SCOP Python Package Cheat Sheet

SCOP Python Package Cheat Sheet

Overview

SCOP (Solver for Constraint Programming) is a solver designed for solving large-scale constraint optimization problems efficiently. It supports hard and soft constraints with weights to prioritize certain conditions.

Installation

Install SCOP using pip:

```
pip install scop
```

Core Concepts

Constraint Satisfaction Problem (CSP)

- **Variable**: The unknown to be determined by optimization.
- **Domain**: The set of possible values a variable can take.
- **Constraint**: Conditions limiting the combinations of variable values.
  - Types: Linear, Quadratic, Alldiff.

Weighted Constraints

- **Hard Constraint**: Must be satisfied; weight = 'inf'.
- **Soft Constraint**: May be violated; minimize weighted deviation.

Key Classes and Methods

Model Class

Used to define and solve a problem.

Attributes

- `name`: Name of the model.
- `variables`: List of variables in the model.
- `constraints`: List of constraints in the model.
- `Params`: Parameters for optimization (e.g., time limit, randomness).

Methods

- `addVariable(name, domain)`: Add a single variable.
- `addVariables(names, domain)`: Add multiple variables.
- `addConstraint(con)`: Add a constraint.
- `optimize()`: Solve the optimization problem.

Example

```
from scop import *

model = Model(name='example')
model.addVariable('x', [0, 1])
model.addVariable('y', [1, 2])
solution, violated = model.optimize()
print('Solution:', solution)
print('Violated Constraints:', violated)
```

Variable Class

Defines variables in a model.

Attributes

- `name`: Name of the variable.
- `domain`: Possible values for the variable.
- `value`: Assigned value after optimization.

Example

```
var = Variable(name='X', domain=[1, 2, 3])
print(var)
```

Constraint Classes

Define constraints for the model.

Linear Class

Represents linear constraints of the form:

```
coeff1 * x1 + coeff2 * x2 <= rhs
```

Example

```
L = Linear(name='linear_constraint', weight='inf', rhs=10, direction='<=')
L.addTerms([1, 2], [x1, x2], [1, 2])
```

Quadratic Class

Represents quadratic constraints of the form:

```
coeff * x1 * x2 <= rhs
```

Example

```
Q = Quadratic(name='quadratic_constraint', rhs=20)
Q.addTerms([3], [x1], ['A'], [x2], ['B'])
```

Alldiff Class

Ensures all variables in a set take unique values.

Example

```
A = Alldiff(name='unique_values')
A.addVariables([x1, x2, x3])
```

Parameters Class

Control optimization behavior.

Attributes

- `TimeLimit`: Time limit for optimization (default: 600 seconds).
- `OutputFlag`: Log verbosity (default: False).
- `RandomSeed`: Random seed for reproducibility (default: 1).

Example

```
params = Parameters()
params.TimeLimit = 300
print(params)
```

Plotting Optimization Performance

SCOP includes a `plot_scop` function to visualize optimization performance.

Example

```
from scop import plot_scop
fig = plot_scop('scop_out.txt')
fig.show()
```

Debugging Tips

1. Use `print` to inspect variables, constraints, and the model.
2. Check `scop_input.txt` and `scop_output.txt` for input/output validation.
3. Start with small test cases and gradually increase complexity.

Command-Line Execution

Run SCOP directly from the terminal:

```
./scop < scop_input.txt
```

Options include:

- `-time`: Set time limit.
- `-interval`: Set log display interval.

Example Problem: Job Assignment

Assign workers to jobs to minimize cost:

Job \ Worker	A	B	C
0	15	7	25
1	20	15	10
2	30	12	13

Solution

```
model = Model(name='Job Assignment')
A = model.addVariable('A', [0, 1, 2])
B = model.addVariable('B', [0, 1, 2])
C = model.addVariable('C', [0, 1, 2])

L = Linear(name='cost', weight='inf', rhs=1)
L.addTerms([15, 7, 25], [A, B, C], [0, 1, 2])
model.addConstraint(L)

solution, violated = model.optimize()
print('Solution:', solution)
```