Les Cahiers du GERAD ISSN: 0711–2440

Dynamic constraint aggregation for the multiple-depot bus scheduling problem

N. Rasouli, G. Desaulniers, M. Saddoune, F. Soumis

G–2025–82

December 2025
Revised: January 2026

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2025
– Bibliothèque et Archives Canada, 2025

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2025
– Library and Archives Canada, 2025

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Dynamic constraint aggregation for the multiple-depot bus scheduling problem

Nadia Rasouli [a, b]

Guy Desaulniers [a, b]

Mohammed Saddoune [a, b, c]

François Soumis [a, b]

[a] *Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal, (Qc), Canada, H3T 1J4*

[b] *Group for Research in Decision Analysis (GERAD), Montréal (Qc), Canada, H3T 1J4*

[c] *Department of Computer Science, University of Hassan II, FST of Mohammedia, Casablanca, Morocco*

nadia.rasouli@gerad.ca
guy.desaulniers@gerad.ca

December 2025
Revised: January 2026
Les Cahiers du GERAD
G–2025–82

Abstract : Bus scheduling problem is a core optimization problem for public transit agencies. Given a set of timetabled trips to cover during a day and a homogeneous bus fleet assigned to multiple depots, the multiple-depot vehicle scheduling problem (MDVSP) consists in finding least-cost feasible schedules that cover each trip exactly once. To solve large-scale MDVSPs, we develop a column generation (CG) heuristic that is applied to a block-based set-partitioning model, where a block starts and ends at a depot without intermediate returns. To reduce degeneracy and improve performance, we combine CG with an improved dynamic constraint aggregation procedure (IDCA). We further devise a hybrid dual-disaggregation (HDD) step to accelerate convergence. Computational results on real-world instances with up to 6,296 trips show significant speed ups resulting from i) using a block-based model rather than a schedule-based model, ii) integrating CG with IDCA, and iii) applying HDD. Together, the these techniques yield an average speed up factor of 12.4 compared to CG alone applied to a schedule-based model, with only marginal degradation in the solution cost.

Keywords : Public transportation; bus scheduling; integer programming; column generation; dynamic constraint aggregation

Acknowledgements: We would like to thank the personnel of GIRO Inc. for providing the datasets and exchanging with us throughout the project. This work was funded by GIRO Inc. and the Natural Sciences and Engineering Research Council of Canada under the grants RDCPJ 524922–18 and RGPIN-2023–03791. This financial support was greatly appreciated.

1 Introduction

Among the various operations planning problems faced by transit agencies, the bus scheduling problem stands out due to its direct impact on both service quality and operational efficiency. This problem is a specific case of the vehicle scheduling problem (VSP), which is classified as the single-depot VSP (SDVSP) when all buses are assigned to a single depot, and the multiple-depot VSP (MDVSP) when they are shared by multiple depots. While the SDVSP can be solved in polynomial time, the MDVSP is known to be NP-hard (Bertossi et al., 1987).

Given a fixed timetable of trips over a day and a homogeneous fleet of buses, the MDVSP aims at constructing feasible bus schedules, ensuring that each trip is covered exactly once while minimizing the total of fixed vehicle and variable travel costs. A bus schedule is a feasible sequence of trips starting and ending at the same depot. Despite the growing interest in electric bus scheduling due to environmental considerations (Gerbaux et al., 2025; Ricard et al., 2025), we focus on diesel buses which remain widely used by public transit agencies and continue to be the subject of active research (Mousavi et al., 2025; Sadrani et al., 2025). We believe that the main ideas of our work can be applied to a next generation of algorithms for electric bus scheduling and, therefore, we also see our effort as a preliminary step towards these future algorithms.

To reduce the complexity of the MDVSP, several studies have proposed using a time-space network representation and solving the problem either exactly with a mixed-integer programming (MIP) solver (Kliewer et al., 2006) or using heuristics such as fix-and-optimize (Gintner et al., 2005). However, our industrial partner GIRO Inc., a leader in the development of optimization software for public transportation, requires a more flexible network representation that allows penalizing or forbidding some direct connections between trips because the connection time is too long or the connection links trips of two bus lines that should not be connected. In this case, it is difficult to rely on a time-space network. Instead, we adopt a connection-based network that explicitly models all feasible direct connections between trips.

In this work, we address the MDVSP using a connection-based network structure, which introduces significant computational challenges, particularly for large-scale instances involving thousands of trips (more than 6,000 in our case). In this context, column generation (CG) heuristics offer a suitable algorithmic framework for handling the problem's complexity and scale. Within this framework, we develop both modeling and solution strategies to devise an efficient CG approach. Indeed, instead of using a set-partitioning model where each column (variable) is associated with a complete bus schedule that potentially includes intermediate returns to the bus' depot, we propose a model in which columns represent bus blocks (sequences of trips without intermediate depot returns) and additional flow conservation constraints links these blocks to form implicit bus schedules. While the schedule-based model is comprehensive, it results in long columns that significantly increase degeneracy and the computational time. In contrast, the block-based model involves shorter columns, inducing faster CG convergence, reduced computational time in the pricing subproblems and overall, as confirmed by our computational experiments.

To further enhance performance, we integrate CG within an improved dynamic constraint aggregation (IDCA) algorithm that uses a novel hybrid dual disaggregation (HDD) strategy. As its name says, IDCA aggregates the trip covering constraints into clusters of trips that have a high probability of being serviced by the same bus and revises dynamically the proposed aggregation. To generate new variables, IDCA requires disaggregated dual variables, which are typically obtained by solving a linear program, called the complementary problem (CP), that considers a subset of the generated columns. In early IDCA iterations, this column subset is often not sufficiently diversified, resulting in arbitrary disaggregated dual values for many covering constraints. To address this limitation, we post-process the dual values obtained by the CP using HDD. Finally, to obtain integer solutions, the CG-IDCA framework is embedded within a diving heuristic that performs variable and inter-trip fixing.

The rest of this paper is organized as follows. Section 2 reviews the literature relevant to the MDVSP and outlines our main contributions. Section 3 formally defines the problem and presents the proposed block-based formulation. Section 4 describes the proposed solution algorithm, whereas Section 5 reports the computational results obtained on real-life instances. Finally, Section 6 concludes the paper and discusses future research directions.

2 Literature review

Introduced by Bodin et al. (1978), the MDVSP is a well-studied problem as surveyed in Ibarra-Rojas et al. (2015) and Perumal et al. (2022). Several exact algorithms have been devised over the years. The following two main methodologies have emerged: branch-and-price (B&P) embeds CG within a branch-and-bound search tree and branch-and-cut (B&C) as found in commercial MIP solvers. B&P has been first proposed by Ribeiro and Soumis (1994) and further applied in subsequent works on the MDVSP (Hadjar et al., 2006) or the MDVSP with time windows (Desaulniers et al., 1998; Hadjar and Soumis, 2009). All these works exploit a connection-based networks to generate schedule variables. With this methodology, the largest artificial instances solved to optimality contains around 900 trips. On the other hand, Kliewer et al. (2002, 2006) showed that, using time-space networks and an arc-flow model, B&C can solve to optimality real MDVSP instances involving more than 7,000 trips.

With the goal of improving the performance of the CG algorithm embedded in B&P algorithms for the MDVSP, Oukil et al. (2007) and Benchimol et al. (2012) have focused on solving the linear relaxation of a schedule-based model. In Oukil et al. (2007), the SPs rely on connection-based networks and the authors develop a dual variable stabilization (DVS) technique to control the oscillation of the dual values from one CG iteration to the next. This technique requires a good initial dual solution, which is computed by solving one or several SDVSPs. On artificial MDVSP instances with 3 depots and 500 trips, the stabilized CG algorithm yields an impressive speedup factor of more than 1000 for highly degenerate instances (with an average of 25 trips per schedule). This research avenue was pursued by Benchimol et al. (2012) who first designed a dynamic constraint aggregation (DCA) method to speed up CG and proposed to combine DVS and DCA to further accelerate CG. Employing time-space networks for the SPs, their tests on randomly generated instances with 3 depots and up to 1000 trips (around 19 trips per schedule on average) showed a speedup factor of 3.9 induced by DCA over CG. On larger instances with up to 3000 trips, integrating DCA and DVS yielded an average acceleration of 1.5 over DVS alone.

To compute integer solutions in fast computational times, a wide variety of heuristics have also been proposed. In particular, Gintner et al. (2005) developed a two-phase heuristic, called fix-and-optimize, that relies on time-space networks. In the first phase, several problem relaxations are solved and their solutions are examined to identify chains of trips that are present in all these solutions. In the second phase, these chains are fixed and the restricted MDVSP is solved using a MIP solver. This heuristic was able to solve instances with up to 11,062 trips in a few hours of computational time. Pepin et al. (2009) compared five heuristics, namely: CG heuristic, CG-based large neighborhood search (LNS), tabu search, truncated B&C, and Lagrangian heuristic. Their computational experiments on artificial instances with up to 1,500 trips showed that the CG heuristic yields the best solutions when sufficient time (at least 20 minutes) is available. Otherwise, the LNS heuristic outperforms the other ones. CG heuristics were also proposed by Guedes and Borenstein (2015) and Guedes et al. (2016), while Laurent and Hao (2009) designed an iterated local search heuristic that yields better results than the LNS and tabu search metaheuristics of Pepin et al. (2009).

In the last decade, research has focused on electric bus scheduling, considering that electric buses have a limited driving range and may require to recharge at dedicated locations (see Perumal et al., 2022; Gkiotsalitis et al., 2023; Gerbaux et al., 2025). As reported in these papers, some exact algorithms and several types of (meta)heuristics have been proposed for multiple problem variants. We highlight that CG algorithms have been developed in several studies (van Kooten Niekerk et al., 2017; Zhang

et al., 2021; Yıldırım and Yıldız, 2021; Gerbaux et al., 2025), as keeping track of the battery state-of-charge can be better handled in the CG pricing subproblem.

To the best of our knowledge, no bus scheduling studies have compared block-based and schedule-based models solved by CG. On the other hand, such a comparison was conducted by Mingozzi et al. (2013) for the multi-trip vehicle routing problem (MTVRP), where least-cost daily schedules composed of several non-overlapping routes (trips starting and ending at the depot) must be computed to service a set of customers. Their computational results obtained using exact column-and-cut generation algorithms show that it is much easier to solve a route-based model than a schedule-based model due to the length of the columns (a schedule contains much more customers than a route). Applying exact B&P algorithms, Hernandez et al. (2016) draw a similar conclusion for another variant of the MTVRP.

To sum up, CG solution approaches for the MDVSP has considered schedule-based formulations. Moreover, time-space networks reduce the size of arc-flow models, enabling the solution of very large-scale MDVSP instances by MIP solvers, but they lack the flexibility required to incorporate practical connection-specific soft or hard restrictions. In contrast, motivated by the practical needs of our partner GIRO, we adopt a connection-based network for solving large-scale MDVSP instances involving up to 6,296 trips, 3 depots, and 281 bus stations. To manage this complexity and efficiently solve large real-world instances, we introduce a compact block-based model and develop a CG heuristic, enhanced with IDCA and a novel HDD strategy. Despite the high potential of the DVS technique studied in Oukil et al. (2007) and Benchimol et al. (2012), we have decided to not consider DVS in our work because it requires a good initial dual solution. As shown in these works, such a solution is available for the MDVSP but would not be easily available for an electric bus variant of the problem.

3 Problem statement and mathematical formulation

In this section, we state the MDVSP, present a block-based formulation for it, and introduce a time-point aggregation to reduce model size.

3.1 Problem statement

Let T be a set of n bus trips to operate over a one-day time horizon. A trip is defined by a start location ℓ_t^S, a start time h_t^S, an end location ℓ_t^E, and an end time h_t^E. A bus fleet is available to operate these trips and shared by a set of depots D. We assume that a sufficiently large number of buses is available at each depot and that the bus fleet is homogeneous. The proposed methodology can be easily adapted if these assumptions do not hold.

The MDVSP consists in finding least-cost feasible bus schedules such that each trip in T is covered by a single bus. A bus schedule is defined as a complete daily sequence of trips assigned to a bus, starting from a depot and ending at the same depot. A schedule may include one or more intermediate returns to the bus' depot during the day. In this case, the schedule is divided into a sequence of bus blocks, each representing a continuous sequence of trips operated without returning to the depot between them.

A connection occurs between two trips if they are covered consecutively by the same bus. A connection is said to be *direct* if it does not contain a return to the depot between the two trips. Denoting by δ_{ij} the travel time between two locations i and j, a direct connection between trips t_1 and t_2 is feasible if $h_{t_2}^S - h_{t_1}^E \geq \delta_{\ell_{t_1}^E, \ell_{t_2}^S}$. To limit excessive unproductive driver time, connection time $h_{t_2}^S - h_{t_1}^E$ must not exceed a given maximum duration (45 minutes for our tests). In certain cases, specific direct connections may be forbidden or penalized to reflect operational preferences or network restrictions, or to discourage undesirable routing options. Indirect connections are not forbidden or penalized.

The cost structure comprises fixed and variable components. Each bus used incurs a fixed cost C that is sufficiently large to minimize the number of buses used. Variable costs (proportional to travel distance for the buses and to travel and waiting times for the drivers) are accounted for deadheads, pull-outs, pull-ins, and waiting times outside the bus' depot (i.e., during direct connections). A *deadhead* refers to the movement of a bus without passengers, directly from the end of a trip to the start of another. Similarly, a *pull-out* and a *pull-in* refer to movements without passengers between the depot and the start of a trip and between the end of a trip and the depot, respectively. Waiting at a depot is not penalized, as buses are not assigned drivers when they are parked there. Variable costs also include penalties for undesirable connections when applicable. Note that no variable costs are incurred by the trips as their total cost is a constant.

3.2 Block-based formulation

In this section, the MDVSP is formulated as a block-based set partitioning model with additional flow conservation constraints to count the number of buses in operation at any time of the day. The flow conservation constraints at each depot $d \in D$ are defined according to a list of time points that includes two times for each trip $t \in T$: $h_t^S - \delta_{d,\ell_t^S}$, the latest start time from depot d to arrive on time to operate trip t, and $h_t^E + \delta_{\ell_t^E,d}$, the earliest end time at the depot after operating trip t. Let $I = \{1, 2, \ldots, 2n\}$ be the index set of the $2n$ time points. We denote these points by M_i^d, $i \in I$, and we assume without loss of generality that they are sorted in chronological order (i.e., $M_i^d \leq M_j^d$ if $i < j$), positioning end times before start times in case of ties. Let i^+ and i^- denote the indices of the time points immediately following and preceding index i, respectively, with $2n^+ = 1$ and $1^- = 2n$ to indicate the wrap-around between the last and first time points. For each $i \in I$, we define a non-negative integer variable u_{i,i^+}^d to count the number of buses idling at depot d between the consecutive time points M_i^d and $M_{i^+}^d$. In particular, variable $u_{2n,1}^d$ specifies the total number of buses used from depot d. Figure 1 illustrates time points and corresponding variables at a depot d, where the white (resp., grey) nodes represent arrivals (resp., departures).

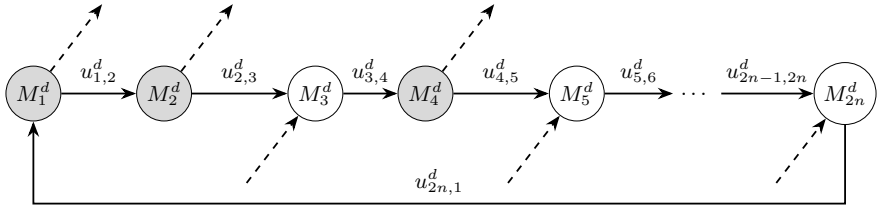

Figure 1: Example of time points and related u_{i,i^+}^d variables at a depot d

The following additional notation is also needed. Let B_d be the set of feasible blocks for buses from depot $d \in D$. For each block $b \in B_d$, we define a binary variable x_b^d that takes value 1 if b is selected in the solution and 0 otherwise, and we denote by c_b its total variable cost. Furthermore, for each trip $t \in T$, the binary parameter $\alpha_{b,t}$ indicates whether block b covers or not trip t, and for each time point index $i \in I$, binary parameter $\beta_{b,i}^{\text{out}}$ (resp., $\beta_{b,i}^{\text{in}}$) is set to 1 if b starts from (resp., ends at) depot d at time point M_i^d. Using this notation, the MDVSP can be formulated as follows:

$$\min \quad \sum_{d \in D} \sum_{b \in B_d} c_b x_b^d + C \sum_{d \in D} u_{2n,1}^d \tag{1a}$$

$$\text{s.t.} \quad \sum_{d \in D} \sum_{b \in B_d} \alpha_{b,t} x_b^d = 1, \qquad \forall t \in T, \tag{1b}$$

$$u_{i^-,i}^d + \sum_{b \in B_d} \beta_{b,i}^{\text{in}} x_b^d - u_{i,i^+}^d - \sum_{b \in B_d} \beta_{b,i}^{\text{out}} x_b^d = 0, \qquad \forall d \in D, i \in I_d, \tag{1c}$$

$$x_b^d \in \{0,1\}, \in \{0,1\}, \qquad \forall d \in D, b \in B_d, \tag{1d}$$

$$u_{i,i+}^d \geq 0, \text{integer}, \quad \forall d \in D, i \in I_d. \tag{1e}$$

The objective function (1a) minimizes the total cost, including block variable costs and bus fixed costs. Constraints (1b) ensure that each trip is covered by a bus exactly once. Setting $I_d = I$ for all $d \in D$, constraints (1c) enforce flow conservation at each time point for each depot. Sets I_d are redefined below. Finally, constraints (1d) and (1e) restrict the domain of the variables.

3.3 Time-point aggregation

For network flow problems especially those defined on time-space networks, node aggregation is commonly used to reduce network size without compromising feasibility and optimality. In our context, we propose to apply time-point aggregation to reduce the number of flow conservation constraints (1c) (and u^d_{i,i^+} variables) in the block-based model. More precisely, for each depot $d \in D$, the sequence of time points $\{M^d_1, M^d_2, \ldots, M^d_{2n}\}$ is partitioned into a set $S_d = \{s_1, s_2, \ldots, s_{m_d}\}$ of m_d maximum length subsequences, with $m_d \leq 2n$, such that each subsequence in S_d contains time points associated with arrivals at depot d, followed by time points associated with departures from d. Given that the operations at a depot starts the day with departures and ends it with arrivals, the first subsequence in S_d contains only departure time points, while its last only arrival time points. Re-defining $I_d = \{1, 2, \ldots, m_d\}$ as the set of subsequence indices for depot d, the time points in a subsequence s_i, $i \in I_d$, are merged into a single aggregated time point τ^d_i, and the variables u^d_{i,i^+} are re-defined for the indices $i \in I_d$.

Figure 2 illustrates this aggregation process for a depot d with 8 initial time points (top part) that are replaced by 3 aggregated time points (bottom part). Thus, for this depot, the numbers of flow conservation constraints and u^d_{i,i^+} variables drop from 8 to 3 each.

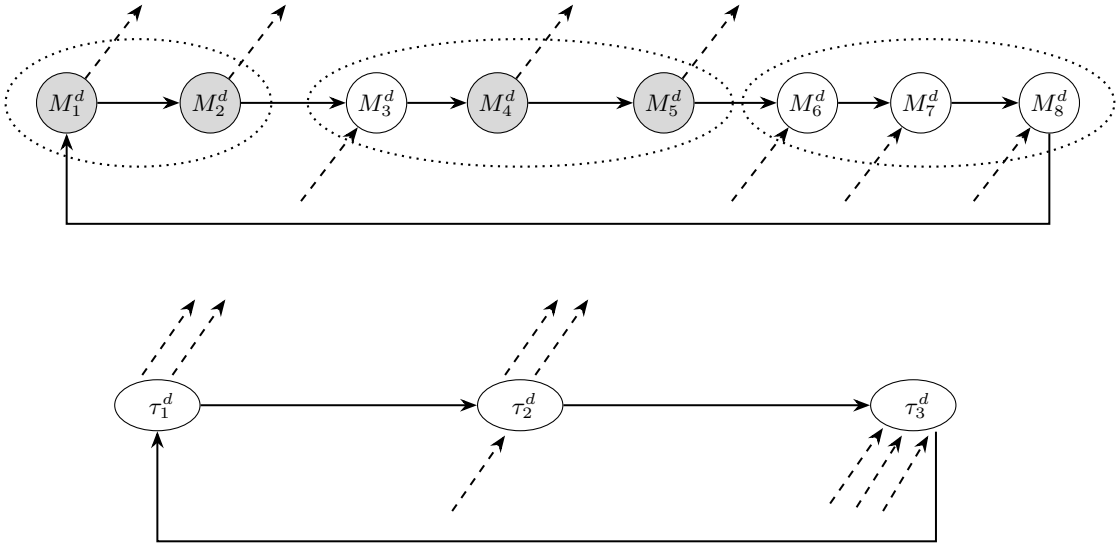

Figure 2: Time-point aggregation (initial on top, aggregated at bottom)

4 Solution algorithm

This section presents the solution algorithm that we propose for solving large-scale practical MDVSP instances. This algorithm combines CG and IDCA, embedded in a diving heuristic. We describe CG in Section 4.1, IDCA in Section 4.2, and the diving heuristic in Section 4.3.

4.1 Column generation

Because block-based model (1a)–(1e) involves a large number of variables (one per feasible bus block), its linear relaxation, also called the master problem, can be solved using CG like in Ribeiro and Soumis

(1994) (see also Desrosiers et al., 2024). CG is an iterative method that solves at each iteration a restricted master problem (RMP) and one pricing subproblem (SP) per depot in our case. The RMP is the master problem restricted to a subset of its variables x_b^d, those that have been generated so far. Solving it yields a pair of primal and dual solutions. The dual solution is then used to generate new columns (variables) with a negative reduced cost by solving the SPs. The generated columns are then added to the current RMP before starting a new iteration. The algorithm continues until no negative reduced cost columns are found, proving that the current RMP primal solution is also optimal for the master problem. In our CG heuristic, the CG algorithm is halted prematurely when the decrease in the RMP optimal value becomes negligible over a predefined number of iterations.

There is one SP per depot $d \in D$. Each SP aims at finding negative reduced cost blocks and is formulated as a shortest path problem defined on an acyclic connection-based network $G^d = (N^d, A^d)$ as illustrated in Figure 3. Node set N^d contains a source node $\bar{o}^d$, a sink node $\underline{o}^d$, and one node for each trip $t \in T$. Arc set A^d contains three types of arcs: pull-out arcs $(\bar{o}^d, t)$, $t \in T$; pull-in arcs $(t, \underline{o}^d)$, $t \in T$; and connection arcs (t_1, t_2), $t_1, t_2 \in T$, when trip t_1 can be directly followed by trip t_2 in a bus block, i.e., if the connection between t_1 and t_2 is feasible. In network G^d, a path from $\bar{o}^d$ to $\underline{o}^d$ represents a feasible block $b \in B_d$. To ensure that its cost is equal to the reduced cost of its corresponding variable x_b^d, we define the *adjusted cost* of an arc $a \in A^d$ as follows:

$$\bar{c}_a = \begin{cases} c_a + \phi^d_{i_a^{\text{out}}} & \text{if } a \text{ is a pull-out arc} \\ c_a - \pi_{t_a^{\text{tail}}} & \text{if } a \text{ is a connection arc} \\ c_a - \pi_{t_a^{\text{tail}}} - \phi^d_{i_a^{\text{in}}} & \text{if } a \text{ is a pull-in arc,} \end{cases} \tag{2}$$

where c_a is the variable cost incurred along arc a, i_a^{out} (resp. i_a^{in}) is the time point index associated with the pull-out (resp., pull-in) arc a, t_a^{tail} is the trip associated with the tail node of arc a, and $(\pi_t)_{t \in T}$ and $(\phi_i^d)_{d \in D, i \in I_d}$ are the dual vectors associated with constraints (1b) and (1c), respectively.

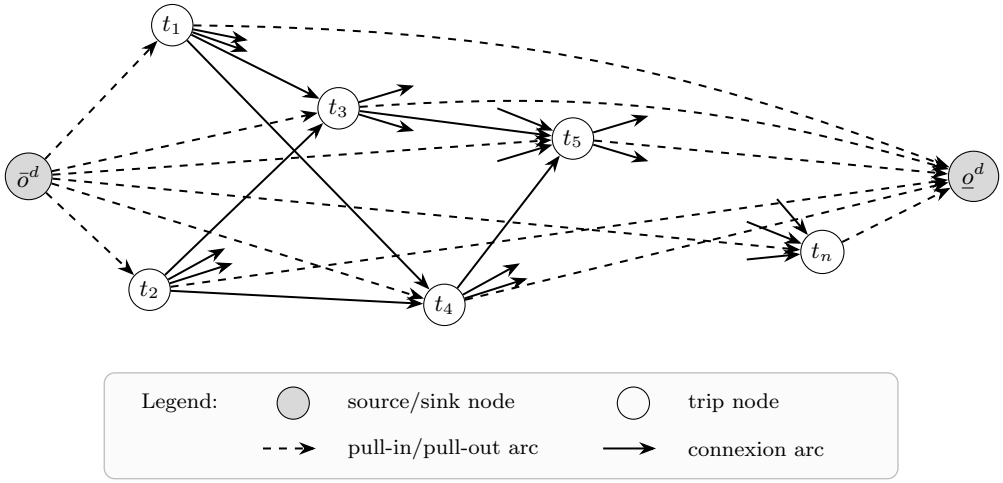

Figure 3: Connection-based network

The SP can be solved using a standard label-setting algorithm for a shortest path problem on an acyclic network. However, this algorithm allows to generate one column per depot at each iteration because a single path (label) is retained at the sink node. It is well known that generating many columns (say, more than 100) at each iteration accelerate the CG process. To do so, instead of keeping only the reduced cost of a path in a label, we also keep track of the number of trips the path covers. The label definition then becomes: a partial path p from $\bar{o}^d$ to a node $j \in N^d$ is represented by a label $L_p = (\bar{Z}_p, V_p)$, where $\bar{Z}_p$ denotes the reduced cost of path p (equal to the sum of the adjusted costs of its arcs) and V_p is the number of trips covered along p. This second label dimension is called a resource (which is unbounded in our case) and transforms the SP into a shortest path problem with

resource constraints. It is solved by a labeling algorithm on an acyclic network (see, e.g., Irnich and Desaulniers, 2005). Omitting label dominance at the sink node $\underline{o}^d$ (i.e., keeping all labels reaching $\underline{o}^d$) allows to generate multiple paths at each iteration.

4.2 Dynamic constraint aggregation

Set partitioning constraints (1b) often induce high degeneracy, especially when the blocks can contain a sufficiently large number of trips, say, more than 10. This issue is more pronounced in large real-world cases involving numerous constraints. To address this challenge and enhance the efficiency of CG, we resort to IDCA (Bouarab et al., 2017), which is an improved varaint of the DCA method designed by Elhallaoui et al. (2005). IDCA mitigates degeneracy by introducing an aggregated RMP (ARMP), which is derived from a partition Q of the set of trips T into disjoint clusters. In the ARMP, each cluster is represented by a single set partitioning constraint. IDCA begins with initial clusters and dynamically adjust partition Q throughout the solution process. In our case, the initial clusters are generated using the following greedy algorithm. Trips are first sorted by their start times for each bus line, and then round trips (i.e., in both directions along a line) are sequentially connected to form clusters, ensuring that trips within a cluster are operationally compatible and belong to the same bus line.

A block b and its associated variable x_b^d are said to be *compatible with a cluster* $q \in Q$ if the trips covered by b include either all or none of the trips in q. They are considered *compatible with partition* Q if they satisfy this condition for every cluster $q \in Q$. Otherwise, they are deemed *incompatible*.

The ARMP involves only variables that are compatible with the current partition Q. Let $B'_d(Q)$ be the subset of generated blocks associated with depot d that are compatible with Q and $\alpha_{b,q}$ be a binary parameter equal to 1 if block b covers all trips in cluster q, and 0 otherwise. The ARMP is formulated as:

$$\min \quad \sum_{d\in D}\sum_{b\in B'_d(Q)} c_b x_b^d + C\sum_{d\in D} u_{2n,1}^d \tag{3a}$$

$$\text{s.t.} \quad \sum_{d\in D}\sum_{b\in B'_d(Q)} \alpha_{b,q} x_b^d = 1, \quad \forall q \in Q, \tag{3b}$$

$$u_{i^-,i}^d + \sum_{b\in B'_d(Q)} \beta_{b,i}^{\text{in}} x_b^d - u_{i,i^+}^d - \sum_{b\in B'_d(Q)} \beta_{b,i}^{\text{out}} x_b^d = 0, \quad \forall d \in D, i \in I_d, \tag{3c}$$

$$x_b^d \geq 0, \quad \forall d \in D, b \in B'_d(Q), \tag{3d}$$

$$u_{i,i^+}^d \geq 0, \quad \forall d \in D, i \in I_d. \tag{3e}$$

Constraints (3b) ensure that each trip cluster $q \in Q$ is covered by one compatible block. All the other constraints and the objective function remain the same as in model (1a)–(1e), except that here, the model is linear and restricted to the set of compatible generated blocks in $B'_d(Q)$, $d \in D$. The ARMP is solved using the simplex algorithm, which provides a primal solution and a so-called *aggregated dual solution* as there is one dual variable $\hat{\pi}_q$, $q \in Q$, for each cluster covering constraint (3b).

When IDCA is combined with CG, the ARMP primal solution computed at a CG iteration is also optimal for the master problem if there exists a disaggregated dual solution $\left((\pi_t)_{t\in T}, (\phi_i^d)_{d\in D, i\in I_d}\right)$ such that the reduced cost of every variable in the master problem (generated or not) is non-negative. To do so, CG-IDCA computes a disaggregated dual solution as discussed below and solves the SPs, searching for negative reduced cost columns. If such columns are found, the compatible ones are always added to the ARMP. The incompatible ones are only added if no negative reduced cost compatible columns have been generated or $\bar{Z}^C/\bar{Z}^I \leq \zeta$, where $\bar{Z}^C$ (resp, $\bar{Z}^I$) denotes the least reduced cost of a compatible (resp., incompatible) column and $\zeta \in]0, 1]$ is a predetermined threshold. In this case, partition Q is updated to ensure the compatibility of all columns in the ARMP. The CG-IDCA

process continues until no more columns with negative reduced cost are generated or, as mentioned in Section 4.1, when the decrease of the ARMP optimal value is not sufficient over a predefined number of iterations (e.g, less than 0.05% over the last five iterations). Note that all incompatible columns that are not added to the ARMP are stored in a pool of incompatible columns and can be added to the ARMP in a subsequent iteration when a partition update makes them compatible. These columns are also used to compute a disaggregated dual solution as explained next.

4.2.1 Dual variable disaggregation

Several algorithms have been devised to compute disaggregated dual values. In the initial DCA algorithm, Elhallaoui et al. (2005) uses a sequence of shortest path problems that aim at finding a dual solution for which a subset of the known incompatible columns, with specific properties, have a non-negative reduced cost. Later, Bouarab et al. (2017) introduces a linear program, called the CP, that seeks a disaggregated dual solution such that all variables with a positive value in the current ARMP solution have a zero reduced cost with respect to this disaggregated solution and that maximizes the reduced cost of the least reduced cost incompatible variable among the known ones. The equivalent primal version of the CP looks for a convex combination of the known incompatible columns that can replace a linear combination of the positive-valued compatible columns, with the objective of minimizing the reduced cost of the former convex combination with respect to the variable disaggregated dual solution. Compared to solving a sequence of shortest path problems, using the CP helps reducing the number of iterations because it can take into account a larger subset of incompatible columns. In the following, we formulate the primal CP that we use.

For each depot $d \in D$, let $B_d^+(Q)$ be the set of the block indices of the variables x_b^d with a positive value in the ARMP solution and let $J_d(Q)$ represent the subset of the block indices of the known incompatible variables with respect to partition Q. For a block $b \in B_d^+(Q) \cup J_d(Q)$, we define the cost coefficient

$$\hat{c}_b = c_b + \hat{\phi}^d_{i_b^{\text{out}}} - \hat{\phi}^d_{i_b^{\text{in}}}, \tag{4}$$

where i_b^{out} and i_b^{in} are the pull-out and pull-in time point indices associated with block b, respectively, and $\left(\hat{\phi}_i^d\right)_{d\in D, i\in I_d}$ are the dual values of constraints (3c) in the ARMP dual solution. Further, for depot $d \in D$, we introduce the non-negative variables χ_j, $j \in J_d(Q)$, and the unrestricted variables λ_b, $b \in B_d^+(Q)$, which are the weights used in the respective convex and linear column combinations. Using this additional notation, the CP writes as follows:

$$\min \quad \sum_{d\in D}\sum_{j\in J_d(Q)} \hat{c}_j \chi_j^d - \sum_{d\in D}\sum_{b\in B_d^+(Q)} \hat{c}_b \lambda_b \tag{5a}$$

$$\text{s.t.} \quad \sum_{d\in D}\sum_{j\in J_d(Q)} \alpha_{j,t}\chi_j^d = \sum_{d\in D}\sum_{b\in B_d^+(Q)} \alpha_{b,t}\lambda_b, \quad \forall t \in T, \tag{5b}$$

$$\sum_{d\in D}\sum_{j\in J_d(Q)} \chi_j^d = 1, \tag{5c}$$

$$\chi_j^d \geq 0, \qquad \forall d \in D, j \in J_d(Q). \tag{5d}$$

Constraints (5b)–(5c) ensure that the selected convex combination of incompatible columns can replace the linear combination of the positive-valued columns. As mentioned above, the objective function seeks to minimize the reduced cost of this convex combination with respect to the computed disaggregated dual solution. The CP is solved using the dual simplex algorithm (shown to be more efficient than the primal simplex algorithm in preliminary tests) to yield dual values $\left(\tilde{\pi}_t\right)_{t\in T}$ for the constraints (5b). The disaggregated dual solution is given by $\left(\left(\pi_t\right)_{t\in T}, \left(\phi_i^d\right)_{d\in D, i\in I_d}\right) = \left(\left(\tilde{\pi}_t\right)_{t\in T}, \left(\hat{\phi}_i^d\right)_{d\in D, i\in I_d}\right)$.

Note that the CP does not take into account an equivalent form of the flow conservation constraints (1c). Although this would have given more leeway to determine the disaggregated dual solution

by not fixing the ϕ_i^d values to those obtained from the ARMP, we have preferred this CP formulation because handling these additional constraints increased significantly the time required to solve the CP in preliminary computational experiments.

The dual variable disaggregation process using the CP depends on the set of available incompatible columns. When this set is too poor (i.e., it does not contain columns that break the clusters in every possible way), some of the disaggregated dual values become somewhat arbitrary, yielding further disaggregation of partition Q and increased degeneracy. As generating a diversified set of incompatible columns is not necessarily an easy task, using the CP helps but has some limitations. Suggested by Elhallaoui et al. (2005) and recently used by Sudoso and Aloise (2025), a simple alternative to compute disaggregated dual values consists in distributing evenly the aggregated dual value of a cluster on the trips it contains. We propose to hybridize these two approaches to disaggregate the aggregated dual values $\left(\hat{\pi}_q\right)_{q\in Q}$ as follows.

Let $q = \{t_1, t_2, \ldots, t_{w_q}\} \in Q$ be a cluster containing w_q trips and assume that they are sorted in chronological order of their start time. An incompatible block $j \in \bigcup_{d\in D} J_d(Q)$ breaks cluster q after a trip $t_\ell \in q$, $\ell < w_q$, if j covers t_ℓ but not $t_{\ell+1}$. These breakpoints allow to partition cluster q into κ_q unbroken sub-clusters $r_1(q), r_2(q), \ldots, r_{\kappa_q}(q)$ of consecutive trips.

The HDD procedure starts by solving the CP to obtain disaggregated dual values $\left(\tilde{\pi}_t\right)_{t\in T}$ for the constraints (5b). Then, for each trip t in each sub-cluster $r(q)$ of each cluster $q \in Q$, the final disaggregated dual value for the corresponding constraint (5b) is equal to the average of the duals $\tilde{\pi}_{t'}$ associated with the trips t' in $r(q)$, i.e.,

$$\pi_t = \frac{\sum_{t'\in r(q)} \tilde{\pi}_{t'}}{|r(q)|}. \tag{6}$$

In this way, the trips in each sub-cluster are equally attractive in the SPs and there are less chance to generate highly incompatible columns that cover strict subsets of trips in several sub-clusters.

4.2.2 Multi-phase dynamic constraint aggregation

To avoid disaggregating partition Q too rapidly when incompatible columns are added to the ARMP, Elhallaoui et al. (2010) has proposed to proceed in phases, each phase restricting the set of columns that can be generated by the SPs less and less, except in the last phase where no restriction is typically applied. More precisely, let $\theta_b(Q)$ be the *degree of incompatibility* of a block b with respect to partition Q, which is computed as the number of times that block b breaks a cluster as defined in Section 4.2.1. In a phase k, only the blocks b with $\theta_b(Q) \leq k$ can be generated. This restriction is imposed in the SPs by adding an additional resource that counts the number of cluster breakpoints induced by the block and that is upper bounded by k. The resulting algorithm is called the multi-phase DCA. In our implementation, we consider four phases, namely, phases 0, 1, 2, and ∞. Phase 0 means that only compatible columns can be generated, whereas there is no restriction in phase ∞. Each phase ends when the CG stopping criterion stated in Section 4.1 is met.

4.3 Diving heuristic

To obtain integer solutions, we use a diving heuristic that explores a single branch of a branch-and-bound search tree (see Sadykov et al., 2019). After solving each linear relaxation (master problem), one or several decisions are permanently fixed before solving the updated linear relaxation, possibly generating new columns. The diving algorithm typically stops when the CG-IDCA algorithm ends with a linear relaxation solution that is integer.

When the solution of a linear relaxation is fractional, two types of decisions can be fixed. When at least one x_b^d variable takes a fractional value greater or equal to a predetermined threshold (set to 0.6 for our tests), we fix to 1 all variables that satisfy this condition. Otherwise, when there exists

at least one connection arc for which the sum of its flows over all depots is greater than or equal to another predefined threshold (set to 0.7), we fix to 1 the total flow of each connection arc satisfying this condition. For arc (t_1, t_2) linking trips t_1 and t_2, this decision is imposed by removing all arcs (t_1, v) with $v \neq t_2$ and all arcs (v, t_2) with $v \neq t_1$ from all networks G^d, $d \in D$. Finally, when the above two cases fail (this rarely occurred in our tests), the x_b^d variable with the largest fractional value is rounded up to 1.

5 Computational results

In this section, we present computational results to assess the proposed CG heuristic and evaluate the performance of some of its components. We first describe the instances used for our tests. We then present results for instances with forbidden direct connections induced only by a maximum unproductive time rule. Finally, we report results on instances that account for additional forbidden/penalized connections.

All our experiments were carried out on a Linux machine equipped with an Intel Core i7-10700 processor (2.90 GHz) and 64 GB of RAM. The implementation was developed in C++ using version 4.5 of the commercial column generation library GENCOL. The RMPs are solved with CPLEX 20.1.0.

5.1 Test instances

We first conducted computational experiments on 8 instances with 1 depot, 6 instances with 2 depots and 3 instances with 3 depots, that are derived from a real trip timetable of the Société des Transports de Montréal (STM). For these instances, all direct connections must respect a maximum duration constraint. No other direct connections are forbidden or penalized. The SDVSP instances were created by selecting each of the 8 STM depots and all the trips of bus lines in the neighborhood of this depot, ensuring that every line is assigned to a single depot. The 2-depot and 3-depot MDVSP instances were obtained by merging two and three SDVSPs. Notice that the SDVSP instances can be solved much more efficiently using a polynomial-time algorithm than with our CG heuristic. They are, however, used here to assess some of the algorithmic components of the proposed algorithm.

The characteristics of these test instances are summarized in Table 1. For each instance, we report the number of depots, the number of bus stations (line ends), the number of trips, and the number of clusters used to initialize IDCA. Furthermore, the last three columns provide the number of rows in the master problem (MP) for the CG algorithm (without IDCA), in the first iteration of CG-IDCA, and on average in all the root node CG iterations of CG-IDCA (with HDD), respectively. The largest 3-depot instances has close to 6,300 trips, linking 281 stations. It induces large SP networks involving more than 6,000 nodes and 550,000 arcs. We observe that IDCA reduces significantly the number of rows in the master problem, starting with an average reduction of 51.6% and maintaining an average reduction of 30.2% over all CG iterations at the root node. This helps solving the RMPs faster and mitigating degeneracy.

To further test the proposed methodology, we have conducted a second series of experiments on modified instances. These instances are the same as the ones above, except that we penalize or forbid additional direct connections according to the two following scenarios:

Forbidden Arcs: A portion of the connection arcs involving a deadhead is removed. Specifically, we randomly remove 15% or 30% of these arcs to generate new instances for both SDVSP and MDVSP.

Penalized Arcs: A portion of the connection arcs involving a deadhead (15% or 30%) is randomly selected to be penalized. The penalty cost ρ_a for such a connection arc $a = (t_1, t_2)$ consists of a fixed component γ^F and a variable component, with a maximal value γ^V, that depends on the risk of being late to the subsequent trip. This risk is inversely related to the available

waiting time between the trips involved in the connection. Specifically, if the travel time $\delta_{\ell^E_{t_1},\ell^S_{t_2}}$ occupies most of the connection time $h^S_{t_2} - h^E_{t_1}$, the likelihood of tardiness increases, and the arc is assigned a larger penalty to discourage its selection. For a connection arc $a = (t_1, t_2)$, this penalty is computed as:

$$\rho_a = \gamma^F + \gamma^V \times \left(\frac{\delta_{\ell^E_{t_1},\ell^S_{t_2}}}{h^S_{t_2} - h^E_{t_1}} \right).$$

To be able to compare the impact of forbidding or penalizing such arcs on the performance of the proposed algorithms, we have generated instances for both scenarios that forbid or penalize the same arcs.

Hereafter, we call the first set of instances (listed in Table 1), the *original instances* and the second one, the *instances with additional forbidden/penalized connections*. Note that the latter instances have the same numbers of depots, stations, and trips as the former instances, but the other statistics reported in Table 1 may slightly differ.

Table 1: Instance Characteristics

					MP Rows		
Instance	Depots	Stations	Trips	Initial Clusters	CG	First IDCA	Average IDCA
Depot50	1	119	3,162	926	4,117	1,881	2,846
Depot54	1	141	2,792	922	3,717	1,847	2,744
Depot55	1	98	2,081	380	2,912	1,211	1,946
Depot56	1	116	3,430	900	4,460	1,930	2,843
Depot57	1	107	1,374	454	1,984	1,064	1,512
Depot58	1	124	2,647	767	3,514	1,634	2,477
Depot59	1	90	2,275	406	3,169	1,300	2,121
Depot60	1	78	1,525	318	2,163	956	1,536
Depot50_58	2	223	5,809	1,693	8,095	3,979	5,485
Depot50_60	2	168	4,687	1,242	6,868	3,423	4,870
Depot55_57	2	173	3,455	833	5,564	2,942	4,123
Depot55_58	2	200	4,728	1,144	6,976	3,392	4,710
Depot55_59	2	170	4,356	786	6,647	3,077	4,399
Depot57_59	2	173	3,649	854	5,777	2,982	4,215
Depot55_57_58	3	270	6,102	1,594	9,687	5,179	6,811
Depot55_57_59	3	234	5,730	1,233	9,361	4,864	6,540
Depot57_58_59	3	281	6,296	1,618	9,830	5,152	6,706

5.2 Results for the original instances

In this section, we present computational results obtained on the original instances that will allow to assess the impact of using a block-based model instead of a schedule-based model, of combining CG with IDCA, and of applying HDD to disaggregate the dual variables. To do so, we have solved these instances using the following four solution approaches:

BM/CG: Block-based model (1a)–(1e) solved by the CG heuristic consisting of the CG algorithm and the diving heuristic described in Sections 4.1 and 4.3, respectively.

SM/CG: A schedule-based model solved by a CG heuristic similar to that used for the block-based model. The schedule-based model and the modifications to the CG heuristic are described in Section 1 of the Supplementary Materials.

BM/CG-IDCA: Same as BM/CG except that CG is combined with IDCA (see Section 4.2). The aggregated dual variables are disaggregated using only the CP.

BM/CG-IDCA-HDD: Same as BM/CG-IDCA except that HDD is used to disaggregate the dual variables (see Section 4.2).

As mentioned in Section 4, the CG process ends when there are no more negative reduced cost columns or when the (A)RMP optimal value has not decreased by a minimum percentage ν^D of the current (A)RMP optimal value over a predefined number of iterations η^I. Based on preliminary experiments, we have chosen the following values for ν^D and η^I: $\nu^D = 0.005\%$ and $\eta^I = 5$ for BM/CG; $\nu^D = 0.02\%$ and $\eta^I = 20$ for SM/CG which is more prone to degeneracy; and $\nu^D = 0.05\%$ and $\eta^I = 5$ for BM/CG-IDCA and BM/CG-IDCA-HDD. Note that, for BM/CG-IDCA and BM/CG-IDCA-HDD, this early stopping criterion applies to each phase.

To assess the quality of the computed solutions by the four proposed algorithms, we have solved to optimality the linear relaxation of each test instance to yield a lower bound $\underline{z}$. This lower bound was computed separately using a CG algorithm that stops when negative reduced cost columns cannot be generated. Given an upper bound $\bar{z}$ provided by a feasible integer solution, the optimality gap of this solution is calculated as $Gap = (\bar{z} - \underline{z})/\underline{z}$.

We begin by presenting the results obtained for the SDVSP instances in Table 2. This table is split into four blocks of rows, one for each algorithm. For each algorithm and each instance, the second and third columns indicate the total numbers of CG iterations (Itr.) and nodes explored in the diving heuristic, respectively. The next three columns provide the time in minutes devoted to solving the (A)RMP, solving the SPs, and solving the whole problem (Total). Note that the total time can exceed than the sum of the (A)RMP and SP times because it includes time for other processes (input parsing, loading and unloading the network of each SP in each CG iteration, applying the fixing decisions, etc.). The last three columns report the total cost of the computed solution, its optimality gap in percentage, and the number of buses used.

The results clearly indicate that SM/CG is much slower than the others, yielding the maximum total time for each SDVSP instance. At the opposite, BM/CG-IDCA-HDD requires the least computational time for every tested instance, except one (Depot58) where it is slightly outperformed by BM/CG-IDCA. Contrary to SM/CG where 55% of the total time is dedicated to the RMP, all solution methods relying on the block-based model spend a much large proportion of the total time on the (A)RMP. This can be explained by the use of SP networks that do not allow intermediate depot returns, reducing the average length of the generated paths. We also make the following high-level observations: for all methods, the number of nodes explored in the diving heuristic is quite small (at most 41) when considering the size of the test instances; the optimality gaps are very small for SM/CG and BM/CG, and small for the two methods applying IDCA; and all methods achieve the same number of buses for each instance.

To ease the comparison of the four solution approaches, we provide additional average comparative results in Table 3 for various pairs of approaches. To compare approach A1 versus approach A2, we report the average variations in percentage per instance obtained by A1 with respect to A2 for the number of CG iterations (Itr), the number of nodes explored, and the cost of the computed solution. We also indicate the average speedup factor obtained by A1 over A2 for the (A)RMP time, the SP time and the total time.

The BM/CG versus SM/CG results clearly show that adopting the block-based model yields a large average speedup factor of 3.8 without changing solution quality. This gain is due to easier-to-solve SPs and a faster convergence of the CG process (close to 70% less iterations) incurred by columns with less trips that help to reduce degeneracy and induce a sparser constraint coefficient matrix in the RMP. Because this confirms what has been previously observed in the literature for other types of applications (see Section 2), we will not present other results for the SM/CG approach.

Next, comparing BM/CG-IDCA with BM/CG, we observe that IDCA brings an additional acceleration (average factor of 2.2) but with a slight cost deterioration. This speedup comes from a large average reduction of the (A)RMP time that takes the largest portion of the total time. Note that the average number of iterations increases substantially with IDCA as reported in previous studies (due

to a smaller number of compatible/incompatible columns added to the ARMP at each iteration), but this increase is largely compensated by a much smaller average time per iteration.

Table 2: Detailed SDVSP Results

			Time (min)			Total	Gap	
Instance	Itr.	Nodes	(A)RMP	SPs	Total	Cost	(%)	Buses
SM/CG								
Depot50	662	20	48.8	39.5	93.0	2,124,074	0.000	210
Depot54	463	17	28.3	23.2	54.7	2,104,315	0.000	208
Depot55	411	11	10.0	10.4	21.7	1,323,834	0.001	131
Depot56	568	23	55.9	47.5	107.7	2,094,086	0.001	207
Depot57	337	10	2.8	2.7	6.0	1,124,130	0.001	111
Depot58	389	15	17.7	15.6	35.6	2,213,098	0.000	219
Depot59	1005	18	22.2	25.6	49.1	1,173,996	0.001	116
Depot60	456	18	5.0	4.5	10.6	1,021,711	0.000	101
BM/CG								
Depot50	240	16	25.7	1.2	27.0	2,124,117	0.002	210
Depot54	142	10	11.8	0.5	12.4	2,104,315	0.000	208
Depot55	112	4	7.1	0.3	7.5	1,323,834	0.001	131
Depot56	186	11	24.7	1.4	26.1	2,094,073	0.000	207
Depot57	72	2	1.5	0.1	1.6	1,124,124	0.000	111
Depot58	157	12	9.7	0.5	10.1	2,213,092	0.000	219
Depot59	123	4	9.1	0.7	9.9	1,174,022	0.003	116
Depot60	187	12	3.5	0.2	3.7	1,021,710	0.000	101
BM/CG-IDCA								
Depot50	595	41	18.0	3.1	21.4	2,124,168	0.005	210
Depot54	413	33	8.3	1.3	9.7	2,104,339	0.002	208
Depot55	214	14	1.0	0.5	1.7	1,324,348	0.040	131
Depot56	535	38	28.5	3.5	32.2	2,094,216	0.007	207
Depot57	242	14	0.9	0.2	1.2	1,124,153	0.003	111
Depot58	205	10	1.1	0.7	2.3	2,214,837	0.079	219
Depot59	388	25	2.9	1.1	4.2	1,174,086	0.008	116
Depot60	471	37	2.2	0.3	2.5	1,021,746	0.004	101
BM/CG-IDCA-HDD								
Depot50	472	17	8.7	2.0	11.4	2,124,285	0.010	210
Depot54	326	15	4.7	0.8	5.6	2,104,388	0.004	208
Depot55	252	21	0.7	0.2	1.1	1,323,887	0.005	131
Depot56	347	30	7.4	1.4	9.5	2,094,311	0.012	207
Depot57	199	13	0.5	0.1	0.7	1,124,205	0.007	111
Depot58	277	23	1.4	0.4	2.6	2,213,338	0.011	219
Depot59	311	20	1.2	0.5	2.2	1,174,061	0.006	116
Depot60	471	40	1.4	0.2	1.9	1,021,784	0.007	101

Table 3: Pairwise Comparisons of Solution Approaches for the SDVSP

	Variation (%)			Speedup Factor		
	Itr	Nodes	Cost	(A)RMP	SP	Total
BM/CG vs SM/CG	-69.8	-48.5	0.000	1.9	38.2	3.8
BM/CG-IDCA vs BM/CG	156.4	274.8	0.017	3.2	0.5	2.2
BM/CG-IDCA-HDD vs BM/CG-IDCA	-7.7	3.4	-0.011	2.0	1.9	1.8
BM/CG-IDCA-HDD vs BM/CG	124.4	241.1	0.007	4.9	0.9	3.3
BM/CG-IDCA-HDD vs SM/CG	-33.9	38.9	0.007	9.2	33.6	12.4

The BM/CG-IDCA-HDD versus BM/CG-IDCA comparison indicates that using HDD slightly reduces the average number of iterations and produces another additional average speedup factor of 1.8. This is due to a better dual variable disaggregation process that helps generating columns that are more suitable (often compatible or incompatible with a small degree of incompatibility) to keep

the ARMP well aggregated. Indeed, for these instances, the average number of rows in the ARMP per CG iteration compared to the number of rows in the RMP without aggregation is reduced by 30.2% with HDD and by only 16.1% without HDD. Furthermore, more evenly spread dual values seem to facilitate the solution of the SPs (SP time speedup factor of 1.9).

The last two rows of Table 3 allow to assess the combined average speedup of all proposed techniques. Using BM/CG-IDCA-HDD yields an average speedup factor of 3.3 over BM/CG and of 12.4 over SM/CG. The latter is due to large reductions in the number of CG iterations, in the time devoted to solving the (A)RMP, and, to a larger extent, in the time required by the SPs. Our results indicate that every technique brings a significant contribution to this speedup. They enable to solve the largest SDVSP instance (Depot56 with 3,430 trips and 116 stations) in less than 10 minutes, whereas it requires 107 minutes using the basic SM/CG approach.

Let us mention a final observation about the number of nodes reported in these two tables, which is also valid for the forthcoming MDVSP results. We can see that the average varies from one method to another. It is difficult to identify the reason of this variation but it does not have a direct impact on the total computational time. Indeed, for the SDVP instances, the two solution approaches with the smallest average numbers of nodes (SM/CG and BM/CG) have the largest average total times.

Now, let us present the results obtained for the original MDVSP instances in Tables 4 and 5, which are formatted like the previous two tables. From the detailed results in Table 4, we observe again that BM/CG-IDCA-HDD is faster than the other methods as it yields the least total time for all tested instances. Like for the SDVSP instances, most of the time is devoted to solving the (A)RMP for all solution algorithms. Anew, all algorithms produce solutions with the same number of buses and with optimality gaps that are very small for BM/CG and small for BM/CG-IDCA and BM/CG-IDCA-HDD. For BM/CG-IDCA-HDD, this small cost increase remains acceptable given the large speedup obtained.

The comparative results in Table 5 are similar to those presented in Table 3 for the original SDVSP instances. However, for the comparison BM/CG-IDCA versus BM/CG, we observe that the average speedup factor is less than for the SDVSP instances (1.4 instead of 2.2) because IDCA did not succeed to reduce substantially the ARMP time. On the other hand, the average computational time gain achieved by using HDD being more important for the MDVSP instances (speedup factor of 2.4 instead of 1.8), the overall speedup factor yielded by BM/CG-IDCA-HDD over BM/CG is 3.4 and, thus, very similar to the one observed for the SDVSP instances. Here again, this acceleration results from much less average time dedicated to solving the (A)RMP despite a larger number of CG iterations.

By combining IDCA with CG and using the proposed HDD strategy to reduce degeneracy and generate more suitable columns, the largest original MDVSP instance (Depot57_58_59 with 3 depots, 6,296 trips and 281 stations) is solved in less than 40 minutes with BM/CG-IDCA-HDD compared to around 140 minutes with BM/CG, resulting in a highly significant speedup factor of 3.5.

5.3 Results for instances with additional forbidden/penalized connections

To further analyze the impact of using BM/CG-IDCA-HDD over BM/CG, we have conducted another series of tests on instances where direct connections involving a deadhead are either forbidden or penalized. Four scenarios are considered: scenario Forbidden/15% (resp., Forbidden/30%) where 15% (resp., 30%) of these connections are forbidden, and scenario Penalized/15% (resp., Penalized/30%) where 15% (resp., 30%) of them are penalized.

The detailed results of these experiments are reported in Section 2 of the Supplementary Materials, while average results allowing to compare BM/CG with BM/CG-IDCA-HDD are provided in Table 6. First, we can observe from the detailed results and those reported in the previous section that forbidding/penalizing additional direct connections increases the cost of the computed solutions moderately for most instances and more substantially for a few instances where the number of buses

used increases. This was expected as some of these connections were selected in the solutions of the instances without additional connection restrictions or penalties.

Table 4: Detailed MDVSP Results

Instance	Itr.	Nodes	Time (min) (A)RMP	Time (min) SPs	Time (min) Total	Total Cost	Gap (%)	Buses
BM/CG								
Depot50_58	339	18	114.3	4.1	118.6	4,254,398	0.001	421
Depot50_60	634	70	108.8	3.3	112.0	3,092,075	0.002	306
Depot55_57	153	5	21.2	0.9	23.9	2,395,374	0.002	237
Depot55_58	224	9	56.5	2.2	58.8	3,433,865	0.002	340
Depot55_59	218	7	57.9	3.0	61.1	2,465,551	0.003	244
Depot57_59	180	9	31.9	2.1	34.0	2,245,628	0.002	222
Depot55_57_58	192	8	133.0	4.0	137.1	4,526,267	0.001	448
Depot55_57_59	199	7	118.2	5.1	124.0	3,548,325	0.002	351
Depot57_58_59	181	8	135.4	5.4	141.3	4,387,580	0.002	434
BM/CG-IDCA								
Depot50_58	728	27	101.6	13.0	115.4	4,254,772	0.010	421
Depot50_60	716	25	78.2	7.4	86.1	3,092,305	0.010	306
Depot55_57	384	14	17.7	2.8	20.8	2,395,493	0.007	237
Depot55_58	409	18	33.3	6.2	40.9	3,434,619	0.024	340
Depot55_59	423	17	24.1	8.3	34.6	2,466,053	0.023	244
Depot57_59	413	15	18.6	5.7	24.3	2,245,743	0.007	222
Depot55_57_58	503	26	65.4	11.0	78.5	4,527,126	0.020	448
Depot55_57_59	520	22	69.3	13.3	83.8	3,548,496	0.007	351
Depot57_58_59	566	29	93.5	16.0	110.7	4,387,775	0.006	434
BM/CG-IDCA-HDD								
Depot50_58	459	18	26.2	6.3	34.0	4,255,326	0.023	421
Depot50_60	492	17	26.4	4.4	32.1	3,092,453	0.014	306
Depot55_57	337	13	9.1	1.8	11.7	2,395,442	0.005	237
Depot55_58	350	18	11.1	3.3	15.6	3,434,187	0.011	340
Depot55_59	371	17	8.8	3.2	13.0	2,465,725	0.010	244
Depot57_59	343	10	10.7	2.4	13.9	2,245,705	0.006	222
Depot55_57_58	466	25	32.8	7.3	41.7	4,527,117	0.019	448
Depot55_57_59	401	16	27.5	8.1	37.0	3,548,626	0.010	351
Depot57_58_59	513	37	29.2	9.3	39.2	4,388,390	0.020	434

Table 5: Pairwise Comparisons of Solution Approaches for the MDVSP

	Variation (%) Itr	Variation (%) Nodes	Variation (%) Cost	Speedup Factor (A)RMP	Speedup Factor SP	Speedup Factor Total
BM/CG-IDCA vs BM/CG	124.5	130.8	0.011	1.6	0.4	1.4
BM/CG-IDCA-HDD vs BM/CG-IDCA	-18.2	-12.1	0.001	2.7	1.9	2.4
BM/CG-IDCA-HDD vs BM/CG	86.4	115.8	0.011	4.3	0.7	3.4

From the comparative results in Table 6, we can see that BM/CG-IDCA-HDD still yields a significant average speedup factor over BM/CG for all scenarios, ranging between 2.4 and 2.9. This speedup is again due to a large reduction of the (A)RMP time that greatly compensates the increase of the SP time. For all scenarios, it is less than the speedup factors (3.3 for the SDVSP and 3.4 for the MDVSP) reported in Section 5.2. We believe that removing/penalizing additional connections diminishes degeneracy because slightly shorter columns are generated and, therefore, the tools put in place to fight degeneracy are less useful. Nevertheless, the achieved speedups clearly show that these tools remain highly relevant to achieve the best results.

Table 6: Comparative Results for the Instances with Additional Forbidden/Penalized Connections: BM/CG-IDCA-HDD vs BM/CG

	Variation (%)			Speedup Factor		
Scenario	Itr	Nodes	Cost	(A)RMP	SP	Total
SDVSP						
Forbidden/15%	129.2	234.4	0.007	3.3	0.6	2.5
Forbidden/30%	149.2	236.8	0.006	4.0	0.7	2.8
Penalized/15%	143.1	212.9	0.005	3.9	0.7	2.6
Penalized/30%	133.8	179.0	0.005	3.5	0.6	2.4
MDVSP						
Forbidden/15%	86.6	89.8	0.011	3.4	0.6	2.8
Forbidden/30%	84.1	59.1	0.009	3.4	0.6	2.9
Penalized/15%	111.9	184.2	0.011	2.9	0.5	2.4
Penalized/30%	85.2	63.2	0.009	3.0	0.6	2.6

6 Conclusion

In this paper, we proposed to solve large-scale practical MDVSP instances using a CG-based diving heuristic that combines CG with IDCA and a novel HDD strategy, to mitigate degeneracy. This algorithm is applied to a block-based model that is also less prone to degeneracy than a traditional schedule-based model. Furthermore, in response to the needs of our industrial partner, we use connection-based networks in the pricing SPs to easily handle restrictions or penalties on undesirable direct connections between two trips.

To assess the impact of using all these components, we conducted computational experiments on real instances of the Montreal transit agency, involving up to 3 depots and more than 6,000 trips. The obtained results on the SDVSP instances indicate that the best algorithm, BM/CG-IDCA-HDD, yields an impressive average speedup factor of 12.4 over the basic SM/CG algorithm. Our results also show that, for both the SDVSP and the MDVSP instances, average acceleration factors ranging between 2.4 and 3.4 can be achieved by applying IDCA with HDD on top of CG when adopting a block-based model for both solution approaches. These time gains come with an increase of the solution cost that is deemed negligible (less than 0.011% on average). For example, the BM/CG-IDCA-HDD algorithm can solve the largest tested MDVSP instance with 3 depots and 6,296 in less than 40 minutes, producing a solution with a 0.02% optimality gap.

These results show that the proposed methodology can be highly effective at solving practical large-scale MDVSP instances. Future work will extend this solution method to electric bus scheduling by integrating charging decisions and battery constraints. In particular, adopting a block-based model in this case is challenging because the flow conservation constraints must take into account the state-of-charge of the buses.

References

Pascal Benchimol, Guy Desaulniers, and Jacques Desrosiers. Stabilized dynamic constraint aggregation for solving set partitioning problems. European Journal of Operational Research, 223(2):360–371, 2012. doi: https://doi.org/10.1016/j.ejor.2012.07.004. URL https://www.sciencedirect.com/science/article/pii/S0377221712005255.

A. A. Bertossi, P. Carraresi, and G. Gallo. On some matching problems arising in vehicle scheduling models. Networks, 17(3):271–281, 1987. doi: https://doi.org/10.1002/net.3230170303. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230170303.

Lawrence Bodin, D Rosenfield, and Andy Kydes. UCOST: a micro approach to a transportation planning problem. Journal of Urban Analysis, 5(1):47–69, 1978.

Hocine Bouarab, Issmail El Hallaoui, Abdelmoutalib Metrane, and François Soumis. Dynamic constraint and variable aggregation in column generation. European Journal of Operational Research, 262(3):835–850, 2017.

doi: https://doi.org/10.1016/j.ejor.2017.04.049. URL https://www.sciencedirect.com/science/article/pii/S037722171730396X.

Guy Desaulniers, June Lavigne, and Francois Soumis. Multi-depot vehicle scheduling problems with time windows and waiting costs. European Journal of Operational Research, 111(3):479–494, 1998.

Jacques Desrosiers, Marco Lübbecke, Guy Desaulniers, and Jean Bertrand Gauthier. Branch-and-price. Les Cahiers du GERAD G-2024-36, Groupe d'études et de recherche en analyse des décisions, June 2024. URL https://www.gerad.ca/en/papers/G-2024-36.

Issmail Elhallaoui, Daniel Villeneuve, François Soumis, and Guy Desaulniers. Dynamic aggregation of set-partitioning constraints in column generation. Operations Research, 53(4):632–645, 2005. doi: 10.1287/opre.1050.0222. URL https://doi.org/10.1287/opre.1050.0222.

Issmail Elhallaoui, Metrane Abdelmoutalib, François Soumis, and Guy Desaulniers. Multi-phase dynamic constraint aggregation for set partitioning type problems. Mathematical Programming, 123:345–370, 06 2010. doi: 10.1007/s10107-008-0254-5.

Juliette Gerbaux, Guy Desaulniers, and Quentin Cappart. A machine-learning-based column generation heuristic for electric bus scheduling. Computers & Operations Research, 173:106848, 2025. doi: https://doi.org/10.1016/j.cor.2024.106848. URL https://www.sciencedirect.com/science/article/pii/S0305054824003204.

Vitali Gintner, Natalia Kliewer, and Leena Suhl. Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. OR Spectrum, 27:507–523, 2005.

K. Gkiotsalitis, C. Iliopoulou, and K. Kepaptsoglou. An exact approach for the multi-depot electric bus scheduling problem with time windows. European Journal of Operational Research, 306(1):189–206, 2023. doi: https://doi.org/10.1016/j.ejor.2022.07.017. URL https://www.sciencedirect.com/science/article/pii/S0377221722005707.

Pablo C. Guedes and Denis Borenstein. Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem. Computers & Industrial Engineering, 90:361–370, 2015. doi: https://doi.org/10.1016/j.cie.2015.10.004. URL https://www.sciencedirect.com/science/article/pii/S0360835215003976.

Pablo Cristini Guedes, William Prigol Lopes, Leonardo Rosa Rohde, and Denis Borenstein. Simple and efficient heuristic approach for the multiple-depot vehicle scheduling problem. Optimization Letters, 10:1449–1461, 2016.

Ahmed Hadjar and François Soumis. Dynamic window reduction for the multiple depot vehicle scheduling problem with time windows. Computers & Operations Research, 36(7):2160–2172, 2009. doi: https://doi.org/10.1016/j.cor.2008.08.010. URL https://www.sciencedirect.com/science/article/pii/S0305054808001391.

Ahmed Hadjar, Odile Marcotte, and François Soumis. A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. Operations Research, 54(1):130–149, 2006.

Florent Hernandez, Dominique Feillet, Rodolphe Giroudeau, and Olivier Naud. Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. European Journal of Operational Research, 249(2):551–559, 2016.

O.J. Ibarra-Rojas, F. Delgado, R. Giesen, and J.C. Muñoz. Planning, operation, and control of bus transport systems: A literature review. Transportation Research Part B: Methodological, 77:38–75, 2015. doi: https://doi.org/10.1016/j.trb.2015.03.002. URL https://www.sciencedirect.com/science/article/pii/S0191261515000454.

Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, Column Generation, pages 33–65. Springer US, Boston, MA, 2005.

Natalia Kliewer, Taïeb Mellouli, and Leena Suhl. A new solution model for multi-depot multi-vehicle-type vehicle scheduling in (sub) urban public transport. In Proceedings of the 13th Mini-EURO Conference. Politechnic of Bari, 2002.

Natalia Kliewer, Taïeb Mellouli, and Leena Suhl. A time–space network based exact optimization model for multi-depot bus scheduling. European Journal of Operational Research, 175(3):1616–1627, 2006.

Benoît Laurent and Jin-Kao Hao. Iterated local search for the multiple depot vehicle scheduling problem. Computers & Industrial Engineering, 57(1):277–286, 2009. doi: https://doi.org/10.1016/j.cie.2008.11.028. URL https://www.sciencedirect.com/science/article/pii/S0360835208003148.

Aristide Mingozzi, Roberto Roberti, and Paolo Toth. An exact algorithm for the multitrip vehicle routing problem. INFORMS Journal on Computing, 25(2):193–207, 2013.

Seyedeh Simin Mousavi, Alireza Pooya, Pardis Roozkhosh, and Morteza Pakdaman. A new bi-objective simultaneous model for timetabling and scheduling public bus transportation. Opsearch, 62(1):198–229, 2025.

Amar Oukil, Hatem Ben Amor, Jacques Desrosiers, and Hicham El Gueddari. Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. Computers & Operations Research, 34 (3):817–834, 2007. doi: https://doi.org/10.1016/j.cor.2005.05.011. URL https://www.sciencedirect.com/science/article/pii/S0305054805001590.

Ann-Sophie Pepin, Guy Desaulniers, Alain Hertz, and Dennis Huisman. A comparison of five heuristics for the multiple depot vehicle scheduling problem. Journal of Scheduling, 12:17–30, 2009.

Shyam SG Perumal, Richard M Lusby, and Jesper Larsen. Electric bus planning & scheduling: A review of related problems and methodologies. European Journal of Operational Research, 301(2):395–413, 2022.

Celso C Ribeiro and François Soumis. A column generation approach to the multiple-depot vehicle scheduling problem. Operations research, 42(1):41–52, 1994.

Léa Ricard, Guy Desaulniers, Andrea Lodi, and Louis-Martin Rousseau. Chance-constrained battery management strategies for the electric bus scheduling problem. arXiv preprint arXiv:2503.19853, 2025.

Mohammad Sadrani, Alejandro Tirachini, and Constantinos Antoniou. Bus scheduling with heterogeneous fleets: Formulation and hybrid metaheuristic algorithms. Expert Systems with Applications, 263:125720, 2025. doi: https://doi.org/10.1016/j.eswa.2024.125720. URL https://www.sciencedirect.com/science/article/pii/S0957417424025879.

Ruslan Sadykov, François Vanderbeck, Artur Pessoa, Issam Tahiri, and Eduardo Uchoa. Primal heuristics for branch and price: The assets of diving methods. INFORMS Journal on Computing, 31(2):251–267, 2019.

Antonio M. Sudoso and Daniel Aloise. A column generation algorithm with dynamic constraint aggregation for minimum sum-of-squares clustering. INFORMS Journal on Computing, 0(0):null, 2025. doi: 10.1287/ijoc.2024.0938. URL https://doi.org/10.1287/joc.2024.0938.

Marcel E van Kooten Niekerk, JM van den Akker, and JA Hoogeveen. Scheduling electric vehicles. Public Transport, 9:155–176, 2017.

Şule Yıldırım and Barış Yıldız. Electric bus fleet composition and scheduling. Transportation Research Part C: Emerging Technologies, 129:103197, 2021.

Le Zhang, Shuaian Wang, and Xiaobo Qu. Optimal electric bus fleet scheduling considering battery degradation and non-linear charging profile. Transportation Research Part E, 154:102445, 2021.