**Instruction to the Python script for automatically generating Orbital Junction Process**

**Xiaochen Zheng**

[xiaochen.zheng@epfl.ch](mailto:xiaochen.zheng@epfl.ch)

This document introduces the general logic of the Python script[[1]](#footnote-1) for automatically generating Orbital Junction Process.

## Dummy dataset and Ontology

The dummy dataset is shown below, and the application ontology has been imported to Neo4j following this instruction[[2]](#footnote-2).

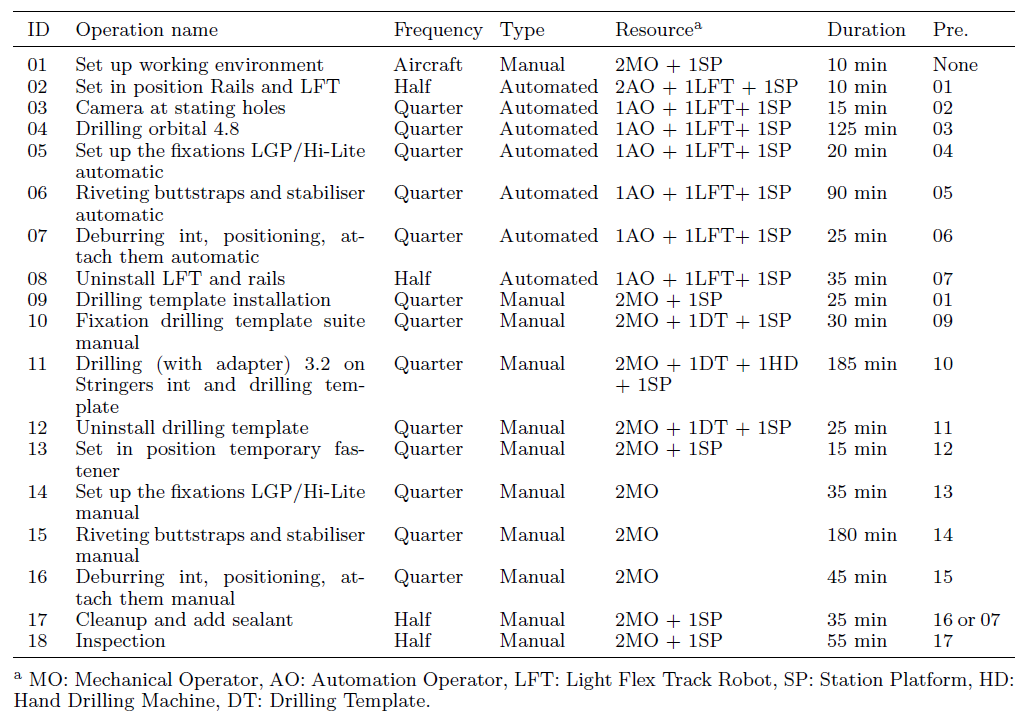


Figure 1 Some typical operations of an aircraft Orbital Junction Process

## Rules for generating Orbital Junction Process solutions

1. Orbital Junction Process consists of two half-orbital subprocesses (1/2), and each subprocess consists of two quarter-orbital subsubprocesses (1/4).
2. The two half-orbital subprocesses (1/2) can be Automatic or Manual.
3. The two half-orbital subprocesses (1/2) can be concurrent or sequential.
4. The two quarter-orbital subsubprocesses (1/4) of the same subprocesses (1/2) can be concurrent or sequential if they are Manual; whereas can only be sequential if they are Automatic due to FlexTrack limitation.

According to these rules, each half-orbital process has 3 options, i.e. Automatic sequential (1/4), Manual sequential (1/4), and Manual concurrent (1/4). And these two half-orbital processes can be concurrent or sequential.

Therefore, the total number of options are: **3 x 3 x 2=18**.

Here we consider upper and lower half-orbital as different processes.

## Explanation to the Python code

Overall workflow of the Python code:

1. Define function to create connection to Neo4j database: class Neo4jConnection
2. Define function to create a main process and add essential operations such as “*Set up working environment*”: Neo4jCreateMainProcess
3. Define functions to create the 3 options for each half-orbital process: Neo4jCreateSubManualProcessSeq, Neo4jCreateSubManualProcessPar, Neo4jCreateSubAutoProcess
4. Define function to add Resource and Time properties to operations: AddResourceTime
5. Define function to connect nodes, e.g. to connect two ¼ subsubprocesses: ConnectNodes
6. Main thread: if \_\_name\_\_ == "\_\_main\_\_":
7. Create connection to Neo4j database

conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="qu4lity")

db = "orbitaljoint"

1. Iterate all possible options for the upper-orbital subprocess with first layer *for* loop:

for p1 in ["SubAuto", "SubManualSeq", "SubManualPar"]:

1. Iterate all possible options for the lower-orbital subprocess with second layer *for* loop:

for p2 in ["SubAuto", "SubManualSeq", "SubManualPar"]:

1. Iterate possible relations between two subprocesses with third *for* loop:

for r in ["sequencial", "parallel"]:

1. Call corresponding functions to create subprocesses and connect them to form a complete process, and then add Resources and Time properties.

Every iteration of the for loop creates an alternative of the orbital junction process.

The complete Python codes are attached below:

#https://towardsdatascience.com/neo4j-cypher-python-7a919a372be7

from neo4j import GraphDatabase

class Neo4jConnection:

def \_\_init\_\_(self, uri, user, pwd):

self.\_\_uri = uri

self.\_\_user = user

self.\_\_pwd = pwd

self.\_\_driver = None

try:

self.\_\_driver = GraphDatabase.driver(

self.\_\_uri, auth=(self.\_\_user, self.\_\_pwd))

except Exception as e:

print("Failed to create the driver:", e)

def close(self):

if self.\_\_driver is not None:

self.\_\_driver.close()

def query(self, query, db=None):

assert self.\_\_driver is not None, "Driver not initialized!"

session = None

response = None

try:

session = self.\_\_driver.session(

database=db) if db is not None else self.\_\_driver.session()

response = list(session.run(query))

except Exception as e:

print("Query failed:", e)

finally:

if session is not None:

session.close()

return response

#Create main process

def Neo4jCreateMainProcess(conn, db, namep):

#namep="N3"

str1 = ("MATCH (p:n4sch\_\_Class{n4sch\_\_label: 'Orbital Joining Process'})-[:hasEssentialOperation]->(e:n4sch\_\_Class) "

"CREATE (op:Process{name: 'R"+namep+ "' + p.n4sch\_\_label}), (op)-[:isIndividualOf]->(p), (oe:Operation{name: '"+namep+ "' + e.n4sch\_\_label}), (oe)-[:isIndividualOf]->(e), (op)-[:hasOperation]->(oe) "

"RETURN oe.name" )

str2 = ("MATCH (p:n4sch\_\_Class{n4sch\_\_label: 'Orbital Joining Process'})-[:hasSubprocess]->(sc:n4sch\_\_Class)-[:hasEssentialOperation]->(e:n4sch\_\_Class) "

"WITH DISTINCT e "

"CREATE (oe1:Operation{name: '"+namep+ "' + e.n4sch\_\_label + '\_1'})-[:isIndividualOf]->(e), (oe2:Operation{name: '"+namep+ "' + e.n4sch\_\_label + '\_2'})-[:isIndividualOf]->(e) "

"WITH \* MATCH (e)-[:hasPredecessor]->(c1)<-[:isIndividualOf]-(pr1), (c1)<-[:isIndividualOf]-(pr2)"

"WHERE pr1.name ENDS WITH '\_1' AND pr2.name ENDS WITH '\_2' AND pr1.name STARTS WITH '" +namep+ "' AND pr2.name STARTS WITH '" +namep+ "' "

"MERGE (oe1)-[:hasPredecessor]->(pr1) MERGE (oe2)-[:hasPredecessor]->(pr2) "

"RETURN pr1.name, pr2.name, oe1.name, oe2.name " )

#conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="qu4lity")

np = conn.query(str1, db) # np is a list [<Record oe.name='N3Set up working environment'>]

np1 = np[0] #np1 is 'neo4j.data.Record' <Record oe.name='N3Set up working environment'>

op0name = np1[0] #name of the first operation "N3Set up working environment"

np2 = conn.query(str2, db) # np is a list

op21name = np2[0][0] #'N3Cleanup and add sealant\_1'

op22name = np2[0][1] #'N3Cleanup and add sealant\_2'

op23name = np2[0][2]

op24name = np2[0][3]

return op0name, op21name, op22name, op23name, op24name

#Create automatic subprocess

def Neo4jCreateSubAutoProcess(conn, db, namep):

#namep ="N3"

str1 = ("MATCH (p:n4sch\_\_Class{n4sch\_\_label: 'Orbital Joining Process'})-[:hasSubprocess]->(sc:n4sch\_\_Class)-[:hasOptionalAutoOperation]->(Oop:n4sch\_\_Class) "

"WITH DISTINCT Oop CREATE (nOop:Operation{name: '"+namep+ "' + Oop.n4sch\_\_label})-[:isIndividualOf]->(Oop) "

"WITH \* MATCH (p:n4sch\_\_Class{n4sch\_\_label: 'Orbital Joining Process'})-[:hasSubprocess]->(sc)-[:hasSubprocess]->(ssc)-[:hasOptionalAutoOperation]->(Oop1) "

"WITH DISTINCT Oop1 CREATE (nOop1:Operation{name: '"+namep+ "' + Oop1.n4sch\_\_label + '\_1'})-[:isIndividualOf]->(Oop1), (nOop2:Operation{name: '"+namep+ "' + Oop1.n4sch\_\_label + '\_2'})-[:isIndividualOf]->(Oop1) "

"WITH \* MATCH (Oop1)-[:hasPredecessor]->(c1)<-[:isIndividualOf]-(pr1), (c1)<-[:isIndividualOf]-(pr2) WHERE pr1.name ENDS WITH '\_1' AND pr2.name ENDS WITH '\_2' AND pr1.name STARTS WITH '" +namep+ "' AND pr2.name STARTS WITH '" +namep+ "' "

"MERGE (nOop1)-[:hasPredecessor]->(pr1) MERGE (nOop2)-[:hasPredecessor]->(pr2) "

"WITH \* MATCH (p1:Operation), (p2:Operation), (p3:Operation), (p4:Operation), (p5:Operation), (p6:Operation) WHERE p1.name CONTAINS '"+namep+ "Camera at stating holes\_1' AND p2.name CONTAINS '"+namep+ "Deburring int, positioning, attach them automatic\_1' AND p3.name CONTAINS '"+namep+ "Camera at stating holes\_2' AND p4.name CONTAINS '"+namep+ "Deburring int, positioning, attach them automatic\_2' AND p5.name CONTAINS '"+namep+ "Set in position Rails and LFT' AND p6.name CONTAINS '"+namep+ "Deinstall LFT and rails' "

"MERGE (p1)-[:hasPredecessor]->(p5) MERGE (p3)-[:hasPredecessor]->(p2) MERGE (p6)-[:hasPredecessor]->(p4) "

"RETURN p5.name, p6.name" )

#conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="qu4lity")

np = conn.query(str1, db)

firstname = np[0][0] #name of the starting operation of the subauto process 'N3Set in position Rails and LFT'

lastname = np[0][1] ##name of the last operation of the subauto process 'N3Deinstall LFT and rails'

return firstname, lastname

#Create manual sequencial subprocess

def Neo4jCreateSubManualProcessSeq(conn, db, namep):

#namep = "N3"

str1 = ("MATCH (p:n4sch\_\_Class{n4sch\_\_label: 'Orbital Joining Process'})-[:hasSubprocess]->(sc)-[:hasSubprocess]->(ssc)-[:hasOptionalManualOperation]->(Oop1) "

"WITH DISTINCT Oop1 CREATE (nOop1:Operation{name: '"+namep+ "' + Oop1.n4sch\_\_label + '\_1'})-[:isIndividualOf]->(Oop1), (nOop2:Operation{name: '"+namep+ "' + Oop1.n4sch\_\_label + '\_2'})-[:isIndividualOf]->(Oop1) "

"WITH \* MATCH (Oop1)-[:hasPredecessor]->(c1)<-[:isIndividualOf]-(pr1), (c1)<-[:isIndividualOf]-(pr2) WHERE pr1.name ENDS WITH '\_1' AND pr2.name ENDS WITH '\_2' AND pr1.name STARTS WITH '" +namep+ "' AND pr2.name STARTS WITH '" +namep+ "' "

"MERGE (nOop1)-[:hasPredecessor]->(pr1) MERGE (nOop2)-[:hasPredecessor]->(pr2) "

"WITH \* MATCH (p1:Operation), (p2:Operation),(p0:Operation), (p3:Operation) WHERE p1.name CONTAINS '"+namep+ "Deburring int, positioning, attach them manual\_1' AND p2.name CONTAINS '"+namep+"Drilling template install\_2' AND p0.name CONTAINS '"+namep+ "Drilling template install\_1' AND p3.name CONTAINS '"+namep+"Deburring int, positioning, attach them manual\_2'"

"MERGE (p2)-[:hasPredecessor]->(p1) "

"RETURN p0.name, p3.name")

#conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="qu4lity")

np = conn.query(str1, db)

op10name = np[0][0] #name of the starting operation of the SubManualSeq process 'N3Drilling template install\_1'

op11name = np[0][1]

return op10name, op11name

#Create manual parallel subprocess

def Neo4jCreateSubManualProcessPar(conn, db, namep):

#namep = "N3"

str1 = ("MATCH (p:n4sch\_\_Class{n4sch\_\_label: 'Orbital Joining Process'})-[:hasSubprocess]->(sc)-[:hasSubprocess]->(ssc)-[:hasOptionalManualOperation]->(Oop1) "

"WITH DISTINCT Oop1 CREATE (nOop1:Operation{name: '"+namep+ "' + Oop1.n4sch\_\_label + '\_1'})-[:isIndividualOf]->(Oop1), (nOop2:Operation{name: '"+namep+ "' + Oop1.n4sch\_\_label + '\_2'})-[:isIndividualOf]->(Oop1)"

"WITH \* MATCH (Oop1)-[:hasPredecessor]->(c1)<-[:isIndividualOf]-(pr1), (c1)<-[:isIndividualOf]-(pr2) WHERE pr1.name ENDS WITH '\_1' AND pr2.name ENDS WITH '\_2' AND pr1.name STARTS WITH '" +namep+ "' AND pr2.name STARTS WITH '" +namep+ "' "

"MERGE (nOop1)-[:hasPredecessor]->(pr1) MERGE (nOop2)-[:hasPredecessor]->(pr2) "

"WITH \* MATCH (p1:Operation), (p2:Operation), (p3:Operation),(p4:Operation) WHERE p1.name CONTAINS '"+namep+ "Drilling template install\_1' AND p2.name CONTAINS '"+namep+ "Drilling template install\_2' AND p3.name CONTAINS '"+namep+ "Deburring int, positioning, attach them manual\_1' AND p4.name CONTAINS '"+namep+ "Deburring int, positioning, attach them manual\_2' "

"RETURN p1.name, p2.name, p3.name, p4.name")

#conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="qu4lity")

np = conn.query(str1, db) # np is a list [<Record oe.name='N3Set up working environment'>]

op01name = np[0][0] #'N3Drilling template install\_1'

op02name = np[0][1] #'N3Drilling template install\_2'

op10name = np[0][2] #'N3Deburring int, positioning, attach them manual\_1'

op11name = np[0][3] #'N3Deburring int, positioning, attach them manual\_2'

#print(op0name)

return op01name, op02name, op10name,op11name

#connect two nodes

def ConnectNodes(conn, db, name1, name2):

str = ("MATCH (p1:Operation), (p2:Operation) WHERE p1.name CONTAINS '" + name1 + "' AND p2.name CONTAINS '" + name2 + "' MERGE (p2)-[:hasPredecessor]->(p1)" )

#conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="qu4lity")

conn.query(str, db)

def AddResourceTime(conn, db, pid):

str1 = ("MATCH (op:Operation)-[:isIndividualOf]->(cl:n4sch\_\_Class)-[:requiresResource]->(res:n4sch\_\_Class) WHERE op.name STARTS WITH '" + pid + "' "

"WITH DISTINCT res CREATE (rob:Resource{name: '" + pid + "' + res.n4sch\_\_label, n4sch\_\_label: '" + pid + "' + res.n4sch\_\_label})-[:isIndividualOf]->(res) "

"WITH \* MATCH (op:Operation)-[:isIndividualOf]->(cl:n4sch\_\_Class)-[r]->(res:n4sch\_\_Class)<-[:isIndividualOf]-(rob) WHERE op.name STARTS WITH '" + pid + "' "

"CREATE (op)-[:requiresResource{number: r.number}]->(rob)")

str2 = ("MATCH (op:Operation)-[:isIndividualOf]->(cl:n4sch\_\_Class) WHERE op.name STARTS WITH '" + pid + "' "

"SET op.duration = cl.duration, op.op\_type=cl.op\_type")

str3 = ("MATCH (op) WHERE op.name STARTS WITH '" + pid + "' "

"SET op.n4sch\_\_label = op.name")

#conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="qu4lity")

conn.query(str1, db)

conn.query(str2, db)

conn.query(str3, db)

if \_\_name\_\_ == "\_\_main\_\_":

conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="qu4lity")

db = "orbitaljoint"

#Remote server

#conn = Neo4jConnection(uri="bolt://52.157.75.156:7687", user="neo4j", pwd="qu4lity")

#db = "neo4j"

i = 1

for p1 in ["SubAuto", "SubManualSeq", "SubManualPar"]:

for p2 in ["SubAuto", "SubManualSeq", "SubManualPar"]:

for r in ["sequencial", "parallel"]:

pid = f"N{i:02}"

op0, clean1, clean2, inspec1, inspec2 = Neo4jCreateMainProcess(conn, db, pid) #3 essential operations

##subProcess1

if p1 == "SubAuto":

subauto0,subauto1 = Neo4jCreateSubAutoProcess(conn, db, pid+"U\_")

ConnectNodes(conn, db, op0, subauto0)

ConnectNodes(conn, db, subauto1, clean1)

elif p1 == "SubManualSeq":

SubManualSeq0, SubManualSeq1 = Neo4jCreateSubManualProcessSeq(conn, db, pid+"U\_")

ConnectNodes(conn, db, op0, SubManualSeq0)

ConnectNodes(conn, db, SubManualSeq1, clean1)

elif p1 == "SubManualPar":

SubManualPar01, SubManualPar02, SubManualPar11, SubManualPar12 = Neo4jCreateSubManualProcessPar(conn, db,pid+"U\_")

ConnectNodes(conn, db, op0, SubManualPar01)

ConnectNodes(conn, db, op0, SubManualPar02)

ConnectNodes(conn, db, SubManualPar11, clean1)

ConnectNodes(conn, db, SubManualPar12, clean1)

else:

print("error!")

##subProcess1

if p2 == "SubAuto":

subauto2,subauto3 = Neo4jCreateSubAutoProcess(conn, db, pid+"L\_")

ConnectNodes(conn, db, subauto3, clean2)

if r == "sequencial":

ConnectNodes(conn, db, inspec1, subauto2)

AddResourceTime(conn, db, pid)

print("---Process No. {} Created!!".format(i))

i = i+1

elif r == "parallel":

ConnectNodes(conn, db, op0, subauto2)

AddResourceTime(conn, db, pid)

print("---Process No. {} Created!!".format(i))

i = i+1

elif p2 == "SubManualSeq":

SubManualSeq2, SubManualSeq3 = Neo4jCreateSubManualProcessSeq(conn, db, pid+"L\_")

ConnectNodes(conn, db, SubManualSeq3, clean2)

if r == "sequencial":

ConnectNodes(conn, db, inspec1, SubManualSeq2)

AddResourceTime(conn, db, pid)

print("---Process No. {} Created!!".format(i))

i = i+1

elif r == "parallel":

ConnectNodes(conn, db, op0, SubManualSeq2)

AddResourceTime(conn, db, pid)

print("---Process No. {} Created!!".format(i))

i = i+1

elif p2 == "SubManualPar":

SubManualPar11, SubManualPar12, SubManualPar21, SubManualPar22 = Neo4jCreateSubManualProcessPar(conn, db, pid+"L\_")

ConnectNodes(conn, db, SubManualPar21, clean2)

ConnectNodes(conn, db, SubManualPar22, clean2)

if r == "sequencial":

ConnectNodes(conn, db, inspec1,SubManualPar11)

ConnectNodes(conn, db, inspec1,SubManualPar12)

AddResourceTime(conn, db, pid)

print("---Process No. {} Created!!".format(i))

i = i+1

elif r == "parallel":

ConnectNodes(conn, db, op0,SubManualPar11)

ConnectNodes(conn, db, op0,SubManualPar12)

AddResourceTime(conn, db, pid)

print("---Process No. {} Created!!".format(i))

i = i+1

else:

print("error!")

1. <https://github.com/zhengxiaochen/ontology_aircraft_system/blob/main/auto_generate_process.py> [↑](#footnote-ref-1)
2. <https://github.com/zhengxiaochen/ontology_aircraft_system/blob/main/Instroduction%20to%20ontology%20import%20and%20edit%20with%20Neo4j.docx> [↑](#footnote-ref-2)