OptSeq Python Package Cheat Sheet

OptSeq Python Package Cheat Sheet

Overview

OptSeq is a Python package for solving scheduling optimization problems. It is specifically designed to efficiently solve generalized resource-constrained scheduling models, leveraging metaheuristics to quickly find high-quality solutions even for large-scale problems.

Installation

Install OptSeq with the following command:

```
pip install optseq
```

Official documentation and practice problems can be found on the support page.

Key Classes and Methods

Model Class

Defines the overall scheduling problem.

Main Methods

- `addActivity`: Adds an activity to the model.
- `addResource`: Adds a resource to the model.
- `addTemporal`: Adds a temporal constraint to the model.
- `addState`: Adds a state to the model.
- `optimize`: Runs the optimization process.
- `write`: Saves the optimization results as a Gantt chart.

Example

```
from optseq import Model

model = Model(name='Sample Model')
activity = model.addActivity(name='A', duedate=100)
resource = model.addResource(name='Res', capacity=10)
model.addTemporal(pred=activity, succ='sink', delay=5)
model.optimize()
model.write('output.txt')
```

Activity Class

Defines an activity.

Attributes

- `name`: Name of the activity.
- `duedate`: Due date.
- `weight`: Penalty for tardiness.
- `modes`: List of execution modes for the activity.

Example

```
activity = model.addActivity(name='Task1', duedate=50, weight=10)
```

Mode Class

Defines an execution mode for an activity.

Main Methods

- `addResource`: Specifies required resources.
- `addBreak`: Configures interruptible execution.
- `addParallel`: Configures parallel execution.

Example

```
mode = Mode(name='Mode1', duration=5)
mode.addResource(resource, requirement=2)
activity.addModes(mode)
```

Resource Class

Defines a resource.

Attributes

- `name`: Name of the resource.
- `capacity`: Resource capacity.
- `rhs`: Right-hand side constant for nonrenewable resource constraints.

Example

```
resource = model.addResource(name='Machine', capacity=10)
resource.addCapacity(start=0, finish=10, amount=5)
```

Temporal Class

Defines temporal constraints.

Attributes

- `pred`: Predecessor activity.
- `succ`: Successor activity.
- `delay`: Time lag.

Example

```
temporal = model.addTemporal(pred=activity, succ='sink', delay=10)
```

Parameters Class

Defines parameters to control the optimization process.

Key Parameters

- `TimeLimit`: Maximum computation time (seconds).
- `RandomSeed`: Random seed for reproducibility.
- `OutputFlag`: Level of logging verbosity.

Example

```
model.Params.TimeLimit = 300
model.Params.OutputFlag = 1
```

Example: Simple Job Scheduling

Below is an example of a simple job scheduling problem.

```
from optseq import Model, Mode

# Create the model
model = Model(name='Job Scheduling')

# Add activities and modes
job1 = model.addActivity(name='Job1', duedate=10)
mode1 = Mode(name='Mode1', duration=5)
job1.addModes(mode1)

# Add resources
machine = model.addResource(name='Machine', capacity=3)
mode1.addResource(machine, requirement=1)

# Run optimization
model.optimize()
model.write('schedule.txt')
```

Debugging Tips and Tricks

1. Use `print` to inspect details of the model and classes.
2. Check output files to verify schedules and resource usage.
3. Start with small test cases before scaling up.

For more detailed information and advanced usage examples, refer to the official documentation.