Or-tools学习

Abracadabra 黄含驰的机器学习与优化打怪路 今天

【摘要】 Ortools简介lortools是google的开源优化算法包，支持线性规划、整数规划，可以方便的求解Routing、Bin packing、Network flows、Assignment、Scheduling等问题。
l 官网地址为：https://developers.google.com/optimization l 开源代码地址为https://github.com/google/o...

ORTOOLS简介

- ortools是google的开源优化算法包，支持线性规划、整数规划，可以方便的求解Routing、Bin packing、Network flows、Assignment、Scheduling等问题。
- 官网地址为：https://developers.google.com/optimization
- 开源代码地址为 https://github.com/google/or-tools
- 算法包支持java、c++、c#、python。
- Python安装算法包方式：
  - pip install ortools

pip install ortools-7.5.7466-cp37-cp37m-win_amd64.whl(要先从pypi官网下载到本地，用于无法直接pip安装的备用安装方式)

ORTOOLS基本求解器

1.约束优化求解器（**CONSTRAINT PROGRAMMING**）CP-SAT、ORIGINAL-CP 2.线性规划和混合整数规划求解器（接口）（**LINEAR AND MIXED-INTEGER PROGRAMMING**），包括 CBC、CLP、GLOP、GLPK、GUROBI、CPLEX 和SCIP。

3.图算法（**GRAPH ALGORITHMS**）(最短路径、最小成本、最大流量、线性求和分配)。

4.经典旅行推销员问题和车辆路径问题（**VEHICLE ROUTING**）。

5.排产问题和指派问题。

6.经典装箱和背包算法。

ORTOOLS的使用

1.**定义要使用的求解器**

使用线性规划求解器：

```
FROM ORTOOLS.LINEAR_SOLVER IMPORT PYWRAPLP

SOLVER = PYWRAPLP.SOLVER('LINEAREXAMPLE',PYWRAPLP.SOLVER.GLOP_LINEAR_PROGRAMMING)
```

使用混合整数规划求解器：

```
FROM ORTOOLS.LINEAR_SOLVER IMPORT PYWRAPLP

SOLVER = PYWRAPLP.SOLVER('SOLVEINTEGERPROBLEM',PYWRAPLP.SOLVER.CBC_MIXED_INTEGER_PROGRAMMING)
```

2.**定义要使用的变量** `X = SOLVER.NUMVAR(-SOLVER.INFINITY(), SOLVER.INFINITY(), 'X')` （定义实数类型变量）

`X = SOLVER.INTVAR(0.0, SOLVER.INFINITY(), 'X')` （定义整数类型变量）

3.**定义约束**

```
# X + 7 * Y <= 17.5
```

CONSTRAINT1 = SOLVER.CONSTRAINT(-SOLVER.INFINITY(), 17.5) CONSTRAINT1.SETCOEFFICIENT(X, 1)

CONSTRAINT1.SETCOEFFICIENT(Y, 7)

4.定义目标函数

MAXIMIZE X + 10 * Y.

OBJECTIVE = SOLVER.OBJECTIVE()OBJECTIVE.SETCOEFFICIENT(X, 1) OBJECTIVE.SETCOEFFICIENT(Y, 10) OBJECTIVE.SETMAXIMIZATION()

5.求解

SOLVER.SOLVE()

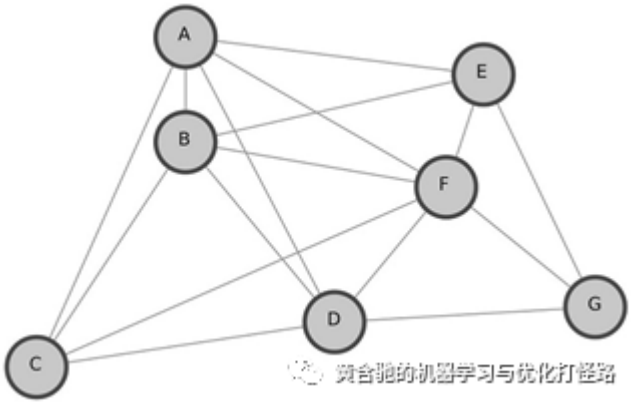
6.获得求解器的得到的变量值和目标函数解

PRINT('X = ', X.SOLUTION_VALUE())

PRINT('Y = ', Y.SOLUTION_VALUE())

PRINT('OPTIMAL OBJECTIVE VALUE = %D' % SOLVER.OBJECTIVE().VALUE())

ORTOOLS路径问题



一.定义数据集 •DATA = CREATE_DATA_MODEL()

二.定义节点索引管理类

•负责算法内部对城市地点索引的管理和计算

•manager = pywrapcp.RoutingIndexManager(len(data['locations']),

data['num_vehicles'],

data['depot'])

三.创建路径模型

•routing = pywrapcp.RoutingModel(manager)

四.为routing对象指定获取距离值的回调方法

•transit_callback_index = routing.RegisterTransitCallback(distance_callback)

五.设定算法搜索参数

• search_parameters = pywrapcp.DefaultRoutingSearchParameters()

• search_parameters.first_solution_strategy = (routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC)

六. 调用元启发式策略

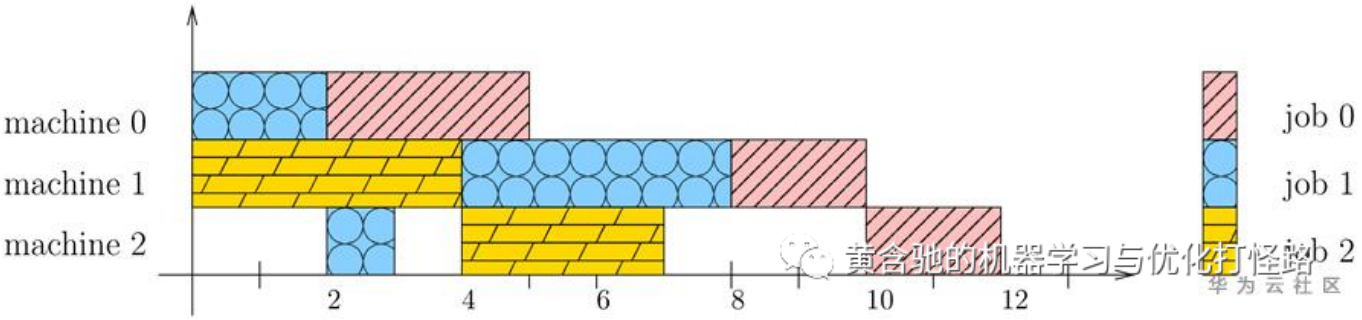
•search_parameters.local_search_metaheuristic = (routing_enums_pb2.LocalSearchMetaheuristic.GUIDED_LOCAL_SEARCH) •search_parameters.time_limit.seconds = 20 •search_parameters.log_search = True •GREEDY_DESCENT、GUIDED_LOCAL_SEARCH、SIMULATED_ANNEALING、TABU_SEARCH •

七.调用计算方法得到结果

•assignment = routing.SolveWithParameters(search_parameters)

ORTOOLS车间作业调度问题

问题介绍:



•job 0 = [(0, 3), (1, 2), (2, 2)]

•job 1 = [(0, 2), (2, 1), (1, 4)]

•job 2 = [(1, 4), (2, 3)]

•task(i, j)表示作业i序列中的第j个任务

•t_i,j表示task(i, j)的开始时间

问题建模：

•优先约束：对于同一作业中的任意两个连续任务，必须在启动第二个任务之前完成第一个任务。

•没有重叠的约束：机器不能同时处理两个任务。

•目标函数：最小化从作业的最早开始时间到最近结束时间的时间长度。

1.定义要使用的模型

使用Constraint programming（约束优化）模型：

```
from ortools.sat.python import cp_model model = cp_model.CpModel()
```

2.定义数据

```
jobs_data = [[(0, 3), (1, 2), (2, 2)],  # Job0

[(0, 2), (2, 1), (1, 4)],  # Job1

[(1, 4), (2, 3)] # Job2 ]
```

3.定义约束

```
# Create and add disjunctive constraints.

for machine in all_machines:

model.AddNoOverlap(machine_to_intervals[machine])

# Precedences inside a job.

for job_id, job in enumerate(jobs_data):

    for task_id in range(len(job) - 1):

        model.Add(all_tasks[job_id, task_id +1].start >= all_tasks[job_id, task_id].end)
```

4.定义目标函数

```
# Makespan objective.
```

```
model.Minimize(obj_var)
```

5.使用求解器求解

```
# Solve model.

solver = cp_model.CpSolver()

status = solver.Solve(model)
```

6.打印结果

```
# Finally print the solution found.

print('Optimal Schedule Length: %i' % solver.ObjectiveValue())

print(output)
```

登录后可下载附件，请登录或者注册

转载自华为云-Abracadabra

https://bbs.huaweicloud.com/blogs/167519