Constraint Programming Algorithms in Generative AI

Shishir Biyyala

October 26th, 2024

Outline

What is Constraint Programming (CP)?

What is Constraint Programming?

- CP is a declarative paradigm used to solve combinatorial problems
- It involves specifying variables, constraints, and using search algorithms to find solutions
- Sub-field of AI that intersects with optimization and decision-making

CP Techniques in Generative AI

- Constraint Satisfaction Problems (CSPs) and Constraint Optimization Problems (COPs)
- Techniques like Backtracking, Constraint Propagation, and Local Search
- How CP is used in structured generation, such as automated scheduling, text generation, and design automation

Constraint Propagation

- Constraint Propagation is a key CP technique used to reduce the search space
- It removes invalid possibilities early in the search, leading to faster solutions
- Relevance to Generative AI: Helps AI systems make structured decisions in real-time

Conference Scheduling Problem

Conference Scheduling

- Problem setup: Assign talks to rooms and time slots with constraints
- Constraint: No overlapping talks in the same room, buffer time between sessions
- CP approach allows for fast query responses with constraints being propagated

Conference Scheduling: Setting up the Model

```
// Create a new model
Model model = new Model("Conference Scheduling");
// Define variables-talks are assigned time slots and rooms
int numTalks = 4; int numRooms = 3; int numSlots = 4;
// Define variables for rooms and time slots for each talk
IntVar[] roomAssignments = new IntVar[numTalks];
IntVar[] slotAssignments = new IntVar[numTalks];
// Each talk is assigned a room from 0 to numRooms-1
IntStream.range(0, numTalks).forEachOrdered(i -> {
  roomAssignments[i]=model.intVar("Room"+i, 0, numRooms-1);
  slotAssignments[i]=model.intVar("Slot"+i, 0, numSlots-1);
});
```

Step 2: Adding Constraints - No Overlap in the Same Room

- Next, we add a constraint to ensure that no two talks in the same room overlap.
- If two talks are in the same room, they must be in different time slots.

```
for (int i = 0; i < numTalks; i++) {
    for (int j = i + 1; j < numTalks; j++) {
        model.ifThen(
            model.arithm(roomAssignments[i], "=", roomAssign
            model.arithm(timeSlotAssignments[i], "!=", timeS
        );
    }
}
```

Step 3: Adding Constraints - Buffer Time Between Rooms

- Now, we add a constraint to ensure that talks in different rooms have buffer time between them.
- This allows attendees to move between rooms.

```
for (int i = 0; i < numTalks; i++) {
    for (int j = i + 1; j < numTalks; j++) {
        model.ifThen(
            model.arithm(roomAssignments[i], "!=", roomAssig
            model.distance(timeSlotAssignments[i], timeSlotA
        );
    }
}
```

Step 4: Solving the Model

- Finally, we use the solver to find a valid assignment for rooms and time slots.
- The solver will attempt to find a solution that satisfies all the constraints.

```
Solver solver = model.getSolver();
if (solver.solve()) {
    for (int i = 0; i < numTalks; i++) {
        System.out.println("Talk " + i + " assigned to Room
                " at TimeSlot " + timeSlotAssignments[i].get
    }
} else {
    System.out.println("No valid schedule found!");
}
```

Applying CP to Real-World Systems

- Constraint Propagation in intelligent systems, such as scheduling, resource allocation, and logistics
- CP's power to handle complex rules while enabling fast, reliable solutions in Generative AI systems
- Applications in areas like supply chain optimization, AI-driven design, and autonomous decision-making

Conclusion

- Summary: CP is a powerful tool for solving complex problems in AI
- Constraint Propagation allows for rapid, efficient query handling in Generative AI
- Invitation to explore more use cases and applications of CP in intelligent systems

Q&A