Jangeun Kim 20236471

Term Project Progress Presentation

[Resource-Constrained Project Scheduling Problem, RCPSP]

Contents

- 1. What is the RCPSP?
- 2. Problem Definition
- 3. Future plans

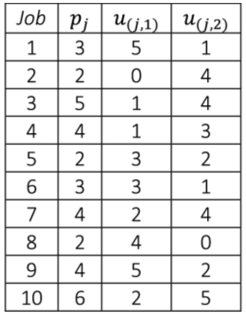
1. What is the RCPSP?

1. What is the RCPSP?

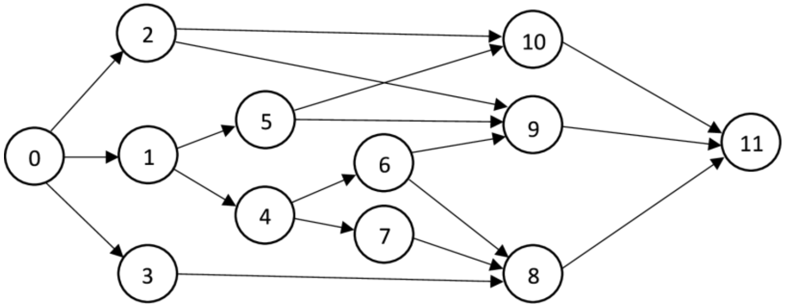
- Definition

✓ **Resource-Constrained Project Scheduling Problem (RCPSP)** is a combinatorial optimization problem in the field of operations research and project management.

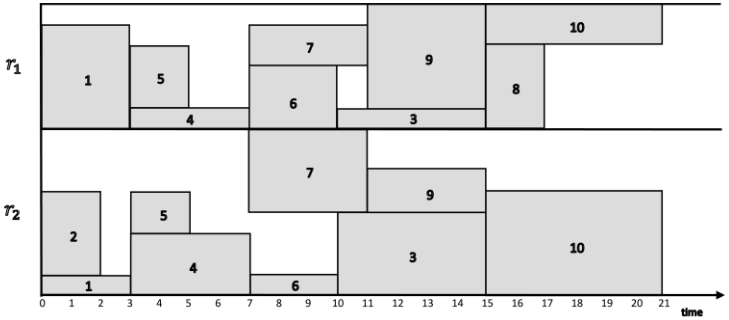
✓ It involves scheduling a set of activities or tasks with given durations and resource requirements in such a way that the project is completed as quickly as possible while respecting resource constraints.



Job	p_j	$u_{(j,1)}$	$u_{(j,2)}$
1	3	5	1
2	2	0	4
3	5	1	4
4	4	1	3
5	2	3	2
6	3	3	1
7	4	2	4
8	2	4	0
9	4	5	2
10	6	2	5



AoN network for project instance



optimal schedule for instance

1. What is the RCPSP?

- Key considerations (Constraints)

✓ Activities

These are tasks or jobs that need to be scheduled.

☞ Each activity has a defined duration, a set of required resources, and precedence relationships with other activities.

✓ Resources

There are limited resources available for executing the activities.

☞ Resources can include labor, machinery, materials, or any other constraints that can affect the scheduling

✓ Precedence Relationships

These dependencies are represented as precedence relationships.

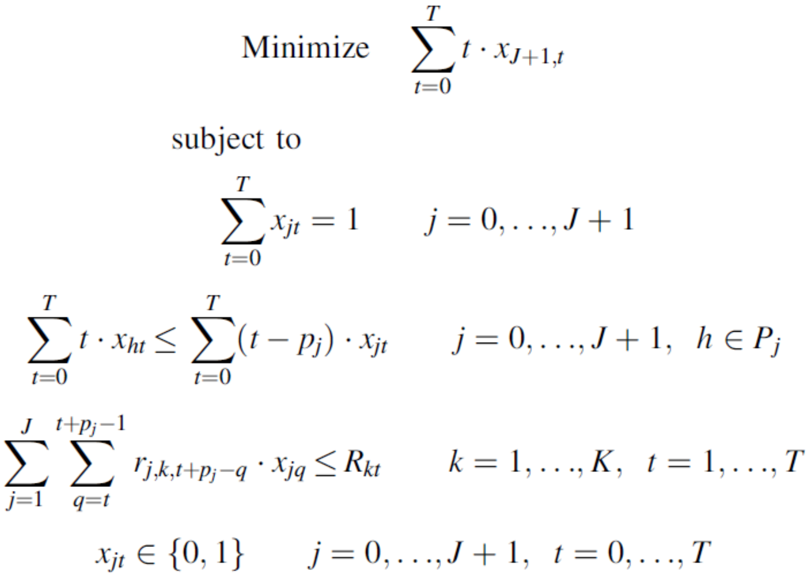
☞ Activities may have dependencies on each other, meaning that certain activities must be completed before others can start.

1. What is the RCPSP?

- Main objective

  ✓ The objective in RCPSP is to find a schedule that **minimizes the project's makespan** while satisfying the resource constraints and respecting the precedence relationships between activities

  ✓ RCPSP is known to be an **NP-hard problem**, which means that finding an optimal solution can be computationally challenging, especially for large and complex projects.



$$\text{Minimize} \quad \sum_{t=0}^{T} t \cdot x_{J+1,t}$$

subject to

$$\sum_{t=0}^{T} x_{jt} = 1 \qquad j = 0, \ldots, J+1$$

$$\sum_{t=0}^{T} t \cdot x_{ht} \leq \sum_{t=0}^{T} (t - p_j) \cdot x_{jt} \qquad j = 0, \ldots, J+1, \quad h \in P_j$$

$$\sum_{j=1}^{J} \sum_{q=t}^{t+p_j-1} r_{j,k,t+p_j-q} \cdot x_{jq} \leq R_{kt} \qquad k = 1, \ldots, K, \quad t = 1, \ldots, T$$

$$x_{jt} \in \{0, 1\} \qquad j = 0, \ldots, J+1, \quad t = 0, \ldots, T$$

1. What is the RCPSP?

- What is the difference between RCPSP and JSSP?

Job Shop Scheduling Problem (JSSP)

✓ Scope

- RCPSP is primarily concerned with **scheduling activities in a project environment**.
  The activities are tasks that need to be executed to complete a project.
- JSSP is focused on **scheduling jobs in a manufacturing environment**, particularly in job shops where different jobs require different sequences of operations on various machines.

✓ Constraints

- RCPSP : Activities, Resources, Precedence
- JSSP : Jobs and Operations , Machines, Precedence

✓ Objective

- RCPSP/JSSP :
  minimize the makespan

✓ Applications

- RCPSP is commonly used in **project management scenarios**, such as **construction projects**, **software development projects**, and manufacturing projects where tasks are interdependent.
- JSSP is commonly used in **manufacturing settings**, such as **job shops**, where different types of jobs with varying processing requirements need to be scheduled on **available machines.**

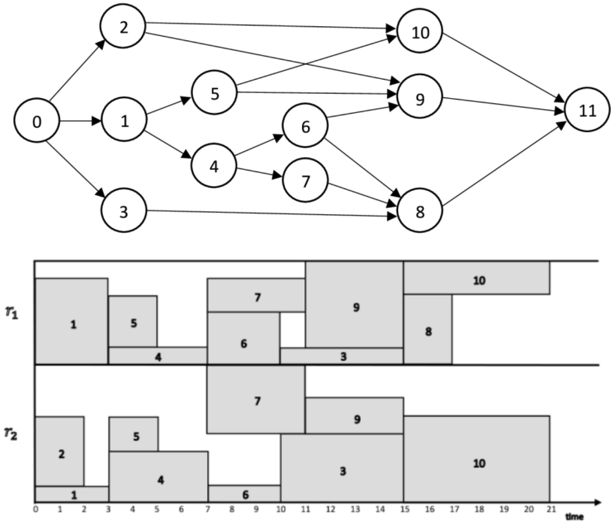
1. What is the RCPSP?

- What is the difference between RCPSP and JSSP?

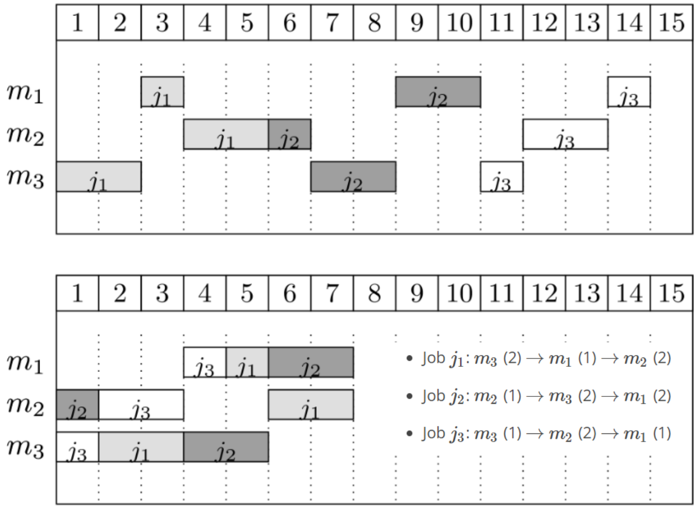
✓ In summary,

- **RCPSP** is more aligned with **project-based scheduling**, where tasks have precedence relationships, and the goal is to optimize the overall project completion time.

- **JSSP** is focused on **manufacturing environments**, where jobs involve multiple operations on different machines, and the objective is often to minimize makespan.



RCPSP


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
m_1
m_2
m_3
j_1
j_2
j_3
• Job j_1: m_3 (2) → m_1 (1) → m_2 (2)
• Job j_2: m_2 (1) → m_3 (2) → m_1 (2)
• Job j_3: m_3 (1) → m_2 (2) → m_1 (1)

JSSP

2. Problem Definition

2. Problem Definition

- Problem Description

```
[ 0,  7, 10,  7,  7,  2,  8,  6,  1,  7, 10,  4,  5,  4,  6,  5,  1,
  5,  7,  6,  6, 10,  9,  2,  4,  5,  2, 10,  3,  6,  9,  1,  5,  2,
  8,  5,  6,  6,  1, 10,  3,  2,  7,  2,  1,  1,  3,  6,  1,  6,  4,
  9, 10,  5,  3,  3,  4,  3, 10,  5,  9,  3,  3,  9,  2,  8,  2,  7,
  1,  4,  8,  9,  5,  2,  5,  3,  4,  2,  9,  8,  7, 10,  4,  2,  2,
  7,  6,  7,  9,  9,  5,  0]
```

[Duration for each activity]

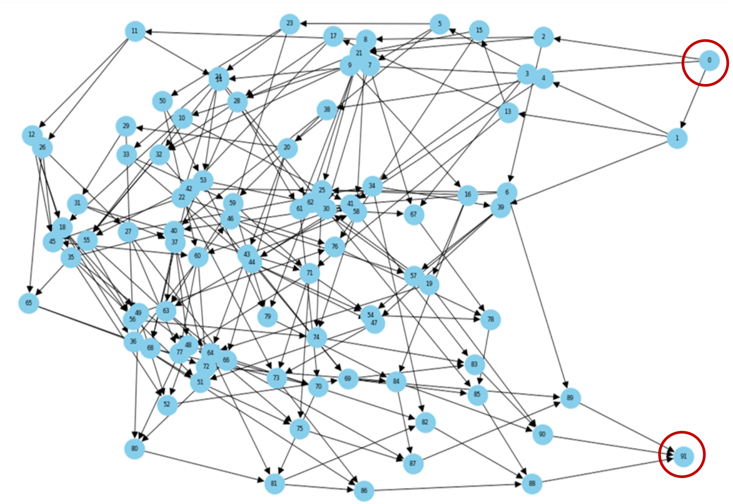
✓ **Project / Activities / Sum of duration** : 1 / 92 (including dummy) / 477 days

[Resource requirement for each activity]

✓ **Resources type / total amount** : 4 / [18, 21, 20, 18]

```
[[ 0,  0,  0,  0],
 [ 0,  0,  5,  6],
 [ 1,  1,  8,  0],
 [10,  9,  2,  8],
 [10,  6,  1,  3],
 [ 4,  9,  1,  1],
 [ 8,  7,  0,  0],
 [ 9,  7,  7,  4],
 [ 1,  3,  0,  2],
 [ 3,  3,  6,  0],
 [ 6,  4,  5,  0],
 [ 8,  5,  1,  0],
 [ 2,  4,  0,  6],
 [ 0,  6,  4,  0],
 [ 0,  3, 10,  9],
 [ 1,  0,  1,  5],
 [ 5,  5,  3,  2],
 [10,  0,  0,  3],
 [ 4,  0,  6,  7],
 [ 4,  3,  6,  0],
 [ 1,  5,  0,  1],
 [ 7,  2,  1,  1],
 [ 0,  0,  4,  8],
 [ 4,  3,  3,  5],
 [ 0, 10,  0,  2],
 [ 5,  5,  5,  0],
 [10,  9,  0,  0],
 [ 9,  0,  2,  7],
 [ 9,  0,  9,  1],
 [ 4, 10,  0,  0],
 [ 0,  7,  2,  2],
 [ 0,  0,  2,  1],
 [ 8,  2,  0,  8],
 [ 8, 10,  5,  2],
 [ 9,  6,  9,  0],
 [ 0,  0,  6,  6],
 [ 6,  2,  0,  1],
 [ 0,  6,  2, 10],
 [10,  4, 10,  0],
 [ 0,  7,  1,  3],
 [ 7,  0,  2,  2],
 [ 0,  7, 10,  3],
 [ 4,  0,  4,  4],
 [ 0,  8,  9,  7],
 [ 0, 10,  0,  0],
 [ 6,  7,  3,  1],
 [ 4,  8,  1,  0],
 [ 8,  0,  0,  8],
 [ 2,  3,  0,  5],
 [10,  6, 10,  2],
 [ 0,  0,  7,  9],
 [ 9,  8,  8,  9],
 [ 1,  3,  1,  8],
 [ 8,  0, 10, 10],
 [ 8,  0,  7,  8],
 [ 4,  5,  8,  1],
 [ 8,  4,  6,  7],
 [ 0,  7,  6,  0],
 [ 9,  7,  8, 10],
 [ 3,  8,  0,  0],
 [ 2, 10,  4,  2],
 [ 6,  8,  5,  0],
 [ 7,  0,  1,  4],
 [ 3,  3,  8,  0],
 [ 0,  8,  3,  0],
 [ 5, 10,  9,  0],
 [ 5,  1,  4,  3],
 [ 2,  8,  0,  4],
 [ 4,  6,  4,  3],
 [ 0,  6,  1,  2],
 [ 0,  3,  6,  7],
 [ 0,  7,  8,  4],
 [ 0,  3, 10,  0],
 [ 0,  7,  0,  9],
 [ 0,  0,  4,  7],
 [10,  5,  8,  8],
 [ 0,  6,  4,  1],
 [ 0,  9,  8,  7],
 [10,  0,  9,  4],
 [ 0,  3,  4,  6],
 [10,  6,  7,  0],
 [ 2,  5,  5,  8],
 [ 0,  0,  7, 10],
 [ 2,  6,  0,  1],
 [ 2,  3,  0,  4],
 [ 9,  9,  6,  0],
 [ 1,  1,  8,  4],
 [ 0, 10,  5,  5],
 [10,  9,  8,  6],
 [ 4,  2,  0,  5],
 [ 0,  2,  1,  4],
 [ 0,  0,  0,  0]]
```

✓ **Precedence Relationships**



```
[[],
 [0],
 [0],
 [0],
 [1],
 [4],
 [2],
 [5],
 [2],
 [3],
 [9],
 [8],
 [11],
 [1],
 [7, 11],
 [13],
 [9],
 [13],
 [10, 12],
 [16],
 [5],
 [2, 15],
 [14],
 [5],
 [88, 89, 90]]

[23],
[6, 16, 21],
[11],
[26],
[17, 21],
[20],
[7, 19, 24],
[29],
[9, 14, 28],
[24],
[3, 27],
[12, 32],
[18],
[28, 31],
[4],
[1, 34],
[30, 35, 38],
[4, 10],
[17, 25],
[20],
[32, 33, 34],
[12, 22, 26],
[23, 36],
[39, 44],
[37, 45],

[18],
[23],
[35, 36, 47],
[35, 48, 49],
[24, 50],
[39, 44, 46],
[10, 32, 53],
[18, 29, 55],
[39, 46],
[8, 43, 53],
[33, 38],
[41, 45, 50],
[6, 8, 28],
[14, 21, 57],
[40, 41, 59],
[19, 43, 55],
[26, 55],
[20, 27, 63],
[3, 7, 61],
[37, 42, 65],
[43, 44, 52],
[61, 66, 68],
[13, 31, 59],
[37, 60, 65],
[22, 47, 71],

[25, 56, 58],
[51, 64, 71],
[15, 22, 49],
[27, 60, 71],
[54, 67, 76],
[53, 59, 62],
[49, 66, 77],
[64, 70, 80],
[34, 48, 81],
[41, 69, 74],
[16, 72, 79],
[30, 69, 78],
[36, 73, 81],
[66, 75, 84],
[82, 85, 86],
[6, 69, 87],
[19, 83, 84],
```

[Precedence relationships for each activity]

2. Problem Definition

- How to solve this problem?

  ✓ Use GA as one of the metaheuristic algorithm model

  ✓ Design the GA using the Python

  - **Initialization population :** making feasible activity string considering constraints

    ☞ chromosome or genotype : Integer string

  - **Fitness Evaluation :** Minimize the makespan (=objective function)

  - **Selection :** Tournament Selection

  - **Crossover :** One-Point Crossover

    ☞ The approach may be subject to change depending on the situation

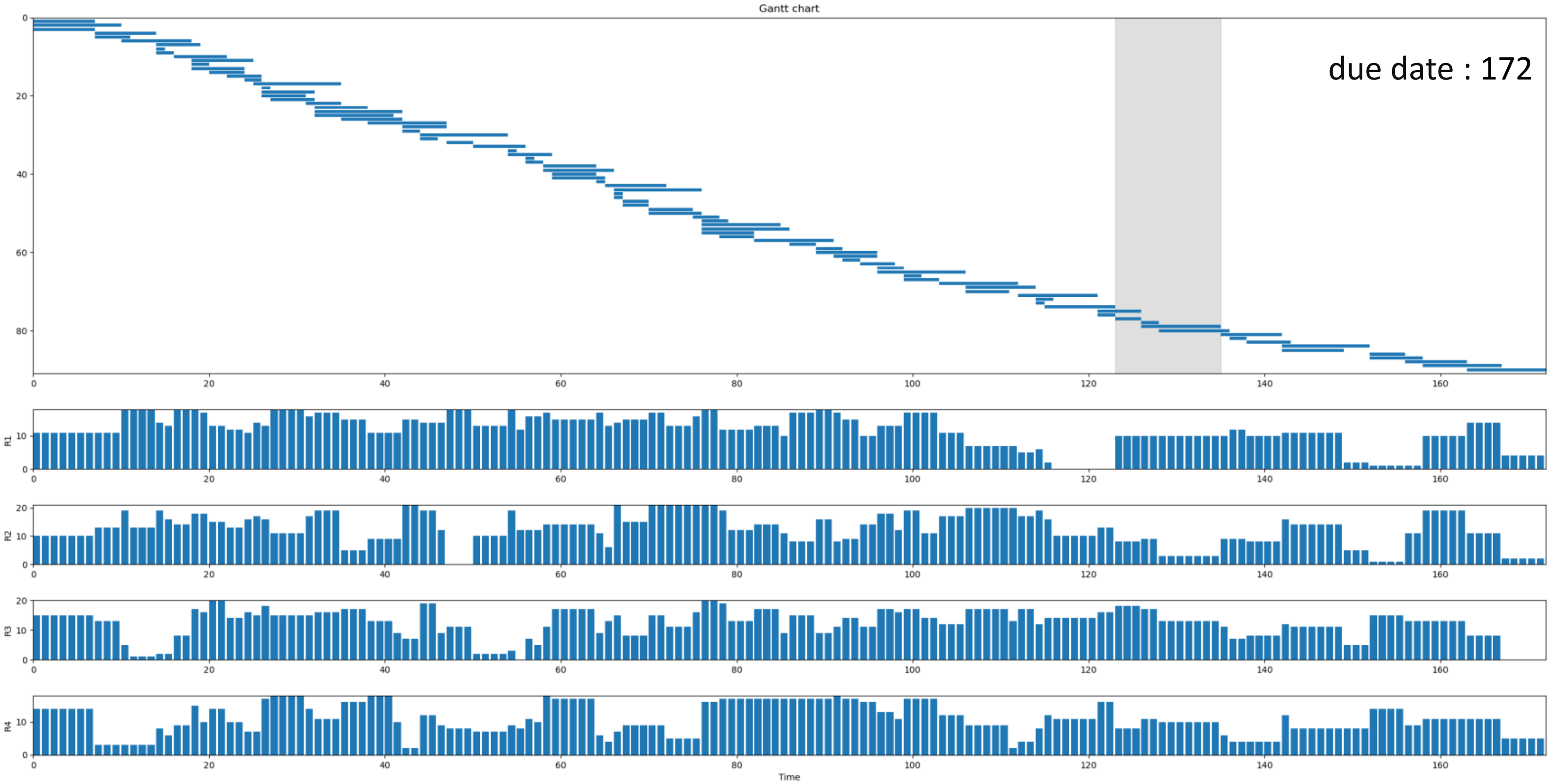
  - **Mutation :** Swap two adjacent activities (considering RCPSP characteristic)

    ☞ The approach may be subject to change depending on the situation

  - **Termination Criteria :** a number of generations(<=1000)

2. Problem Definition

- How to solve this problem?
  - ✓ Example of feasible activity string



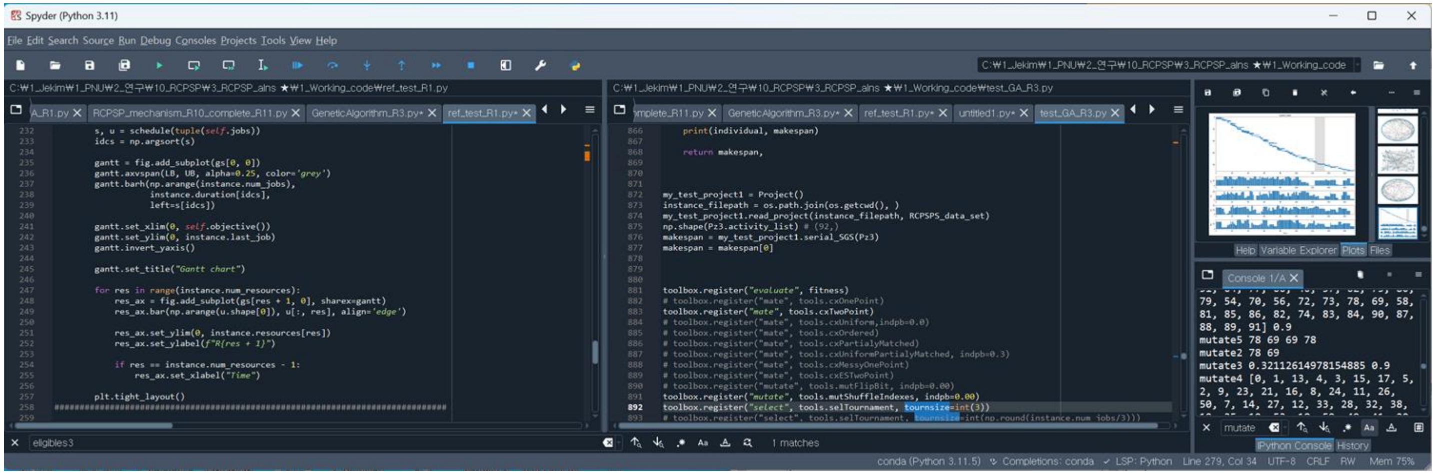
due date : 172

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,   68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91]

3. Future plans

3. Future plans

- Designing and Coding the GA for RCPSP using the python


Spyder (Python 3.11)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:₩1_Jekim₩1_PNU₩2_연구₩10_RCPSP₩3_RCPSP_alns ★₩1_Working_code
C:₩1_Jekim₩1_PNU₩2_연구₩10_RCPSP₩3_RCPSP_alns ★₩1_Working_code₩ref_test_R1.py
A_R1.py
RCPSP_mechanism_R10_complete_R11.py
GeneticAlgorithm_R3.py*
ref_test_R1.py*
C:₩1_Jekim₩1_PNU₩2_연구₩10_RCPSP₩3_RCPSP_alns ★₩1_Working_code₩test_GA_R3.py
mplete_R11.py
GeneticAlgorithm_R3.py*
ref_test_R1.py*
untitled1.py*
test_GA_R3.py
Help
Variable Explorer
Plots
Files
Console 1/A
IPython Console
History
eligibles3
1 matches
mutate
conda (Python 3.11.5)
Completions: conda
LSP: Python
Line 279, Col 34
UTF-8
CRLF
RW
Mem 75%

```python
        s, u = schedule(tuple(self.jobs))
        idcs = np.argsort(s)

        gantt = fig.add_subplot(gs[0, 0])
        gantt.axvspan(LB, UB, alpha=0.25, color='grey')
        gantt.barh(np.arange(instance.num_jobs),
                   instance.duration[idcs],
                   left=s[idcs])

        gantt.set_xlim(0, self.objective())
        gantt.set_ylim(0, instance.last_job)
        gantt.invert_yaxis()

        gantt.set_title("Gantt chart")

        for res in range(instance.num_resources):
            res_ax = fig.add_subplot(gs[res + 1, 0], sharex=gantt)
            res_ax.bar(np.arange(u.shape[0]), u[:, res], align='edge')

            res_ax.set_ylim(0, instance.resources[res])
            res_ax.set_ylabel(f"R{res + 1}")

            if res == instance.num_resources - 1:
                res_ax.set_xlabel("Time")

        plt.tight_layout()
    ##############################################################################
```

```python
    print(individual, makespan)

    return makespan,



my_test_project1 = Project()
instance_filepath = os.path.join(os.getcwd(), )
my_test_project1.read_project(instance_filepath, RCPSPS_data_set)
np.shape(Pz3.activity_list) # (92,)
makespan = my_test_project1.serial_SGS(Pz3)
makespan = makespan[0]



toolbox.register("evaluate", fitness)
# toolbox.register("mate", tools.cxOnePoint)
toolbox.register("mate", tools.cxTwoPoint)
# toolbox.register("mate", tools.cxUniform,indpb=0.0)
# toolbox.register("mate", tools.cxOrdered)
# toolbox.register("mate", tools.cxPartialyMatched)
# toolbox.register("mate", tools.cxUniformPartialyMatched, indpb=0.3)
# toolbox.register("mate", tools.cxMessyOnePoint)
# toolbox.register("mate", tools.cxESTwoPoint)
# toolbox.register("mutate", tools.mutFlipBit, indpb=0.00)
toolbox.register("mutate", tools.mutShuffleIndexes, indpb=0.00)
toolbox.register("select", tools.selTournament, tournsize=int(3))
# toolbox.register("select", tools.selTournament, tournsize=int(np.round(instance.num_jobs/3)))
```

```
79, 54, 70, 56, 72, 73, 78, 69, 58,
81, 85, 86, 82, 74, 83, 84, 90, 87,
88, 89, 91] 0.9
mutate5 78 69 69 78
mutate2 78 69
mutate3 0.32112614978154885 0.9
mutate4 [0, 1, 13, 4, 3, 15, 17, 5,
2, 9, 23, 21, 16, 8, 24, 11, 26,
50, 7, 14, 27, 12, 33, 28, 32, 38,
```

- Coding the MILP for RCPSP using the python
  - ✓ For the comparison between a feasible solution and an optimal solution

Thanks