PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Modèle de programmation par contraintes pour l'agencement scolaire basé sur le curriculum de cours sur demi-blocs
Auteur: Author:	Bérénice Dubois
Date:	2026
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Dubois, B. (2026). Modèle de programmation par contraintes pour l'agencement scolaire basé sur le curriculum de cours sur demi-blocs [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/76285/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/76285/
Directeurs de recherche: Advisors:	Quentin Cappart
Programme: Program:	génie informatique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Modèle de programmation par contraintes pour l'agencement scolaire basé sur le curriculum de cours sur demi-blocs

BÉRÉNICE DUBOIS
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Avril 2026

© Bérénice Dubois, 2026.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Modèle de programmation par contraintes pour l'agencement scolaire basé sur le curriculum de cours sur demi-blocs

présenté par **Bérénice DUBOIS**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Gilles PESANT, président
Quentin CAPPART, membre et directeur de recherche
Alain HERTZ, membre externe

RÉSUMÉ

Certaines écoles secondaires ont une structure d'horaire similaire à celles d'universités, où les étudiants suivent différents curricula et ne sont pas préassignés à l'une des multiples sections, ou groupes, de leurs cours. Dans ces cas, les problèmes d'agencement d'horaire et d'assignation d'élèves aux groupes sont fortement interdépendants. L'agencement d'horaire est le plus souvent réalisé par une méthode de programmation linéaire ou métaheuristique, avant d'assigner les élèves à des groupes une fois l'horaire déterminé.

Cependant, de nombreuses écoles québécoises construisent leur horaire selon un patron de bloc de périodes pour assurer une bonne répartition hebdomadaire du temps d'enseignement. Si cette structure remplace le recours à plusieurs contraintes classiques en agencement d'horaire, elle introduit un défi particulier : les cours occupant un demi-bloc doivent être pairés pour recomposer des blocs complets. Cela permet aux élèves de prendre un ou deux cours sur chaque bloc pour se composer un horaire sans trou. Les cours étant donnés en même temps facilitent alors le mélange des élèves entre cours sur blocs entiers et demi-blocs respectivement. L'agencement des cours sur demi-bloc est une étape cruciale de l'agencement d'horaire, car elle a un impact direct sur l'équilibre des groupes après l'assignation des élèves.

Basé sur ce contexte, ce mémoire présente une approche de programmation par contraintes pour générer les horaires de cours sur demi-blocs d'écoles offrant plusieurs curricula dans le but de favoriser une répartition équilibrée des élèves. Le modèle est développé pour être intégré au processus d'agencement du partenaire industriel de ce projet ; Solution Informatiques Dash, une compagnie montréalaise réalisant l'agencement annuel de plus d'une centaine d'horaires d'écoles québécoises. Ce modèle est présenté dans un article de conférence soumis à la 32^{e} Conférence Internationale sur les Principes et Pratiques de la Programmation par Contraintes, se déroulant du 20 au 23 juillet 2026 à Lisbonne, au Portugal.

ABSTRACT

In high school timetabling, some institutions adopt a university-like structure in which students follow different curricula and multiple sections are offered for each course, without pre-assigning students to specific sections. In such settings, student sectioning and timetabling are strongly interdependent. The timetabling step is usually carried out with linear programming or metaheuristics, before assigning students to sections on the fixed schedule.

However, several Quebec high schools construct their schedules according to predefined block patterns designed to ensure a balanced distribution of instructional time. While this structure eliminates many classical timetabling constraints, it introduces a specific challenge: courses occupying half-blocks must be paired to form full blocks. This allows students to have the rest of their courses on well balanced, full blocks, thus easing student mixing between sections for improved student sectioning.

The pairing of half-block courses is a critical step, as it directly impacts the balance of students across sections. Based on this context, this paper introduces a constraint programming approach to generate a feasible half-block courses schedule in schools with multiple curricula leading to good student balancing. The model is to be integrated to the scheduling process of the industrial partner Dash Computer Solutions, a Montreal-based company responsible for the annual design of over a hundred high school schedules in Quebec. This constraint programming model is designed to enhance computational efficiency, accommodate additional practical constraints, and reduce the need for manual adjustments by scheduling operators. This work was presented in an article submitted to The 32$^{\text{nd}}$ International Conference on Principles and Practice of Constraint Programming taking place in Lisbon, Portugal, from July 20$^{\text{th}}$ to 23$^{\text{rd}}$.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

CB-CTT	*Curriculum-based Course Timetabling*, Agencement d'horaire scolaire basé sur le curriculum
CP	*Constraint Programming*, Programmation par contraintes
Dash	Solutions Informatiques Dash
HTT	*High-School Timetabling*, Agencement d'horaire d'école secondaire
ITC	*Internationnal Timetabling Competition*, Compétition internationnale d'agencement scolaire
MIP	*Mixed Integer Programming*, Programmation mixte en nombres entiers

CHAPITRE 1 INTRODUCTION

Chaque été, un casse-tête d'agencement d'horaire scolaire d'une complexité grandissante attend les 72 centres de services scolaires et commissions scolaires québécoises [1]. Entre la pénurie d'enseignants [2], les fermetures de locaux dues à la vétusté des écoles [3] et le resserrement des maximums d'élèves par groupe [4], les pressions sur les ressources posent de réels défis d'organisation. L'agencement d'horaires scolaires de près d'un million [5] d'élèves du primaire et du secondaire en 2025, qui était traditionnellement fait à la main par les directions scolaires, est de plus en plus livré à des services d'agencement externes spécialisés en agencement. De plus, la structure d'horaire par blocs, commune au Québec, est peu documentée dans la littérature. Ce chapitre introduit donc le cadre de ce projet menant à la modélisation et l'implémentation d'un algorithme d'agencement scolaire. Les termes propres au contexte sont définis avant l'explication des éléments de la problématique. Les objectifs de la recherche ainsi que le plan du mémoire sont ensuite présentés.

1.1 Définitions et concepts de base

Au Québec, les écoles secondaires publiques peuvent avoir une structure d'horaire similaire à celles des universités : chaque *cours*, soit une matière enseignée, peut avoir plus d'une section, c'est-à-dire être offert plus d'une fois pour des groupes d'élèves différents. Les sections d'un cours sont appelées des *groupes* au secondaire. Un *cours-groupe* désigne alors l'un des groupes d'un cours précis. Les élèves peuvent s'asseoir dans n'importe quel groupe des cours à leur choix de cours. Les groupes sont alors dits *ouverts*, puisque les élèves peuvent s'asseoir dans n'importe quel groupe. Chaque combinaison distincte de cours d'un choix de cours forme un *curriculum* qui peut être suivi par un ou plusieurs élèves. Pour les élèves du deuxième cycle du secondaire, soient les élèves de secondaire 3 à 5 (âgés de 14 ans à 17 ans), la promotion d'une année à l'autre se fait par matière. Cela signifie que les élèves redoublent uniquement les matières échouées. De plus, ces niveaux comportent plus de cours au choix pour les élèves qu'en secondaire 1 et 2, comme par exemple les cours de mathématiques et sciences avancées versus régulières. La promotion par matière amène donc également à mixer les élèves de différents niveaux dans les différents groupes et augmente ainsi le nombre de curriculums possibles.

Dans les écoles considérées, plusieurs paramètres sont fixés avant l'agencement d'horaire. La direction de l'école récupère les choix de cours des étudiants pour la prochaine année scolaire et décide du nombre de groupes à ouvrir pour chaque cours à partir de ces données. S'il y

a lieu, les résultats des cours d'été peuvent aussi généralement être bien estimés à l'avance selon le nombre d'inscrits et des seuils de réussite historiquement assez fixes. Chaque cours-groupe se voit attribuer un enseignant et une salle précise. Avec ces paramètres, l'agencement de l'horaire-maître à groupes ouverts consiste à fixer deux types de variables : assigner des périodes de rencontre à tous les cours-groupes et assigner à chaque élève un groupe pour chacun de ses cours. Ainsi, l'agencement d'horaire-maître à groupes ouverts se divise en deux problèmes interdépendants. Le premier est *l'agencement d'horaire* : l'assignation aux cours-groupes de périodes de rencontres, soit en anglais le *timetabling problem*. Le second problème est *l'assignation des élèves* : la répartition des élèves dans un groupe spécifique de leurs cours, appelé le *student sectioning problem* dans la littérature. Le but de la résolution de ces problèmes est l'obtention d'un horaire d'école qui respecte un ensemble de contraintes *dures* obligatoires, tout en ayant des groupes bien *équilibrés*. Un cours bien équilibré devrait avoir des groupes contenant tous le même nombre d'élèves. La plupart des méthodes d'agencement d'horaire documentées dans la littérature visent également à respecter un maximum de contraintes *souples* facultatives dans le même processus, comme d'éviter des changements de salles aux élèves ou d'empêcher les élèves de suivre deux fois le même cours dans la même journée. Pour réduire le nombre de contraintes souples à imposer sur l'horaire, plusieurs écoles québécoises adoptent une structure d'horaire par patron, dit horaire *par bloc*. L'horaire scolaire est alors agencé sur un cycle de 6 à 9 jours d'école qui est répété jusqu'à la fin de l'année scolaire. Le cycle est lui-même divisé en 6 à 9 blocs. Un *bloc* est un ensemble prédéfini et fixe de périodes de rencontre bien distribuées à travers le cycle. Agencer un cours-groupe consiste alors à lui assigner un bloc (ou portion de bloc) plutôt que de spécifier chacune de ses périodes de rencontres individuellement. Ces blocs garantissent une bonne distribution des périodes d'un cours à travers le cycle, mais favorisent aussi plusieurs autres qualités d'un bon horaire. En effet, les blocs empêchent d'agencer une matière deux fois dans la même journée pour les élèves, mais permettent aussi d'interchanger les salles de cours plus facilement entre deux cours agencés sur le même bloc.

Les contraintes peuvent alors être imposées au niveau des cours-groupes, et non au niveau des périodes individuelles de rencontre d'un cours-groupe, ce qui simplifie le problème d'agencement d'horaire. Selon les formulations de problèmes d'agencement scolaire décrites par Ceschia et al. (2023) [6], le problème considéré s'apparente alors à un problème d'agencement d'horaire universitaire, et plus particulièrement à un *problème d'horaire universitaire basé sur les curricula* ou en anglais le *curriculum-based course timetabling problem* (CB-CTT).

1.2 Éléments de la problématique

Bien que la structure par blocs permette d'agencer des horaires de haute qualité, cette structure entraîne un défi particulier. Les blocs ont tous le même nombre de périodes, qui correspond au nombre de périodes requis pour la majorité des cours, mais certains cours sont enseignés sur un peu plus ou un peu moins de périodes de rencontre. Les cours comme les arts ou le sport occupent plutôt un *demi-bloc*, soit un sous-ensemble contenant la moitié des périodes d'un bloc. Au contraire, les cours comme les mathématiques ou le français occupent un bloc régulier plus un demi-bloc. Cependant, pour qu'un élève ait exactement un cours par bloc à son horaire, chaque cours occupant un demi-bloc doit être complété par un autre cours sur le reste du demi-bloc.

Pairer ainsi les cours sur demi-bloc deux par deux permet d'agencer les autres cours sur des blocs entiers, ce qui procure plus de granularité pour agencer l'horaire-maître. Avoir beaucoup de cours sur blocs complets offerts simultanément permet aussi une meilleure distribution des élèves dans les groupes enseignés en même temps lors de l'assignation des élèves, et de même pour les groupes sur demi-bloc. C'est pourquoi les cours sur demi-blocs sont concentrés le plus possible sur une minorité de blocs, et leur agencement est particulièrement critique pour une répartition optimale des élèves. En effet, considérant que les cours sur demi-blocs sont à eux seuls source d'une grande variété de curricula possibles pour les secondaire 3, 4 et 5, ils sont généralement les plus difficiles à agencer. De plus, les cours comme les arts et le sport, qui en font partie, utilisent généralement des ressources (enseignants et locaux) partagés par l'école complète. L'agencement de ces cours impose alors des contraintes sur les horaires des autres niveaux qui en partagent les ressources. L'horaire des cours sur demi-blocs détermine donc la qualité du reste de l'horaire et est généralement agencé en premier.

En outre, l'agencement de l'horaire doit être complété d'une assignation bien équilibrée des élèves aux groupes. Or, l'assignation des élèves est une tâche NP-dure lorsqu'il y a plus qu'un curriculum [7]. Devant le nombre de groupes ouverts par cours dans les écoles, les combinaisons exponentielles d'horaires possibles ne peuvent toutes être explorées. Pour une école plutôt petite avec 4 cours par curriculum, 2 groupes par cours et 3 niveaux (secondaire 3, 4 et 5) à agencer, il y a 24 cours-groupes à agencer sur un minimum de 4 demi-blocs, ce qui fait au minimum $4^{24} = 281474976710656$ horaires à considérer sans contraintes.

Par ailleurs, pour ces horaires ouverts où les élèves sont libres de s'asseoir dans n'importe quel groupe, il y a des contraintes additionnelles sur l'assignation des élèves aux groupes. D'abord, les capacités des salles imposent des limites inflexibles sur la taille des groupes. Ensuite, peu importe la taille de la salle, les écoles doivent payer des pénalités financières si

le nombre d'élèves d'un groupe dépasse le ratio maître-élève maximum pour leur commission scolaire [8]. Les groupes qui dépassent ce ratio sont non seulement une plus grande charge pour les enseignants, mais réduisent le temps que chaque élève peut recevoir dans les sections plus chargées. Par exemple, dans une école donnée, les salles de secondaire 5 pourraient toutes avoir 35 pupitres et la limite du ratio maître-élève pourrait être de 32 élèves. Dans ce cas, il vaut mieux avoir deux sections d'un cours de mathématique remplies à 32 élèves chacune, plutôt qu'une à 31 et l'autre à 33 élèves, et ce bien qu'aucune contrainte dure sur la capacité des salles ne soit violée dans les deux cas. Les directions scolaires désirent donc, dans un horaire idéal, que les sections de chaque cours accueillent le même nombre d'élèves. Dans les écoles où la pénurie d'enseignants force à faire de grands groupes [4], les coûts de débordement de groupes peuvent être particulièrement sensibles à la répartition des élèves dans les sections.

Solutions Informatiques Dash (Dash), une compagnie responsable de l'agencement annuel de plus d'une centaine d'horaire-maîtres pour des écoles québécoises, a mis au point une heuristique efficace pour faire l'assignation des élèves après l'agencement des cours. L'assignation peut même se faire sur un horaire incomplet et être progressivement mise à jour au fur et à mesure que l'horaire se remplit. Leur algorithme retourne un assignation des élèves aux groupes presque instantanément. Si la compagnie est satisfaite des résultats de cet algorithme d'assignation des élèves, ces résultats dépendent d'abord de la qualité de l'horaire agencé. Avec cette méthode, l'équilibre des groupes ne peut pas être prédit exactement avant de construire l'horaire. Et comme l'assignation des élèves peut faire déborder des locaux, rendant un horaire innaceptable, Dash se base actuellement sur des méthodes empiriques pour construire des horaires qui favorisent un bon équilibre des élèves par la suite.

Chez Dash, l'agencement des cours sur demi-blocs est fait en premier dans les horaires par blocs. Cette partie de l'horaire requiert aussi une bonne planification avec l'école pour valider qu'une bonne répartition des élèves est possible. Suite à cela, l'agencement des cours sur demi-blocs se fait par recherche locale guidée par l'humain. L'opérateur construit une solution initiale à la main, qui est par la suite améliorée par des échanges locaux de cours-groupes, jusqu'à l'obtention d'une solution localement optimale. Bien que le processus puisse générer des horaires de haute qualité, il n'y a cependant aucune garantie quant à l'optimalité globale de la solution trouvée, et le processus prend beaucoup de temps. En moyenne, trois à douze heures ouvrables sont requises pour l'agencement selon l'expérience de l'opérateur et la qualité de la préparation de l'horaire (assignation préalable des enseignants et locaux, nombre de sections à ouvrir par cours, *etc.*). C'est sans compter le temps de prétraitement des données, la maintenance et les autres tâches opérationnelles. Un bon horaire pour les cours sur demi-blocs ne garantit pas non plus la qualité globale de l'horaire-maître une fois les cours à blocs

entiers ajoutés. La partie de l'horaire sur demi-blocs doit le plus souvent être révisée, c'est-à-dire reconstruite, au moins une fois avant la finalisation de l'horaire, pour essayer de trouver une meilleure solution ou bien pour prendre en compte des mises à jour dans les données. L'agencement de la portion de l'horaire sur demi-blocs prend donc en réalité entre une et une dizaine de journées de travail à l'opérateur.

1.3 Objectifs de recherche

L'objectif principal de cette recherche est la conception et l'implémentation d'un algorithme d'agencement par programmation par contraintes des cours ouverts sur demi-blocs dans les horaires d'écoles secondaires par blocs, qui favorise le bon équilibre des moyennes des groupes. Le modèle présenté est conçu pour être facilement adaptable aux contraintes additionnelles propres aux écoles. Ce modèle vise à améliorer l'efficacité du processus d'agencement de Dash et à diminuer les interventions manuelles des opérateurs en agencement d'horaire. L'approche est évaluée sur quatre ensembles de données anonymisés provenant d'écoles secondaires québécoises de l'année académique 2024-2025. La solution est intégrée au logiciel Dash pour usage dès l'été 2026.

1.4 Plan du mémoire

D'abord, la revue de littérature et présentation du cadre conceptuel, à la section 2, introduit la classification du problème d'agencement d'horaire d'intérêt et les méthodes employées pour la résolution qui se retrouvent dans la littérature. Les méthodes d'assignation d'élèves sont également discutées. Ce chapitre introduit également les principaux éléments de programmation par contraintes et les modèles d'agencement scolaire ayant recours à cette technique de résolution. Le chapitre suivant contient l'article de conférence soumis à la conférence 32^{e} Conférence Internationale sur les Principes et Pratiques de la Programmation par Contraintes, et qui détaille la description du problème, la modélisation de la solution et les principaux résultats. Le dernier chapitre présente des résultats et discussions complémentaires.

CHAPITRE 2 REVUE DE LITTÉRATURE ET CADRE CONCEPTUEL

Le chapitre qui suit présente la revue de littérature des problème d'agencement scolaire et la présentation des concepts de programmation par contraintes. La première partie explique la classification du problème abordé selon une formulation standard reconnue dans la littérature. Les deuxième et troisième parties abordent respectivement les méthodes employées pour résoudre les problèmes interdépendants d'agencement d'horaire et d'assignation d'élèves. La quatrième partie est un survol des concepts de base de la programmation par contraintes, incluant le formalisme et l'impact de la modélisation sur la taille d'un problème, les principes de stratégie de recherche et de propagation dans la résolution et le type de langage utilisé pour l'implémentation. Enfin la dernière section traite des applications de la programmation par contraintes aux problèmes d'agencement d'horaires d'écoles.

2.1 Classification du problème d'agencement d'horaire

L'agencement d'horaire d'école admet une grande variété de formulations différentes du problème. En effet, à l'échelle internationale tout comme régionale, les écoles et le travail des enseignants sont régis par des règles et des structures spécifiques à l'établissement d'enseignement. Pour comparer les méthodes d'agencement d'horaire développées par la communauté scientifique, Ceschia et al. introduisent plusieurs formulations standards du problème (2023) [6]. Les auteurs y présentent une formulation standard pour les écoles secondaires, soit le *High-School Timetabling* (HTT), qui vise à agencer les classes fermées d'une école sans conflit. Puisque notre but est d'agencer des groupes ouverts, notre problème d'agencement est en fait du type *University Course Timetabling* (CTT). Et en effet, lorsque les groupes sont ouverts ainsi, la liste des curricula des élèves sert mieux à déterminer si tous les élèves ont au moins un horaire possible. La structure par bloc du problème permet aussi de formuler des contraintes au niveau des cours plutôt que de vérifier les périodes de rencontres individuellement. Ces deux caractéristiques en font bien un problème de *Curriculum-based Course Timetabling* (CB-CTT) selon les formulations standards de Ceschia et al.

La plupart des méthodes de résolution du CB-CTT décrites dans la littérature visent à satisfaire un ensemble de contraintes *dures*, tout en minimisant le nombre ou le poids total des violations de contraintes *souples* [9]. Les contraintes dures sont celles qui doivent absolument être respectées pour qu'une solution soit considérée valide. Les trois contraintes dures les plus communes visent à empêcher l'horaire de contenir un *conflit de local*, un *conflit d'enseignant* ou un *conflit d'élève*. Les conflits de local ou d'enseignant surviennent à une période dans

l'horaire où un local ou un enseignant respectivement est sollicité par deux cours ou plus agencés à la même période dans l'horaire, ce qui est physiquement impossible à satisfaire. Un conflit d'élève se produit lorsqu'un élève a plus d'un cours à son horaire dans la même période. Les contraintes souples visent à améliorer la qualité d'une solution. Des exemples courants incluent de limiter le nombre de trous, soient les périodes vides sans cours en milieu de journée, dans l'horaire des élèves, ou encore d'assigner toutes les périodes d'un cours-groupe à la même salle [9].

2.2 Méthodes de résolution du problème d'agencement d'horaire

Les méthodes métaheuristiques sont populaires pour résoudre le CB-CTT [10–12]. En effet, ces techniques conviennent à la complexité et la taille de l'agencement d'horaire sur une école complète. La compétition internationale d'agencement scolaire ITC-2007 (*International Timetabling Competition*), offre un ensemble de données de référence provenant principalement de l'école d'Ingénierie d'Udine. L'assignation des élèves n'est pas incluse, et seuls les curricula fournis par l'école doivent pouvoir être suivis [13]. Pour ces données, S. Abdullah et H. Turabieh dépassent des records avec un algorithme génétique hybridé avec une recherche Tabu à large voisinage, en 10 minutes de recherche [14]. Kiefer et al. établissent également des records sur ces données avec un algorithme de recherche à voisinage large adaptative [15]. Cependant, ces méthodes visent également à assigner des locaux aux cours, contrairement au cas des écoles québécoises, où les salles sont généralement préassignées. La qualité des solutions sur ces données est évaluée par le respect des contraintes dures et souples, notamment la capacité des salles, mais pas la taille des groupes, puisqu'elle n'importe pas autant en agencement d'horaires universitaires. Sur les données de la compétition ITC-2019, l'algorithme de recuit-simulé de Sylejmani et Bashi s'avère également compétitif, en employant des sous-algorithmes pour résoudre les contraintes les plus difficiles à respecter [16]. L'un des inconvénients des méthodes métaheuristiques néanmoins est que les algorithmes sont construits sur mesure pour le type de problème qu'ils visent à résoudre, ce qui les rend moins flexibles aux variations de formulation. En outre, la maintenance de ces algorithmes, et notamment de leurs hyperparamètres le cas échéant, requiert une très bonne compréhension des techniques et du problème.

Le problème d'ITC-2019, plus complexe que celui de l'ITC-2007, inclut l'assignation des étudiants [17] et les deux meilleures solutions sont des méthodes de programmation mixte en nombre entiers (MIP). La solution de Holm et al. construit une solution initiale incluant l'agencement d'horaire et l'assignation d'élèves qui est améliorée et réparée par la suite [18]. La solution initiale et la phase de réparation et optimisation utilisent une méthode MIP. Le

modèle de Rappos et al. utilise un algorithme qui branche et borne [19].

La programmation linéaire a aussi été essayée avec des patrons de périodes pour les rencontres d'un même cours [20, 21] et par génération de colonnes [22]. Ces méthodes donnent de bons résultats, garantissent un meilleur respect des contraintes souples et réduisent l'espace de recherche, mais ces modèles utilisent tout de même des patrons beaucoup plus granulaires que ceux standards au Québec. Cela donne lieu à beaucoup plus de possibilités d'horaires que dans notre cas.

Néanmoins, la plupart des contraintes souples considérées par les modèles dans la littérature deviennent inutiles à notre problème. D'abord, les étudiants des écoles québécoises considérées suivent un horaire complet, c'est-à-dire qu'ils n'ont pas de périodes libres ou de trous à l'horaire à minimiser. Ensuite, la structure par bloc des horaires fait en sorte qu'un cours ne puisse pas être agencé deux fois dans la même journée et que les périodes d'un cours soient bien réparties dans le cycle. De plus, contrairement à notre cas, la plupart des méthodes décrites dans la littérature assignent les salles de cours après l'agencement, ce qui permet plus de flexibilité dans la recherche de solutions, car seul le nombre maximal de cours donné en même temps doit être considéré. Il faut cependant veiller à ce que les salles assignées par la suite soient adéquates aux cours qui y sont enseignés [23]. Par exemple, un laboratoire ne peut généralement pas servir pour un cours d'art. Ainsi le cas des écoles secondaires québécoises est aussi différent, car les salles sont pré-assignées aux cours avant l'agencement, contrairement aux cours universitaires. En outre, sur un campus universitaire, il peut aussi y avoir des distances trop longues à parcourir entre deux cours consécutifs dans des pavillons différents. Ce problème ne survient pas dans notre contexte, puisque toutes les écoles secondaires considérées sont suffisamment petites pour que les déplacements entre n'importe quelles salles soient considérés possibles dans les courtes pauses après chaque cours. Enfin, la taille des problèmes universitaires est beaucoup plus grande que les cours de deuxième cycle du secondaire qui sont agencés dans notre recherche. Pour toutes ces raisons, notre cas de CB-CTT est tout de même nettement différent de ceux abordés dans la littérature, avec une structure peu commune.

2.3 Méthodes de résolution du problème d'assignation d'élèves

L'agencement d'horaire de groupes ouverts doit être combiné à une méthode d'assignation des élèves aux cours-groupes. Le problème d'assignation des élèves aux groupes, seul, a été prouvé NP-dur lorsqu'il y a plus d'un curriculum [7]. L'assignation des élèves peut se faire avant, pendant et après l'agencement de l'horaire [7, 24]. Des approches itératives, alternant entre des phases d'agencement d'horaire et des phases d'assignation d'élèves ont également

pour but de progressivement raffiner la solution [25, 26].

Dans le cas de notre partenaire industriel, Dash, une heuristique efficace est utilisée pour réaliser l'assignation des élèves après l'agencement de l'horaire. L'assignation peut même se faire sur un horaire incomplet et être progressivement mise à jour au fur et à mesure que l'horaire se remplit. L'algorithme retourne alors une bonne assignation des élèves en quelques secondes selon l'horaire fourni.

2.4 Programmation par contraintes

Cette section regroupe les notions de base servant à la compréhension de la modélisation présentée dans le chapitre suivant. Les explications sont adaptées des livres *Handbook of Constraint Programming* de Rossi, van Beek et Walsh (2006) [27], et *Principles of Constraint Programming* de Apt (2003) [28]. La première sous-section présente l'écriture d'un modèle de programmation par contraintes, tandis que la deuxième partie résume le fonctionnement de la résolution de ce type de modèle.

2.4.1 Structure d'un modèle d'optimisation

La programmation par contraintes est tout indiquée pour la résolution de problèmes combinatoires régis par des règles. L'écriture du modèle requiert des choix dans la définition des éléments du problème combinatoire, comme présenté ci-après.

Formalisation d'un problème combinatoire

Un problème de satisfaction de contraintes est défini par le triplet $\langle X, D, C \rangle$. Leur signification est la suivante :

- X est l'ensemble fini des variables de décision du problème.
- D est l'ensemble fini qui regroupe les valeurs possibles que peuvent prendre les variables de décision du problème. On note $D(x_i)$ le domaine de la variable x_i, soit l'ensemble des valeurs que la variable x_i peut prendre.
- C est l'ensemble fini des contraintes du problème. Chaque contrainte correspond à une relation à respecter entre les variables d'un sous-ensemble de variables $X(c) \subseteq X$. Une contrainte regroupe les valeurs des k variables qui doivent respecter la contrainte : $c \subseteq D(x_1) \times D(x_2) \times ... \times D(x_k)$.

Une solution au problème de satisfaction de contraintes est alors définie comme étant une affectation τ de valeurs aux variables telle que toutes les contraintes soient respectées. Faire une affectation de valeurs aux variables signifie qu'à la fin de la résolution, les variables ont

exactement une valeur dans leur domaine, qui leur est affectée. Les autres valeurs ont été retirées de leur domaine durant la résolution.

$$\tau : x \in X \mapsto v \in D(x) \mid \tau|_{X(c)} \in c \; \forall c \in C$$

Dans le cas d'un problème d'optimisation, une fonction objectif f qui évalue la valeur de l'objectif pour les solutions est aussi nécessaire. $f\left((\tau(x))_{x \in X}\right) \in \mathbb{R}$ est alors la valeur de l'objectif, qui est minimale pour la meilleure solution. Le problème d'optimisation est alors défini par $\langle X, D, C, f \rangle$.

Modélisation d'un problème d'optimisation

La modélisation du problème à résoudre présente les paramètres du problème, spécifie le calcul de l'objectif et définit les variables, leur domaine initial et les contraintes à respecter. La modélisation décrit donc l'espace de recherche initial. Restreindre le nombre des variables et leur domaine dès la définition du problème peut alors être un moyen de réduire l'espace de recherche, afin de trouver plus efficacement des solutions.

Par exemple, prenons un problème de satisfaction de contraintes dont le but d'un problème est d'agencer l'horaire sur une semaine d'une classe fermée, pour laquelle les élèves prennent tous les mêmes cours ensemble. Les paramètres sont les suivants :

- C : l'ensemble des cours enseignés à la classe
- J : l'ensemble des jours sur lesquels l'horaire est agencé
- P : l'ensemble des périodes

Un premier modèle pourrait avoir les variables binaires énoncées ci-dessous, qui valent 1 si le cours c est enseigné le jour j à la période p, et valent 0 sinon.

$$x_{c,j,p} \in \{0, 1\} \tag{2.1}$$

Dans ce cas, il faut $|C| \times |J| \times |P|$ variables pour définir le problème, et le nombre de solutions possibles s'élève à $2^{|C| \cdot |J| \cdot |P|}$, si $\sum_{J,P} x_{c,j,p} \forall c \in C$, avant d'imposer des contraintes additionnelles. Cependant, on peut aussi modéliser de manière plus efficace avec des variables qui prennent directement comme valeur le cours enseigné. La variable $c_{j,p}$ prend alors la valeur du cours enseigné à la période p du jour j.

$$c_{j,p} \in C \tag{2.2}$$

Le nombre de variables est alors égal au nombre de périodes sur l'horaire complet $|J| \times |P|$, tandis que le nombre de solutions sans contrainte s'élève à $|C|^{|J|\cdot|P|}$. Or $2^{|C|} > |C|$ dès que $|C| \geq 2$, d'où $2^{|C|\cdot|J|\cdot|P|} > |C|^{|J|\cdot|P|}$: il y a moins de solutions à évaluer avec cette deuxième formulation.

Dans le cas où chaque cours est enseigné au même nombre de périodes, et que ces périodes doivent être placées sur des blocs prédéfinis, une troisième formulation simplifie encore l'espace de recherche. En effet, agencer l'horaire revient alors à associer un cours à chaque bloc. La variable b_c désigne ainsi le bloc sur lequel est agencé le cours c, parmi un ensemble de blocs $\mathcal{B}$ prédéfini.

$$b_c \in \mathcal{B} \tag{2.3}$$

Avec le type de variable présenté ci-dessus, il faut utiliser $|C|$ variables. Le nombre de solutions possibles est de $|\mathcal{B}|^{|C|}$, et s'il y a autant de blocs que de cours cela donne $|C|^{|C|}$ solutions. Comme le nombre de bloc $|C|$ est plus petit que le nombre de périodes total sur une semaine $|J| \cdot |P|$, il y a encore une réduction du nombre de solutions possibles $|C|^{|J|\cdot|P|} > |C|^{|C|}$.

Le tableau 2.1 illustre un exemple simplifié du patron par bloc sur 5 jours, où chaque jour (colonne) est divisé en 5 périodes (lignes). Les blocs `A, B, C, D` et `E` sont des groupes de 5 périodes placées en diagonale. Un cours agencé sur le bloc `A` se donnerait en période 1 du jour 1, période 2 du jour 2 et ainsi de suite jusqu'à la période 5 du jour 5. Dans ce cas, le premier modèle admet 125 variables et plus de $4,2 \times 10^{37}$ solutions. Le deuxième introduit 25 variables pour 33 554 432 solutions potentielles. Le troisième modèle aurait 5 variables et 3125 solutions potentielles. Cet exemple illustre l'impact de la définition du modèle.

Jour / Période	J1	J2	J3	J4	J5
P1 (8h00 à 9h00)	`A1`	`B2`	`C3`	`D4`	`E5`
P2 (9h00 à 10h00)	`E1`	`A2`	`B3`	`C4`	`D5`
P3 (10h00 à 11h00)	`D1`	`E2`	`A3`	`B4`	`C5`
P4 (13h00 à 14h00)	`C1`	`D2`	`E3`	`A4`	`B5`
P5 (14h00 à 15h00)	`B1`	`C2`	`D3`	`E4`	`A5`

Tableau 2.1 Exemple de patron d'agencement par bloc simplifié.

Enfin, la *symétrie* du modèle peut aussi changer significativement l'espace de recherche. En effet, si une solution valide est transformable en une autre solution valide de même qualité selon la modélisation du problème, il y a lieu de s'interroger sur la pertinence de générer les

deux. Le solveur génère deux solutions symétriques qu'il considère différentes mais qui sont en réalité équivalentes pour le problème. En agencement scolaire, dans la plupart des formulations, l'ordre des jours de la semaine peut facilement être inversé sans affecter la qualité de l'horaire. Dans certains cas, l'ordre des périodes peut aussi être interverti sans changer la qualité de l'horaire. Le solveur considère alors un nombre exponentiellement plus grand de solutions. Pour empêcher de générer autant de solutions additionnelles, il faut restreindre les symétries lorsque possible dans la définition du modèle, ou bien ajouter des contraintes forçant un ordre particulier dans les solutions. Par exemple, le patron d'agencement par bloc force un ordre des jours et des périodes dans l'horaire, contrairement à l'assignation d'un cours par période, mais il laisse la flexibilité d'échanger les blocs entre les cours si rien d'autre n'est spécifié.

2.4.2 Résolution par programmation par contraintes

Pour trouver une solution au problème combinatoire, il faut donc une affectation de valeurs aux variables qui respecte toutes les contraintes. La recherche est effectuée par un solveur de programmation par contraintes. Trois principaux éléments influencent l'efficacité de la recherche de solutions. Le premier est la modélisation du problème, qui comme indiqué précédemment, peut déterminer la taille de l'espace de recherche. Le deuxième est la stratégie de recherche et le troisième est la propagation des contraintes.

Stratégie de recherche

Au début de la recherche, les variables n'ont aucune valeur. Les solutions peuvent être construites en parcourant un arbre de recherche, où chaque branchement correspond à *l'instanciation* d'une variable, soit au choix d'une valeur pour une variable. L'arbre a donc une profondeur maximale correspondant au nombre de variables à instancier. L'ordre dans lequel l'arbre est parcouru est déterminé par la *stratégie de recherche*. L'une des stratégies de recherche les plus courantes est la recherche avec *retours en arrière*. Cela consiste à faire une recherche en profondeur dans l'arbre jusqu'à ce qu'une contrainte ne soit plus respectée ou jusqu'à l'atteinte d'une solution. Si une contrainte n'est pas respectée, le retour en arrière consiste à revenir au dernier branchement et prendre une valeur différente. Pour un problème d'optimisation, il est commun de *brancher et borner*, qui est une exploration prenant en compte la valeur de l'objectif minimisé dans la recherche. Cette stratégie s'appuie sur le fait qu'il ne sert à rien d'explorer une branche où l'objectif ne puisse pas être amélioré par rapport à la meilleure solution trouvée jusqu'alors, pour couper des branches entières de l'arbre de recherche.

Ces stratégies de recherche générales sont pairées à une heuristique de recherche qui détermine dans quel ordre exact les variables sont explorées et quelle valeur leur assigner. Par exemple, l'agencement d'horaire ayant longtemps été réalisé à la main, il est plus facile dans ces cas d'essayer d'agencer les cours les plus difficiles à placer en premier. L'agenceur donc sélectionne à la main le cours pour lequel il y a le moins de périodes possibles et fait un choix arbitraire de période pour l'agencement. Dans ce processus, l'agenceur choisi donc la variable avec le plus petit domaine et lui donne une valeur arbitraire.

Propagation de contraintes

La *propagation* des contraintes permet également de retirer certaines valeurs des domaines des variables plus rapidement. Pour comprendre la propagation, il faut cependant introduire la notion de *cohérence*. La cohérence est une propriété atteignable à plusieurs niveaux. L'exemple suivant permet d'illustrer plusieurs de ces niveaux.

Le tableau 2.2 montre une courte liste de cours donnés à une classe d'élèves. Les trois premières colonnes indiquent les paramètres du problème : le type de cours, l'enseignant qui le donne et le local dans lequel il est donné. Dans cet exemple, chaque variable correspond au bloc sur lequel on agence chaque cours. La colonne "variable" indique donc le nom de la variable associée au cours. La colonne "domaine" indique, en un instant donné de la résolution, les valeurs que les variables peuvent encore prendre. Le problème est accompagné des contraintes suivantes :

1. Le Français doit être enseigné en bloc `A` : $x_1 = \texttt{A}$
2. Le local 1 ne peut être utilisé que par un cours à la fois : $x_1 \neq x_2$
3. La classe ne suit qu'un cours à la fois : $x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_2 \neq x_3$

La cohérence de nœud est atteinte lorsque le domaine d'une variable respecte une contrainte unaire, soit une contrainte qui ne concerne qu'une seule variable. On dit que chaque valeur de la variable concernée a un *support*, soit une solution qui respecte la contrainte. Dans cet exemple, la contrainte 1 est une contrainte unaire, et dans le Tableau 2.2, le seul bloc possible est bien le bloc `A`. Donc la cohérence de noeud est atteinte pour la contrainte 1.

Cours	Enseignant	local	variable	domaine
Français	Prof 1	local 1	x_1	$x_1 \in \{\texttt{A}\}$
Mathématiques	Prof 2	local 1	x_2	$x_2 \in \{\texttt{A,B,C}\}$
Histoire	Prof 1	local 2	x_3	$x_3 \in \{\texttt{A,C}\}$

TABLEAU 2.2 Exemple de cours donnés sur blocs à une classe fermée et les variables associées

La cohérence d'arc concerne les contraintes binaires, soit celles qui impliquent deux variables. La contrainte 2 en est un exemple. Dans le Tableau 2.2, le local 1 est utilisé par les deux premiers cours. Or, les variables x_1 et x_2 ont le bloc `A` en commun dans leur domaine. Pour que la cohérence d'arc soit respectée, il faudrait retirer le bloc `A` du domaine de x_2, pour avoir $x_1 \in \{\texttt{A}\}, x_2 \in \{\texttt{B,C}\}$.

Enfin la cohérence d'arc généralisée est appliquée aux contraintes qui concernent plus de deux variables. Dans cet exemple il s'agit de la troisième contrainte. Il faut alors filtrer le bloc `C` du domaine de x_2 et filtrer `A` du domaine de x_3 pour atteindre la cohérence d'arc généralisée avec $x_1 \in \{\texttt{A}\}, x_2 \in \{\texttt{B}\}, x_3 \in \{\texttt{C}\}$.

Cette dernière contrainte stipule que toutes les variables doivent être différentes. En langage déclaratif, ces contraintes qui impliquent plus de deux variables peuvent être écrites avec des *contraintes globales*. Dans ce cas, cette contrainte correspond à la contrainte globale alldifferent(X), qui indique que les valeurs doivent être différentes. Le langage de modélisation déclaratif permet d'énoncer les contraintes à haut niveau, ce qui est plus facile à comprendre et à maintenir. Ce langage peut s'écrire dans un environnement de résolution comme Minizinc [29], qui traduit les contraintes à l'exécution pour faire appel au solveur. Le solveur applique alors la stratégie de recherche et l'heuristique de recherche, tout en propageant les contraintes à chaque étape.

2.5 Modèles de programmation par contraintes en agencement d'horaire d'école

Ce paragraphe présente des modèles de *Constraint Programming* (CP) ou CP hybrides développés pour l'agencement d'horaires d'école, bien que non applicables à notre contexte spécifique.

Une application similaire à la nôtre est le modèle de Nogareda et Camacho (2017) [26], qui proposent une approche itérative alternant entre une phase d'agencement d'horaire par CP et une phase d'assignation des étudiants par une métaheuristique de type colonie de fourmis.

Dimitsas et al. (2022) [30] proposent un modèle hybride MILP où la CP est utilisée pour chercher des améliorations locales et pour vérifier le respect de contraintes non-linéaires qui ne peuvent être modélisées dans le modèle linéaire. Ces approches opèrent cependant au niveau des périodes individuelles, et non des cours.

Demirović and Stuckey (2018) [31] présentent un modèle CP complet avec initialisation à chaud (*warm start*) pour l'agencement d'horaire d'école secondaire. Leur méthode présentent de bons résultats sur des instances de taille similaire à celle que nous considérons, mais les classes sont fermées, c'est-à-dire que les élèves sont préassignés à des groupes et le modèle ne

fait pas d'assignation d'élève.

Enfin, Valouxis et Housos [32] proposent un modèle hybride qui itère entre une étape d'agencement par programmation par contraintes et une étape de réduction de l'espace de recherche par l'emploi de la recherche locale autour de la solution trouvée par CP. La méthode retourne de bon résultats, mais est évaluée sur des instances significativement plus petites que dans notre contexte. Là aussi, les contraintes sont validées à la période, plutôt que de considérer un cours dans son ensemble.

CHAPITRE 3 ARTICLE 1 : CONSTRAINT PROGRAMMING FOR CURRICULUM-BASED HIGH SCHOOL TIMETABLING WITH HALF BLOCKS

Le chapitre qui suit est le contenu de l'article de conférence écrit par Bérénice Dubois, Quentin Cappart et Stephen Walsh qui présente la solution de programmation par contraintes développée dans le cadre de ce projet de recherche. L'article a été soumis le 14 mars 2026 à la conférence CP 2026 se déroulant du 20 au 23 juillet 2026. À titre d'auteure principale, j'ai présenté le problème, conçu et présenté la modélisation de la solution ainsi qu'effectué les tests et analyses présentées, le tout sous la supervision du professeur Quentin Cappart, directeur de recherche de ce projet. Le professeur Cappart a révisé la description du problème, la formulation du modèle et la présentation des résultats, en plus de son accompagnement dans le processus de conception du modèle. Stephen Walsh, fondateur et chef de la direction chez Solutions Informatiques Dash a complété dans l'article les informations relatives au cadre d'implémentation et à la portée du projet. En tant que représentant du partenaire industriel, il a aussi contribué à la définition des objectifs et à l'analyse des résultats durant le projet. Suite à l'abstract, l'article introduit la description formelle du problème traité, la modélisation de la solution par contraintes appliquée et l'analyse des principaux résultats sur les instances fournies par Dash.

3.1 Abstract

In high school timetabling, some institutions adopt a university-like structure in which students follow different curricula and multiple sections are offered for each course, without pre-assigning students to specific sections. In such settings, student sectioning and timetabling are strongly interdependent. The timetabling step is usually carried out with linear programming or metaheuristics, before assigning students to sections on the fixed schedule. However, several Canadian high schools construct their schedules according to predefined block patterns designed to ensure a balanced distribution of instructional time. While this structure eliminates many classical timetabling constraints, it introduces a specific challenge : courses occupying half-blocks must be paired to form full blocks. This allows students to have the rest of their courses on well balanced, full blocks, thus easing student mixing between sections for improved student sectioning. The pairing of half-block courses is a critical step, as it directly impacts the balance of students across sections. Based on this context, this paper introduces a constraint programming approach to generate a feasible half-block courses

schedule leading to good student balancing. The model is designed to enhance computational efficiency, accommodate additional practical constraints, and reduce the need for manual adjustments by scheduling operators. The approach is evaluated on anonymized real-world data from Canadian high schools for the 2024-2025 academic year, provided by Dash Computer Solutions, a company responsible for the annual design of many high school schedules in Quebec, and is integrated with their existing student sectioning algorithm. Our results show that our approach provides solutions of comparable quality to a human-guided local search with a hand-crafted initial solution, while reducing the designing time from 10 hours to an hour in the best case.

3.2 Introduction

In the province of Quebec, Canada, public high schools can have a university-like structure : students can sit in any groups in accordance with their course selection and mix between sections. Each unique course-selection combination forms a *curriculum*. In secondary 3 to 5 (roughly for students between 14 to 17 years old), students are promoted by course, meaning that they may have to retake one or more individual courses from the previous year [33]. For these grade levels, students usually have more choices of courses to take, such as option classes or advanced sciences courses. This contributes to increasing the number of curricula and the mixing of students, and therefore increasing the difficulty to design appropriate schedules.

In these schools, each course section is preassigned a specific teacher and classroom before the timetable is constructed. Students also select their courses in advance. As a result, the overall problem can be decomposed into two interrelated subproblems. The first is the *timetabling problem*, which consists of assigning course sections to periods while respecting the predefined teacher and room allocations. The second is the *student sectioning problem*, which assigns students to course sections in accordance with their course selections. The resulting timetable must satisfy all hard constraints while also supporting an effective distribution of students, ensuring that the sections of a given course have similar enrollment sizes.

Focusing on the timetabling problem, the schools considered adopt a pattern-based schedule known as a *block schedule*. In this structure, the timetable is organized as a cycle of 6 to 9 school days repeated throughout the year. The cycle is also divided into 6 to 9 blocks. A *block* corresponds to a predefined set of meeting periods distributed across the timetable cycle. Courses are scheduled by assigning them to blocks (or portions of blocks) rather than to individual periods. According to the timetabling formulations described by Ceschia et al. (2023) [6], the problem considered here is similar to a university course timetabling problem (CTT), more specifically a *curriculum-based course timetabling problem* (CB-CTT). Although

scheduling occurs after student enrollment, the block structure results in constraints being defined at the course level rather than at the individual lecture level.

In most CB-CTT problems described in the literature, all hard constraints must be satisfied while the total weight of unsatisfied soft constraints is minimized. Hard constraints prevent infeasible situations, such as teacher schedule conflicts. Soft constraints aim to improve timetable quality, for instance by reducing gaps in students' schedules, limiting the number of daily lectures, or assigning all lectures of a course to the same room [9]. Existing approaches for solving the CB-CTT formulation mainly rely on metaheuristics [10–12], hyper-heuristics [34], mixed-integer programming [21, 35], and hybrid approaches [9] reflecting the large scale and computational complexity of these problems. However, many soft constraints typically considered in CB-CTT formulations become irrelevant when applied to this specific case of high school timetabling. Students in these schools follow a full schedule, which eliminates the possibility of free periods. Also, the predefined block structure ensures that each course section is scheduled at most once per day, thereby preventing double lectures. Furthermore, room suitability and travel distance [23] are not major concerns in this context, since preassigned rooms are only changed in the event of exceptional conflicts. Solving approaches for pattern-based timetabling methods include mixed-integer programming [20] and column generation approaches using Dantzig–Wolfe reformulation [21]. While these approaches help mitigate violations of soft constraints, they generally explore a much larger set of possible schedules than those arising in the block scheduling context considered here.

While the block structure may lead to high-quality timetables, it introduces a specific challenge : courses occupying half-blocks must be matched to form full blocks. A *half-block* is a subset of periods within a block used for most courses that require less instructional time. The motivation behind the use of half-blocks is that they increase the granularity level when building the schedule, allowing the remaining courses to be scheduled on well-balanced full blocks. Schools can therefore adjust the length of class periods so that courses requiring less instructional time are scheduled for exactly half of the periods within a block. As long as the minimum instructional time for each subject is respected, there is some flexibility to implement this. This structure facilitates the mixing of students across sections and therefore improves the effectiveness of the student sectioning process. However, the pairing of half-block courses is a critical step, as it directly impacts the balance of students across sections. Courses occupying half-blocks are most often elective courses, such as arts and STEM courses. These courses account for most of the variations in student curricula and are generally the most difficult to schedule. Moreover, the student sectioning problem has been shown to be NP-hard when more than one curriculum is involved [7]. Student sectioning can be performed before, during, or after the timetabling process [7, 24]. Some approaches also

adopt an iterative strategy, alternating between timetabling and student sectioning stages in order to progressively refine the solution [25, 26]. Balancing course sections is important to ensure that room capacity limits are respected. It is also essential from an educational perspective, as highly imbalanced classes may disadvantage students and place additional burden on teachers. The practical importance of maintaining balanced sections, combined with the computational difficulty of the student sectioning problem, makes this problem particularly relevant to study.

Contributions. Our work is motivated by this practical context. *Dash Computer Solutions* is a company responsible for the annual design of more than a hundred schedules in Quebec. Currently, these schedules are built manually, based on the expert knowledge accumulated by operators over the years. The process typically begins with a hand-crafted initial solution, which is then iteratively improved through locally designed adjustments performed by human operators. Although this approach is capable of producing high-quality schedules, it is time-consuming : generating a schedule can require more than ten person-days per school, excluding additional time spent on maintenance, data processing and other operational tasks. Our work aims to improve this process. We propose a constraint programming (CP) approach to generate feasible schedules for half-block courses that promote effective student balancing. In particular, the model incorporates student course selections in order to favor well-balanced student sectioning. The proposed model is designed to improve computational efficiency, accommodate additional practical constraints, and reduce the need for manual adjustments by scheduling operators. The approach is evaluated on four anonymized real-world datasets from Canadian high schools for the 2024-2025 academic year, provided directly by Dash Computer Solutions, our industrial partner. The results show that our approach produces solutions of comparable quality to those obtained through a human-guided local search starting from a hand-crafted initial solution, while reducing the designing time from 10 hours to an hour in the best case. Finally, the proposed approach has been integrated with their existing student sectioning algorithm and will be used starting in 2026 to generate schedules for the upcoming academic years.

Related timetabling tasks with CP. This paragraph reviews CP approaches that, while relevant, cannot be directly applied to the specific context considered in this work. Dimitsas et al. (2022) [30] address a post-enrollment timetabling problem of a scale comparable to those encountered in Quebec. Their approach relies on a mixed-integer programming model strengthened by CP, which is used to search for local improvements and to verify non-linear constraints. Nogareda and Camacho (2017) [26] propose an iterative approach in which

the timetabling problem is solved using a CP model and the student sectioning problem is addressed using an ant colony metaheuristic. However, both of these hybrid approaches operate at the level of individual lectures, whereas the problem considered in this work is defined at the course-block level. Also using CP, Demirović and Stuckey (2018) [31] present a complete model with warm starts for high school timetabling, achieving good results on instances of similar size to those considered in this work. In their setting, students are already assigned to sections, and therefore the model addresses only the timetabling problem and does not consider the student sectioning component. Finally, Valouxis and Housos [32] propose a hybrid approach that iterates between generating a timetable using a CP model and reducing the search space through local search around the obtained timetable. While this method yields good results, it has been evaluated on smaller instances and operates at the level of individual course periods.

Structure of the paper. The paper is organized as follows. Section 3.3 provides a detailed description of the problem. Section 3.4 presents the proposed CP model, and Section 3.5 reports and discusses the results.

3.3 Problem description

High schools with a sufficient number of students may offer multiple sections of the same course. In this context, a specific section of a course is referred to as a *course-group*. A course-group is identified by the course code and a section number. For example, if two sections of an art course are offered in Secondary 3, they may be labeled `ART302-01` and `ART302-02`. In high school timetabling, the periods assigned to a course-group should be well distributed throughout the timetable cycle. In addition, the total instructional time for each course must comply with the requirements set by the governing board [36]. To satisfy these constraints, many schools adopt a timetable cycle typically spanning 6 or 9 days, during which students follow a full schedule with no free periods. The cycle is then divided into predefined fixed patterns, called *blocks*, as illustrated in Table 3.1. In this example, there are 9 blocks (`A, B, C, D, E, F, G, H, J`). Each block contains an even number of meeting periods that are well distributed across the cycle. This configuration, with a 9-day timetable cycle and a fixed block pattern, is commonly observed in high schools in Quebec.

Introducing half-blocks. With this structure, the timetabling task consists of assigning course-groups to blocks. Most courses require a number of periods corresponding to a regular block, which represents four periods in the example shown in Table 3.1. However, some

Period	D1	D2	D3	D4	D5	D6	D7	D8	D9
1 (8 :30am to 9 :30am)	**A1**	E1	J1	D2	H2	C3	G3	B4	F4
2 (9 :45am to 10 :45am)	B1	F1	**A2**	E2	J2	D3	H3	C4	G4
3 (1 :00pm to 2 :15pm)	C1	G1	B2	F2	**A3**	E3	J3	D4	H4
4 (2 :30pm to 3 :45pm)	D1	H1	C2	G2	B3	F3	**A4**	E4	J4

TABLEAU 3.1 Block pattern for a timetable cycle of 9 days and 4 periods per day. A course-group scheduled on block **A** will be given at period 1 on day 1, period 2 on day 3, period 3 on day 5 and period 4 on day 7.

courses require fewer or more meeting periods. For course-groups that require fewer periods, a half pattern will be used. Courses assigned to half-block patterns are frequently elective courses such as arts or sports. These courses are scheduled on *half-blocks* and are referred to as *half-block courses*.

These half-block courses are typically designed so that they can be combined to form a complete block. For example, in Table 3.1, a student could take an *arts* course-group in block periods A1 and A3 and a *sports* course-group in block periods A2 and A4, thereby completing block A. This structure ensures that half-block courses remain evenly distributed across the timetable cycle. In this example, periods A1 and A3 correspond to the first half-block of block A, denoted A13, while periods A2 and A4 correspond to the second half-block, denoted A24. The half-block A24 is the *complement* of A13, and vice versa. Half-blocks such as A13, B13, and C13 are referred to as *odd blocks*, because they contain periods with an odd number, whereas the other half-blocks (A24, B24, and C24) are referred to as *even blocks*.

With this structure, the timetable can be represented as columns of clusters of course-groups assigned to the same half-block, as illustrated in Table 3.2. Each student is then assigned to exactly one course-group per half-block (e.g. ART302 to half-block A13). In this example, two thirds of the students would take a combination of arts (ART302) and sports (SPO302) in block A, and history (HIS304) in block B, while the remaining third would follow the inverse schedule. Note that the specific assignment of arts sections (i.e. ART302-01, ART302-02, and ART302-03) to half-blocks does not affect student sectioning, since students can move freely between sections. Moreover, these sections are taught by the same teacher and take place in the same room. As a result, permutations of these sections correspond to *symmetrical solutions*.

Introducing multiple curricula. In this example, all students take ART302, SPO302, and HIS304, meaning that there is only one *curriculum*. However, when multiple curricula are

Tableau 3.2 Timetable example for one curriculum : the × mark shows the course-group assigned to the corresponding half-block. All the course-group seat 32 students.

	students	A13	A24	B13	B24	C13	...	teacher	room
ART302-01	32	×						t1	r1
ART302-02	32		×					t1	r1
ART302-03	32			×				t1	r1
SPO302-01	32	×						t2	r2
SPO302-02	32		×					t3	r2
SPO302-03	32				×			t3	r2
HIS304-01	32	×	×					t4	r3
HIS304-02	32			×	×			t4	r3
HIS304-03	32			×	×			t5	r4

present, the pairing of half-block courses becomes a critical step, as it directly affects the balance of students across sections. Table 3.3 illustrates a case with two different curricula : all students take a sports course (SPO302), but they also choose either a music course (MUS302) or a visual arts course (ART302). Students' course selections are known in advance and remain fixed. Table 3.3 also shows an example with a good balance in the number of students per section. In practice, grouping half-block courses into as few blocks as possible has been observed to improve the balance of students across sections. Additionally, this structure benefits the school operationally, as it makes it easier to accommodate changes in a student's course selection during the year.

Tableau 3.3 Example for 2 curricula : the × mark shows the course-group assigned to the corresponding half-block. Note that there is a room conflict between course-groups SPO302-01 and SPO302-02.

	students	A13	A24	B13	B24	C13	...	teacher	room
ART302-01	34	×						t1	r1
ART302-02	30			×				t1	r1
ART302-03	30				×			t1	r1
MUS302-01	20	×						t2	r2
SPO302-01	27		×!					t3	r3!
SPO302-02	27		×!					t4	r3!
SPO302-03	30			×				t4	r3
SPO302-04	30				×			t4	r3

Introducing conflicts and closed schedules. Unfortunately, the timetable shown in Table 3.3 is infeasible because there is a room conflict between the first two sports groups. Moving one of these groups to half-block A13 would not resolve the issue, since all the

remaining students taking the art and music options in `A13` would then be assigned to the remaining sports section in `A24`, because of the block-pairing constraint. This limitation arises because the students must follow a closed schedule. A *closed schedule* is a schedule for which all half-blocks courses are matched with a course on the complementing half-block.

Forbidding full-block splitting. Splitting a full-block course across multiple half-blocks is theoretically possible, but it is generally discouraged. The first reason is that breaking a block increases the risk of creating poorly distributed schedules, potentially resulting in two meeting periods of the same course occurring on the same day, which is commonly forbidden. Another reason is that keeping full-block courses on their intended base pattern simplifies the management of rooms and teacher workloads, as these resources can be more easily allocated and adjusted. It also improves student sectioning, particularly when students are repeating courses from a previous grade, as sections of the same course from different grades can then be aligned more easily. Finally, doing so helps prevent conflicts with other parts of the timetable that will be assigned later. As commonly done, schools may also impose additional specific constraints. For example, they may require that two courses have at least one concurrent section in order to facilitate changes in students' course selections.

Summary. The problem considered in this paper can now be summarized as follows. The objective is to determine clusters of half-block course-groups that allow students to be distributed as evenly as possible across sections, while keeping the flexibility to potentially accommodate additional school-specific constraints. All solutions must satisfy the following hard constraints :

1. A teacher cannot teach more than one course at the same time.
2. A room cannot be assigned to more than one course-group at the same time.
3. A student cannot attend more than one course in the same half-block.
4. The number of blocks used in the timetable cannot exceed the fixed number defined by the block pattern (e.g. 9 blocks in Table 3.1).

3.4 Designing the CP model

High schools in Quebec typically involve, for grade levels secondary 3 to 5, between 140 and 1350 students, 30 to 180 different courses, 6 to 60 possible curricula, and 6 to 14 half-blocks. Considering the scale of these instances, the existing combinatorial constraints, and the need for a flexible modeling framework capable of accommodating practical constraints, constraint

programming appears to be a suitable paradigm for addressing this real-world problem. This section describes the CP model we designed for this purpose.

3.4.1 Decision variables

Briefly, solving this problem consists of assigning a half-block to each half-block course-group. Let $\mathcal{G}$ be the set of course-groups to schedule, and let $\mathcal{B}$ be the set of available blocks. The block b from set $\mathcal{B}$ is a subset of periods. These blocks can be split in two, to form half-blocks. The half-blocks of a block b are denoted by $h \in \mathcal{H}_b$ ($|\mathcal{H}_b| = 2 \; \forall b \in \mathcal{B}$). Each half-block $h \in \mathcal{H}_b$ of block b has a complementing half-block $\overline{h}$. This complementing half-block can be formally defined as $\overline{h} = b \backslash h$ for all $h \in \mathcal{H}_b$. The global set of all half-blocks $h \in \mathcal{H}$ represent the clusters to which half-block course-groups may be assigned, and cover the available blocks, i.e. $\bigcup_{h \in \mathcal{H}} h = \bigcup_{b \in \mathcal{B}} b$.

In Table 3.2, there are six half-block course-groups : `ART302-01`, `ART302-02`, `ART302-03`, `SPO302-01`, `SPO302-02`, and `SPO302-03`. Note that the history course (`HIS304`) occupies a full block and is therefore not part of the half-block assignment. These six half-block course-groups can be grouped into a minimum of three clusters of paired groups. However, since schedules must remain closed (i.e. half-blocks must be paired to form complete blocks), the number of clusters must be even. Consequently, the number of clusters is rounded to four, identified by two available blocks ($|\mathcal{B}| = |$`A, B`$| = 2$), each subdivided in two :

— $b_1 = $ `A` $= \bigcup_{h \in \mathcal{H}_{\texttt{A}}} h =$ `A13` $\cup$ `A24`,

— $b_2 = $ `B` $= \bigcup_{h \in \mathcal{H}_{\texttt{B}}} h =$ `B13` $\cup$ `B24`.

The number of blocks used for half-block courses $|\mathcal{B}|$ should nevertheless be kept as small as possible in order to leave as many blocks as possible available for full-block courses, which has empirically been shown to improve overall scheduling quality. In practice, the number of available half-blocks can be bounded by the maximum number of periods required by either the teacher who teaches the most, the most used room, or the longest curriculum.

Equation (3.1) defines decision variables $\textsc{hblock}[g]$ that specify, for each course-group $g \in \mathcal{G}$, the half-block $h \in \mathcal{H}$ to which the course-group is assigned.

$$\textsc{hblock}[g] \in \mathcal{H} \qquad \forall g \in \mathcal{G} \tag{3.1}$$

Let $\mathcal{C}$ be the set of curricula and $\mathcal{S}$ the set of courses from different subjects (e.g. arts, sports, etc.). For each course-group $g \in \mathcal{G}$, let $s_g \in \mathcal{S}$ denote the course associated with g. Let $\mathcal{G}_s \subseteq \mathcal{G}$ denote the set of course-groups of course s. For each curriculum $c \in \mathcal{C}$, let $\mathcal{S}_c \subseteq \mathcal{S}$ denote the set of courses included in curriculum c. Note that several course-groups may correspond to

the same course. In order to ensure that half-blocks can be combined within each curriculum to form complete blocks, additional decision variables are required. These variables ensure that at least one timetable is feasible per curriculum. Equation (3.2) defines the decision variables $\text{SCHED}[c, s]$, indicating the course-group from course $s \in \mathcal{S}_c$ in which a student from curriculum $c \in \mathcal{C}$ can sit without a conflict in their schedule.

$$\text{SCHED}[c, s] \in \mathcal{G}_s \qquad \forall c \in \mathcal{C}, \forall s \in \mathcal{S}_c \tag{3.2}$$

Intuitively, the sequence $(\text{HBLOCK}[\text{SCHED}[c, s]])_{s \in \mathcal{S}_c}$ gives a timetable for a student following curriculum c. Table 3.4 provides a summary of the sets and notation used in the model. Some of these elements are introduced later in the text when they become necessary.

TABLEAU 3.4 Main sets and notation used in the model. All sets are defined and explained throughout the model description.

Symbol	**Description**
n	Number of students
$\mathcal{C}$	Set of curricula
$\mathcal{L}$	Set of grade levels
$\mathcal{B}$	Set of available blocks
$\mathcal{H}$	Set of all half-blocks in the blocks of $\mathcal{B}$
$\mathcal{H}_b$	Set of half-blocks in block b
$\mathcal{H}_l$	Subset of half-blocks on which the solution is spread for level $l \in \mathcal{L}$
$\overline{h}$	Complementing half-block of half-block h
$\mathcal{T}$	Set of teachers preassigned to the course-groups
t_g	Teacher preassigned to the course-group g
$\mathcal{R}$	Set of rooms preassigned to the course-groups
r_g	Room preassigned to the course-group g
$\mathcal{S}$	Set of courses (course titles)
$\mathcal{S}_c$	Set of courses included in curriculum $c \in \mathcal{C}$
s_g	Course to which course-group g belongs
$low[s]$	Minimum enrollment allowed for a section of course s
$cap[s]$	Maximum enrollment allowed for a section of course s
A	Enrollment matrix, $A[s, s']$ is the number of students taking both s and s'
$\mathcal{G}$	Set of course-groups to be scheduled
$\mathcal{G}_s$	Set of course-groups corresponding to course $s \in \mathcal{S}$
$\mathcal{G}_t$	Set of course-groups taught by teacher $t \in \mathcal{T}$
$\mathcal{G}_r$	Set of course-groups taught in room $r \in \mathcal{R}$

3.4.2 Constraints

This section introduces the constraints enforced in our model. We consider three types of constraints : those aimed at *preventing conflicts*, those designed for *balancing students*, and finally those used for *breaking symmetries*. Although symmetry-breaking constraints are not required for correctness, they are necessary to keep the execution time tractable.

Constraints for preventing conflicts

The school assigns the teachers and rooms with a structure in mind that should allow for a feasible schedule. The following constraints define a feasible schedule.

Constraint 1 : No teacher schedule conflict. No teacher is solicited by two course-groups at the same time. Constraint C1 specifies that all periods assigned to teacher t are different, where $\mathcal{G}_t \subseteq \mathcal{G}$ denotes the course-groups that are assigned to teacher t.

$$\texttt{allDifferent}\Big(\text{HBLOCK}[g] \;\Big|\; g \in \mathcal{G}_t\Big) \qquad \forall t \in \mathcal{T} \qquad \text{(C1)}$$

Constraint 2 : No room conflict. No room is solicited by two course-groups at the same time. Constraint C2 specifies that all periods assigned to room r are different, where $\mathcal{G}_r \subseteq G$ denotes the course-groups that are assigned to room r.

$$\texttt{allDifferent}\Big(\text{HBLOCK}[g] \;\Big|\; g \in \mathcal{G}_r\Big) \qquad \forall r \in \mathcal{R} \qquad \text{(C2)}$$

Constraint 3 : Minimum number of half-blocks for a curriculum. A feasible schedule must assign course-groups to distinct half-blocks. Consequently, the course-groups selected for a feasible timetable must cover at least as many half-blocks as there are courses in the schedule. For each curriculum $c \in \mathcal{C}$, let $\langle s_1^c, s_2^c, \ldots, s_{|c|}^c \rangle$ denote the courses in curriculum c, sorted in increasing order of the number of available sections. Intuitively, courses with more sections would span over more half-blocks in the timetable. This leads to the following progressive requirement : the course-groups of course s_1^c must cover at least one half-block, the course-groups of courses $\{s_1^c, s_2^c\}$ must cover at least two half-blocks, and more generally, the course-groups of courses $\{s_1^c, s_2^c, \ldots, s_k^c\}$ must cover at least k distinct half-blocks. The course-groups associated with the set of courses $\{s_1^c, s_2^c, \ldots, s_k^c\}$ correspond to all $g \in \mathcal{G}$ such that $s_g \in \{s_1^c, s_2^c, \ldots, s_k^c\}$. Let us define the function $\texttt{nvalue}(E)$ returning the number of

different values in the collection E. The constraints is as follows.

$$\texttt{nvalue}\Big(\textsc{hblock}[g] \;\Big|\; g \in \mathcal{G} \wedge s_g \in \{s_1^c, ..., s_k^c\}\Big) \geq k \qquad \forall c \in \mathcal{C}, \forall k \leq |c| \qquad \text{(C3)}$$

Constraint 4 : Feasible schedules have all different half-blocks. A student cannot attend two course-groups simultaneously. Therefore, Constraint C4 enforces that the course-groups selected for a given curriculum are assigned to distinct half-blocks.

$$\texttt{allDifferent}\Big(\textsc{hblock}\Big[\textsc{sched}[s, c]\Big] \;\Big|\; s \in \mathcal{S}_c\Big) \qquad \forall c \in \mathcal{C} \qquad \text{(C4)}$$

Constraint 5 : Feasible schedules are closed. Feasible schedules must be closed, meaning that every half-block course-group must be paired with a course-group assigned to its complementary half-block. To enforce this constraint, auxiliary variables $\mathcal{H}_c$ are first introduced.

$$\mathcal{H}_c = \Big\{\textsc{hblock}\Big[\ \textsc{sched}[s, c]\ \Big] \;\Big|\; s \in \mathcal{S}_c\Big\} \qquad \forall c \in \mathcal{C} \qquad (3.3)$$

These variables represent the set of half-blocks to which the course-groups of a feasible schedule for curriculum c are assigned through the decision variable HBLOCK. For every half-block h in $\mathcal{H}_c$, the complementary half-block $\overline{h}$ must also belong to the schedule. Constraint C5 therefore ensures that half-blocks are paired to form complete blocks using $\texttt{count\_eq}(E, v, c)$. This constraint ensures that value v is found exactly c times in array E.

$$\texttt{count\_eq}(\mathcal{H}_c, \overline{h}, 1) \qquad \forall c \in C, \forall h \in \mathcal{H}_c \qquad \text{(C5)}$$

3.4.3 Constraints for balancing students

This second set of constraints aims to balance students across sections. For each course s, the school may specify the minimum and maximum number of students allowed in any section, denoted by the parameters $low[s]$ and $cap[s]$, respectively.

Constraint 6 : At most all students can take a half-block course at the same time. We introduce the auxiliary variable sets $\mathcal{G}_h$, regrouping all course-groups assigned to half-block h.

$$\mathcal{G}_h = \Big\{\ g \in \mathcal{G} \;\Big|\; \textsc{hblock}[g] = h\ \Big\} \qquad \forall h \in \mathcal{H} \qquad (3.4)$$

Constraint C6 then bounds the number of seats offered in each half-block by the total number of students (n).

$$\sum_{g \in \mathcal{G}_h} low[s_g] \leq n \qquad \forall h \in \mathcal{H} \qquad \text{(C6)}$$

Constraint 7 : Complementing course-groups are linked by the student course selection. A co-enrollment matrix A, provided by the school prior to scheduling, indicates for each pair of courses (s, s') the number of students $A[s, s']$ enrolled in both courses. To obtain closed schedules, the course-groups $\mathcal{G}_h$ assigned to a half-block h must correspond to courses whose students are also enrolled in at least one different course from the course-groups $\mathcal{G}_{\overline{h}}$ assigned to the complementary half-block $\overline{h}$.

$$\sum_{g' \in \mathcal{G}_{\overline{h}} \mid s_{g'} \neq s_g} \Big(A[s_g, s_{g'}] \Big) \;\geq\; low[s_g] \qquad \forall h \in \mathcal{H}, g \in \mathcal{G}_h \qquad \text{(C7)}$$

Constraint 8 : Bounds on acceptable number of students per cluster. This constraint ensures that the minimum number of students potentially assigned to one half-block never exceed the maximum number of students that can be accommodated in the complementary half-block. The minimum number of students assigned to a half-block h is obtained by summing the minimum enrollments $low[s_g]$ for the courses corresponding to the course-groups in $\mathcal{G}_h$. Similarly, the maximum number of students that can be assigned to the complementary half-block $\overline{h}$ is obtained by summing the maximum enrollments $cap[s_{g'}]$ for the courses corresponding to the course-groups in $\mathcal{G}_{\overline{h}}$.

$$\sum_{g \in \mathcal{G}_h} low[s_g] \leq \sum_{g' \in \mathcal{G}_{\overline{h}}} cap[s_{g'}] \qquad \forall h \in \mathcal{H} \qquad \text{(C8)}$$

3.4.4 Constraints for breaking symmetries

A specificity of this problem is that the particular half-block assigned to a course-group (e.g., `ART302-01` or `ART302-02`) does not affect the feasibility, nor the quality, of the solution when half-blocks courses are scheduled first. This creates many symmetries, since courses may have multiple sections and students have no preference regarding the specific section to which they are assigned. Consequently, identifying and breaking these symmetries can significantly reduce the search time. To break symmetries, we consider courses in increasing order of the number of sections they offer. Within each course, the corresponding course-groups are already indexed and sorted by section number (e.g. `ART302-01` before `ART302-02`).

Table 3.5 illustrates an example of a schedule that is progressively reordered as the symmetry-breaking constraints are applied. It can be assumed that every teacher has their own room, in which they teach all their course-groups. In this example, MUS302 is the course with the fewest sections, followed by ART302, and then SPO302. The course-groups are considered from top to bottom. The initial clusters of course-groups in this example are : {ART302, SPO302, SPO302}, {MUS302, ART302, SPO302}, {ART302}, and {SPO302}.

TABLEAU 3.5 Example of symmetry breaking constraints applied progressively. Underlined marks shows difference from the schedules of the previous situation.

(a) Base example without symmetry constraints.

	A13	A24	B13	B24
MUS302-01		×		
ART302-01			×	
ART302-02		×		
ART302-03	×			
SPO302-01		×		
SPO302-02	×			
SPO302-03				×
SPO302-04	×			

(b) Adding Constraint C9 on schedule a).

A13	A24	B13	B24	teacher
	×			t1
	$\underline{\times}$			t2
		$\underline{\times}$		t2
×				t3
$\underline{\times}$				t4
	$\underline{\times}$			t4
$\underline{\times}$				t5
			$\underline{\times}$	t5

(c) Adding Constraints C9 and C10 on schedule a).

	A13	A24	B13	B24
MUS302-01				×
ART302-01	$\underline{\times}$			
ART302-02				×
ART302-03			$\underline{\times}$	
SPO302-01			×	
SPO302-02				×
SPO302-03		×		
SPO302-04			×	

(d) Adding Constraints C9, C10 and C11 on schedule a).

A13	A24	B13	B24	teacher
$\underline{\times}$				t1
×				t2
		$\underline{\times}$		t2
	$\underline{\times}$			t3
×				t4
	×			t4
	×			t5
			$\underline{\times}$	t5

Constraint 9 : Identical course-groups are strictly increasing. In a school timetable, the same teacher often teaches multiple groups of the same course in the same room. From the students' perspective, these groups are identical. As a result, exchanging these groups across half-blocks produces symmetrical solutions. To eliminate such symmetries,

these course-groups are assigned to strictly increasing half-block indices, as enforced by Constraint C9, where $t_g \in \mathcal{T}$ and $r_g \in \mathcal{R}$ are the teacher and room preassigned to the course-group g, respectively.

$$\texttt{strictly\_increasing}\Big(\textsc{hblock}[g] \;\Big|\; g \in \mathcal{G}_s \wedge t_g = t \wedge r_g = r\Big) \quad \forall s,t,r \in (\mathcal{S},\mathcal{T},\mathcal{R}) \quad \text{(C9)}$$

Example in Table 3.5b shows the schedule after breaking this symmetry. Compared with Table 3.5a, the identical course-groups `ART302-01` and `ART302-02` have been reordered. Similarly, course-groups `SPO302-01` and `SPO302-02`, as well as `SPO302-03` and `SPO302-04`, are reordered.

Constraint 10 : Symmetries between blocks. This constraint indicates that no distinction should be made between blocks. We formalize it using the MiniZinc predicate `value_precede_chain`(c, X), which ensures that the value $c[i]$ appears before $c[i+1]$ in the array of variables X. Constraint C10 applies this predicate to an array of increasing odd half-blocks. It enforces that the first occurrence of each odd half-block appears in increasing order within the HBLOCK variable array. The HBLOCK array itself is ordered according to increasing course-group identification numbers.

$$\texttt{value\_precede\_chain}\Big(\Big[h \in \mathcal{H} \mid h \text{ is an odd half block}\Big], \textsc{hblock}\Big) \quad \text{(C10)}$$

We note that blocks with even identifiers cannot be considered for this constraint, since the uniqueness of the solution lies in the pairing between each odd block and its complementary even block. The example in Table 3.5b shows that the odd half-block `A13` has its first course-group, `ART302-03`, appearing after the first course-group in odd half-block `B13`, namely `ART302-02`. Therefore, these blocks are exchanged, together with their complementary half-blocks, to satisfy Constraint C10, yielding the configuration shown in Table 3.5c. In addition, to satisfy Constraint C9, some course-groups must also now exchange clusters : `ART302-01` with `ART302-02`, and `SPO302-03` with `SPO302-04`.

Constraint 11 : Symmetries between complementing half-blocks. Similarly, Constraint C11 ensures that paired half-blocks appear in a canonical order in the solutions. Specifically, for each pair of complementary half-blocks $(h, \overline{h})$, the odd half-block h must contain the course-group with the smallest identification number compared to those assigned to the even half-block $\overline{h}$. Lexicographic constraint `value_precede`(s, t, E) ensures that value s precedes value

t in collection E.

$$\texttt{value\_precede}\Big(h, \overline{h}, \text{HBLOCK}\Big) \qquad \forall h \in \mathcal{H} | h \text{ is an odd half block} \tag{C11}$$

In Table 3.5c, this constraint is satisfied for block `A` : the odd half-block `A13` contains the course-group `ART302-01`, which precedes `SPO302-03` assigned to the complementary half-block `A24` in the course-group ordering. Therefore, the relative order between these clusters remains unchanged in Table 3.5d. However, the symmetry must be broken for half-blocks `B13` and `B24`. Their first course-groups appear in reversed order, which violates the constraint. As a result, the course-groups in these two columns are exchanged. Furthermore, to satisfy Constraint C9, the odd half-blocks `A13` and `B13` must also be reordered. Finally, the sections of identical courses are reordered accordingly.

Constraint 12 : Fixing the first half-block variable. The first course-group is assigned to the first half-block. In Table 3.5d, `MUS302-01` is therefore placed on the first half-block.

$$\text{HBLOCK}[1] = \texttt{A13} \tag{C12}$$

3.4.5 Objectives

We recall that the objective is to obtain a well-balanced schedule. In an ideal solution, each section of a course would contain the same number of students. However, student sectioning is not performed within the CP model, as including it would make the problem too large to solve within an acceptable time (i.e., it would introduce an additional decision variable for each student) and may require school-specific constraints. Instead, an alternative metric is used to evaluate the solutions.

Objective 1 : Theoretical balance. Solving the model described above yields an assignment of course-groups to half-blocks. Assuming an ideal student sectioning, all sections of a course would contain the same number of students. Consequently, in a suitable timetable, the total ideal number of students per group on a half-block should be close to the average number of seats offered per half-block for each grade level considered. Maintaining this balance facilitates the subsequent student sectioning process. We refer to this metric as the

theoretical balance.

$$\textsc{theoreticalBalance} = \sum_{l \in \mathcal{L}} \sum_{h \in \mathcal{H}} \left| \underbrace{\left(\sum_{\substack{g \in \mathcal{G}_l \\ \textsc{hblock}[g]=h}} \frac{A[s_g, s_g]}{|\mathcal{G}_{s_g}|} \right)}_{\text{Ideal number of students per group}} - \underbrace{\left(\frac{\sum_{s \in \mathcal{S}_l} A[s,s]}{|\mathcal{H}_l|} \right)}_{\text{Number of seats offered}} \right| \tag{3.5}$$

For each level $l \in \mathcal{L}$ and each block $h \in \mathcal{H}$, the model computes the difference between the ideal number of students expected in the groups assigned to that half-block and the number of seats that should ideally be offered in that half-block. These differences are then aggregated over all levels and blocks. The ideal number of students per group corresponds to the average enrollment of a course assuming that students are evenly distributed across all its sections. The number of seats that should be offered in a block is obtained by summing the available seats for all courses at level l and dividing this total by $|\mathcal{H}_l|$, the number of half-blocks used for that level. This metric is also used by our industrial partner to build a first solution prior the student sectioning phase.

Objective 2 : Actual spread after student sectioning. The solution produced by the CP model is then passed to the student sectioning algorithm of the industrial partner. This algorithm iteratively assigns students using a greedy heuristic that has historically produced excellent results within seconds. After student sectioning, the *spread* metric is computed. The spread of a course is defined as the difference between the number of students in its largest and smallest sections, as formalized in Equation 3.6, where $N_{g,s}$ denotes the number of students enrolled in group g for course s. This value corresponds to the actual imbalance observed in practice once all students have been assigned to sections. Consequently, it can only be computed *a posteriori* after student sectioning. The objective is to minimize the spread, thereby improving the balance of students among course sections.

$$\textsc{spread} = \sum_{s \in S} \Big(\max(N_{g,s} \mid g \in \mathcal{G}_s) - \min(N_{g,s} \mid g \in \mathcal{G}_s) \Big) \tag{3.6}$$

3.5 Experimental results

The goal of the experiments is to evaluate the performance of the proposed CP model in generating suitable schedules. The solutions are assessed using both the theoretical balance, which is the metric minimized by the model, and the actual spread observed after student sectioning. The student course selections and course-group information are provided by our

industrial partner.

The CP model is implemented in MiniZinc [29] and solved using Gecode 6.3.0. The solver configuration is kept to its default settings and heuristics. The input data for the CP model are preprocessed using a Python script, which then calls the MiniZinc model through their Python API. The output produced by the CP model is subsequently passed to the student sectioning algorithm of the industrial partner to complete the timetable construction. Finally, all experiments were conducted on a laptop equipped with an Intel i7-1065G7 CPU (1.30 GHz) and 8 GB of RAM.

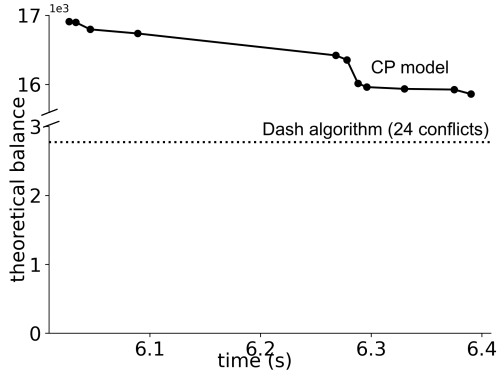
Description of the instances and baselines. Our approach is evaluated on four real-world instances from high schools in the province of Quebec for the 2024–2025 academic year. Their characteristics are summarized in Table 3.6. The results are compared with the schedules that were accepted and used by the schools during that year, which we refer to as the *historical solutions*. It is difficult to provide an exact estimate of the time that has been required to produce these schedules, as it depends on the operator guiding the search as well as on the number of iterations (or restarts) needed following consultations with the school. As a rough estimate, one iteration typically takes between 3 and 12 hours, and at least 2 iterations are required. For example, instance `school-2` took over 5 hours per iteration to schedule and took two iterations, for a total of a day and a half of work for the operator. Moreover, these solutions may violate some constraints due to late requirements introduced by the schools and may therefore not be reachable by our CP model.

Tableau 3.6 Test instances features : numbers of students (n), courses ($|\mathcal{S}|$), total course-groups ($|\mathcal{G}|$), curricula ($|\mathcal{C}|$), half-blocks ($|\mathcal{H}|$), teachers ($|\mathcal{T}|$), rooms ($|\mathcal{R}|$), groups per curriculum ($|\mathcal{G}|/(|\mathcal{C}|$), and course-groups per curriculum ($size(c)$).

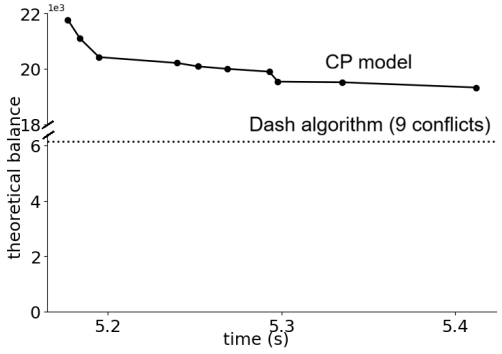
Instance	n	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{H}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{G}\rvert/\lvert\mathcal{C}\rvert$	$size(c)$
`School-1`	553	20	108	38	10	32	27	2.84	2 to 6
`School-2`	714	19	97	34	12	28	23	2.85	4
`School-3`	781	31	112	97	12	27	29	1.15	2 to 6
`School-4`	399	18	46	25	10	18	17	1.84	2 to 6

Our method is also compared with the hand-crafted approach used by the industrial partner to find a first feasible solution, which we refer to as the *Dash algorithm*. This algorithm iteratively assigns course-groups to blocks by greedily minimizing the theoretical balance while ensuring that no teacher or room conflicts occur. However, this approach may generate conflicts at the student level (i.e. conflicts on Constraints C3, C4, C6, C7 and C8).

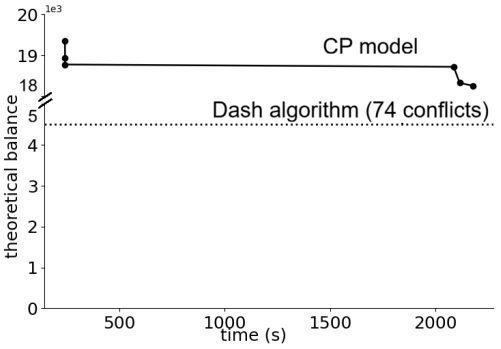
Result : Analysis of the theoretical balance. Figure 3.1 shows the evolution of the theoretical balance, used as the objective function, during the search for each instance. Instances `School-1` and `School-2` reach their best found solution quickly (in roughly 6 seconds). Instances `School-3` and `School-4` continue to produce improved solutions until approximately 37 and 10 minutes, respectively. These two instances are more challenging, mainly because they contain fewer groups per curriculum (lower ratio $|\mathcal{G}|/|\mathcal{C}|$). Consequently, students have fewer groups available for each course, which reduces the number of feasible scheduling options. Optimality has not been proven for any of the instances. Dash algorithm provides a solution almost instantly, with a significantly better theoretical balance. However, it generates many conflicts, i.e., students without a feasible schedule (74 for `School-3` and up to 221 for `School-4`), whereas no conflicts are produced by our CP model.



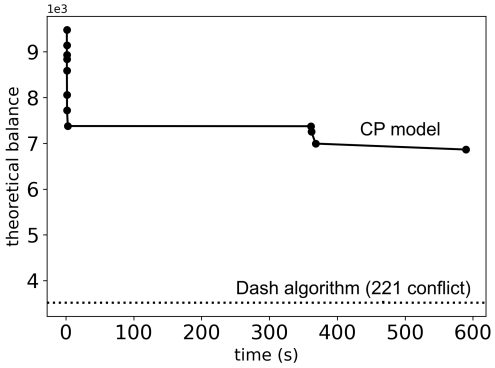
(a) Results for `School-1`.



(b) Results for `School-2`.



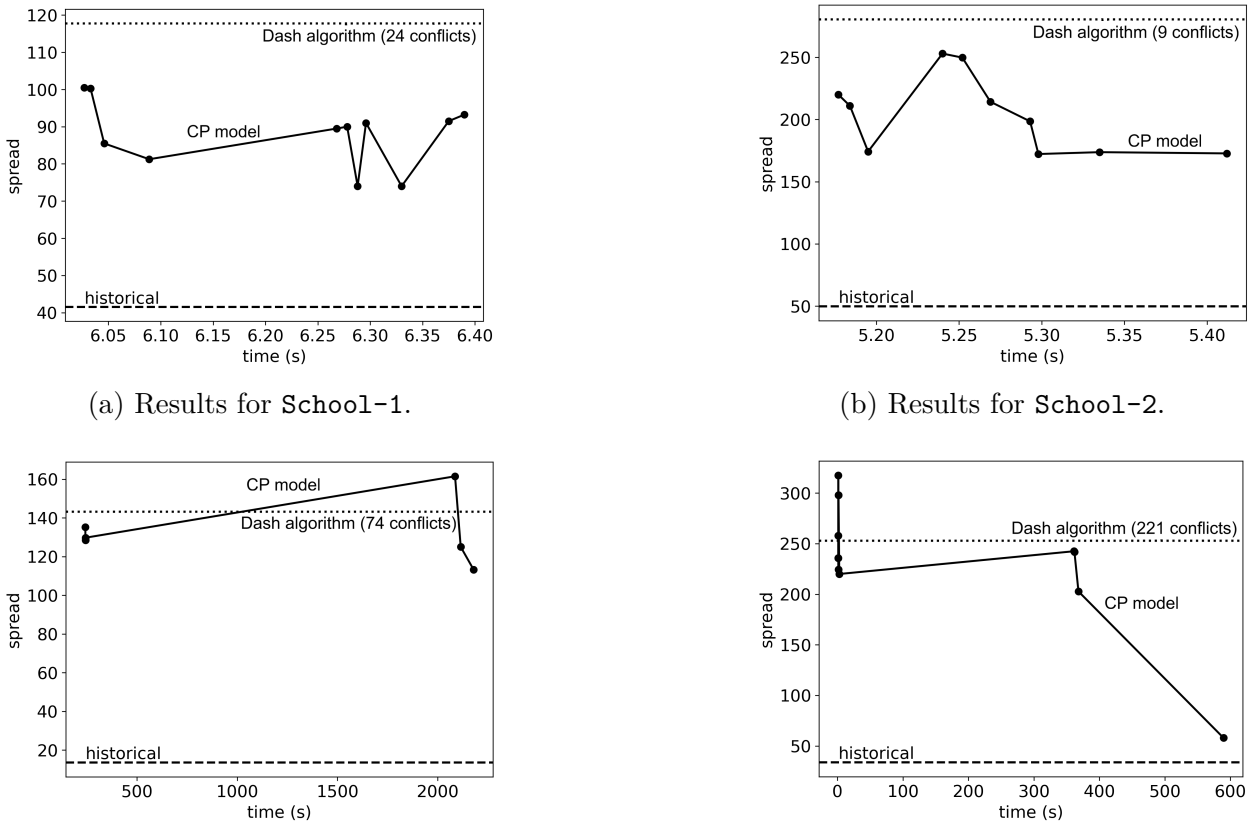
(c) Results for `School-3`.



(d) Results for `School-4`.

FIGURE 3.1 Theoretical balance in 60 min of search time. No improvements were observed after : (a) 6.39 s, (b) 5.33 s, (c) 2179 s, and (d) 590 s. Figures are truncated accordingly.

Result : Analysis of the actual spread. Figure 3.2 presents the actual *spread* obtained after student sectioning. As baselines, we report both the historical solution and the results of *Dash algorithm*. The results support the hypothesis that lower values of the theoretical balance tend to yield a lower spread. The CP model achieves a better total spread than the Dash algorithm on all instances, notably because there is no conflict that must be manually handled. The timetable obtained for `School-4` closely match the historical result and could be finalized with only a few minor manual adjustments during the student sectioning process.



(a) Results for `School-1`.

(b) Results for `School-2`.

(c) Results for `School-3`.

(d) Results for `School-4`.

FIGURE 3.2 Actual spread in 60 min of search time. No improvements were observed after : (a) 6.39 s, (b) 5.33 s, (c) 2179 s, and (d) 590 s. Figures are truncated accordingly. Thresholds indicate historical spread and spread with the Dash algorithm.

3.6 Conclusion

This paper addresses a specific variant of the curriculum-based timetabling problem arising in institutions that rely on predefined block patterns. We formulate and solve the half-block course-group clustering problem using constraint programming. The model was evaluated on

real instances from high schools, involving between 46 and 112 course-groups to schedule. On the instance 4, it produced a timetable of comparable quality to the historically approved solution for the 2024–2025 academic year roughly six time faster than the current manual process. Overall, the proposed approach generates solutions of moderate quality compared those obtained through human-guided local search, but does so with significantly reduced computation time. This makes the model useful for validating teacher and room preassignments, as well as the number of sections offered for each course. Moreover, the model provides a flexible framework that can be extended to incorporate additional practical constraints, such as enforcing that certain course-groups be scheduled simultaneously or restricting specific teachers from teaching during certain periods.

The approach has been integrated in the student sectioning algorithm of our industrial partner and will be used starting in 2026 to generate schedules for the upcoming academic years. As future work, alternative metrics derived from the co-enrollment matrix could be investigated to guide the CP search toward solutions that provide greater scheduling flexibility for students and further improve the quality of the resulting student sectioning.

CHAPITRE 4 RÉSULTATS COMPLÉMENTAIRES

Le chapitre qui suit présente des résultats complémentaires. La distribution des valeurs d'écarts, en anglais les valeurs de *spread*, est aussi obtenue par les tests effectués pour l'article de la section précédente, et est un autre indicateur de la qualité des solutions. Ces distributions sont analysées dans la section 4.1. Des analyses additionnelles générales sont présentées à la section 4.2.

4.1 Analyse de la distribution des valeurs d'écarts

La distribution des valeurs d'écarts, à la Figure 4.1 indique, après l'assignation des élèves, le nombre de cours par valeur d'écart. Les valeurs d'écart "P" signifient que les cours-groupes d'un cours sont parfaitement équilibrés. Les valeurs "2-4" correspondent au nombre de cours pour lesquels la différence d'élèves entre les plus petit et plus grand groupes correspond à une valeur entre 2 et 4 incluses. Il en va de même pour les autres valeurs d'écarts. Les valeurs dépassant 14, indiquées par "14+" sont toutes regroupées ensemble. Des valeurs au delà de 14 sont rarement acceptables, dépendamment du nombre d'élèves exact.

La figure 4.1 présente la distribution des valeurs d'écarts pour la solution ayant le meilleur écart total réel (*spread*) sur les instances testées dans le chapitre 3. Les valeurs historiquement acceptées (*historic score*), le résultat du modèle CP (*CP model score*) et les résultats de l'algorithme Dash (*Dash algorithm*) y sont présentés. Dans un horaire idéal, la distribution des écarts serait concentrée sur la gauche autour de "P" et tous les cours seraient presque parfaitement équilibrés. Cependant, lorsque ce n'est pas possible, il est préférable pour les écoles de distribuer les déséquilibres le plus possible. Par exemple, trois cours moyennement équilibrés valent mieux que : deux cours parfaitement équilibrés plus un cours extrêmement déséquilibré.

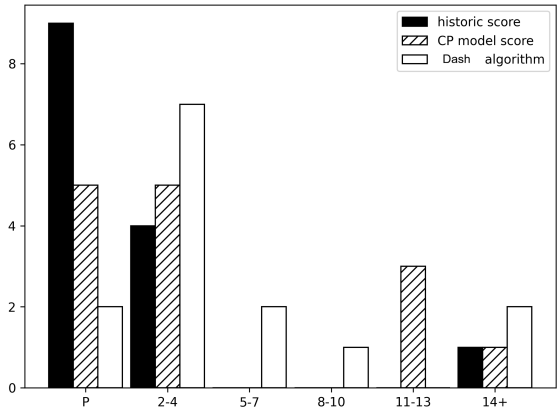
La figure 4.1 révèle que, bien que sur les autres instances les résultats du modèle CP ne soient pas près des résultats historiques, l'instance 1 est résolue avec un bon équilibre des écart par le modèle CP. En effet, la valeur du pire écart, à 15 élèves, pourrait être acceptable et la majorité des autres autres cours sont bien équilibrés. L'instance 4 pourraient aussi admettre une solution acceptable et bien équilibrée pour le modèle CP, si ce n'était de la pire valeur d'écart à 21 élèves. Les instances 1 et 4 ont en commun que leur solution par le modèle CP est construite sur moins de demi-blocs $|\mathcal{H}|$ que les autres. Les cours sont alors plus concentrés sur les demi-blocs, ce qui permet de mixer davantage d'élèves. Cela confirme la pratique

de concentrer les cours à demi-bloc sur le moins de demi-blocs possible, que Dash a su lier empiriquement à de meilleures solutions.

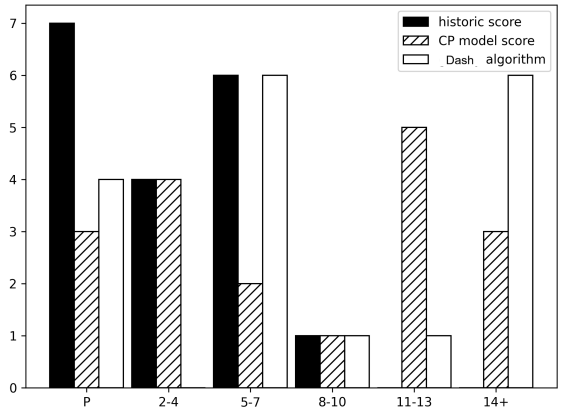
La distribution des élèves par le modèle CP est la pire pour l'instance 2, mais dans ce cas la solution CP a été construite sur 12 demi-blocs, contrairement aux 24 demi-blocs utilisés par la solution historique. Utiliser moins de demi-blocs est généralement mieux pour l'ensemble de l'horaire, comme il y a plus de blocs sur lesquels les élèves peuvent prendre des cours à bloc entier, mais ces résultats comparent uniquement l'horaire des cours sur demi-blocs. En outre, en regardant l'évolution de l'écart pour cette instance, à la figure 3.1 (b), il y a d'autres solutions d'écart similaire qui pourraient offrir une distribution différente. Globalement, le modèle CP offre une bien meilleure distribution que l'algorithme Dash, excepté sur l'instance 3. Cependant, comme l'algorithme Dash produit des conflits d'élèves, ces solutions ne sont en fait pas acceptables.

4.2 Discussions additionnelles

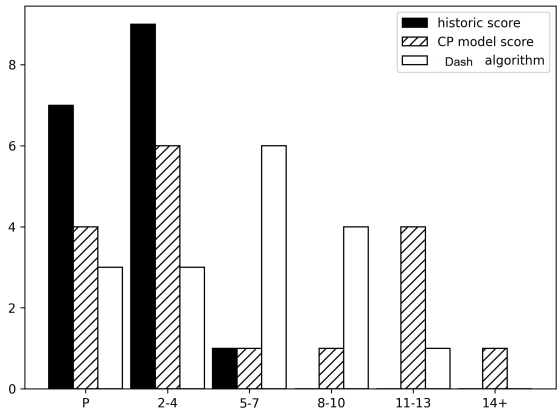
Outre les métriques discutées précédemment, le modèle CP a également des qualités non quantitatives. En effet, si le modèle ne retournait aucune solution, alors il permet d'invalider les pré-assignations d'enseignants et de locaux faites par la direction scolaire, puisque cela veut dire que l'agencement d'horaire n'est pas possible. Le modèle peut donc être utilisé rapidement à des fins de validation et aider l'opérateur à valider la structure d'horaire préparée. En outre, le modèle CP ne requiert pas d'intervention humaine et peut générer une solution initiale de meilleure qualité que l'algorithme Dash qui puisse être améliorée par l'opérateur selon les besoin, sauvant du temps de travail aux intervenants humains. En plus, le modèle CP ne génère pas de conflits d'élèves, contrairement à l'algorithme Dash, ce qui permet de valider également les choix de cours. L'algorithme pourrait aussi être adapté assez simplement pour minimiser le nombre de conflits d'élèves et indiquer à la direction scolaire le minimum de choix de cours nécessaires de changer. Par ailleurs, l'utilisation de l'algorithme CP ne demande pas autant de connaissances en optimisation de la part de l'opérateur comparé aux solutions historiques construites par une recherche locale guidée par l'opérateur. Enfin, des contraintes peuvent être ajoutées à la suite du modèle CP pour prendre en compte les demandes particulières des écoles, comme d'agencer des cours spécifiques au même moment par exemple.



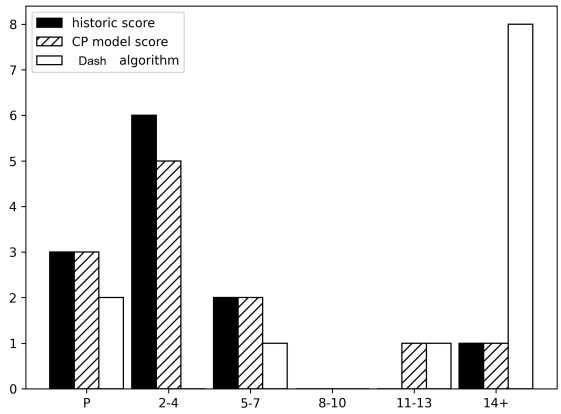
(a) Instance 1 : pire écart historique = 16, pire écart CP = 15, pire écart Dash = 37, conflicts d'élèves Dash = 8



(b) Instance 2 : pire écart historique = 9, pire écart CP = 19, pire écart Dash = 28



(c) Instance 3 : pire écart historique = 5, pire écart CP = 29, pire écart Dash = 13, conflicts d'élèves Dash = 65



(d) Instance 4 : pire écart historique = 15, worse CP écart = 21, worse Dash écart = 68, conflicts d'élèves Dash = 92

FIGURE 4.1 Nombre de cours-groupes par valeur d'écart. Les écarts de plus de 14 sont regroupés ensemble et la valeur exacte du pire écart est spécifiée sous l'instance.

CHAPITRE 5 CONCLUSION

Ce travail présente la modélisation et l'implémentation d'un modèle de programmation par contraintes pour le problème spécifique d'agencement basé sur le curriculum des cours ouverts sur demi-blocs qui vise à favoriser l'équilibre des groupes.

5.1 Synthèse des travaux

L'algorithme a été modélisé avec deux types de variables pour assurer l'agencement des cours et la possibilité pour les élèves de tous les curriculum de s'asseoir. Les paramètres du problème, notamment la matrice de choix de cours, servent à éliminer des solutions pouvant être agencées mais dont l'équilibre des groupes est inacceptable. L'article *Constraint Programming for Curriculum-based High School Timetabling with Half-Blocks* présente le problème, la modélisation, ainsi que les résultats sur les données de quatre écoles secondaires québécoises pour l'année scolaire 2024-2025. Les écoles testées comportent de 46 à 112 cours-groupes à agencer, et, pour la plus petite instance, le modèle de programmation par contrainte atteint une solution de bonne qualité, en un laps de temps environ six fois plus rapide. Globalement, le modèle génère des solutions de qualité modérée lorsque comparées à celles créées par la recherche locale guidée par un opérateur humain, mais en un temps significativement plus court. Le modèle présenté permet de valider rapidement la structure d'horaire préparée par la direction scolaire et demeure assez flexible pour que des contraintes spécifiques à l'école soient ajoutées au besoin.

5.2 Limitations de la solution proposée

La performance du modèle, et notamment la qualité des solutions dans le temps imparti est meilleure sur les écoles plus petites ou agencées sur moins de demi-blocs, compte tenu du nombre plus restreint de solutions possibles. L'amélioration de l'équilibre théorique ne donne pas non plus systématiquement un meilleur équilibre d'élèves réel, et le nombre de demi-blocs est fixé d'avance.

5.3 Améliorations futures

L'objectif du modèle pourrait être modifié afin d'améliorer les résultats, notamment en tenant compte de la matrice de sélection des choix de cours des élèves. L'utilisation de démarrage

à chaud, par exemple avec l'algorithme Dash corrigé pour éviter les conflits d'élève serait également une piste intéressante pour accélérer la recherche. En outre, le modèle présenté pourrait être généralisé pour être applicable à d'autres patrons de blocs. Dans certaines écoles, plutôt que de séparer les blocs en deux, les blocs sont plutôt séparés en trois, ce qui changerait la modélisation de la solution d'agencement. Enfin, l'explicabilité des résultats a beaucoup d'intérêt pour Dash, qui doit planifier la structure d'horaire avec la direction scolaire et justifier les impossibilités de respect de contraintes lorsque nécessaire. À cette fin, il serait utile de pouvoir extraire du modèle les contraintes qui causent l'échec ou qui réduisent le plus le nombre de solutions possibles.

RÉFÉRENCES

[1] Gouvernement du Québec, "Mise à jour des données : 1331 postes d'enseignantes et d'enseignants sont à pourvoir," September 2023. [En ligne]. Disponible : https://www.quebec.ca/nouvelles/actualites/details/mise-a-jour-des-donnees-1331-postes-denseignantes-et-denseignants-sont-a-pourvoir-50298

[2] Centrale des syndicats du Québec (CSQ), "Pénurie de main-d'œuvre : un défi urgent pour l'avenir du système scolaire," February 2025. [En ligne]. Disponible : https://www.lacsq.org/actualite/penurie-de-main-doeuvre-un-defi-urgent-pour-lavenir-du-systeme-scolaire/

[3] Gouvernement du Québec, "Plan québécois des infrastructures 2022-2023," Secrétariat du Conseil du trésor, Rapport technique, 2022. [En ligne]. Disponible : https://www.tresor.gouv.qc.ca/fileadmin/PDF/budget_depenses/22-23/6-Plan_quebecois_infrastructures.pdf

[4] M. Homsy et S. Savard, "Pénurie d'enseignants au québec : portrait et pistes de solution," Institut du Québec, Rapport technique, September 2019. [En ligne]. Disponible : https://institutduquebec.ca/wp-content/uploads/2019/09/201909-IDQ-PENURIEENSEIGNANTS.pdf

[5] Gouvernement du Québec, "Tableau de bord de l'éducation préscolaire, primaire et secondaire," 2025. [En ligne]. Disponible : https://www.quebec.ca/education/indicateurs-statistiques/prescolaire-primaire-secondaire/tableau-de-bord

[6] S. Ceschia, L. Di Gaspero et A. Schaerf, "Educational timetabling : Problems, benchmarks, and state-of-the-art results," *European Journal of Operational Research*, vol. 308, n°. 1, p. 1–18, 2023.

[7] M. Dostert, A. Politz et H. Schmitz, "A complexity analysis and an algorithmic approach to student sectioning in existing timetables," *Journal of Scheduling*, vol. 19, n°. 3, p. 285–293, 2016.

[8] Centre de services scolaire de Montréal, "Ratio maître-élèves," 2026. [En ligne]. Disponible : https://www.cssdm.gouv.qc.ca/formation-jeunes/ratio-maitre-eleves/

[9] A. Bettinelli, V. Cacchiani, R. Roberti et P. Toth, "An overview of curriculum-based course timetabling," *Top*, vol. 23, n°. 2, p. 313–349, 2015.

[10] J. S. Tan, S. L. Goh, G. Kendall et N. R. Sabar, "A survey of the state-of-the-art of optimisation methodologies in school timetabling problems," *Expert Systems with Applications*, vol. 165, p. 113943, 2021.

[11] M. A. A. Lam, F. Balderas, N. Rangel-Valdez, J. A. Martinez-Flores, J. A. Pérez-Vázquez *et al.*, "University timetabling problem : An analysis of the state of the art," *International Journal of Combinatorial Optimization Problems and Informatics*, vol. 16, n°. 3, p. 489, 2025.

[12] J. D. F. T. Filgueira, H. V. Siqueira et C. J. A. B. Filho, "A survey of the state of the art of educational timetabling problems," dans *Anais do XVII Congresso Brasileiro de Inteligência Computacional (CBIC 2025)*, F. G. aes, D. Ferreira et G. Barreto, édit. Belo Horizonte, MG : SBIC, 2025, p. 1–7.

[13] International Timetabling Competition 2007, "Curriculum-based course timetabling : Data and instances," 2007. [En ligne]. Disponible : https://www.eeecs.qub.ac.uk/itc2007/curriculmcourse/course_curriculm_index.htm

[14] S. Abdullah et H. Turabieh, "On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems," *information sciences*, vol. 191, p. 146–168, 2012.

[15] A. Kiefer, R. F. Hartl et A. Schnell, "Adaptive large neighborhood search for the curriculum-based course timetabling problem," *Annals of Operations Research*, vol. 252, n°. 2, p. 255–282, 2017.

[16] K. Sylejmani, E. Gashi et A. Ymeri, "Simulated annealing with penalization for university course timetabling," *Journal of Scheduling*, vol. 26, n°. 5, p. 497–517, 2023.

[17] International Timetabling Competition 2019, "Itc 2019 – international timetabling competition," 2019. [En ligne]. Disponible : https://www.itc2019.org/home

[18] D. S. Holm, R. Ø. Mikkelsen, M. Sørensen et T. J. R. Stidsen, "A mip based approach for international timetabling competition 2019," 2020.

[19] E. Rappos, E. Thiémard, S. Robert et J.-F. Hêche, "A mixed-integer programming approach for solving university course timetabling problems," *Journal of Scheduling*, vol. 25, n°. 4, p. 391–404, 2022.

[20] L. Saviniec, M. O. Santos, A. M. Costa et L. M. dos Santos, "Pattern-based models and a cooperative parallel metaheuristic for high school timetabling problems," *European Journal of Operational Research*, vol. 280, n°. 3, p. 1064–1081, 2020.

[21] N.-C. F. Bagger, G. Desaulniers et J. Desrosiers, "Daily course pattern formulation and valid inequalities for the curriculum-based course timetabling problem," *Journal of Scheduling*, vol. 22, n°. 2, p. 155–172, 2019.

[22] V. Cacchiani, A. Caprara, R. Roberti et P. Toth, "A new lower bound for curriculum-based course timetabling," *Computers & Operations Research*, vol. 40,

nº. 10, p. 2466–2477, 2013. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0305054813000506

[23] A. Bonutti, F. De Cesco, L. Di Gaspero et A. Schaerf, "Benchmarking curriculum-based course timetabling : formulations, data formats, instances, validation, visualization, and results," *Annals of Operations Research*, vol. 194, nº. 1, p. 59–70, 2012.

[24] E. Steiner, U. Pferschy et A. Schaerf, "Curriculum-based university course timetabling considering individual course of studies," *Central European Journal of Operations Research*, vol. 33, nº. 1, p. 277–314, 2025.

[25] V. Robert et A. Hertz, "How to decompose constrained course scheduling problems into easier assignment type subproblems," dans *International Conference on the Practice and Theory of Automated Timetabling*, 1995, p. 364–373.

[26] A.-M. Nogareda et D. Camacho, "Optimizing satisfaction in a multi-courses allocation problem combined with a timetabling problem," *Soft Computing*, vol. 21, nº. 17, p. 4873–4882, 2017.

[27] F. Rossi, P. van Beek et T. Walsh, édit., *Handbook of Constraint Programming*. Amsterdam : Elsevier, 2006.

[28] K. R. Apt, *Principles of Constraint Programming*. Cambridge : Cambridge University Press, 2003.

[29] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck et G. Tack, "Minizinc : Towards a standard cp modelling language," dans *International conference on principles and practice of constraint programming*. Springer, 2007, p. 529–543.

[30] A. Dimitsas, V. Nastos, C. Gogos et C. Valouxis, "An exact based approach for the post enrollment course timetabling problem," dans *Proceedings of the 26th Pan-Hellenic Conference on Informatics*, 2022, p. 77–82.

[31] E. Demirović et P. J. Stuckey, "Constraint programming for high school timetabling : A scheduling-based model with hot starts," dans *International conference on the integration of constraint programming, artificial intelligence, and operations research*. Springer, 2018, p. 135–152.

[32] C. Valouxis et E. Housos, "Constraint programming approach for school timetabling," *Computers & Operations Research*, vol. 30, nº. 10, p. 1555–1572, 2003.

[33] Gouvernement du Québec, "I-13.3, r. 8 — basic school regulation for preschool, elementary and secondary education, art. 28," https://www.legisquebec.gouv.qc.ca/en/version/cr/I-13.3%2C%20r.%208%20?code=se%3A28, Légis Québec, 2025, article 28 : Evaluation and promotion of students, updated 1 July 2025. [En ligne]. Dispo-

nible : https://www.legisquebec.gouv.qc.ca/en/version/cr/I-13.3%2C%20r.%208%20?code=se%3A28

[34] I. G. A. Premananda, A. Tjahyanto et A. Muklason, "Timetabling problems and the effort toward generic algorithms : A comprehensive survey," *IEEE Access*, vol. 12, p. 143 854–143 868, 2024.

[35] G. Colajanni et P. Daniele, "A new model for curriculum-based university course timetabling," *Optimization Letters*, vol. 15, n°. 5, p. 1601–1616, 2021.

[36] Gouvernement du Québec, "Loi sur l'instruction publique," https://www.legisquebec.gouv.qc.ca/fr/document/lc/i-13.3, 1988, rLRQ, c. I-13.3.