Ecole Centrale Casablanca

Facility and Work Design

Mini-Projet : Implémentation d'Approches par Programmation Linéaire Entière et Algorithmique pour l'Equilibrage de Lignes d'Assemblage

---

Réalisé par :
DIA Mohamadou ,
EL FAHSI Aymane,
Amar Serigne Mbaye Sy,
Tanoh Anet Fergus
Elèves-ingénieurs 2ème année -
Majeure "Modélisation et Industrie"

Encadrants :
Mr. RIANE Fouad,
Mme. JGHAMOU Afaf

Année Unniversitaire 2023-2024

Table des matières

1 Mise en contexte

Le travail consiste en une analyse comparative de deux approches différentes pour résoudre le problème d'équilibrage de lignes de production dans l'industrie textile, spécifiquement pour la fabrication de chemises et de pantalons. Les deux méthodes sont issues de recherches distinctes, l'une basée sur un modèle de programmation mathématique (IP) et l'autre sur une procédure algorithmique, utilisant le Ranked Positional Weight Method (RPWM).

Le premier article se penche sur l'équilibrage d'une ligne de production de pantalons en utilisant la méthode Ranked Positional Weight. L'étude empirique a été réalisée dans une installation de fabrication de vêtements chez Southern Range Nyanza Limited (NYTIL). Les résultats suggèrent que la méthode Ranked Positional Weight est plus adaptée pour l'équilibrage de lignes sans contraintes sur les ressources. Cependant, pour des lignes de production de vêtements complexes avec différents types de machines, cette méthode se révèle pratiquement inefficace. Ainsi, l'article souligne la nécessité d'explorer des approches plus avancées, telles que l'optimisation basée sur la simulation, pour améliorer l'efficacité des lignes de production complexes

Le second article se focalise sur l'équilibrage de la ligne de production de chemises au sein du département de couture d'une entreprise textile. Cette approche repose sur un modèle de programmation linéaire entière, conçu pour optimiser l'équilibre de la ligne tout en tenant compte de diverses contraintes comme le temps de cycle, les relations de précédence, l'adéquation des machines pour certaines tâches et la disponibilité des opérateurs. De plus, l'article procède à une comparaison entre les résultats obtenus via l'optimisation mathématique et ceux issus de la méthode Ranked Positional Weight Method (RPWM).

Pour simplifier notre analyse, nous avons entièrement mis en œuvre et testé l'approche RPWM décrite dans le premier article. En revanche, pour le second article, nous avons uniquement implémenté et testé la méthode de Programmation Linéaire en Nombres Entiers (PLNE) pour vérifier si nous pouvions reproduire les résultats présentés dans les deux modèles 1 et 7.

2 La méthode algorithmique de l'article Improvement of garment assembly line efficiency using line balancing technique

2.1 Contexte et approche de résolution

2.1.1 Approche Générale

L'approche adoptée dans cette recherche se focalise sur l'analyse détaillée des opérations de la chaîne d'assemblage, en évaluant les temps de travail normaux et standards pour chaque tâche. En utilisant ces données, l'étude a pu identifier des moyens d'optimiser la distribution des tâches et de réduire les délais inutiles tout en maintenant la qualité et l'efficacité de la production.

2.1.2 La méthode d'équilibrage de ligne par l'algorithme "Ranked Positional Weight"

L'article décrit l'application de l'algorithme de poids positionnel classé pour équilibrer la chaîne d'assemblage. Cette méthode algorithmique a été choisie pour sa capacité à fournir une répartition efficace des tâches en tenant compte de divers facteurs tels que le temps de cycle et les contraintes de précédence. L'algorithme a joué un rôle clé dans l'identification des goulots d'étranglement potentiels et la proposition de solutions pour améliorer l'efficacité globale.

2.1.3 Mesures de Performance

L'étude a évalué l'efficacité de la chaîne d'assemblage en utilisant des indicateurs tels que le nombre de postes de travail, le temps de cycle, l'efficacité de la ligne, le retard d'équilibre et l'indice de fluidité. Ces mesures ont permis d'apprécier l'impact des améliorations et d'identifier les ajustements nécessaires pour optimiser la ligne d'assemblage.

2.1.4 Les deux scénarios étudiés

Scénario 1 : Équilibrage de la Ligne avec Contraintes de Temps de Cycle et de Précédence
Dans ce scénario, l'équilibrage de la ligne d'assemblage a respecté les contraintes de temps de cycle et de précédence. L'organisation des tâches a été optimisée pour ne pas excéder le temps alloué par poste tout en maintenant l'ordre de précédence, visant ainsi à améliorer l'efficacité globale de la ligne.

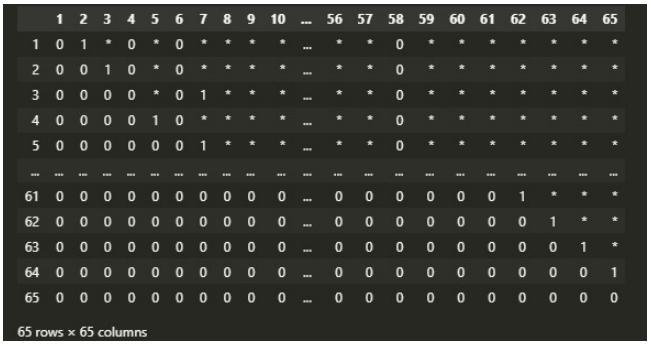
Scénario 2 : Équilibrage de la Ligne avec Contraintes Étendues
Le second scénario a pris en compte, en plus des contraintes du premier, les types de machines et les ressources disponibles. L'accent a été mis sur l'utilisation optimale des machines pour chaque tâche, sans compromettre le temps de cycle et l'ordre de précédence, pour une optimisation plus complète de la chaîne de montage.

2.2 Notre implémentation de l'approche algorithmique, son application et comparaison de nos résultats avec ceux de l'article

Le fichier `implementation_2.ipynb` contient notre implémentation et ses résultats, avec des graphes présentant l'assignement des taches aux différentes stations de travail.

Nous avons procédé à l'évaluation des performances de notre ligne d'assemblage. Pour ce faire, nous avons fixé le nombre de stations à 61 (n_stations) et calculé le temps de cycle (CT), basé sur une production quotidienne requise de 250 unités. Nous avons ensuite déterminé le nombre minimal de stations nécessaires (min_n_stations) en divisant le temps total de travail par le temps de cycle, et arrondi au nombre entier supérieur. Par la suite, le temps total d'inactivité (total_idle_time) a été calculé comme la différence entre le temps de cycle multiplié par le nombre de stations et le temps total de travail. En utilisant ces valeurs, nous avons évalué le retard d'équilibre (balance_delay) et l'efficacité de la ligne (line_efficiency), offrant ainsi une mesure quantitative clé de l'efficacité de notre modèle d'équilibrage.

Ensuite, nous avons transformé nos données initiales en DataFrame pour une analyse plus claire, en identifiant le nombre total de tâches. Ensuite, une matrice de précédence a été construite pour visualiser les relations entre les tâches. Cette matrice, mise à jour via une fonction personnalisée, reflète les relations directes et indirectes de précédence entre les tâches, fournissant une base pour planifier l'ordonnancement dans la chaîne d'assemblage. Cette matrice (figure 1) serait utilisé par l'algorithme de création de stations de travail dans la suite, comme fait dans l'article de base.



	1	2	3	4	5	6	7	8	9	10	...	56	57	58	59	60	61	62	63	64	65
1	0	1	*	0	*	0	*	*	*	*	...	*	*	0	*	*	*	*	*	*	*
2	0	0	1	0	*	0	*	*	*	*	...	*	*	0	*	*	*	*	*	*	*
3	0	0	0	0	*	0	1	*	*	*	...	*	*	0	*	*	*	*	*	*	*
4	0	0	0	0	1	0	*	*	*	*	...	*	*	0	*	*	*	*	*	*	*
5	0	0	0	0	0	0	1	*	*	*	...	*	*	0	*	*	*	*	*	*	*
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
61	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	*	*	*
62	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	*	*
63	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	*
64	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
65	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

65 rows × 65 columns

FIGURE 1 – Matrice de précédences

Nous avons créé un graphe dirigé pour illustrer les relations de précédence entre les tâches avant tout procédure algorithmique. En utilisant la bibliothèque NetworkX, chaque tâche a été ajoutée comme un nœud, et les liens de précédence, dérivés de notre matrice, ont été représentés sous forme d'arêtes. Le graphe a ensuite été visualisé avec un agencement et une taille spécifiques, permettant une compréhension visuelle immédiate des dépendances entre les tâches de notre scénario.

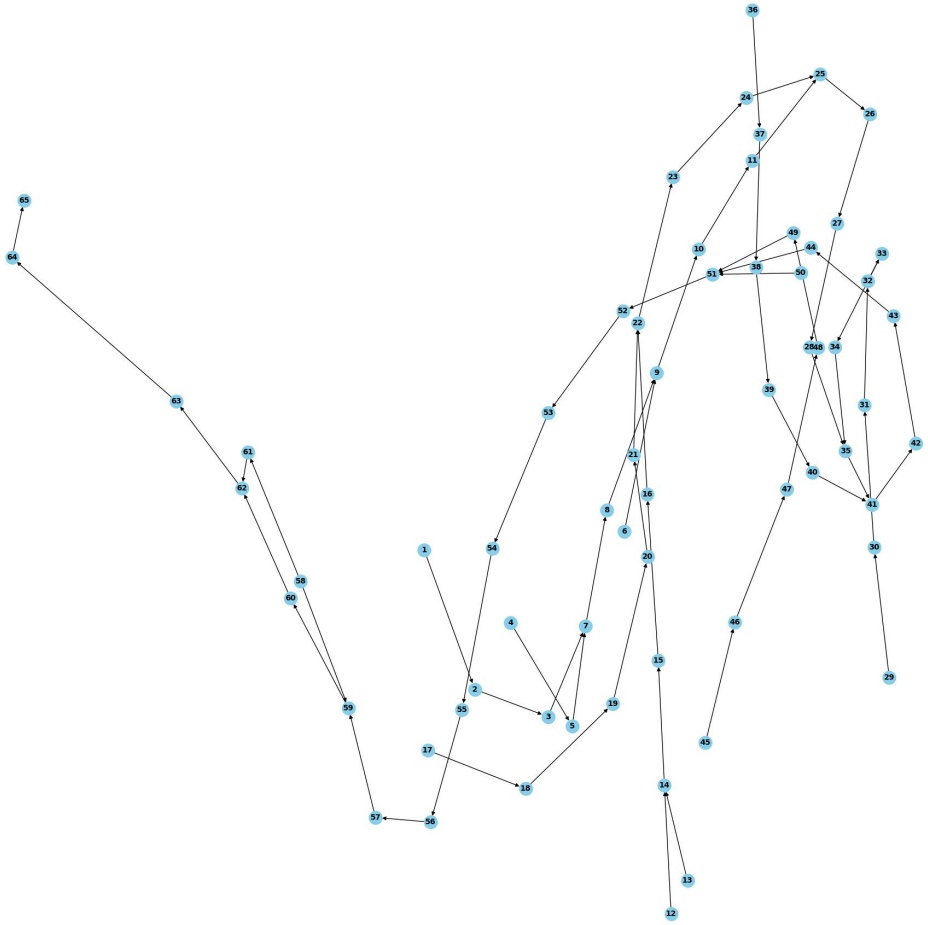


FIGURE 2 – Configuration initiale

Nous avons développé une fonction calculate_rpw pour calculer le Poids Positionnel Classé (RPW) de chaque tâche dans notre graphe. Cette fonction identifie tous les chemins possibles de chaque tâche jusqu'à la tâche finale et calcule la somme des temps standards le long de ces chemins, retenant la somme maximale. Ensuite, nous avons calculé le RPW pour chaque tâche du graphe et ajouté ces valeurs à notre DataFrame matrix$_{d}f.Pouruneutilisationfutureetuneanalysedétaillée$,

2.2.1 Notre implémentation du 1er scénario

Dans ce scénario, les contraintes liées au type de machine et aux opérateurs ne sont pas prises en compte.

Nous avons initialisé nos données dans data_scen1, regroupant les numéros des tâches, leurs relations de précédence, et les temps standards par tâche. Cette structuration préliminaire est essentielle pour analyser l'équilibrage de la ligne d'assemblage. En affichant le temps standard de la première tâche et en calculant le temps total nécessaire pour toutes les tâches, nous avons

établi une base pour l'évaluation des performances de notre modèle d'équilibrage.

Nous avons trié les tâches par leur Poids Positionnel Classé (RPW) en ordre décroissant, déterminant ainsi l'ordre d'assignation des tâches aux postes de travail. En utilisant le temps de cycle défini, nous avons élaboré une planification pour les postes de travail, en veillant à ce que l'ajout de chaque nouvelle tâche ne dépasse pas le temps de cycle imparti.

Un processus itératif a été mis en œuvre pour attribuer chaque tâche au poste de travail actuel ou à un nouveau poste de travail si le temps de cycle était dépassé. Cette méthode a permis de minimiser le nombre de postes de travail nécessaires tout en respectant le temps de cycle. Enfin, nous avons affiché le nombre de postes de travail requis après équilibrage pour le premier scénario, ainsi que la répartition des tâches pour chaque poste.

2.2.2 Notre implémentation du 2ème scénario

Dans le deuxième scénario, nous avons suivi une approche similaire à celle du premier, tout en introduisant de nouvelles données pour refléter les contraintes supplémentaires. Nous avons créé un autre dictionnaire data_scen2, comprenant les numéros des tâches, les relations de précédence, et les temps standards comme dans le premier scénario, mais avec l'ajout d'une nouvelle dimension : la ressource associée à chaque tâche. Cela inclut le type de machine nécessaire et la présence d'aide (par exemple, un assistant). Cette extension des données nous permet de prendre en compte des contraintes plus complexes et réalistes dans notre modèle d'équilibrage de ligne.

Pour le deuxième scénario, nous avons affiné notre méthode d'affectation des tâches aux postes de travail, en tenant compte des contraintes de ressources. Après avoir trié les tâches par leur RPW, nous avons initié notre planification des postes de travail (workstation_schedule2_b), en commençant par assigner la première tâche.

Nous avons également suivi le statut de chaque tâche (status) pour savoir si elle était déjà assignée. Pour chaque tâche suivante, nous avons cherché des postes de travail compatibles avec les contraintes de ressources de la tâche, en veillant à ne pas dépasser le temps de cycle. Si aucun poste compatible n'était disponible, ou si l'ajout d'une nouvelle tâche dépassait le temps de cycle, nous avons créé un nouveau poste de travail.

Ce processus itératif a assuré que chaque tâche était assignée à un poste de travail compatible avec les contraintes de ressources, optimisant ainsi la distribution des tâches tout en respectant les contraintes de cycle et de ressources.

2.2.3 Nos résultats et comparaison avec les résultats de l'article

Workstation Number	Tasks Assigned
1	13, 1
2	2, 3, 12, 4
3	14, 5, 17, 7, 18
4	15, 19, 16
5	8, 20, 6, 21
6	9, 29, 22
7	30, 10
8	23, 11, 24, 31
9	25, 32
10	26
11	33
12	34
13	27, 28, 36
14	35, 37, 38
15	39, 40
16	41
17	42
18	43, 45, 46, 44, 47
19	48, 50, 49
20	51
21	52
22	53
23	54
24	55, 56, 57, 58
25	59, 60
26	61, 62, 63
27	64, 65

TABLE 1 – Postes de travail et tâches associées - scénario 1

Workstation ID	Type	Tasks Assigned
1	Helper	13, 12, 17, 45
2	BH	1, 18, 37, 49, 60
3	nO/L	2, 5, 16, 6
4	SN/L	3, 14, 19
5	F/L	4
6	SN/L	7, 8, 21, 38
7	AWM	15
8	TM	20, 39, 47
9	SN/L	9, 22, 40
10	Helper	29, 11, 24, 25, 28, 58
11	SN/L	30, 10
12	SN/L	23, 31, 46, 48, 61
13	nO/L	32, 26
14	SN/L	33
15	Iron press	34
16	F/A	27, 57
17	Helper	36, 56, 63
18	SN/L	35
19	SN/L	41
20	BT	42
21	nO/L	43, 55
22	DN/L	44
23	LM	50
24	SN/L	51
25	SN/L	52, 62
26	SN/L	53
27	SN/L	54
28	SN/L	59
29	SN/L	64
30	BT	65

TABLE 2 – Postes de travail et tâches associées - scénario 2

Nous avons construit un graphe dirigé pour représenter l'affectation des tâches aux postes de travail après l'équilibrage, en visualisant les relations de précédence entre eux. Chaque nœud représentait un poste de travail avec ses tâches assignées. Des arêtes ont été ajoutées pour illustrer les dépendances entre les tâches de postes différents. Cette visualisation a clairement montré la distribution des tâches et la structure de précédence dans notre modèle d'équilibrage pour les deux scénarios post-traitement.

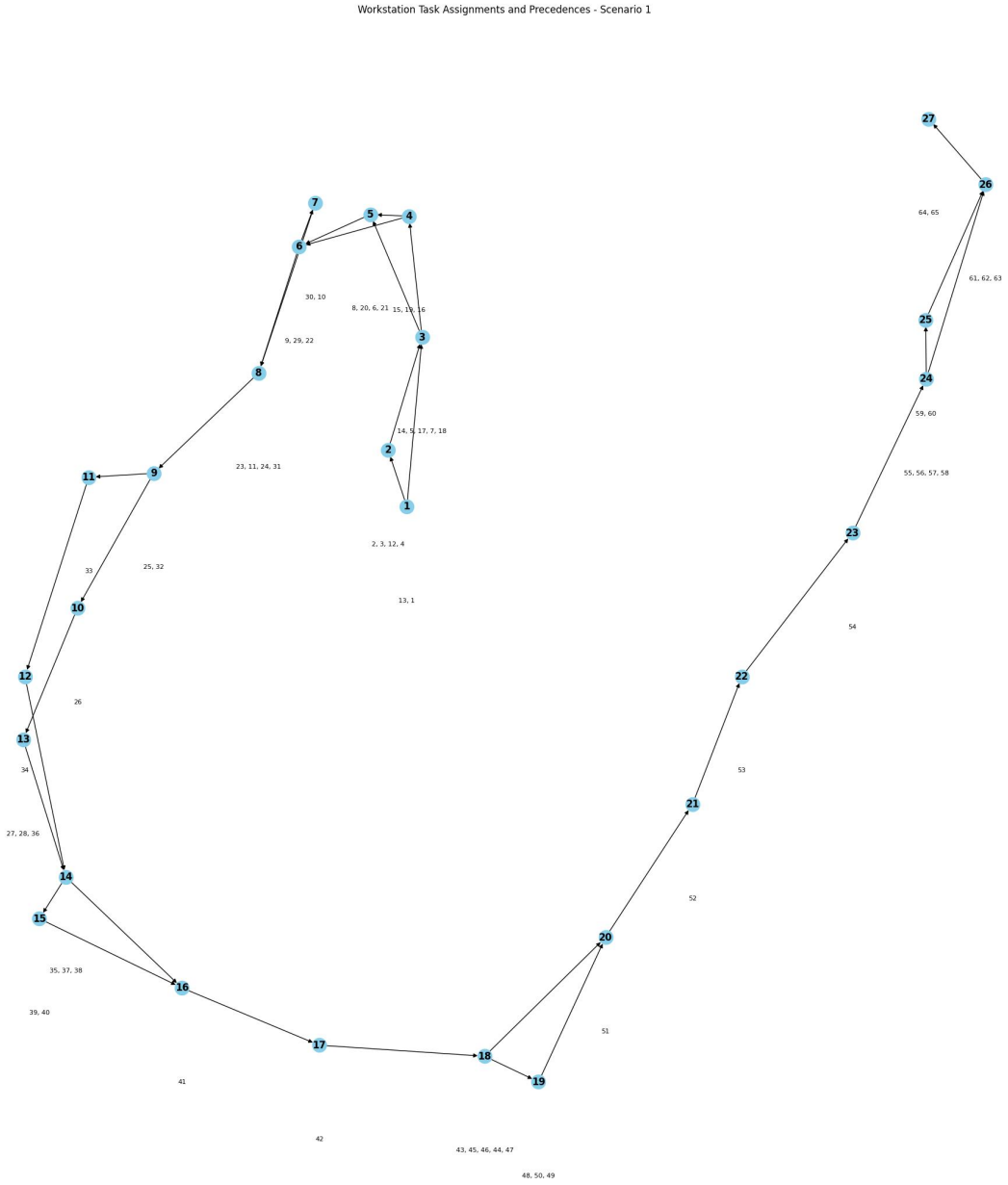


FIGURE 3 – Scénario 1 optimisé par méthode RPW

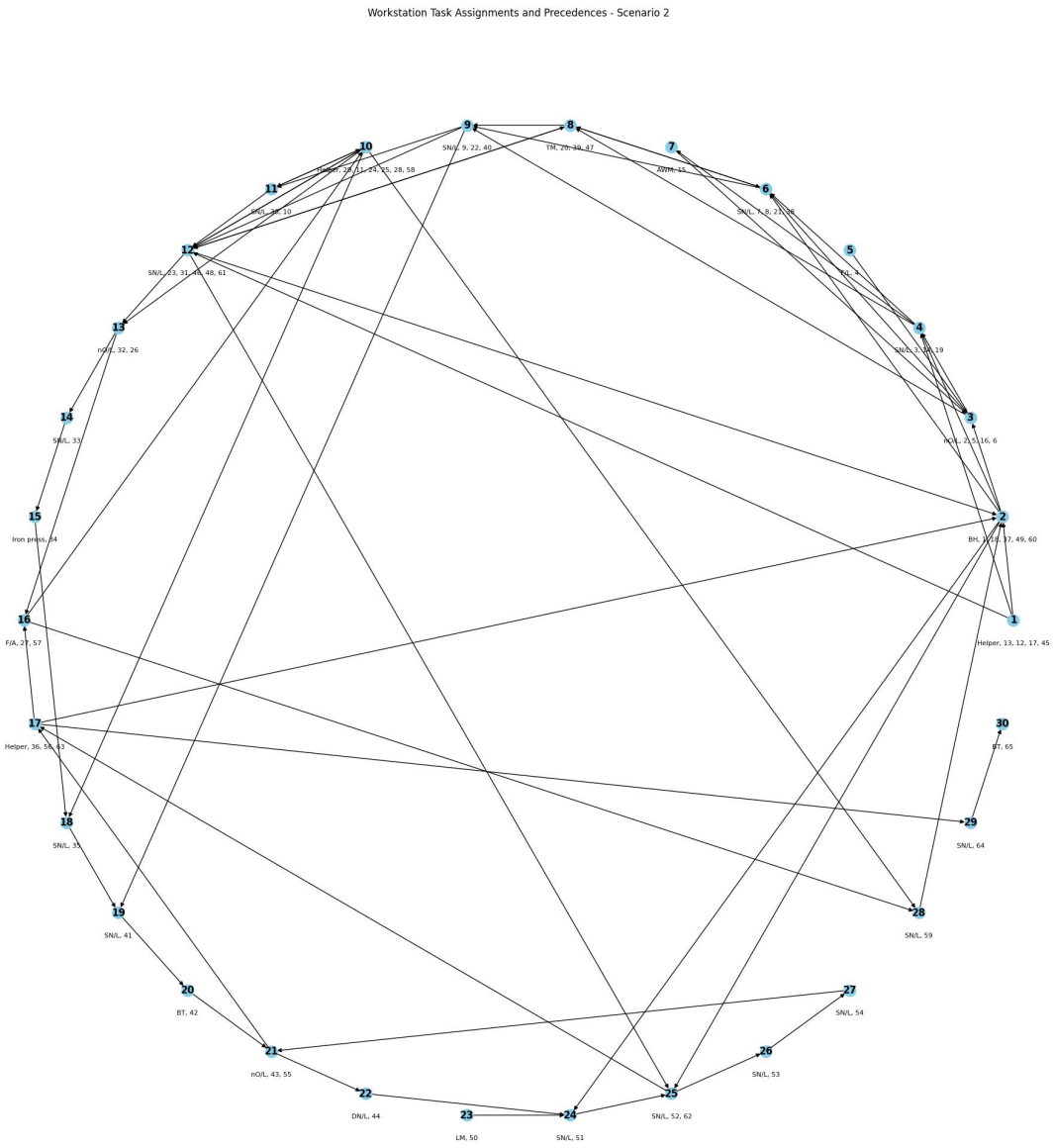


FIGURE 4 – Scénario 2 optimisé par méthode RPW

Voici un tableau récapitulatif qui compare les performances des deux scénarios sur la base de plusieurs métriques clés. Ces métriques incluent le nombre de postes de travail, l'indice de fluidité, l'efficacité de la ligne et le retard d'équilibre, permettant ainsi une évaluation quantitative des résultats de notre modèle d'équilibrage pour chaque scénario :

Les résultats présentés dans le tableau montrent des différences notables entre les scénarios 1 et 2, en termes d'efficacité et d'équilibrage de la ligne d'assemblage. Le nombre de postes de travail augmente de 27 à 30 en passant du scénario 1 au scénario 2, indiquant une augmentation de la complexité et des ressources nécessaires pour gérer les contraintes supplémentaires. L'indice de fluidité passe de 2.54 à 4.03, révélant une distribution moins uniforme des charges de travail entre les postes dans le scénario 2.

Metric	Scénario 1	Scénario 2
Number of Workstations	27	30
Smoothness Index	2.54	4.03
Line Efficiency(%)	80.56	72.50
Balance Delay(%)	19.44	27.49
Total Idle Time(min)	10.08	15.84

TABLE 3 – Metrics for Scenario 1 and Scenario 2

Intéprétation des résultas :
L'efficacité de la ligne diminue de 80.56% à 72.50%, ce qui montre une réduction de l'optimisation globale sous des contraintes plus strictes. Le retard d'équilibre augmente également, passant de 19.% à 7.49%, signalant une hausse du temps inutilisé et une moins bonne exploitation des postes de travail. En outre, le temps total d'inactivité augmente, passant de 10.08 minutes à 15.84 minutes, ce qui confirme une perte d'efficacité dans le scénario 2. Ces résultats suggèrent que les contraintes supplémentaires du scénario 2 ont un impact négatif significatif sur l'équilibrage et l'efficacité de la ligne d'assemblage.

Compraison de nos résultats avec ceux de l'article :
Nous constatons une légère différence entre nos résultats et ceux de l'article, probablement du à l'exploration du graphe initiale de prédecession pour le calcul des poids RPW. Néanmoins, nous constatons une très forte ressemblance entre nos résultats et ceux présentés dans l'article, nos conclusions se concordants avec ceux des auteurs de l'article.

3 La méthode mathématique de l'article Balancing The Shirt Production Line Under Different Operational Constraints Using An Integer Programming Model"

3.1 Contexte de l'application dans l'article et l'approche de résolution

3.1.1 Le contexte

L'article propose une approche avancée pour l'équilibrage de la ligne de production de chemises. Nous allons examiner en détail les deux approches comparées, le 'Modèle 1' et le 'Modèle 7', et explorer notre implémentation basée sur la méthode de Programmation Linéaire en Nombres Entiers (PLNE) uniquement par contrainte de temps.

Les aspects opérationnels tels que les temps de traitement des tâches et les capacités des postes de travail sont pris en compte pour optimiser la production.

3.1.2 Les deux approches comparées dans l'article

L'article introduit une perspective comparative en examinant deux approches distinctes. Nous nous contentons de comparer les deux modèles, le "Model 1" et le "Model 7", le modèle 7 étant le modèle le plus affiné en allant de 1 à 7 pour une optimisation plus avancée et complexe.

3.1.3 1ère situation : "Model 1" dans l'article

Le "Model 1" cherche à minimiser le nombre total de postes de travail ouverts, une stratégie traditionnelle mais néanmoins efficace.
Les contraintes du "Model 1" garantissent une allocation appropriée des tâches tout en respectant les capacités des postes de travail.

3.1.4 2ème situation : "Model 7" dans l'article

L'intégration du "Model 7" dans notre implémentation étend la modélisation au-delà des approches classiques. Cette deuxième situation, inspirée de l'article, repose sur une formulation mathématique sophistiquée pour prendre en compte les écarts entre les temps d'exécution des tâches dérivées. Explorons en détail cette extension et son impact sur la résolution du problème.

3.2 Notre implémentation de l'approche "Multiple Integer Programming" son application et comparaison de nos résultats avec ceux de l'article

Nous avons fais la transition du code LINGO au code Python dans la fin de l'article de base, et nous avons utilisé le module Gurobipy de Gurobi Optimization avec licence académique vu la complexité du problème d'optimisation linéaire entière.

Le code est dûment commenté et les résultats se trouvent dans le notebook `implementation_1.ipynb`.

3.2.1 Le modèle simplifié : "Model 1" dans l'article

L'implémentation du "Model 1" dans notre étude représente la première situation analysée dans l'article, visant à minimiser le nombre total de postes de travail ouverts. Cette approche, bien que classique, est utile pour établir une base de comparaison solide avec des stratégies plus complexes telles que le "Model 7".

3.2.2 le modèle affiné complexe : "Model 7" dans l'article

Le "Model 7" introduit des variables supplémentaires pour capturer les écarts entre les temps d'exécution des tâches dérivées. La fonction objective du modèle s'ajuste en conséquence, visant à minimiser ces écarts tout en tenant compte des contraintes opérationnelles.

3.2.3 Nos résultats et comparaison avec les résultats de l'article

Résultats pour le modèle 1 :

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (win64 - Windows 10.0 (19045.2))

CPU model: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 4 logical processors, using up to 4 threads

Optimize a model with 2643 rows, 1344 columns and 52812 nonzeros
Model fingerprint: 0xec4cef50
Variable types: 0 continuous, 1344 integer (1152 binary)
Coefficient statistics:
  Matrix range     [4e-02, 3e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 4e+01]
Presolve removed 1921 rows and 714 columns
Presolve time: 0.15s
Presolved: 722 rows, 630 columns, 3771 nonzeros
Variable types: 0 continuous, 630 integer (583 binary)

Root relaxation: objective 2.373933e+01, 1235 iterations, 0.04 seconds (0.02 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   23.73933    0   84          -   23.73933      -     -    0s
     0     0   23.79125    0  159          -   23.79125      -     -    0s
...
x_40_22: 0.0
x_40_23: 0.0
x_40_24: 0.0
Objective Value for model 1: 30
```

FIGURE 5 – Optimisation du modèle 1

Le résultat de l'optimisation est 30 stations de travail ouvertes, contre 36 mentionnés comme résultat dans l'article (table 10).

Résultats pour le modèle 7 :

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (win64 - Windows 10.0 (19045.2))

CPU model: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 4 logical processors, using up to 4 threads

Optimize a model with 2643 rows, 1344 columns and 52812 nonzeros
Model fingerprint: 0xaf404b71
Variable types: 0 continuous, 1344 integer (1152 binary)
Coefficient statistics:
  Matrix range     [4e-02, 3e+01]
  Objective range  [1e-01, 5e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 4e+01]

Loaded MIP start from previous solve with objective 35

Presolve removed 1921 rows and 714 columns
Presolve time: 0.12s
Presolved: 722 rows, 630 columns, 3771 nonzeros
Variable types: 0 continuous, 630 integer (583 binary)

Root relaxation: objective 2.578728e+01, 848 iterations, 0.02 seconds (0.02 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
...
x_40_22: 0.0
x_40_23: 0.0
x_40_24: 0.0
Objective Value for model 7: 33
```

FIGURE 6 – Optimisation du modèle 7

Le résultat de l'optimisation est 33 stations de travail ouvertes, excactement égal au résultat de l'article (table 10).

References

- Bongomin O, Mwasiagi JI, Nganyi EO, Nibikora I. Improvement of garment assembly line efficiency using line balancing technique. Engineering Reports. 2020 ;2 :e12157. Article 2

- Şeyda Topaloğlu Yıldız, Gülseren Karabay. Balancing The Shirt Production Line Under Different Operational Constraints Using An Integer Programming Model. TEKSTİL VE KONFEKSİYON. VOL : 32, NO. 4. Article 1