Rapport Technique : Planificateur de Quarts de Travail

Vue d'ensemble de l'Application

Description Générale

Le **Planificateur de Quarts de Travail** est une application d'optimisation intelligente des horaires du personnel pour le commerce de détail. Elle utilise l'optimisation mathématique (programmation linéaire en nombres entiers) via le solveur **Gurobi** pour générer automatiquement des plannings optimaux qui minimisent les coûts de main-d'œuvre tout en respectant les contraintes opérationnelles et les besoins en personnel.

Objectif Principal

L'application résout le problème complexe de la planification des horaires en :

• **Minimisant les coûts** de main-d'œuvre
• **Maximisant la couverture** des besoins en personnel
• **Respectant les contraintes** légales et opérationnelles
• **Optimisant l'utilisation** des ressources humaines

Fonctionnalités de l'Interface

1. **Onglet "Employés" (Gestion du Personnel)**

Fonctionnalités :

• **Ajout d'employés** : Nom, taux horaire, heures maximales par jour/semaine
• **Gestion des disponibilités** : Définition des heures où chaque employé peut travailler
• **Compétences** : Attribution de compétences spécifiques (Caisse, Stock, Manager, etc.)
• **Visualisation** : Tableau récapitulatif avec toutes les informations des employés

Données stockées pour chaque employé :

• Identifiant unique
• Nom
• Taux horaire (salaire)
• Heures maximales par jour (par défaut : 8h)

• Heures maximales par semaine (par défaut : 40h)
• Disponibilités horaires (ensemble d'heures de 0 à 23)
• Liste de compétences

2. **Onglet "Demande" (Configuration de la Demande Client)**

Fonctionnalités :

• **Configuration du magasin** :
• Heures d'ouverture et de fermeture
• Ratio personnel/client (ex: 0.05 = 1 employé pour 20 clients)
• Personnel minimum requis par heure

• **Visualisation graphique** : Graphique en barres montrant la demande horaire attendue

• **Motifs prédéfinis** :
• **Plat** : Demande constante toute la journée
• **Pic matinal** : Pic d'affluence le matin
• **Pic déjeuner** : Pic entre 11h-13h
• **Pic soirée** : Pic en fin de journée
• **Bimodal** : Deux pics (déjeuner + soirée)
• **Week-end** : Pattern typique du week-end

• **Contrôles manuels** : Sliders et spinboxes pour ajuster la demande heure par heure

• **Outils** :
• Échelle x2 : Double la demande
• Échelle ÷2 : Divise la demande par 2
• Réinitialiser : Remet toutes les valeurs à zéro

Calcul automatique : Pour chaque heure, le système calcule le nombre d'employés requis :

```
Employés requis = max( (Demande clients × Ratio), Personnel minimum )
```

3. **Onglet "Planification" (Optimisation et Résultats)**

Fonctionnalités :

Paramètres d'optimisation :

• **Objectif** :
• **Minimiser le coût** : Optimise pour réduire les coûts de main-d'œuvre
• **Maximiser la couverture** : Optimise pour maximiser le nombre d'employés présents

• **Contraintes de quarts** :
• Durée minimale d'un quart (par défaut : 4h)
• Durée maximale d'un quart (par défaut : 8h)
• Autoriser les heures supplémentaires (optionnel)

• **Temps limite** : Limite de temps pour le calcul (par défaut : 60 secondes)

Résultats de l'optimisation :

1. **Graphique Gantt** : Visualisation des horaires de chaque employé sous forme de barres temporelles
• Axe horizontal : Heures de la journée (8h à 20h)
• Axe vertical : Liste des employés planifiés
• Barres bleues : Périodes de travail avec heures affichées

2. **Tableau récapitulatif** :
• **Employé** : Nom de l'employé
• **Horaires** : Liste des quarts (ex: "8:00-12:00, 14:00-18:00")
• **Heures Total** : Nombre total d'heures travaillées
• **Coût** : Coût total pour cet employé

3. **Statistiques du planning** :
• Nombre d'employés planifiés / Total
• Heures totales de travail
• Moyenne d'heures par employé
• Analyse de couverture :
• Heures avec couverture parfaite
• Heures sous-effectif (manque de personnel)
• Heures sur-effectif (trop de personnel)
• Coût total du planning

Mécanisme d'Optimisation Détaillé

Architecture de l'Algorithme

L'application utilise la **Programmation Linéaire en Nombres Entiers (PLNE)** via Gurobi pour résoudre un problème d'optimisation combinatoire.

Variables de Décision

Le modèle définit trois types de variables binaires (0 ou 1) :

1. Variables `x[e, h]` (Travail horaire)
• **Définition** : `x[e, h] = 1` si l'employé `e` travaille à l'heure `h`, sinon `0`
• **Type** : Variable binaire
• **Exemple** : `x[1, 10] = 1` signifie que l'employé #1 travaille à 10h

2. Variables `y[e, h]` (Début de quart)
• **Définition** : `y[e, h] = 1` si l'employé `e` commence un quart à l'heure `h`, sinon `0`
• **Type** : Variable binaire

• **Utilité** : Permet de garantir la continuité des quarts

3. Variables `s[h]` (Effectif par heure)

• **Définition** : Nombre total d'employés travaillant à l'heure `h`
• **Type** : Variable entière
• **Calcul** : `s[h] = Σ x[e, h]` pour tous les employés `e`

4. Variables de déviation (contraintes souples)

• `shortage[h]` : Nombre d'employés manquants à l'heure `h` (si sous-effectif)
• `surplus[h]` : Nombre d'employés en excès à l'heure `h` (si sur-effectif)

Contraintes du Modèle

1. **Comptage du personnel par heure**

```
s[h] = Σ x[e, h]  pour tous les employés e
```

Garantit que `s[h]` représente bien le nombre total d'employés présents.

2. **Couverture de la demande (contrainte souple)**

```
s[h] + shortage[h] - surplus[h] = Demande_requise[h]
```

• Si `s[h] < Demande_requise[h]` → `shortage[h] > 0` (pénalité)
• Si `s[h] > Demande_requise[h]` → `surplus[h] > 0` (pénalité légère)
• Permet au modèle de trouver une solution même si la demande ne peut pas être parfaitement satisfaite

3. **Disponibilité des employés**

```
Si employé e non disponible à l&#x27;heure h :
    x[e, h] = 0
```

Respecte les contraintes de disponibilité de chaque employé.

4. **Limite d'heures par jour**

```
Σ x[e, h] ≤ max_hours_per_day  pour chaque employé e
```

Empêche un employé de dépasser ses heures maximales quotidiennes.

5. **Continuité des quarts (durée minimale)**

```
Si y[e, h] = 1 (début de quart à l&#x27;heure h) :
    Alors x[e, h], x[e, h+1], ..., x[e, h+min_shift_length-1] = 1
```

Garantit qu'un quart dure au moins `min_shift_length` heures consécutives.

6. **Un seul début de quart par employé**

```
Σ y[e, h] ≤ 1  pour chaque employé e
```

Chaque employé ne peut commencer qu'un seul quart par jour (mais peut avoir plusieurs quarts séparés).

7. **Limite de durée maximale d'un quart**

```
Σ x[e, h] ≤ max_shift_length  pour chaque employé e
```

Un quart ne peut pas dépasser `max_shift_length` heures.

Fonction Objectif

Le modèle peut optimiser selon deux objectifs :

A. **Minimiser le Coût** (Objectif principal)

```
Minimiser :
    Coût_main_d&#x27;œuvre + Pénalité_shortage + Pénalité_surplus

Où :
    Coût_main_d&#x27;œuvre = Σ (x[e, h] × taux_horaire[e])  pour tous e, h
    Pénalité_shortage = 1000 × Σ shortage[h]  (pénalité élevée)
    Pénalité_surplus = 10 × Σ surplus[h]  (pénalité légère)
```

Stratégie :

• Minimise d'abord le coût total de main-d'œuvre

• Pénalise fortement les manques de personnel (facteur 1000)

• Pénalise légèrement le sur-effectif (facteur 10)

B. **Maximiser la Couverture**

```
Maximiser :
    1000 × Σ s[h] - Coût_main_d&#x27;œuvre

Où :
    Σ s[h] = Nombre total d&#x27;heures-personnel
```

Stratégie :

• Priorise la maximisation du nombre d'employés présents

• Minimise le coût comme objectif secondaire

Processus de Résolution

1. **Construction du modèle** :

• Création des variables de décision

• Ajout de toutes les contraintes

• Définition de la fonction objectif

2. **Résolution avec Gurobi** :

• Gurobi utilise des algorithmes avancés (Branch-and-Bound, Cutting Planes)

• Recherche de la solution optimale ou meilleure solution dans le temps limite

• Retourne le statut : Optimal, Time Limit Reached, Infeasible, etc.

3. **Extraction de la solution** :

• Lecture des valeurs des variables `x[e, h]`

• Regroupement des heures consécutives en quarts

• Calcul des statistiques (coûts, heures, couverture)

Exemple Concret

Scénario :

• 5 employés avec taux horaires différents
• Demande : Pic à 12h (85 clients), autres heures (40 clients)
• Ratio : 0.05 employé/client
• Personnel minimum : 1

Calcul de la demande :

• Heure 12h : `max(85 × 0.05, 1) = max(4.25, 1) = 5 employés requis`
• Autres heures : `max(40 × 0.05, 1) = max(2, 1) = 2 employés requis`

Solution optimale :

Le modèle trouve automatiquement :

• Quels employés assigner à quelles heures
• Comment former des quarts continus de 4-8h
• Comment minimiser le coût total
• Comment respecter toutes les disponibilités

Résultat :

• Planning avec coût minimal
• Couverture de la demande maximale
• Respect des contraintes légales et opérationnelles

Avantages de cette Approche

1. **Optimisation Mathématique Rigoureuse**

• Garantit la meilleure solution possible (ou proche de l'optimum)
• Prend en compte toutes les contraintes simultanément
• Évite les solutions sous-optimales des méthodes manuelles

2. **Flexibilité**

• Paramètres ajustables (durée min/max, objectifs)
• Gestion des contraintes souples (shortage/surplus)
• Support des heures supplémentaires optionnelles

3. **Efficacité**

• Résolution rapide (généralement < 1 seconde pour des problèmes moyens)
• Gestion de problèmes avec plusieurs dizaines d'employés
• Limite de temps configurable

4. **Visualisation et Analyse**

• Graphique Gantt pour visualiser les horaires
• Statistiques détaillées sur la couverture
• Calcul automatique des coûts

Cas d'Utilisation

Commerce de Détail

• Magasins avec variations de demande horaire
• Optimisation des coûts de main-d'œuvre
• Respect des contraintes légales (heures max)

Restaurants

• Gestion des pics de service (déjeuner, dîner)
• Optimisation des équipes de cuisine et service
• Réduction des coûts opérationnels

Services Clients

• Centres d'appels avec demande variable
• Optimisation des équipes selon les pics d'appels
• Réduction des temps d'attente clients

Technologies Utilisées

• **PyQt6** : Interface graphique moderne et responsive
• **Gurobi Optimizer** : Solveur de programmation mathématique de pointe
• **Python 3.8+** : Langage de programmation
• **Architecture MVC** : Séparation claire des responsabilités

Conclusion

Le Planificateur de Quarts de Travail est une application complète qui transforme un problème complexe de planification en un processus automatisé et optimisé. En utilisant l'optimisation mathématique, elle garantit des solutions optimales qui minimisent les coûts tout en respectant toutes les contraintes opérationnelles et légales. L'interface intuitive permet aux gestionnaires de configurer facilement leurs paramètres et d'obtenir des plannings optimaux en quelques secondes.

Rapport généré automatiquement - Planificateur de Quarts de Travail v1.0