In a constraint satisfaction problem (CSP), the use of the heuristic of selecting the most constrained variable (often called the Minimum Remaining Values, or MRV, heuristic) and setting it to the least constraining value is an efficient search improvement technique. Choosing the most constrained variable is choosing the variable with the smallest number of legal values remaining, which allows for early failure detection: if a variable has no legal choice, it's better to discover that as soon as possible and backtrack right away instead of squandering time on dead ends. By doing this, the search attacks the hardest areas of the problem first, making it more likely that the search space will be cut down early on. Having selected a variable, assigning it the value which is most relaxing to wit, the one which rules out the fewest possibilities for the variables to its left

will allow flexibility to be saved in the remainder of the search. It keeps other variables with as many options as possible, reducing the potential for subsequent conflict and backtracking. Together, these heuristics temper aggressive cutting with cautious assignment, making the search both wiser and more efficient.