Constraint Logic Programming and Vehicle Routing Problem

Mikael Arakelian

ezhome

May 24, 2017

Constraint Programming

So what is Constraint Programming?

Constraint Programming

So what is Constraint Programming?
You can think of it as a declarative computational paradigm, where you express the target solution in terms of variables and constraints between them.

Constraint Programming

So what is Constraint Programming?
You can think of it as a declarative computational paradigm, where you express the target solution in terms of variables and constraints between them.
So you don't specify the sequence of steps of finding a solution, but rather define the properties you want the solution to satisfy.

Constraint Satisfaction Problem Definition

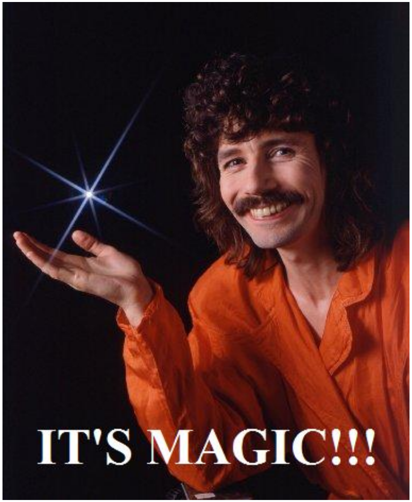
Given a set of *variables* $\{X_1, X_2, \ldots, X_n\}$ where with each variable we associate its *domain* $D(X_i)$ and a set of *constraints* $\{C_1, C_2, \ldots, C_m\}$ we need to assign each variable to a value, such that all constraints are satisfied

Constraint Optimization Problem Definition

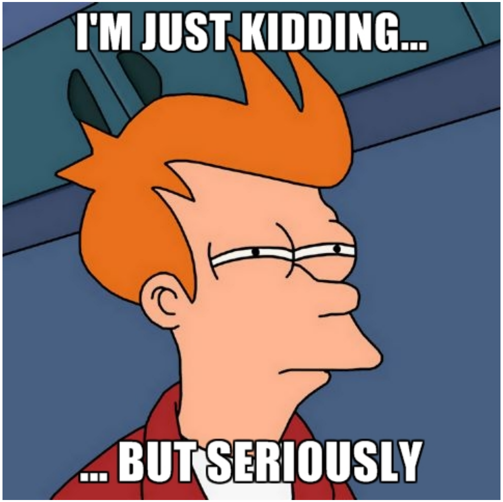
Basically almost the same as the CSP, with an objective function defined, which needs to be minimized (maximized).

So... How does it work?

So... How does it work?



So... How does it work?



Main Approaches

There are 2 main approaches for solving the CSP and COP

1. Complete Search (Backtracking and Constraint Propagation)
2. Local Search

Complete Search

1. Use constraints to reduce the set of values each variable can take without losing (Constraint Propagation or Pruning)
2. Choose a value for a variable when no further deduction can be made. If the choice did not result in a solution then backtrack (Backtracking)

Complete Search

1. Use constraints to reduce the set of values each variable can take without losing (Constraint Propagation or Pruning)
2. Choose a value for a variable when no further deduction can be made. If the choice did not result in a solution then backtrack (Backtracking)

Pros: If a solution exists, we will necessarily find it

Cons: It has an exponential behaviour

Local Search

1. Make an initial assignment of variables (possibly violating some of constraints)
2. Iteratively change the assigned values of the variables by making a small pertubation of assigned values (looking at the neighbourhood of current solution), maximizing the number of constraints satisfied (or depending on the objective function to be optimizer)

Local Search

1. Make an initial assignment of variables (possibly violating some of constraints)
2. Iteratively change the assigned values of the variables by making a small pertubation of assigned values (looking at the neighbourhood of current solution), maximizing the number of constraints satisfied (or depending on the objective function to be optimizer)

Pros: Much faster, suits better for using CP for optimization (in optimization we may forbid visiting solutions that violate constraints)
Cons: We might not find a solution, even if it exists. In optimization problems optimality is not guaranteed

Example 1

A famous example in the constraint programming community is cryptarithmetic.

Example 1

A famous example in the constraint programming community is cryptarithmetic.
It is a mathematical game where the digits of some numbers in an equation are represented by letters. Each letter represents a unique digit. The goal is to find the digits such that a given mathematical equation is verified.

Example 1

A famous example in the constraint programming community is cryptarithmetic.
It is a mathematical game where the digits of some numbers in an equation are represented by letters. Each letter represents a unique digit. The goal is to find the digits such that a given mathematical equation is verified.
We are going to solve the following cryptarithmetic:

$$GREEN + GARDEN = EZHOME$$

GREEN + GARDEN = EZHOME

To model the problem in terms of CSP we need to answer the following questions

1. What are the variables in the problem? What are their domains?
2. What are the constraints of the problem?

GREEN + GARDEN = EZHOME

Let's model our problem the following way:

1. We have a variable for each letter in the equation. All of them should be digits, so the domain of each variable is the set $\{0, 1, \ldots, 9\}$. Let's denote the variables as X_{letter}
2. The constraints are: $X_G \neq 0, X_E \neq 0$, $X_{l_i} \neq X_{l_j}$, and the actual equation

$$\sum_{l_i \in GREEN} 10^i * X_{l_i} + \sum_{l_i \in GARDEN} 10^i * X_{l_i} = \sum_{l_i \in EZHOME} 10^i * X_{l_i}$$

Example 2

Let's imagine that some engineers have decided to go to Berlin. They are going to stay in a hotel in twin-rooms to get to know each other better. Unfortunately, some of the engineers are really bad snorers and some just cannot stand it.

Example 2

Let's imagine that some engineers have decided to go to Berlin. They are going to stay in a hotel in twin-rooms to get to know each other better. Unfortunately, some of the engineers are really bad snorers and some just cannot stand it.
The first question we may ask is: is it possible to split them into m rooms? (CSP version)

Example 2

Let's imagine that some engineers have decided to go to Berlin. They are going to stay in a hotel in twin-rooms to get to know each other better. Unfortunately, some of the engineers are really bad snorers and some just cannot stand it.

The first question we may ask is: is it possible to split them into m rooms? (CSP version)

We can also pose the problem in a different way: what is the minimal number of necessary rooms? (COP version)

The Room Sharing Problem (CSP)

Let's model the problem:

- For each person we have a variable $Room_i$ and $D(Room_i) = \{0, 1, \ldots, m-1\}$
- Constraints: $\sum_{i=1}^{n}(Room_i = j) \leq 2$, $Room_i \neq Room_j$ for all (i, j) that don't want to be in the same room,

The Room Sharing Problem (COP)

In the optimization version we don't know the of rooms beforehand, so the previous model won't work

The Room Sharing Problem (COP)

In the optimization version we don't know the of rooms beforehand, so the previous model won't work
We can think of the problem the following way: for each person we pick the person he is paired up with, so our variable should represent the person we paired him up.

The Room Sharing Problem (COP)

In the optimization version we don't know the of rooms beforehand, so the previous model won't work

We can think of the problem the following way: for each person we pick the person he is paired up with, so our variable should represent the person we paired him up.

As a result, here is our new model:

1. N_i represents the neighbour of ith person. $D(N_i) = \{1, 2, \dots, n\}$
2. N_i's must be all different, preferences and $N_{N_i} = i$,
3. Objective function: $\sum_{i=1}^{n}(N_i = i) \to \min$

Vehicle Routing Problem Definition

The Vehicle Routing Problem is a combinatorial optimization problem which asks the following question: "What is the optimal set of routes for a fleet of vehicles delivering goods or services to various locations?"

Vehicle Routing Problem Definition

The Vehicle Routing Problem is a combinatorial optimization problem which asks the following question: "What is the optimal set of routes for a fleet of vehicles delivering goods or services to various locations?" There are various generalizations of this problem, such as Capacitated VRP (the total demand of the locations on a vehicle's route cannot exceed its capacity), VRP with Time Windows and a whole zoo of them

How we think about a VRP

Suppose we have n customers and m vehicles. We usually think about the VRP in terms of a graph on $n + 2m$ nodes. In terms of this graph we can reformulate the problem as follows: given a graph $G = (V, E)$ and $2m$ nodes $\{start_1, start_2, \ldots, start_m\}$ and $\{end_1, end_2, \ldots, end_m\}$ from V break down G into m simple paths where the ith path starts at $start_i$ and ends at end_i.

CP formulation of Classical VRP

So how can we reformulate graph representation of VRP in terms of CP?

CP formulation of Classical VRP

So how can we reformulate graph representation of VRP in terms of CP?

1. Let's introduce the variables $Next_i, Vehicle_i, Active_i$, where $Next_i$ corresponds the number of the next of ith node, $Vehicle_i$ corresponds to which vehicle node i corresponds to and $Active_i$ shows whether the ith node belongs to any route or not
2. $D(Next_i) = V$, $D(Active_i) = \{0, 1\}$,
   $D(Vehicle_i) = \{-1, 0, 1, \ldots, m-1\}$
3. Necessary constraints in VRP are: $AllDifferent(Next_i)$,
   $Vehicle_i = Vehicle_{Next_i}$ and a special constraint called $NoCycle$

VRP with Time Windows

Many vehicle routing problems involve scheduling deliveries or service calls to customers. A natural constraint for these problems is that a customer can only receive a delivery or service in a specified window of time. For example, a company might need to schedule a time window when a customer will be at home.

VRP with Time Windows

Many vehicle routing problems involve scheduling deliveries or service calls to customers. A natural constraint for these problems is that a customer can only receive a delivery or service in a specified window of time. For example, a company might need to schedule a time window when a customer will be at home.
How can we model this?

Modelling Time Windows in VRP

For simplicity, let's assume that all of the vehicles happen at the same day. For each node let us create an additional variable called $Time_i$ (arrival time in seconds) with its domain equal to
$D(Time_i) = \{0, 1, \ldots, 24 * 60 * 60 - 1\}$ and constraints
if $Active_i = 1$ then $Time_{Next_i} = Time_i + duration_i + driving\_time_i$.

Modelling Time Windows in VRP

For simplicity, let's assume that all of the vehicles happen at the same day. For each node let us create an additional variable called $Time_i$ (arrival time in seconds) with its domain equal to $D(Time_i) = \{0, 1, \dots, 24 * 60 * 60 - 1\}$ and constraints if $Active_i = 1$ then $Time_{Next_i} = Time_i + duration_i + driving\_time_i$. This variables enable us to model all types of constraints on time. For example:

1. $12 * 3600 <= Time_i <= 14 * 3600$ node i must be served after $12:00PM$ and before $14:00PM$
2. $Time_i isnotin\{(0, 7.5 * 360001), (13 * 3600, 24 * 60 * 60 - 1)\}$
3. $Time_{end_i} \leq 16 * 3600$

Modelling Capacity in VRP

Capacity can be modelled in VRP very similar to time

Thank You!