Optimasi penjadwalan urutan kerja pada ship plane block flow-line

Alpha Roissul
1806202531
UAS Optimasi Produksi Kapal

CONTENTS OF PRESENTASION

- Latar Belakang
- Tujuan & Asumsi Penelitian
- Metode Algoritma

Latar Belakang

Introduction

Pada industri perkapalan, proses perencanaan dan penjadwalan pekerjaan merupakan komponen yang penting dalam proses produksi dan konstruksi karena akan mempengaruhi tingkat efisiensi produksi dan persaingan. Dengan pengaplikasian model pembangunan kapal yang modern, pembangunan tipe plane block flowline sudah diperkenalkan ke galangan - galangan untuk meningkatkan efisiensi dalam konstruksi section kapal. Akan tetapi karena fakta kalau galangan-galangan tersebut masih menggunakan metode penjadwalan tradisional yang dilakukan langsung di lokasi, hasil penjadwalan tersebut kurang optimal sehingga menyebabkan efiisensi produksi yang rendah untuk proses produksi sistem plane block flowline.

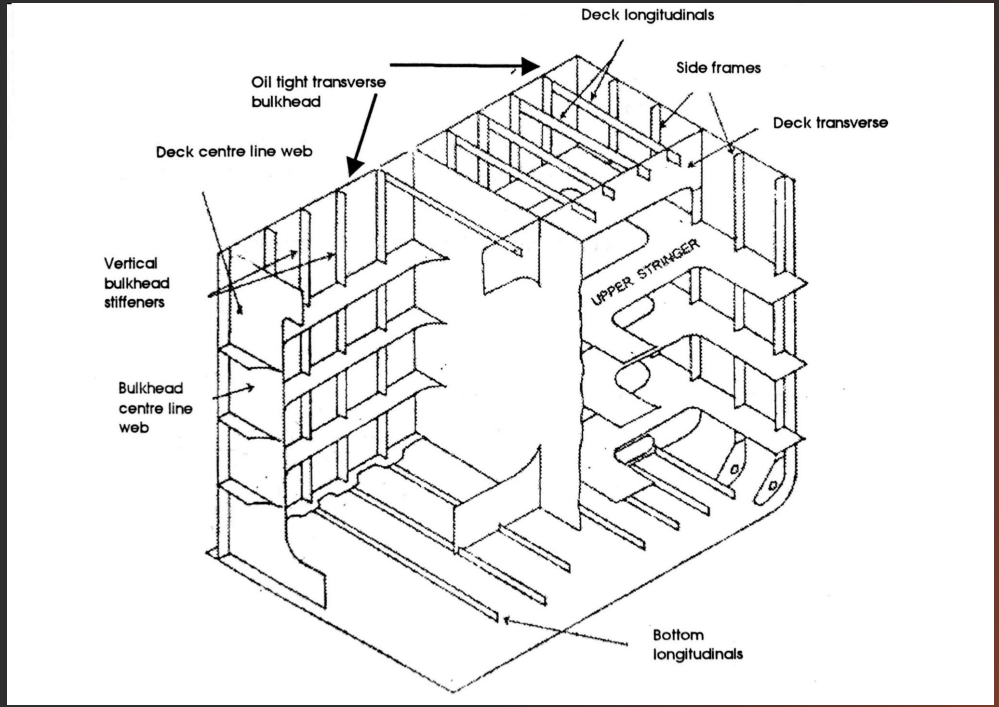
Selain itu karena industri galangan kapal adalah industri manufaktur yang intens maka ada beberapa masalah dalam proses manufaktur kapal seperti, material sisa yang terbuang karena pemilihan dan pemakaian yang kurang tepat, polusi udara, tanah , dan juga air dan dengan meningkatnya perhatian terkait pemanasan global dan target zero net emission, industri galangan kapal perlu mulai menerapkan proses manufaktur yang bersifat hijau (Green Manufacturing)

Plane Block

Plane Block adalah proses pembuatan blok dimana proses tersebut akan merakit dan mengelas bagian dari struktur dari kapal terlebih dahulu.

Dan berdasarkan metode konstruksinya ada dua tipe kategori :

1. Frame Method
2. Longitudinal First Installation method (penilitian Ini) karena lebih mudah dalam aplikasi teknologi automatic dan semi automatic welding.



Plane Block Flow

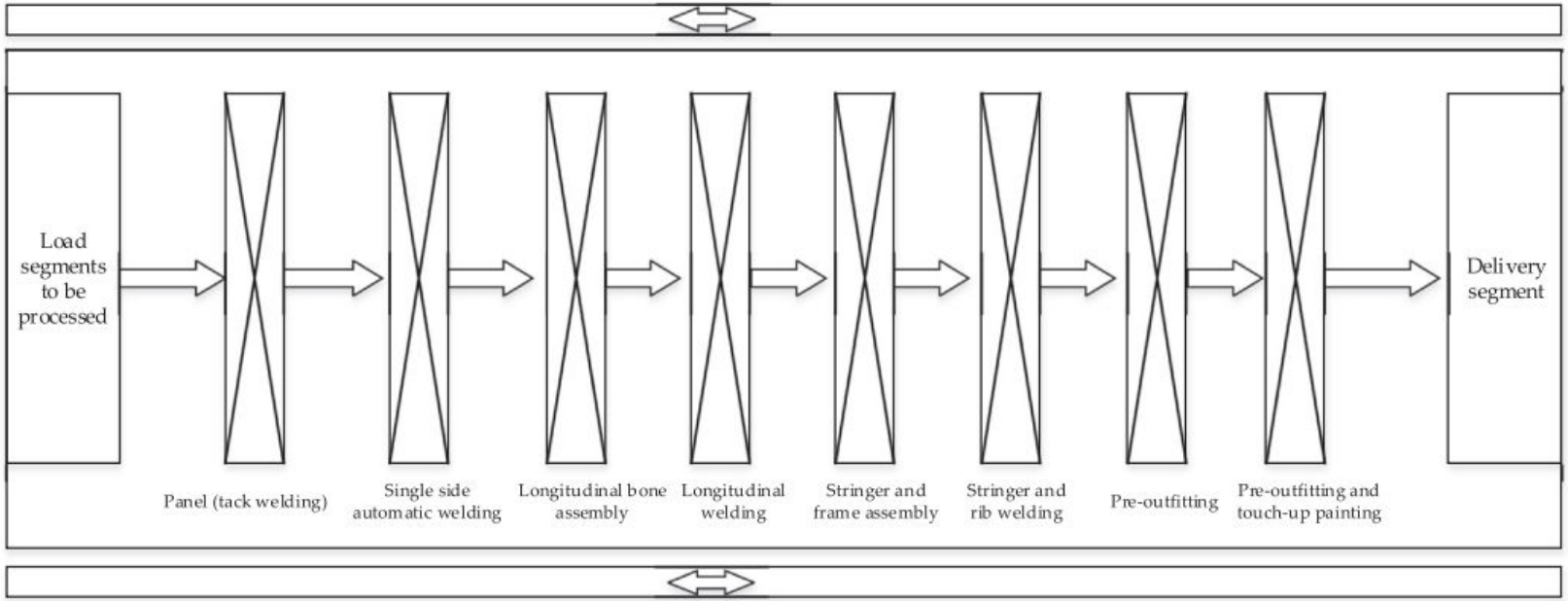
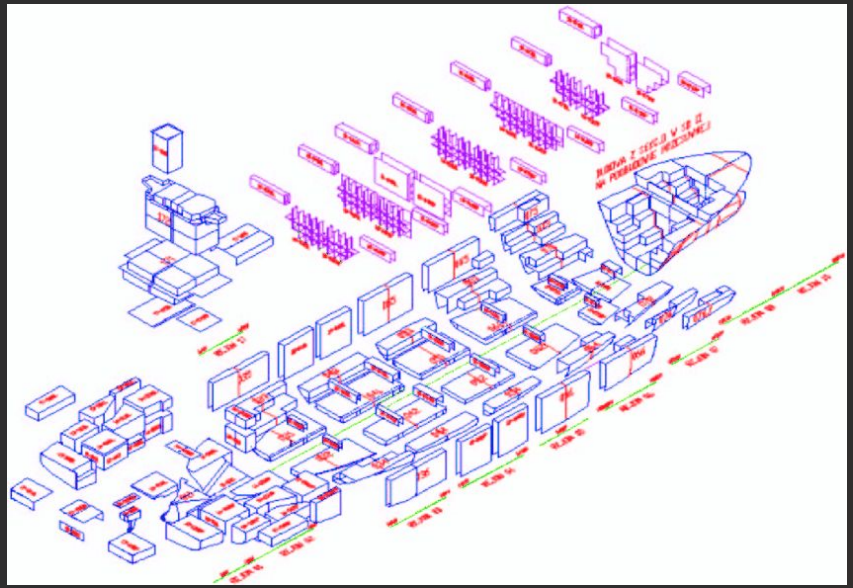


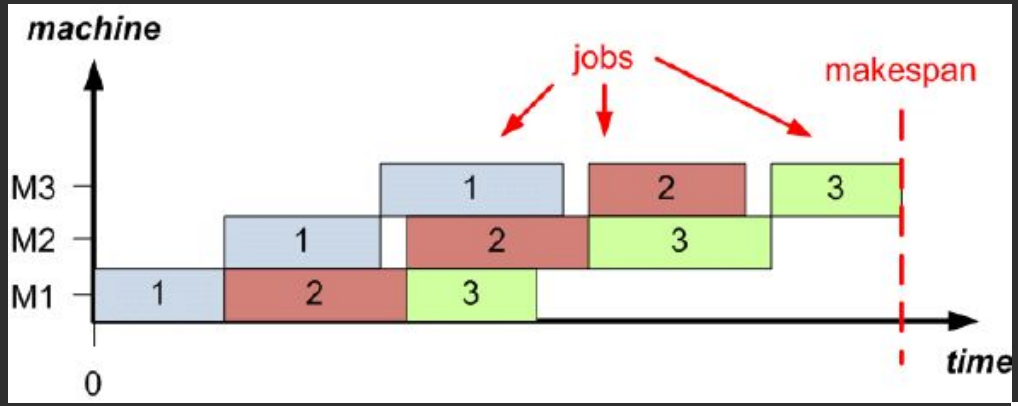
Fig. 1. Flow line diagram of a shipyard's plane block workshop.

Rumusan Masalah

Kapal banyak bloknya kalau dibagi dengan sistem blok dan beda - beda ukuranya urutan kerja yang paling cepat yang kayak gimana ?



Flow Shop Scheduling Problem



Machine = Workstation

Jobs = Blocks

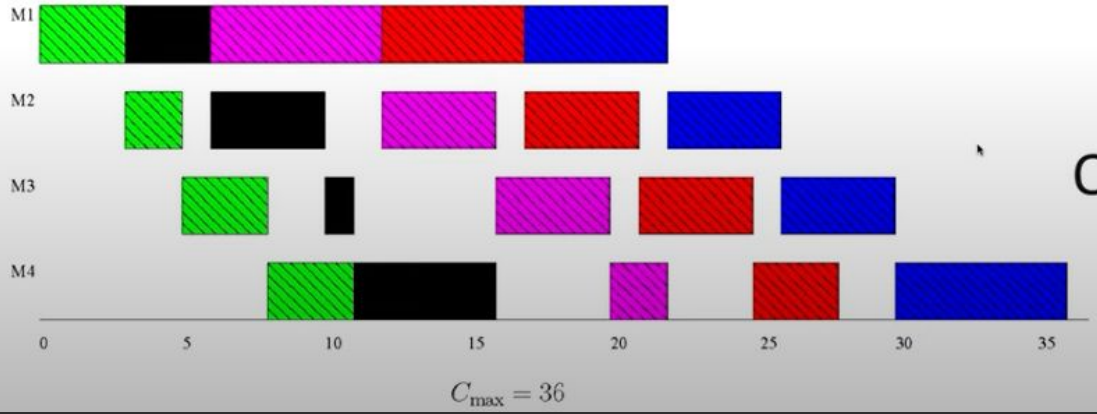
An **flow shop problem** is a general shop problem in which

- each job i consists of m operations O_{ij} with processing time p_{ij} $(j = 1,..., m)$ where O_{ij} must be processed on machine M_j,
- there are precedence relations of the form $O_{ij} \rightarrow O_{ij+1}$ $(i = 1,..., m - 1)$ for each $i = 1,..., n$, i.e. each job is first processed on machine 1, then on machine 2, then on machine 3, etc.

Thus, the problem is to find a **job order** π_j for each machine M_j.

	M1	M2	M3	M4
Job 1	5	4	4	3
Job 2	5	4	4	6
Job 3	3	2	3	3
Job 4	6	4	4	2
Job 5	3	4	1	5

p(j,i):Processing time of job j in machine i.



ORDER of JOBS: 3,5,4,1,2

NP Hard J-1!

Karakterisitik Unik Dari Masalah

GREEN BLOCK FLOW SCHEDULING PROBLEM (GBFSSP)

- Zero Buffer. Karena volume dan berat dari plane section dari kapal yang besar kita tidak bisa menyimpanya di tempat lain terlebih dahulu sehingga section yang lagi dibuat akan tetap diam di current workstation hingga workstation yang berikutnya sudah selesai beroperasi
- Preparation time . Karena ukuran dan bentuk dari block bervariatif terdapat waktu tambahan yang terjadi ketika akan menyiapkan suatu workstation setelah memproses blok dengan tipe yang berbeda.

(1) Zero buffer. Due to the large volume and weight of the plane section of the ship, it cannot be stored or moved after processing in a certain station. If the processing task of the next station is not completed, the section can only stay in the current station until the next station completes the processing task.
(2) Switch the preparation time. For the ship plane block, because the processing tasks of each station are different and the shape and size of the block are not exactly the same, there is a certain preparation time for the station after finishing the processing of a block.

Tujuan & Asumsi Penilitian

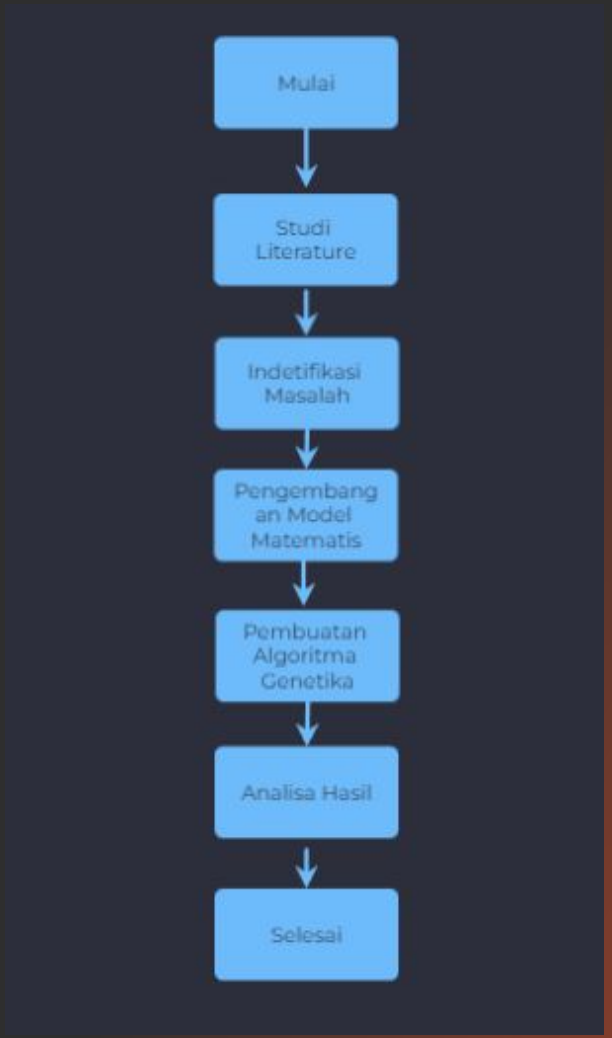
Tujuan Penilitian

Karena Subjek dari riset ini adalah flowline ship plane block maka kita ingin mendapatkan urutan kerja yang membutuhkan waktu yang paling cepat dalam memproduksi plane block.

Selain dari itu karena ikut mengkonisderasi aspek dari green manufacturing maka kita ingin menambahkan selain jadwal urutan kerja yang paling cepat tetapi juga yang paling ramah lingkungan, untuk membedakanya dengan referensi [1] dalam perhitungan kita mengkonisder faktor deteroriasi, dimana faktor tersebut adalah faktor yang meperhitungkan keahlian dari operator, kondisi fisik operator dan *equipment wear.*[2]

Sehingga tujuan penilitan bisa disimpulkan sebagai berikut :

- Mendapatkan jadwal urutan kerja yang paling optimal dengan algoritma yang diajukan
- Membandingkan hasil jadwal optimal dengan yang mengkonsiderasi faktor deteriorasi dengan yang tidak
- Melihat perbedaan hasil jadwal optimasi dengan perhitungan green index dengan yang tidak



Asumsi Penelitian

- Masing - masing block diproses pada flow-line dengan urutan proses yang berurutan
- Waktu yang dibutuhkan untuk memproses masing - masing block pada tiap workstation telah ditentukan.
- Blok tidak bisa diganggu dengan block lain ketika lagi di proses di workstation
- Setelah block selesai di proses, block akan menunggu di workstasion sekarang hingga workstasion berikutnya kosong
- Waktu pergerakan dari block antara workstasion diabaikan

$$L_{j_n,m} = \sum_{n=1}^{J} x^{k}_{j_n j_{n-1}} \times S_{j_n,m} + \sum_{n=1}^{J} x^{k}_{j_n j_{n-1}} \times P^{m}_{j_n j_{n-1}} + T_{j_n,m}$$

$$= 2, \ldots, J$$

$$C_{\max} = L_{jj,M}$$

(1) Each block is processed on the flow-line according to a certain processing sequence.
(2) The processing time of each block in each station is determined, and the equipment switching preparation time of the processing block is set as the preparation time based on the processing sequence of the block.
(3) After each block arrives, the preparation time to start block switching.
(4) At the same time, each block can only be processed in a unique position.
(5) Blocks cannot be interrupted by other blocks when they are processed in one station.
(6) After each block completes a certain process, it will wait on this station until the next station is idle.
(7) The moving time of blocks between stations is not considered.

Fungsi Objektif

$$F = \min(C_{\max} + E)$$

Cmax = Maximum Block Completion Time

E = Green Index (Carbon Emission and Noise Pollution)

Metode Optimasi Particle Swarm

INTRO

Particle Swarm Optimization (PSO) dikembangkan oleh Kennedy dan Eberchart pada tahun 1995. Algoritma ini didasarkan pada perilaku sekawanan burung atau ikan. Setiap individu atau partikel berperilaku dengan cara menggunakan kecerdasannya (intelligence) sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh di kelompok tersebut.

Rumusan Algoritma

Update Posisi

$$X^i(t+1) = X^i(t) + V^i(t+1)$$

Update velocities

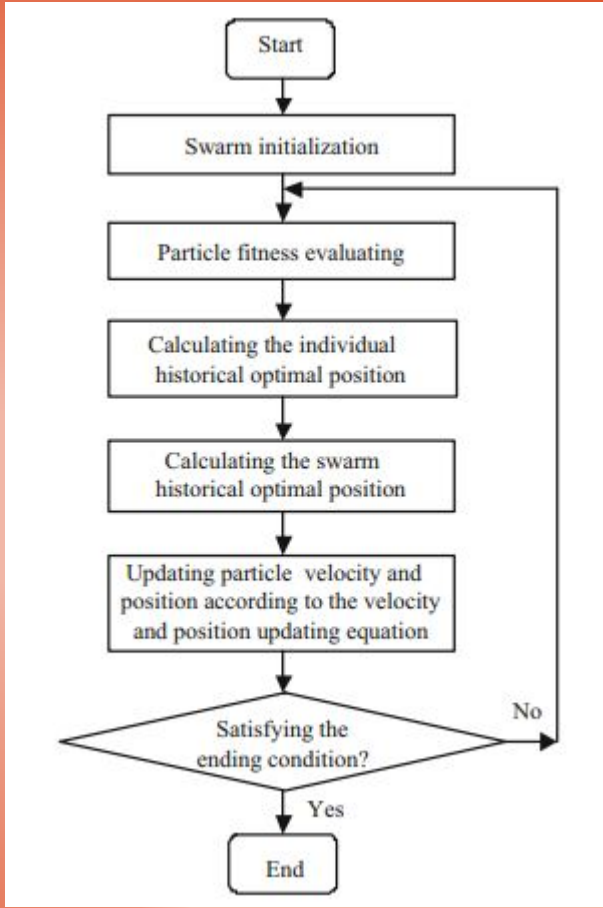
$$V^i(t+1) = wV^i(t) + c_1 r_1(pbest^i - X^i(t)) + c_2 r_2(gbest - X^i(t))$$

$X^i(t)$ Posisi partikel i pada iterasi t

$V^i(t)$ Kecepatan partikel i pada iterasi t

R1, r2 = bilangan random
C1 = learning factor
C2 = Social learning factor
W = inertia weight

PSWO FLOW CHART

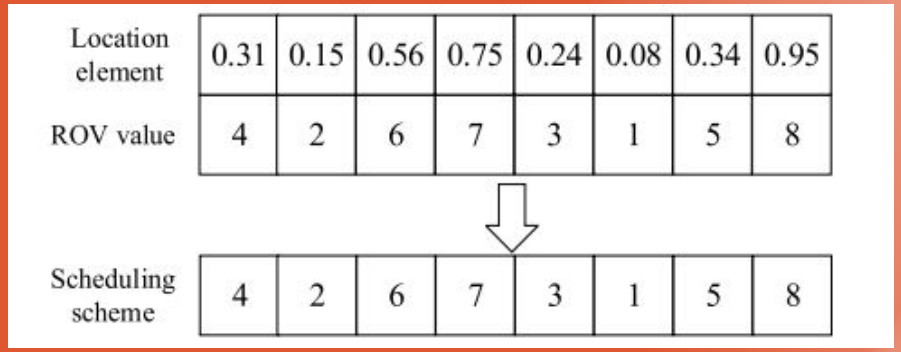


Representasi Particle (Solusi)

TABLE 9. **Processing time of waiting to be scheduled block in each station.**

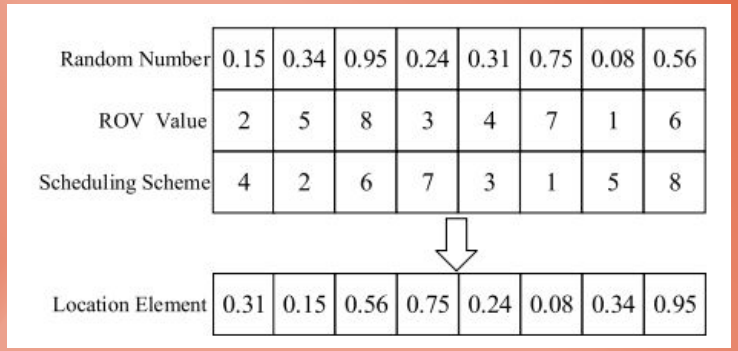
Block	Block type	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
1	G_1	3	2	4	1	0	2	3	5
2	G_1	3	2	4	1	0	2	3	5
3	G_1	3	2	4	1	0	2	3	5
4	G_1	3	2	4	1	0	2	3	5
5	G_1	3	2	4	1	0	2	3	5
6	G_2	5	4	6	3	0	0	0	0
7	G_2	5	4	6	3	0	0	0	0
8	G_2	5	4	6	3	0	0	0	0
9	G_2	5	4	6	3	0	0	0	0
10	G_2	5	4	6	3	0	0	0	0
11	G_3	6	3	5	1	0	1	4	2
12	G_3	6	3	5	1	0	1	4	2
13	G_3	6	3	5	1	0	1	4	2
14	G_3	6	3	5	1	0	1	4	2
15	G_3	6	3	5	1	0	1	4	2
16	G_4	2	5	1	4	0	0	0	0
17	G_4	2	5	1	4	0	0	0	0
18	G_4	2	5	1	4	0	0	0	0
19	G_4	2	5	1	4	0	0	0	0
20	G_4	2	5	1	4	0	0	0	0

Proses Encoding Metode ROV



Location element	0.31	0.15	0.56	0.75	0.24	0.08	0.34	0.95
ROV value	4	2	6	7	3	1	5	8
Scheduling scheme	4	2	6	7	3	1	5	8

Dari lokasi jadi jadwal



Random Number	0.15	0.34	0.95	0.24	0.31	0.75	0.08	0.56
ROV Value	2	5	8	3	4	7	1	6
Scheduling Scheme	4	2	6	7	3	1	5	8
Location Element	0.31	0.15	0.56	0.75	0.24	0.08	0.34	0.95

Jadwal jadi lokasi

Hasil Codingan

```python
#Swarm Intialization
n_pop = 100 #Jumlah partikel
n_blocks = 5
n_workstation = 4
#Xp = np.array([np.random.permutation(range(1,n_blocks + 1)) for _ in range(n_pop)]) # Array of particle need connversion
X = np.random.uniform(0, n_blocks, size= (n_pop,n_blocks))
V = np.random.uniform(0, n_blocks, size= (n_pop,n_blocks))

#data for fitness evaluation
sample_prep_time = np.random.randint(n_blocks, size = (n_blocks,n_workstation))
sample_process_time = np.random.randint(n_blocks, size = (n_blocks,n_workstation))
```

```python
#Hyperparameter
w = 0.9 #Interial Weight
c1 = 2  #Cognitive Learning Factor
c2 = 2  #Social Leraning Factor

#Conversion Based on Ranked Order Value
def encoding_pop(solutions,reverse = True):
  X_encoded = []
  for solution in solutions :
      X_encoded.append(np.argsort(solution) + 1)
  return np.array(X_encoded)

def encoding_part(solution) :
  return np.argsort(solution) + 1
```

```python
#Fitness Evaluation
def calculate_obj(sol,sample_process_time,n_blocks,m_station):
  qTime = queue.PriorityQueue()
  qMachines = []
  for i in range(m_station):
      qMachines.append(queue.Queue())

  for i in range(n_blocks):
      qMachines[0].put(sol[i])

  busyMachines = []
  for i in range(m_station):
      busyMachines.append(False)

  time = 0
  job = qMachines[0].get()
  qTime.put((time + sample_process_time[job-1][0], 0, job))
  busyMachines[0] = True

while True:
    time, workstation , block = qTime.get()
    #print(f"it's processing block {block} at time interval {time} in workstasion {workstation+1} current work stasion condition : {busyMachines}")
    # Remove the hashtage on upper code to see the calculation processes
    if block == sol[n_blocks-1] and workstation == m_station-1:
        break
    busyMachines[workstation] = False
    if not qMachines[workstation].empty():
            j = qMachines[workstation].get()
            qTime.put((time+sample_process_time[j-1][workstation], workstation, j))
            busyMachines[workstation] = True
    if workstation < m_station-1:
        if busyMachines[workstation+1] == False:
            qTime.put((time+sample_process_time[block-1][workstation], workstation+1, block))
            busyMachines[workstation+1] = True
        else:
            qMachines[workstation+1].put(block)
return time
```

```python
def fitness(converted_solutions,cal_obj,process_time,n_blocks,n_workstation):
  c_max = [] #Maximum Block Completion Time
  for solution in converted_solutions :
    c_max.append(calculate_obj(solution,process_time,n_blocks,n_workstation))
  return np.array(c_max)
```

```
# Initialize data
pbest = X
pbest_obj = fitness(encoding_pop(X),calculate_obj,sample_process_time,n_blocks,n_workstation)
gbest = pbest[pbest_obj.argmin()]
gbest_obj = pbest_obj.min()
```

```
#Updating Particle
def update():
    "Function to do one iteration of particle swarm optimization"
    global V, X, pbest, pbest_obj, gbest, gbest_obj
    # Update params
    r1, r2 = np.random.rand(2)
    V = w * V + c1*r1*(pbest - X) + c2*r2*(gbest.reshape(-1,n_blocks)-X)
    Xn = X + V
    obj = fitness(encoding_pop(Xn),calculate_obj,sample_process_time,n_blocks,n_workstation)
    pbest[(pbest_obj >= obj)] = X[(pbest_obj >= obj)]
    pbest_obj = np.array([pbest_obj, obj]).min(axis=0)
    gbest = pbest[pbest_obj.argmin()]
    gbest_obj = pbest_obj.min()
```

Code Link

https://colab.research.google.com/drive/1f3nFE-bad-lonhiI751mdElDYYaHJD8b?usp=sharing

Referensi

- Green scheduling optimization of ship plane block flow line considering carbon emission and noise. *Hui Guo & Friends* (2020)
- A Hybrid Whale Optimization Algorithm for Plane Block Parallel Blocking Flowline Scheduling Optimization With Deterioration Effect in Lean Shipbuilding. *Jinghua Li & Hui Guo* (2021)
- *IMPROVED PARTICLE SWARM ALGORITHM FOR PERMUTATION FLOW SHOP SCHEDULING PROBLEMS*, Manal Abdulkareem Zeidan* , Manar Abdulkareem Al-AbajI and Manaf Hazim Ahmed University of Mosul, Irak.

Terima Kasih

https://www.youtube.com/watch?v=eBW04vH1BxU

Assume we have P particles and we denote the position of particle i at iteration t as $X^i(t)$, which in the example of above, we have it as a coordinate $X^i(t) = (x^i(t), y^i(t))$. Besides the position, we also have a velocity for each particle, denoted as $V^i(t) = (v_x^i(t), v_y^i(t))$. At the next iteration, the position of each particle would be updated as

$$X^i(t+1) = X^i(t) + V^i(t+1)$$

or, equivalently,

$$x^i(t+1) = x^i(t) + v_x^i(t+1)$$
$$y^i(t+1) = y^i(t) + v_y^i(t+1)$$

t kecil adalah iterasi kebarapa i adalah particle ke berapa dan X^i(t) nujukkin posisi dari suatu partikel pada iterasi t

Q : How do you intialize the population as a solution also how do you determine the position value and the velocity value