****

**IND8200 – Organisation industrielle**

**Hiver 2022**

**TP No. 6**

**Groupe 01**

**1949477 – Ming Xiao Yuan**

**1955913 – Andi Podgorica**

**Soumis à : M. Arman Aksoy**

**15 avril 2022**

**Problème à remettre 1**: « FlowShop »

**Solution 1 : Résolution par algorithme**

Nous avons 4 machines, donc ceci correspond à une FlowShop à plus de 2 machines. Alors, nous avons utilisé l’algorithme de Campell, Dudek & Smith. Ceci nous a permis de trouver les séquences d’ordonnancement minimisant le makespan ainsi que le temps moyen passé dans le système. Pour ce faire, nous avons regroupé les machines et avons appliqué l’algorithme de Johnson traditionnel.

Une image contenant table

Description générée automatiquement

Figure 1 : Tableau donné avec les Machines selon les Jobs

**Itération 1 : Johnson sur M1 et M4**

Étape 1 : {J6, …}

Étape 2 : {J6, J1, …}

Étape 3 : {J6, J1, …, J4}

Étape 4 : {J6, J1, …, J3, J4}

Étape 5 : {J6, J1, J2, …, J3, J4}

Étape 6 : {J6, J1, J2, J5, J3, J4}

*Séquence : {J6, J1, J2, J5, J3, J4}*

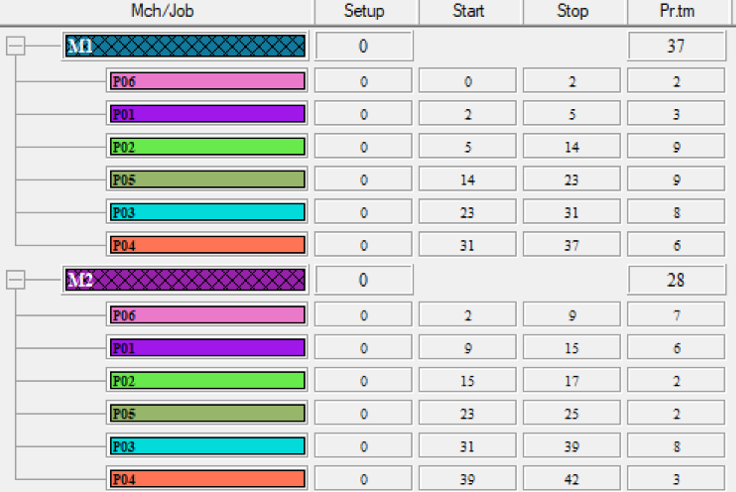
Makespan : 55 sec

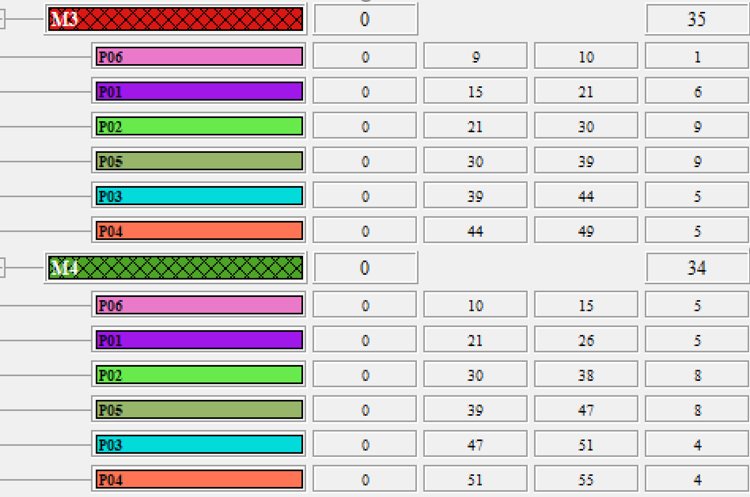
Temps passé dans le système : 232/6 = 38,6667 sec

Timeline

Description automatically generated with medium confidence

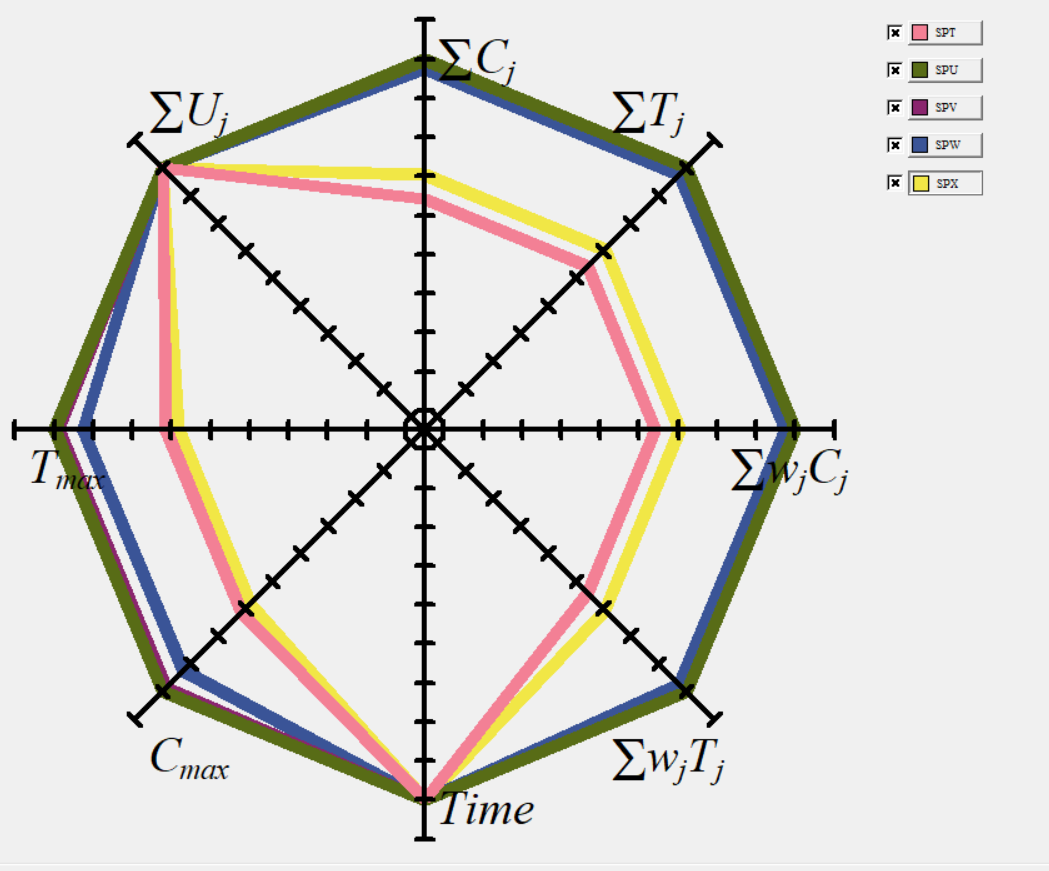
Diagramme de Gantt





Une image contenant table

Description générée automatiquement



**Itération 2 : Johnson sur M’1 = M1+M2 ainsi que M’2 = M3+M4**

|  |  |  |
| --- | --- | --- |
| Jobs | M’1 | M’2 |
| J1 | 9 | 11 |
| J2 | 11 | 17 |
| J3 | 16 | 9 |
| J4 | 9 | 9 |
| J5 | 11 | 17 |
| J6 | 9 | 6 |

Étape 1 : {…, J6}

Étape 2 : {J4, …, J6}

Étape 3 : {J4, J1, …, J6}

Étape 4 : {J4, J1, …, J3, J6}

Étape 5 : {J4, J1, J2, …, J3, J6}

Étape 6 : {J4, J1, J2, J5, J3, J6}

*Séquence : {J4, J1, J2, J5, J3, J6}*

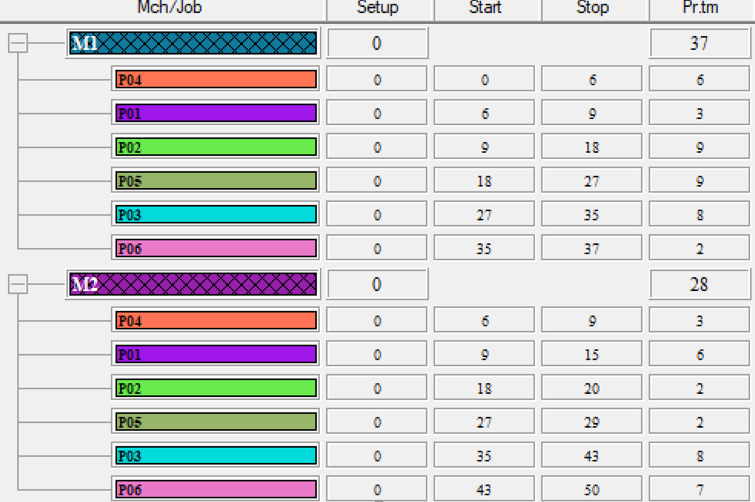
Makespan : 57 sec

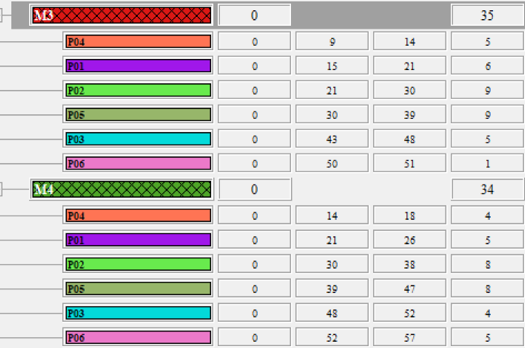
Temps passé dans le système : 238/6 = 39,6667 sec

Chart

Description automatically generated with low confidence

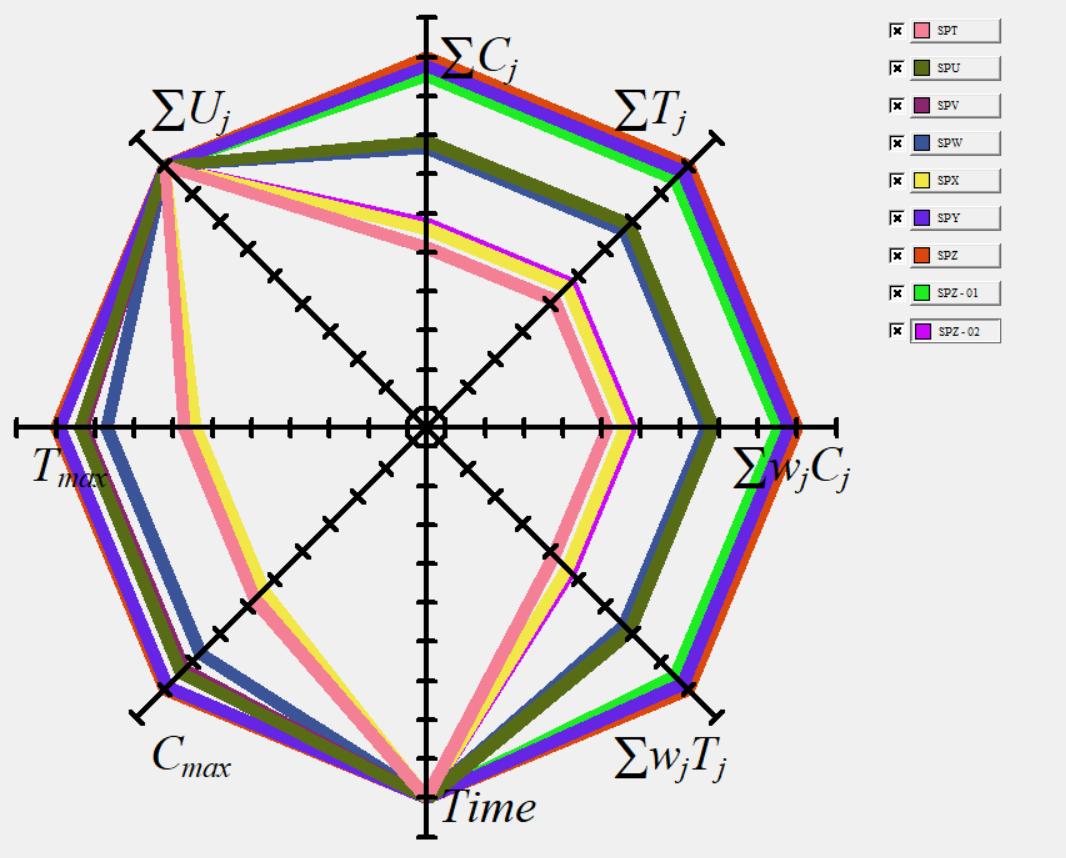
Diagramme de Gantt





Table

Description automatically generated



**Itération 3 : Johnson sur M’’1 = M1+M2+M3 ainsi que M’’2 = M2+M3+M4**

|  |  |  |
| --- | --- | --- |
| Jobs | M’’1 | M’’2 |
| J1 | 15 | 17 |
| J2 | 20 | 19 |
| J3 | 21 | 17 |
| J4 | 14 | 12 |
| J5 | 20 | 19 |
| J6 | 10 | 13 |

Étape 1 : {J6, …}

Étape 2 : {J6, … J4}

Étape 3 : {J6, J1, … J4}

Étape 4 : {J6, J1, …, J3, J4}

Étape 5 : {J6, J1, J5, …, J3, J4}

Étape 6 : {J6, J1, J5, J2, J3, J4}

*Séquence : {J6, J1, J5, J2, J3, J4}*

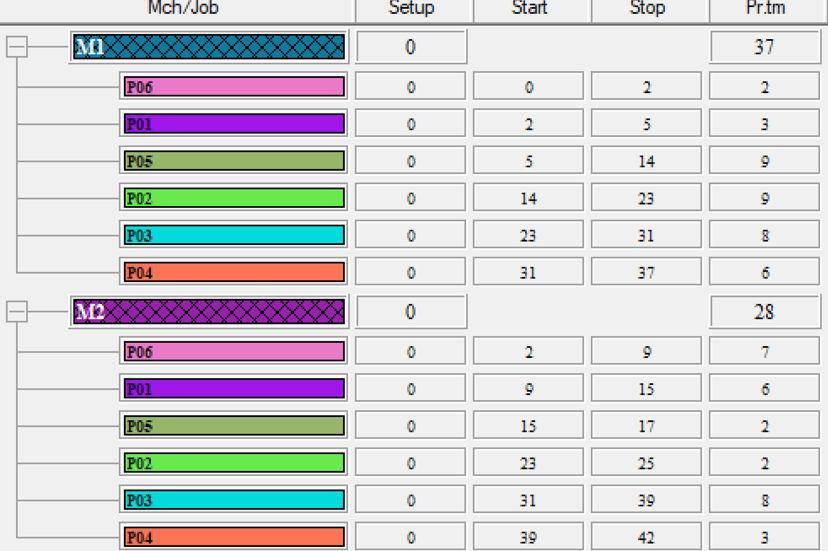
Makespan : 55 sec

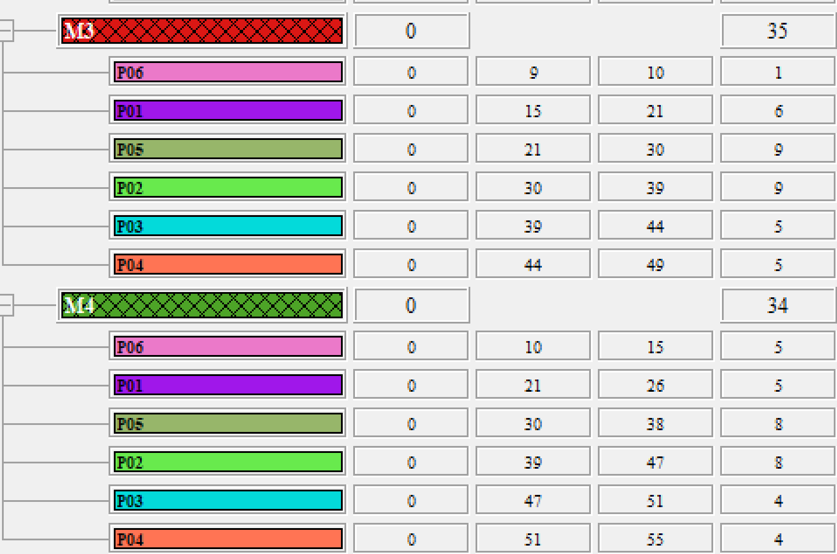
Temps moyen passé dans le système : 232/6 = 38,6667 sec

Graphical user interface

Description automatically generated with medium confidence

Diagramme de Gantt





Table

Description automatically generated

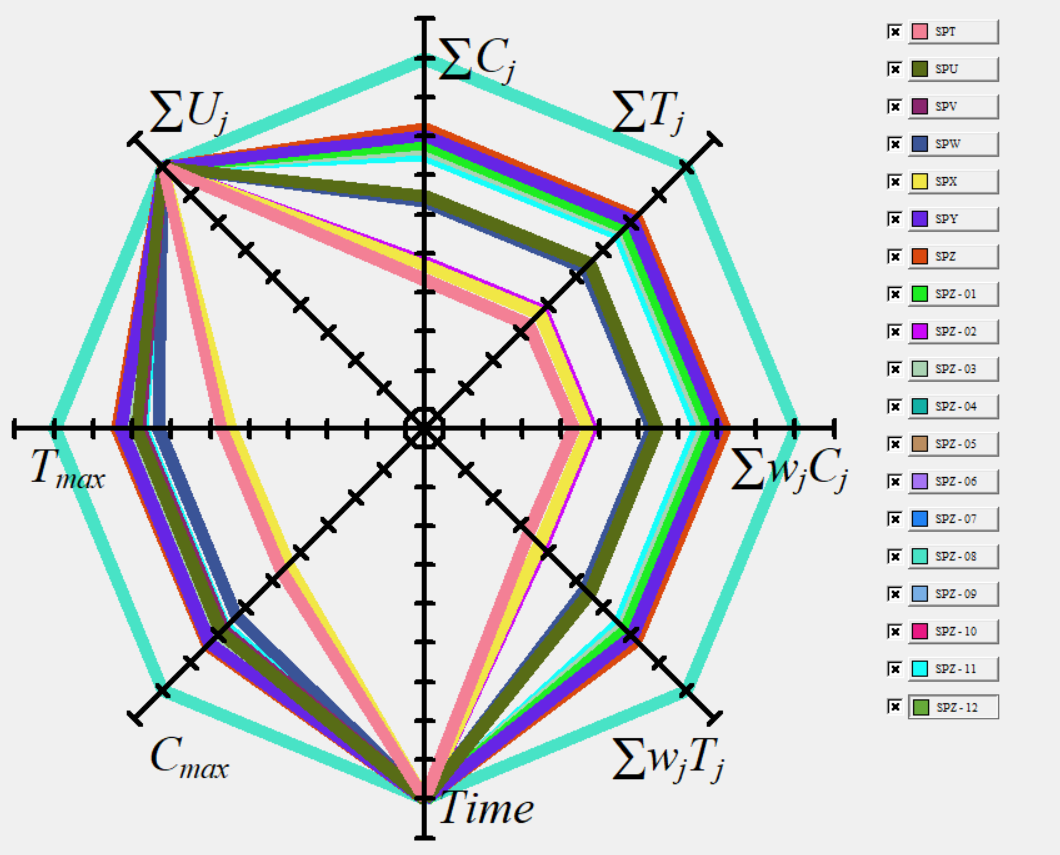


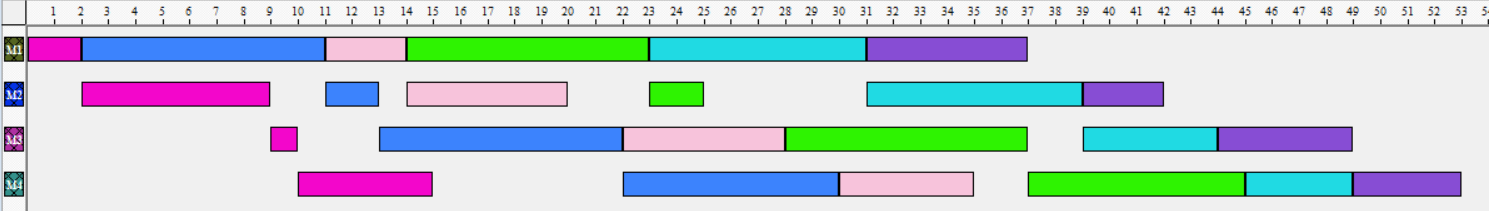
Tableau 1 : Les trois séquences selon leur temps moyen passé dans le système et le temps de fin des opérations

|  |  |  |  |
| --- | --- | --- | --- |
| Obj. | Séquence 1 | Séquence 2 | Séquence 3 |
| Makespan | 55 | 57 | 55 |
| Temps moyen passé système | 38,67 | 39,67 | 38,67 |

Il est clair donc que la première et la troisième itération permettent d’avoir des meilleurs résultats pour minimiser le makespan et le temps moyen passé dans le système.

**Solution 2 :**

* Règle ou heuristique utilisée : Local Search
* Séquence obtenue : {J6; J5; J1; J2; J3; J4}
* Makespan : 53 sec
* Diagramme de Gantt :



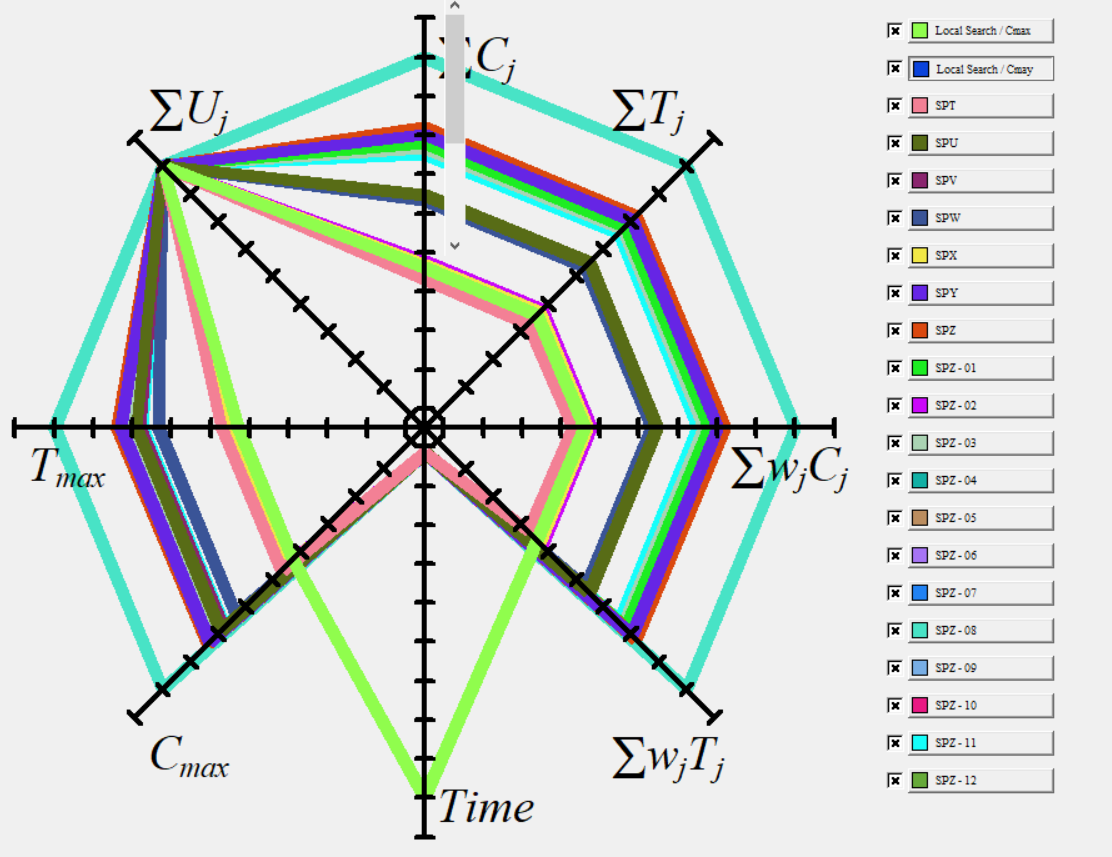
Une image contenant table

Description générée automatiquement

Une image contenant table

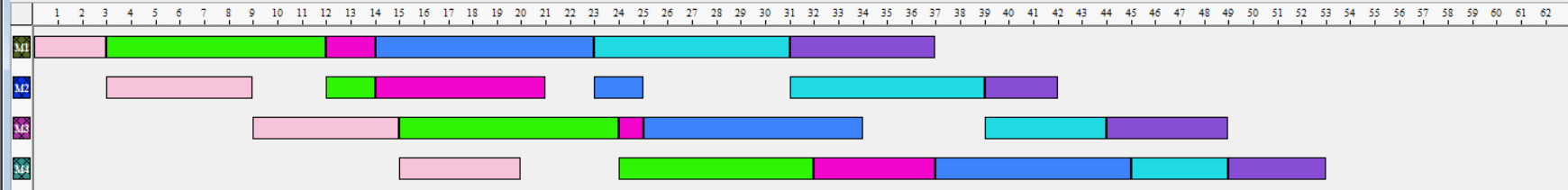
Description générée automatiquement

* Graphique des mesures de performance :



Solution 3 :

* Règle ou heuristique utilisée : Shifting Bottleneck / Tmax
* Séquence obtenue : {J1; J2; J6; J5; J3; J4}
* Makespan : 53
* Diagramme de Gantt :



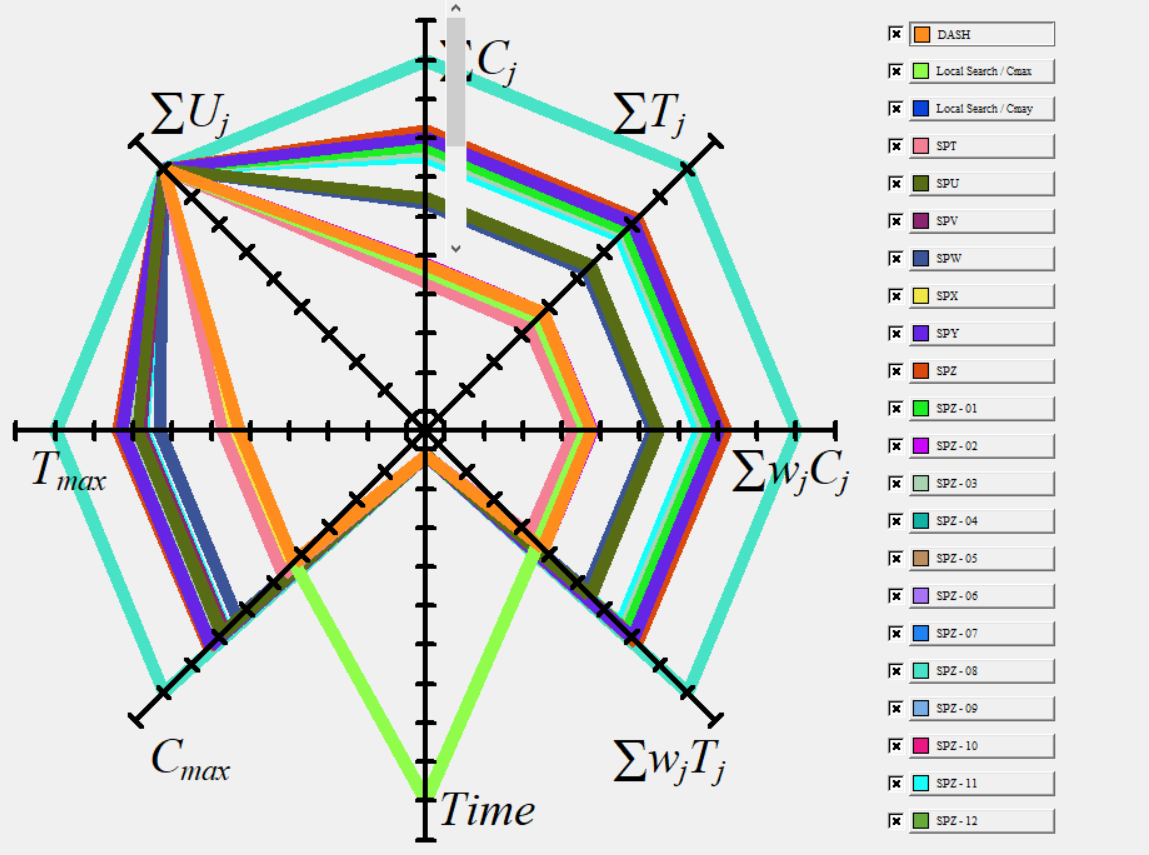
Une image contenant table

Description générée automatiquement

Une image contenant table

Description générée automatiquement

* Graphique des mesures de performance :



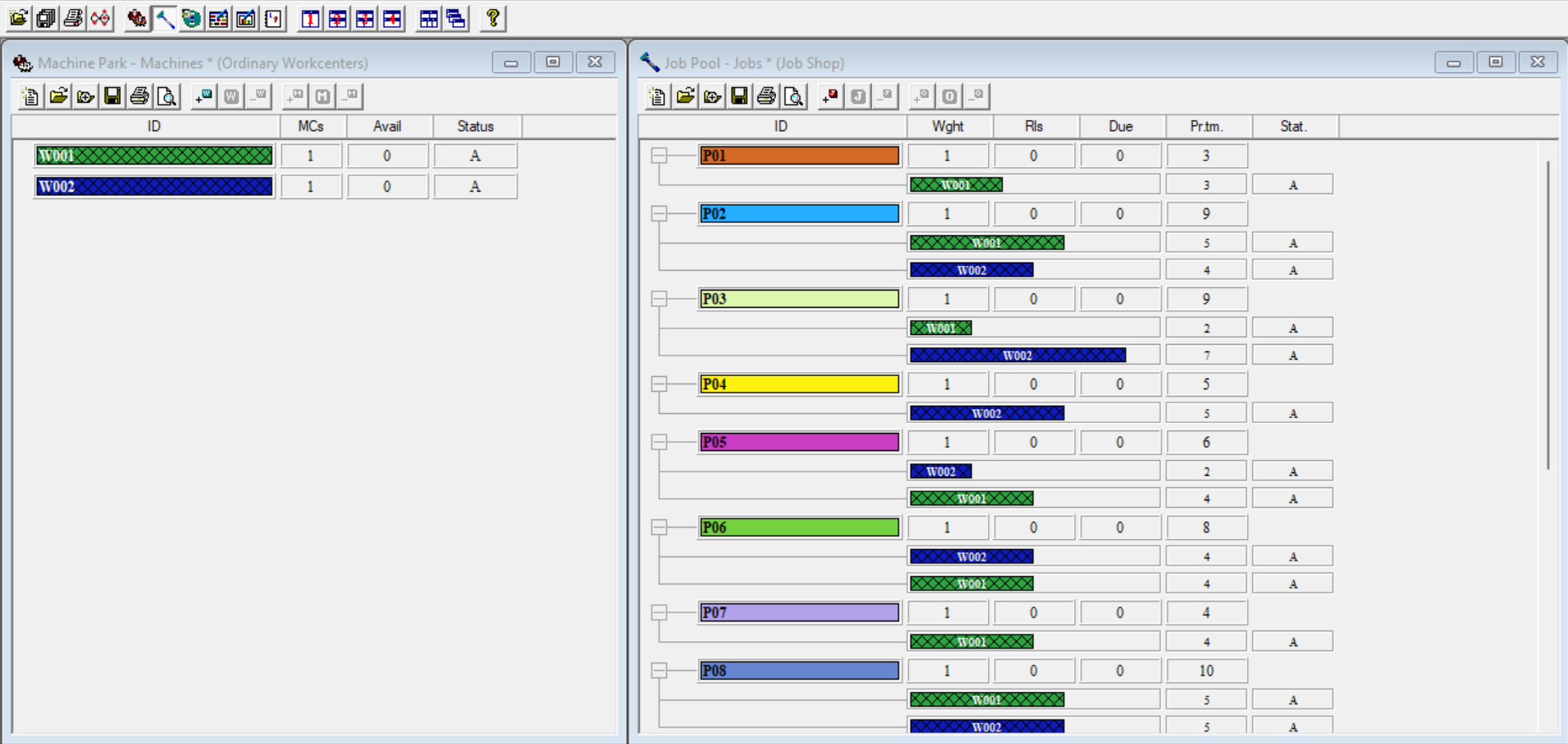
**Choix final de solution** :

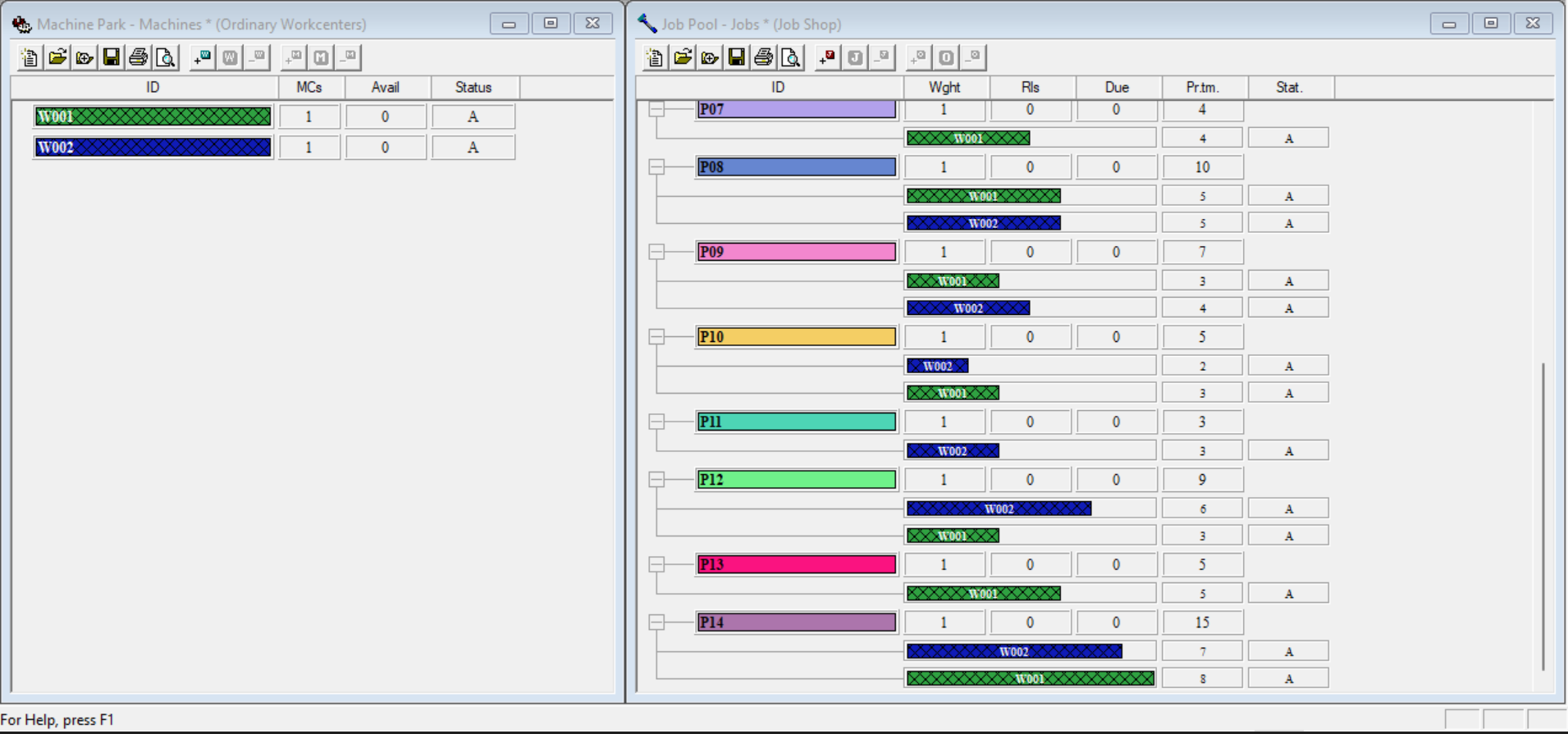
Après l’analyse de tous les résultats, nous constatons que l’heuristique Local Search est la meilleure car c’est elle qui a le Makespan ainsi que le temps moyen passé dans le système les plus petits.

**Problème à remettre 2** : « JobShop »

Solution 1 :

Pour la solution 1, comme nous avons un problème Jobshop avec 14 commandes sur 2 machines de l’atelier, notre première solution sera d’employer l’algorithme Jackson, qui est basé sur Johnson.





Voici l’ajout des 14 jobs dans le modèle Jobshop.

Avec l’étape 0 de l’algorithme de Jackson, nous trouvons 4 ensembles. Soit E1 : {P1, P7, P13} avec des taches comme une seule opération sur A. E2 : {P4, P11} avec des tâches comportant une seule tâche sur B. E3 : {P2, P3, P8, P9} comportant une première opération sur A, une deuxième sur B et finalement E4 : {P5, P6, P10, P12, P14} comportant une première opération sur B, une deuxième sur A.

Avec l’étape 1, nous procédons à ordonner E3 avec l’algorithme de Johnson.

|  |  |  |
| --- | --- | --- |
| E3 | A | B |
| P2 | 5 | 4 |
| P3 | 2 | 7 |
| P8 | 5 | 5 |
| P9 | 3 | 4 |

* Étape1 : {P3, …}
* Étape2 : {P3, P9, …}
* Étape3 : {P3, P9, …, P2}
* Étape4 : {P3, P9, P8, P2}
* Seq1 : {P3, P9, P8, P2}

Avec l’étape 2, nous procédons à ordonner similairement E4 avec l’algorithme de Johnson.

|  |  |  |
| --- | --- | --- |
| E4 | B | A |
| P5 | 2 | 4 |
| P6 | 4 | 4 |
| P10 | 2 | 3 |
| P12 | 6 | 3 |
| P14 | 7 | 8 |

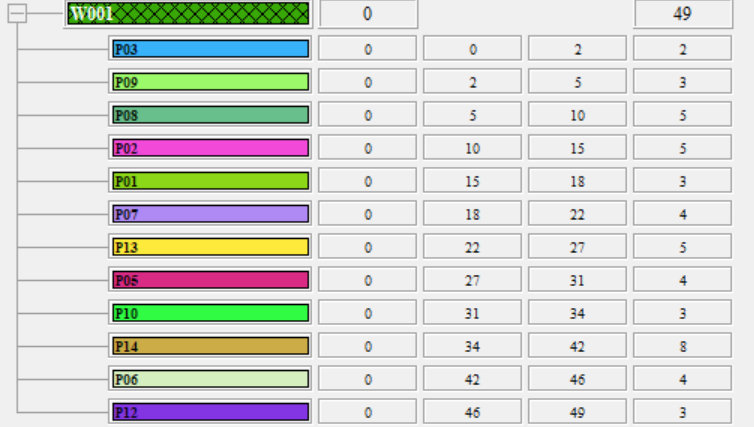
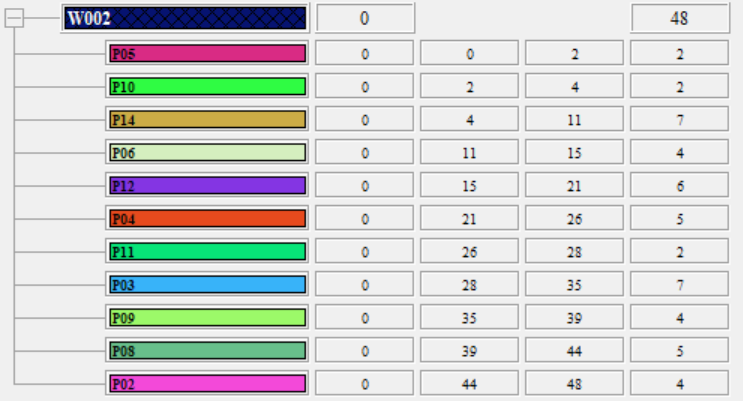
* Étape1 : {P5, …}
* Étape2 : {P5, P10, …}
* Étape3 : {P5, P10, …, P12}
* Étape4 : {P5, P10, …, P6, P12}
* Étape5 : {P5, P10, P14, P6, P12}
* Seq1 : {P5, P10, P14, P6, P12}

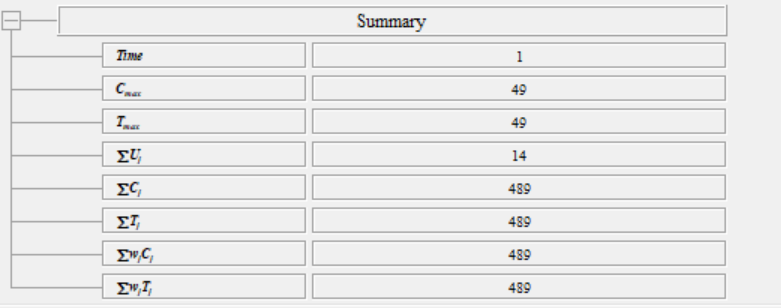
Avec l’étape 3, nous procédons à effectuer les tâches sur A dans l’ordre E3 E1 E4. Nous avons donc A : P3, P9, P8, P2, P1, P7, P13, P5, P10, P14, P6, P12 (12 tâches).

Similairement avec l’étape 4, nous précédons à effectuer les tâches sur B dans l’ordre E4, E2, E3. Nous avons donc B : P5, P10, P14, P6, P12, P4, P11, P3, P9, P8, P2 (11 tâches)

**Règle de priorité utilisé :** SPT (TOC) Shortest Processing Time

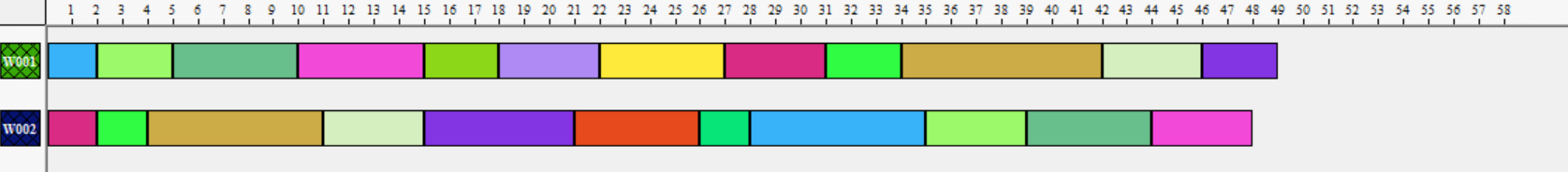
**Séquence obtenue** :

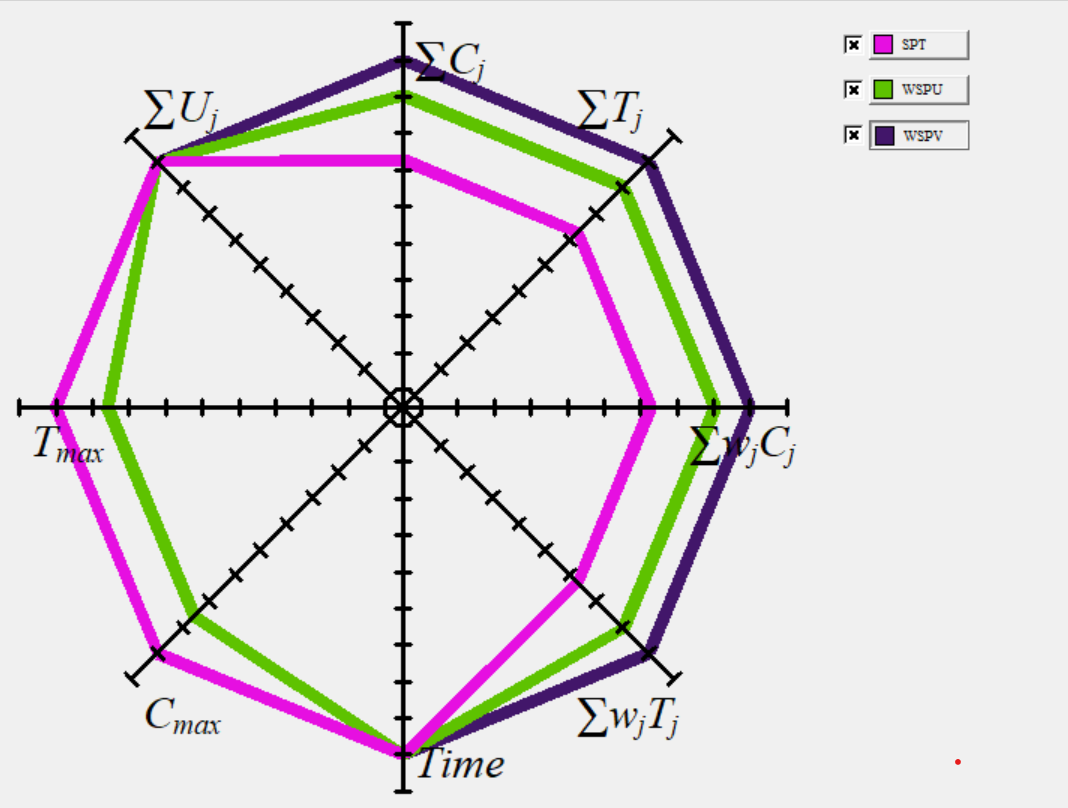


À la suite de l’utilisation de SPT (Shortest Processing Time) avec l’algorithme de Jackson, nous remarquons une séquence de A : {P3, P9, P8, P2, P1, P7, P13, P5, P10, P14, P6, P12} et une séquence de B : {P5, P10, P14, P6, P12, P4, P11, P3, P9, P8, P2}.

**Diagramme de Gantt**



**Graphe avec mesures de performance**



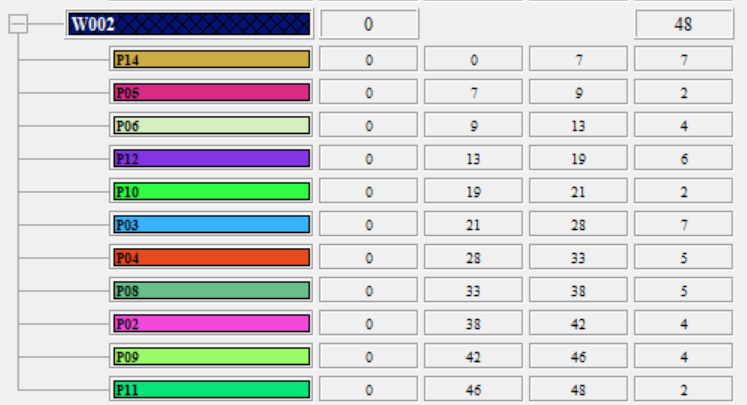
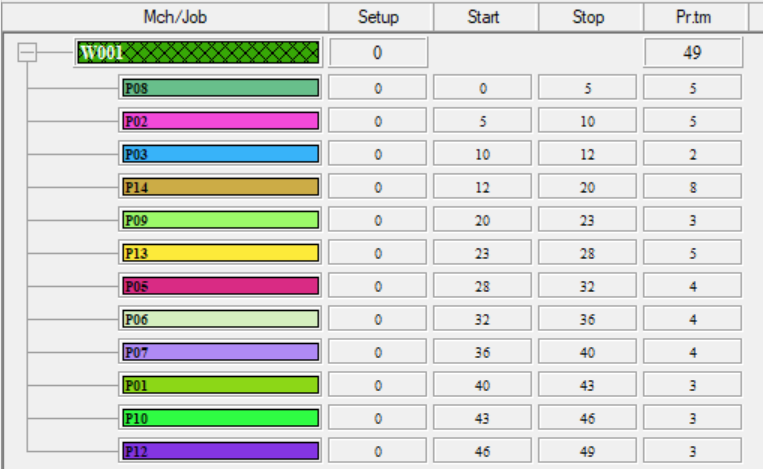
Solution 2 :

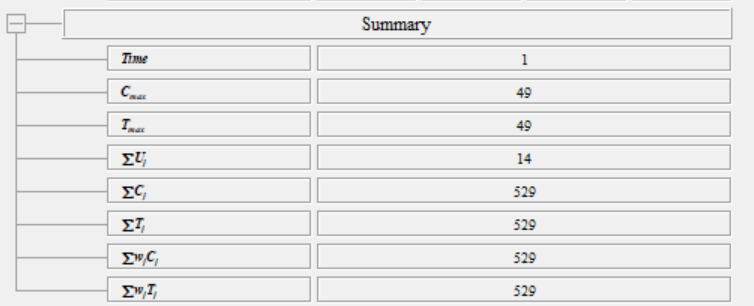
Voici l’ordonnancement de Job Shop avec SPT (TOC) Shortest Processing Time avec l’heuristique General SB Routine.

**Règle de priorité utilisé** : SPT (TOC) Shortest Processing Time

**Heuristiques LEKIN** : General SB Routine

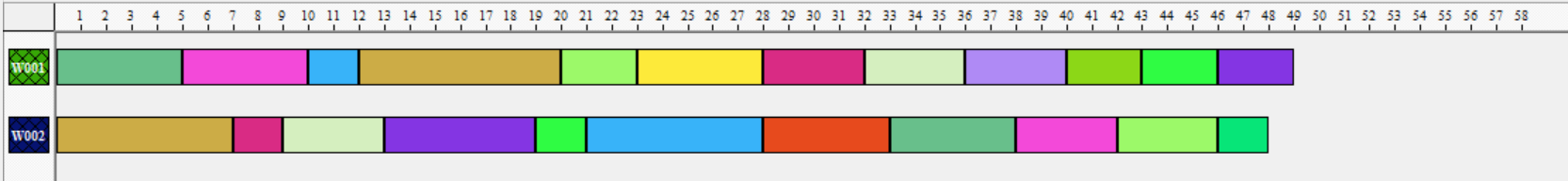
**Séquence obtenue** :



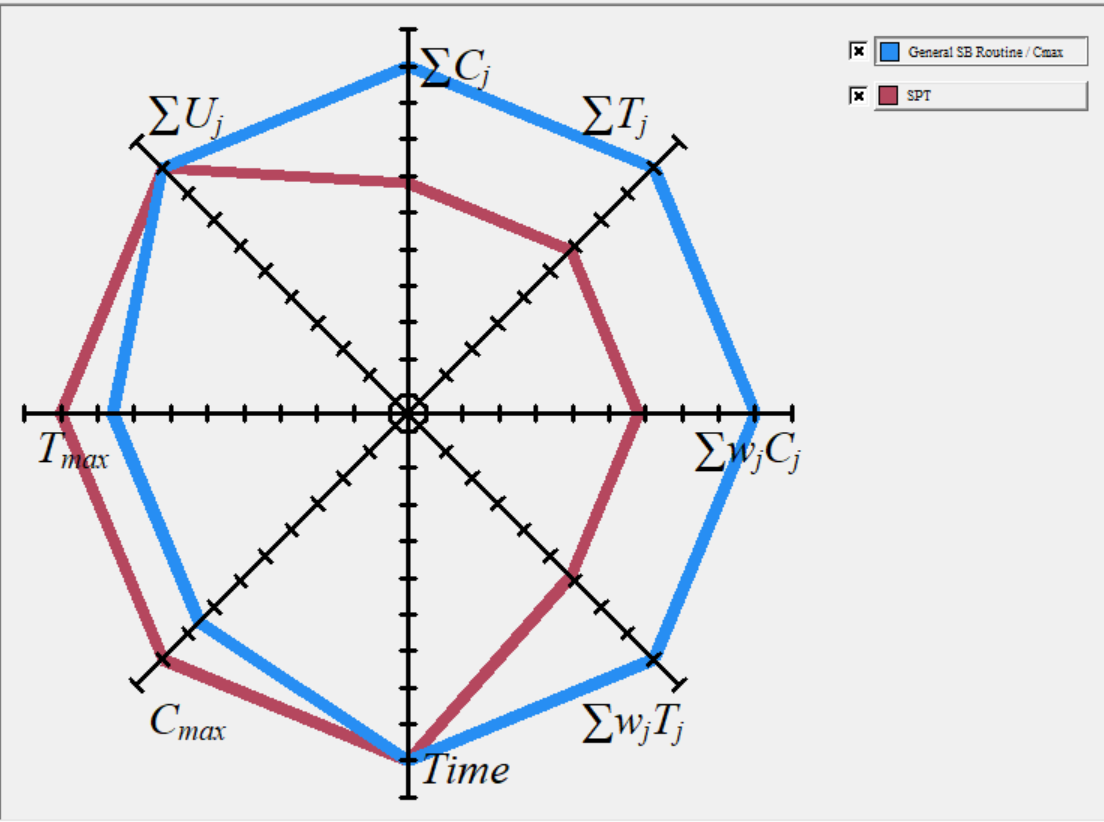


À la suite de l’utilisation de SPT (Shortest Processing Time) et de l’heuristique General SB Routine, nous remarquons une séquence de A : {P8, P2, P3, P14, P9, P13, P5, P6, P7, P1, P10, P12} et une séquence de B : {P14, P5, P6, P12, P10, P3, P4, P8, P2, P9, P11}.

**Diagramme de Gantt**

****

**Graphe avec mesures de performance**

****

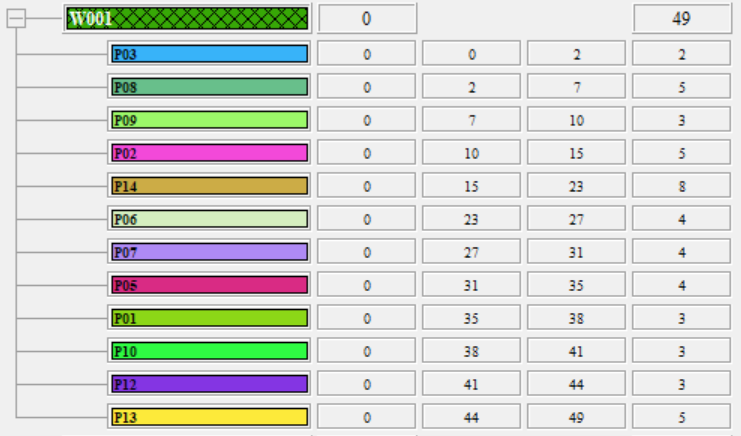
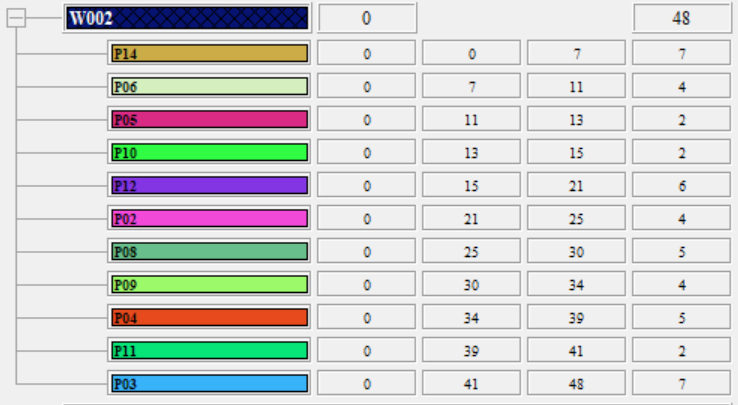
Solution 3 :

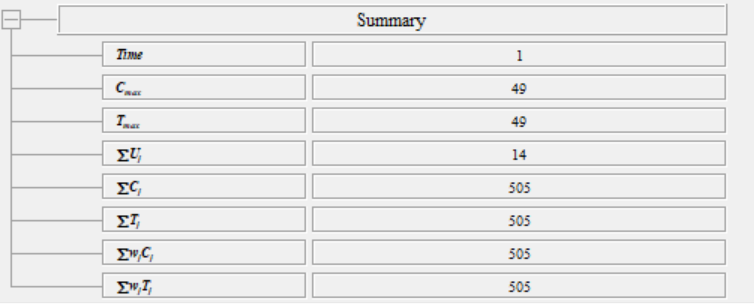
Voici l’ordonnancement avec FCFS (PEPES) First-Come First Serve avec l’heuristique Shifting Bottleneck/Tmax.

**Règle de priorité utilisé** : FCFS (PEPES) First-Come First Serve

**Heuristiques LEKIN**: Shifting Bottleneck/Tmax.

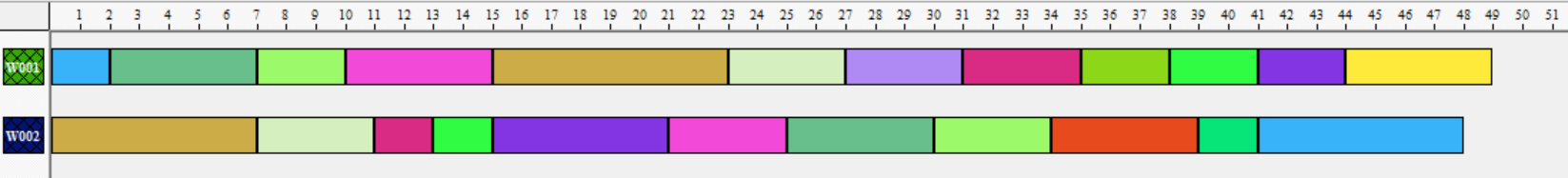
**Séquence obtenue** :

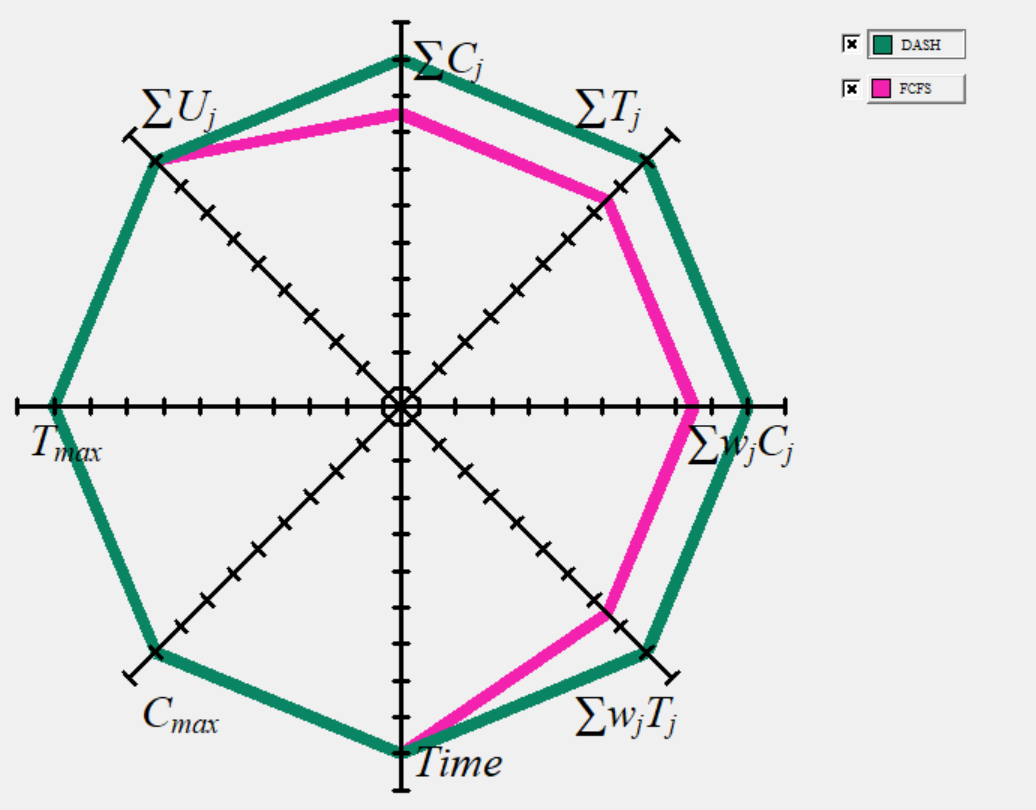


À la suite de l’utilisation de FCFS (First-Come First Serve) et de l’heuristique Shifting Bottleneck/Tmax, nous remarquons une séquence de A : {P3, P8, P9, P2, P14, P6, P7, P5, P1, P10, P12, P13 } et une séquence de B : {P14, P6, P5, P10, P12, P2, P8, P9, P4, P11, P3}.

**Diagramme de Gantt**

****

**Graphe avec mesures de performance**

****

**Justification**

Il serait intéressant de pointer que la solution 1 utilise l’algorithme de Jackson tandis que les solutions 2 et 3 emploient des heuristiques et des règles de priorités différentes. Pour la solution 1, nous avons employé le SPT et l’algorithme de Jackson. Ce dernier permet de minimiser le temps moyen dans le système en utilisant le Shortest Processing Time. La solution 2 emploie le SPT avec l’heuristique General SB Routine. Cette heuristique a été utilisé car nous voulons optimiser le temps moyen dans le système et le Makespan. Cet heuristique fonctionne dans notre cas car ce dernier optimise la majorité des objectifs dans un cas de Job Shop. Pour le scénario 3, nous avons employé la règle de priorité First-Come First Serve avec l’heuristique Shifting Bottleneck/Tmax. Ce dernier est utile car nous voulons optimiser le Makespan alors que le Tmax optmise à la fois le Maximum Tardiness mais aussi le Makespan. Il est donc pratique d’employer cette combinaison. Les trois solutions permettent à la machine A à terminer le plus tôt possible car comme nous pouvons le remarquer, il n’y a aucun endroit dans les 3 diagrammes de Gantt que la machine A prend une pause. De plus, nous remarquons que les 3 solutions offrent le même temps pour la machine A pour terminer les 14 tâches. Nous pouvons donc affirmer que c’est bel et bien le temps le plus tôt que la machine A peut terminer.