Système de Gestion d'Événements avec Résolution de Conflits Temporels

AmrDroid

12 décembre 2025

Table des matières

1 Introduction

Ce rapport présente un système de gestion d'événements développé avec PyQt6 qui intègre un solveur d'optimisation pour résoudre les conflits de temps entre événements. Le système permet aux utilisateurs de créer, gérer et organiser des événements quotidiens tout en assignant automatiquement des classes non-conflictuelles aux événements qui se chevauchent.

2 Interface Utilisateur (UI)

2.1 Architecture de l'Interface

L'interface utilisateur est divisée en deux sections principales :

— **Menu latéral gauche** : Navigation temporelle (aujourd'hui, hier, demain, etc.) et accès au calendrier

— **Corps principal** : Affichage de la date courante, table des événements, et boutons d'action

2.2 Fonctionnalités Principales

1. **Navigation temporelle** : Boutons pour naviguer rapidement entre les jours
2. **Calendrier interactif** : Sélection de dates via une interface calendrier moderne
3. **Gestion d'événements** :
   — Ajout d'événements avec nom et durée (heure début → heure fin)
   — Suppression d'événements sélectionnés
   — Suppression de tous les événements d'une date
4. **Résolution automatique de conflits** : Bouton "Refresh" qui lance l'algorithme d'optimisation

2.3 Design et Accessibilité

L'interface utilise un thème à contraste élevé avec :

— Palette de couleurs accessible (bleu #0b61d8, vert #16a34a, orange #ff7a00, rouge #ef4444)

— Typographie claire (Segoe UI, tailles 10-18px)

— Indicateurs de focus pour la navigation au clavier

— Boutons avec curseur pointeur et effets de survol

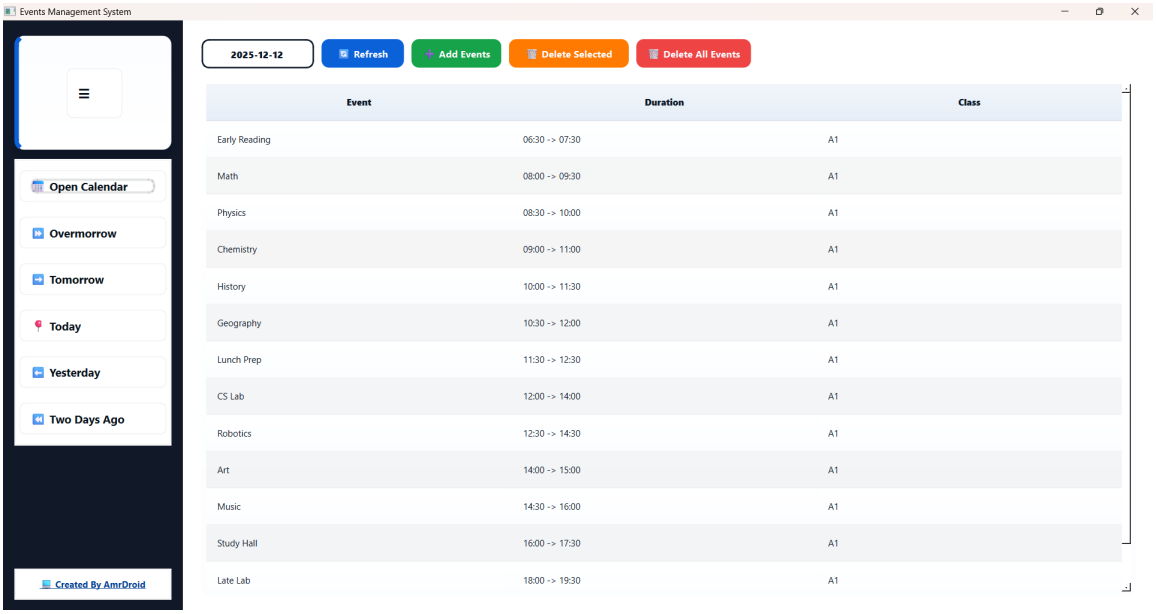

FIGURE 1 – Interface principale du système de gestion d'événements

3 Modélisation Mathématique

3.1 Problème de Coloration de Graphe

Le problème de résolution de conflits temporels est modélisé comme un **problème de coloration de graphe optimal**.

3.1.1 Définitions

Soit :
- $V = \{v_1, v_2, \ldots, v_n\}$ l'ensemble des événements
- $t_i = (s_i, e_i)$ la durée de l'événement v_i avec s_i = heure de début et e_i = heure de fin
- K le nombre maximum de couleurs (classes) disponibles, où $K = |V|$

3.1.2 Graphe de Conflits

Construisons un graphe $G = (V, E)$ où :

$$(v_i, v_j) \in E \iff \text{les événements } v_i \text{ et } v_j \text{ se chevauchent} \tag{1}$$

Deux événements se chevauchent si et seulement si :

$$(s_i < e_j) \wedge (s_j < e_i) \tag{2}$$

3.1.3 Formulation en Programmation Linéaire en Nombres Entiers

Variables de décision :

$$x_{vk} \in \{0,1\} \quad \forall v \in V, k \in \{0,1,\ldots,K-1\} \tag{3}$$

$$x_{vk} = 1 \text{ si l'événement } v \text{ reçoit la couleur } k \tag{4}$$

$$y_k \in \{0,1\} \quad \forall k \in \{0,1,\ldots,K-1\} \tag{5}$$

$$y_k = 1 \text{ si la couleur } k \text{ est utilisée} \tag{6}$$

Fonction objectif :

$$\min \sum_{k=0}^{K-1} y_k \tag{7}$$

Contraintes :

1. Chaque événement reçoit exactement une couleur :

$$\sum_{k=0}^{K-1} x_{vk} = 1 \quad \forall v \in V \tag{8}$$

2. Liaison entre x et y :

$$x_{vk} \leq y_k \quad \forall v \in V, k \in \{0,1,\ldots,K-1\} \tag{9}$$

3. Événements adjacents (conflictuels) ne peuvent pas partager la même couleur :

$$x_{uk} + x_{vk} \leq y_k \quad \forall (u,v) \in E, k \in \{0,1,\ldots,K-1\} \tag{10}$$

3.2 Résolution avec Gurobi

Le modèle est résolu en utilisant Gurobi Optimizer, un solveur commercial de programmation linéaire en nombres entiers qui garantit l'optimalité de la solution.

4 Exemple Pratique

4.1 Données d'Entrée

Considérons une journée avec 5 événements :

Événement	**Début**	**Fin**
Réunion A	08 :00	10 :00
Cours B	09 :00	11 :00
Présentation C	10 :30	12 :00
Déjeuner D	12 :00	13 :00
Atelier E	09 :30	10 :30

TABLE 1 – Événements avec leurs durées

4.2 Construction du Graphe de Conflits

Conversion en valeurs numériques :

$$\text{Réunion A :} \quad [8.0, 10.0] \tag{11}$$
$$\text{Cours B :} \quad [9.0, 11.0] \tag{12}$$
$$\text{Présentation C :} \quad [10.5, 12.0] \tag{13}$$
$$\text{Déjeuner D :} \quad [12.0, 13.0] \tag{14}$$
$$\text{Atelier E :} \quad [9.5, 10.5] \tag{15}$$

Détection des chevauchements :
— A et B se chevauchent : $(8.0 < 11.0) \wedge (9.0 < 10.0)$
— A et E se chevauchent : $(8.0 < 10.5) \wedge (9.5 < 10.0)$
— B et C se chevauchent : $(9.0 < 12.0) \wedge (10.5 < 11.0)$
— B et E se chevauchent : $(9.0 < 10.5) \wedge (9.5 < 11.0)$
— C et E se chevauchent : $(10.5 = 10.5)$ (juste adjacent)
— C et D ne se chevauchent pas : $(12.0 \not< 12.0)$
Arêtes du graphe :

$$E = \{(A, B), (A, E), (B, C), (B, E)\} \tag{16}$$

4.3 Résolution Optimale

Le solveur Gurobi trouve la solution optimale avec **3 couleurs** :

Événement	**Couleur**	**Classe**
Réunion A	0	A1
Cours B	1	A2
Présentation C	0	A1
Déjeuner D	0	A1
Atelier E	2	A3

Table 2 – Assignation optimale des classes

4.4 Vérification de la Solution

Classe A1 (Réunion A, Présentation C, Déjeuner D) :
— A [8 :00-10 :00] et C [10 :30-12 :00] : pas de chevauchement
— C [10 :30-12 :00] et D [12 :00-13 :00] : pas de chevauchement
— A [8 :00-10 :00] et D [12 :00-13 :00] : pas de chevauchement
Classe A2 (Cours B uniquement) : pas de conflit interne
Classe A3 (Atelier E uniquement) : pas de conflit interne
Optimalité : Il est impossible de réduire à 2 couleurs car le sous-graphe $\{A, B, E\}$ forme un triangle (clique de taille 3).

5 Analyse de la Solution

5.1 Complexité Algorithmique

— **Construction du graphe :** $O(n^2)$ où $n = |V|$ (comparaison de toutes les paires)

— **Résolution PLNE :** NP-difficile en théorie générale, mais Gurobi utilise des heuristiques et du branch-and-bound pour une résolution efficace en pratique
— **Pour notre cas d'usage :** Avec $n \leq 50$ événements par jour, le temps de calcul reste inférieur à 1 seconde

5.2 Avantages de l'Approche

1. **Optimalité garantie :** Nombre minimal de classes utilisées
2. **Flexibilité :** S'adapte automatiquement à n'importe quelle configuration d'événements
3. **Scalabilité :** Gurobi gère efficacement des instances de taille modérée
4. **Robustesse :** Gère les cas particuliers (aucun conflit, conflits complets, etc.)

5.3 Interprétation Pratique

Dans le contexte de gestion d'événements :
— **Classe A1** pourrait représenter une salle de conférence
— **Classe A2** pourrait représenter une salle de cours
— **Classe A3** pourrait représenter un atelier
Le système minimise le nombre de ressources (salles, équipes, etc.) nécessaires pour organiser tous les événements sans conflit temporel.

5.4 Limitations et Extensions Possibles

Limitations actuelles :
— Nécessite l'installation de Gurobi (licence commerciale ou académique)
— Pas de contraintes de préférences ou de priorités entre événements
Extensions possibles :

1. Ajouter des poids aux événements pour prioriser certaines assignations
2. Implémenter des heuristiques alternatives (algorithme glouton) pour une version sans Gurobi
3. Permettre des contraintes de pré-assignation (forcer certains événements dans des classes spécifiques)
4. Visualisation graphique du graphe de conflits
5. Export des plannings par classe au format PDF ou calendrier

6 Conclusion

Ce système de gestion d'événements combine une interface utilisateur moderne et accessible avec un algorithme d'optimisation mathématique rigoureux. L'utilisation de la coloration de graphe optimale via Gurobi garantit une utilisation minimale des ressources tout en évitant les conflits temporels.
Le système est particulièrement adapté pour :
— La planification de salles de réunion
— L'organisation d'emplois du temps académiques
— La gestion de ressources partagées
— La coordination d'équipes multiples

L'approche modulaire du code (séparation UI/Backend/Optimisation) facilite la maintenance et l'extension future du système.