

Penalty function-based dual stabilization for column generation with graph neural networks: Application to the vehicle routing problem with stochastic demands

G. Reynal, Q. Cappart, G. Desaulniers, L.-M. Rousseau

G-2026-51

September 2026

La collection *Les Cahiers du GERAD* est constituée des travaux de recherche menés par nos membres. La plupart de ces documents de travail a été soumis à des revues avec comité de révision. Lorsqu'un document est accepté et publié, le pdf original est retiré si c'est nécessaire et un lien vers l'article publié est ajouté.

Citation suggérée : G. Reynal, Q. Cappart, G. Desaulniers, L.-M. Rousseau (Septembre 2026). Penalty function-based dual stabilization for column generation with graph neural networks: Application to the vehicle routing problem with stochastic demands, Rapport technique, Les Cahiers du GERAD G- 2026-51, GERAD, HEC Montréal, Canada.

Avant de citer ce rapport technique, veuillez visiter notre site Web (<https://www.gerad.ca/fr/papers/G-2026-51>) afin de mettre à jour vos données de référence, s'il a été publié dans une revue scientifique.

The series *Les Cahiers du GERAD* consists of working papers carried out by our members. Most of these pre-prints have been submitted to peer-reviewed journals. When accepted and published, if necessary, the original pdf is removed and a link to the published article is added.

Suggested citation: G. Reynal, Q. Cappart, G. Desaulniers, L.-M. Rousseau (September 2026). Penalty function-based dual stabilization for column generation with graph neural networks: Application to the vehicle routing problem with stochastic demands, Technical report, Les Cahiers du GERAD G-2026-51, GERAD, HEC Montréal, Canada.

Before citing this technical report, please visit our website (<https://www.gerad.ca/en/papers/G-2026-51>) to update your reference data, if it has been published in a scientific journal.

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, 2026
– Bibliothèque et Archives Canada, 2026

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, 2026
– Library and Archives Canada, 2026

Penalty function-based dual stabilization for column generation with graph neural networks: Application to the vehicle routing problem with stochastic demands

Gaël Reynal ^{a, c, d}

Quentin Cappart ^{a, b, d}

Guy Desaulniers ^{a, c}

Louis-Martin Rousseau ^{a, d}

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

^b *Ecole polytechnique de Louvain, Université catholique de Louvain, Louvain-la-Neuve, 1348 Belgium*

^c *GERAD, Montréal (Qc), Canada, H3T 1J4*

^d *CIRRELT, Montréal (Qc), Canada, H3T 1T9*

`gael-simon.reynal@etud.polymtl.ca`

`quentin.cappart@polymtl.ca`

`guy.desaulniers@gerad.ca`

`louis-martin.rousseau@polymtl.ca`

September 2026
Les Cahiers du GERAD
G–2026–51

Copyright © 2026 Reynal, Cappart, Desaulniers, Rousseau

Les textes publiés dans la série des rapports de recherche *Les Cahiers du GERAD* n'engagent que la responsabilité de leurs auteurs. Les auteurs conservent leur droit d'auteur et leurs droits moraux sur leurs publications et les utilisateurs s'engagent à reconnaître et respecter les exigences légales associées à ces droits. Ainsi, les utilisateurs:

- Peuvent télécharger et imprimer une copie de toute publication du portail public aux fins d'étude ou de recherche privée;
- Ne peuvent pas distribuer le matériel ou l'utiliser pour une activité à but lucratif ou pour un gain commercial;
- Peuvent distribuer gratuitement l'URL identifiant la publication.

Si vous pensez que ce document enfreint le droit d'auteur, contactez-nous en fournissant des détails. Nous supprimerons immédiatement l'accès au travail et enquêtrons sur votre demande.

The authors are exclusively responsible for the content of their research papers published in the series *Les Cahiers du GERAD*. Copyright and moral rights for the publications are retained by the authors and the users must commit themselves to recognize and abide the legal requirements associated with these rights. Thus, users:

- May download and print one copy of any publication from the public portal for the purpose of private study or research;
- May not further distribute the material or use it for any profit-making activity or commercial gain;
- May freely distribute the URL identifying the publication.

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Abstract : Column generation is a widely used dual-guided approach to solve highly degenerate problems. During the resolution, the degeneracy of the master problem often manifests through dual oscillations, which results in many iterations in which no progress is made towards the optimal solution. To mitigate the impact of degeneracy, dual stabilization techniques can be added to the process, with the aim of preventing strong dual oscillations and speeding up convergence. Such methods involve numerous design choices, including the choice of the initial stability center, which requires accurately estimating the optimal dual values. This paper introduces a machine-learning framework leveraging graph neural networks to predict these values and use them as the initial stability center in a dual stabilization framework with dynamic parameter updates. Our experiments focus on the vehicle routing problem with stochastic demands (40 instances with up to 60 customers), a challenging variant of the classical VRP for which an accurate estimate of the optimal dual values is hard to retrieve. Our method outperforms all algorithm variants considered.

Keywords : Column generation; dual stabilization; vehicle routing; stochastic demands; graph neural networks

Acknowledgements: This research was undertaken, in part, thanks to funding from the Canada Research Chairs Program in Healthcare Analytics. We also acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC RGPIN 2025-04995, RGPIN 2023-03791, RGPIN-2022-03964).

1 Introduction

Column generation (CG) is a widely used algorithm for solving a combinatorial optimization problem formulated as an integer linear program with a very large number of variables. Applications include vehicle routing (see Costa et al. 2019), vehicle scheduling (Oukil et al. 2007, Ben Amor et al. 2009, Benchimol et al. 2012), crew scheduling (Desrochers and Soumis 1989, Kasirzadeh et al. 2017, Quesnel et al. 2022), stock cutting (Valério de Carvalho 2005, Fang et al. 2025), and many more. Column generation is an iterative algorithm that solves the integer program’s linear relaxation, which is referred to as the *master problem*. At each iteration, it solves the so-called *restricted master problem* (RMP), which is the master problem restricted to a subset of its variables, and a pricing problem that seeks to find negative reduced cost columns (variables) to add to the RMP. The algorithm stops when no such column exists. The objective function of the pricing problem depends on the dual solution of the current RMP, and thus varies at each iteration. Throughout the iterations, these dual solutions guide the search towards relevant columns that are progressively added to the RMP. Therefore, the quality of the dual solutions has a clear impact on the efficiency of the CG process.

For several applications, the master problem is often highly degenerate (see, e.g., Lübbecke and Desrosiers 2005). In this case, the RMP has multiple optimal dual solutions and the dual values may oscillate substantially from one iteration to the next, without yielding a significant improvement in the RMP optimal value. This often results in many CG iterations and slow convergence toward optimality. Among the existing strategies for mitigating high degeneracy, a common option is to introduce a stabilization mechanism, such as dual variable stabilization (Du Merle et al. 1999, Valério de Carvalho 2005, Ben Amor et al. 2009, Benchimol et al. 2012, Desrosiers et al. 2026). It aims to keep the dual values in a so-called trust region by applying a penalty for lying outside of it. Many implementation choices must be considered when using this kind of method, such as the size and center of the trust region (called the *stability center*), the type and value of the penalties, and an update strategy for all these parameters.

While the literature contains many stabilization techniques, the practical impact of the initial stability center has received relatively little attention so far. This is mainly because computing a relevant stability center requires an accurate estimate of an optimal dual solution (the closer, the better), a task for which an efficient algorithm is not always available. Moreover, even with a rough estimate of the dual values, stabilization has been shown to accelerate CG convergence. A good choice of initial stability center can, however, result in a significant additional acceleration of the process (Ben Amor et al. 2009, Kraul et al. 2023).

Based on this context, we propose using a supervised learning framework leveraging graph neural networks to predict optimal dual values and use these predictions as a high-quality initial stability center for a dual stabilization algorithm adapted from Benchimol et al. (2012). We evaluate our approach on the vehicle routing problem with stochastic demands (VRPSD, Tillman 1969). This problem features a difficult pricing problem: the stochasticity forces the use of a very large pricing network, making each CG iteration expensive and thus naturally calling for a stabilization mechanism. Moreover, to our knowledge, no efficient algorithm exists to obtain an accurate estimate of the optimal dual values. This makes the problem a relevant testbed for properly assessing the impact of the stability center on the overall convergence.

Our contribution is as follows:

- We introduce a supervised learning architecture that leverages graph neural networks (GNNs) to predict optimal dual values for a VRPSD instance, and we use these predictions as the initial stability center in a dual stabilization framework.
- We experimentally evaluate the accuracy to aim for when computing an initial stability center to ensure that it has a significant impact on the resolution speed. In particular, we show

that stabilization does not require highly accurate dual predictions to still provide substantial computational benefits.

- We adapt the dual stabilization strategy from Benchimol et al. (2012) with a periodic update procedure that mitigates the tailing-off effect observed between stabilization updates and assess the impact of such periodic updates.

The paper is organized as follows. In Section 2, we review relevant works regarding stabilization techniques in CG, dual variable prediction, and machine learning integration in CG. In Section 3, we present a mathematical model for the VRPSD and briefly describe the CG algorithm used to solve it. Next, we detail our dual stabilization method in Section 4, present the experimental results in Section 5, and conclude in Section 6.

2 Related works

Column generation is one of the most successful algorithms for solving large-scale linear programs arising from Dantzig-Wolfe reformulations (Lübbecke and Desrosiers 2005, Desrosiers et al. 2026). Despite its effectiveness, its convergence can be severely affected by primal degeneracy. It often results in large oscillations of the dual variables and a pronounced tailing-off effect, where many iterations are required to achieve marginal improvements in the objective value (see Du Merle et al. 1999, Ben Amor et al. 2009, Pessoa et al. 2018). Over the last decades, a substantial body of work has therefore focused on accelerating the convergence of the algorithm in various ways, including: stabilizing the sequence of dual solutions generated throughout the algorithm (see Section 2.1), reducing the dual space through dual inequalities (see Section 2.1), selecting a subset of the columns generated at each iteration (Morabit et al. 2021), or solving the pricing problem heuristically (see, e.g., Desaulniers et al. 2008, Quesnel et al. 2022, Morabit et al. 2023).

More recently, advances in machine learning for combinatorial optimization have shown that dual information can often be predicted directly from the structure of an optimization problem. In particular, several studies have demonstrated that graph-based neural architectures can predict high-quality Lagrangian multipliers or dual solutions, leading to substantial computational savings (Demelas et al. 2024, Parjadis et al. 2024, Bessa et al. 2025). These developments open new perspectives for incorporating learned dual information into decomposition-based algorithms.

This section first reviews the main CG speedup techniques focusing on dual variables (Section 2.1). We then present several works incorporating machine learning in a CG framework (Section 2.2), before discussing on how these machine learning architectures can learn to predict dual information (Section 2.3).

2.1 Dual variable techniques for column generation

Dual variable techniques in CG are classified into two families: dual variable stabilization procedures, which explicitly control dual value variations between the iterations, and alternative ones, which aim at generating more informative dual solutions or reducing the degeneracy responsible for dual instability. They share the same objective of improving CG convergence by mitigating the adverse effects of dual oscillations and degeneracy. In this section, we provide a brief review of alternative methods before discussing stabilization algorithms.

2.1.1 Alternative dual variable techniques

Alternative methods improve the behavior of CG without explicitly controlling the oscillations of the dual values between the iterations. A first approach consists in selecting more informative dual solutions. Rousseau et al. (2007) observed that the extreme dual points returned by the simplex algorithm favor large dual variable oscillations and proposed to select interior points on the optimal

dual face instead. This strategy produces more stable dual values and improves convergence on highly degenerate vehicle routing problem instances.

Another line of research relies on Dual Optimal Inequalities (DOIs), introduced for cutting-stock problems by Valério de Carvalho (2005) and generalized by Ben Amor et al. (2006). Rather than guiding the dual values, DOIs restrict the dual feasible region while preserving at least one optimal dual solution. They are imposed by considering additional variables in the primal formulation. Gschwind and Irnich (2016) generalized several DOI classes and introduced new DOIs for various packing and coloring problems. Later, for different set-covering applications in computer vision, Yarkony et al. (2020) devised DOIs that vary with the set of columns present in the RMP. In a subsequent work, Haghani et al. (2022) proposed smooth DOI variants requiring less problem-specific information, thereby extending their applicability to broader set-covering formulations.

A complementary technique consists in reducing degeneracy in the RMP itself. For set-partitioning models, dynamic constraint aggregation (DCA, El Hallaoui et al. 2005, 2010) reduces degeneracy by aggregating set-partitioning rows according to the positive-valued variables. The aggregation is dynamically revised to guarantee optimality. Later, Benchimol et al. (2012) combined DCA with the stabilization framework of Oukil et al. (2007), simultaneously reducing degeneracy in the master problem and stabilizing dual variables. Similarly, Costa et al. (2022) proposed dynamically separating aggregated constraints to smooth dual variables and reduce degeneracy, leading to significant reductions in the number of CG iterations and the overall solution time.

These methods illustrate that the dual variable behavior can be restrained through a variety of mechanisms, ranging from centering dual solutions to explicitly restricting the dual space and modifying the master problem structure.

2.1.2 Dual variable stabilization techniques

Dual variable stabilization methods directly regulate the evolution of the dual variables around a reference solution. The underlying principle originates from the *Boxstep* method of Marsten et al. (1975), which constrains dual values within a trust region. Building on this idea, Du Merle et al. (1999) introduced one of the first stabilized CG frameworks that penalizes deviations from a stability center using a fixed trust region that evolves throughout the solution process. The trust region and the penalties are directly expressed in the master problem by adding slack and surplus variables. Their method has yielded substantial speedups on crew pairing instances.

Subsequent work focused on improving both the stabilization mechanism and the initialization of the stability center. Oukil et al. (2007) proposed a piecewise-linear penalty function for the multi-depot vehicle scheduling problem (MDVSP) and initialized the stability center using the dual solution of a related single-depot problem. They reported increasing benefits on larger instances, due to higher accuracy of the dual estimates provided by the single-depot problem variant, highlighting the importance of a high-quality initial stability center. Ben Amor et al. (2009) later evaluated several stabilization schemes within a proximal optimization framework considering both piecewise-linear and quadratic penalties, and demonstrated that the effectiveness of a stabilization technique strongly depends on the penalty function, parameter values, update strategy, and initial dual information, even if speedups can still be observed when the dual estimates are inaccurate.

Compared to what was introduced by Du Merle et al. (1999), two recurring themes emerge from these works. First, several methods rely on a suitable initial stability center, which is not always available. Second, the performance of stabilized CG depends on how the stabilization mechanism is set up and updated throughout the solution process. The computational study of Ben Amor et al. (2009) also illustrates that stabilization does not automatically guarantee faster convergence.

From a nonsmooth optimization perspective, Briant et al. (2008) interpreted CG as a cutting-plane algorithm and compared the classical Kelley method with bundle approaches relying on quadratic

penalties. By exploiting information accumulated during previous iterations, bundle methods produce more stable approximations of the dual function and significantly improve convergence on difficult combinatorial optimization problems.

Another important family of methods relies on dual price smoothing. Instead of penalizing deviations from a stability center, Wentges (1997) proposed using a convex combination of successive dual solutions for pricing, thereby mitigating oscillations. Building on this idea, Pessoa et al. (2018) developed a unified stabilization framework combining smoothing and penalty-based methods with self-adjusting parameter updates. Their computational study showed that automatically managing and combining stabilization mechanisms can outperform either approach used independently. Furthermore, their results indicate that, for vehicle routing problems, the penalty function-based stabilization techniques outperform smoothing-based methods. An explanation for this could be that smoothing methods tend to make an already challenging pricing problem even more difficult. For this reason, we focus in this work on an efficient penalty function-based stabilization algorithm.

Overall, explicit stabilization methods differ in the way they control the dual trajectory, but they consistently identify two key factors governing performance: the quality of the stability center and the management of the stabilization mechanism throughout the solution process. While substantial effort has been devoted to designing adaptive update strategies, little attention has been paid to systematically obtaining high-quality initial stability centers. This is the main goal of our paper.

2.2 Machine learning in column generation

Numerous recent works have shown the potential of combining machine learning and optimization techniques: machine learning is able to exploit high-level information provided by optimization techniques, allowing the prediction of primal solutions (Kool et al. 2019), branching decisions (Gasse et al. 2019), or cutting planes (Paulus et al. 2022). We refer to the surveys of Bengio et al. (2021), Cappart et al. (2023) and of Scavuzzo et al. (2024) for an extended overview of such works. The drawback lies in the probabilistic nature of machine learning models that forces one to adopt a heuristic approach when solving an optimization problem. However, several steps of the CG algorithm do not require an exact model to perform well, and optimality can still be achieved when an exact mechanism is also included in the process.

An example concerns column selection. Morabit et al. (2021) introduced a GNN architecture featuring a bipartite graph to filter the columns returned by the pricing problem by selecting only the most relevant ones. The benefit is twofold: it reduces the number of columns in the RMP, thus making it faster to solve, and the decrease in degeneracy makes the general algorithm converge in fewer iterations. On the other hand, since the pricing problem is solved exactly, the algorithm reaches an optimal solution. Chi et al. (2022) built on this work, focusing on the vehicle routing problem with time windows (VRPTW), and developed a reinforcement learning framework to identify relevant columns to add to the RMP.

Instead of filtering the information provided by the pricing problem, one could guide it toward promising regions of the search space, making the search smaller and faster while maintaining relevance. Following this idea, Morabit et al. (2023) proposed a supervised learning framework based on a GNN to reduce the size of the pricing problem network when solving the VRPTW. They use the resulting reduced network until the pricing problem does not return any negative reduced cost columns before switching back to the original network, thus ensuring optimality. Even if the number of iterations increases, they still observe a time reduction of 40%, and up to 70% for instances with 200 customers. Later, Gerbaux et al. (2025) applied a similar strategy for an electric bus scheduling problem, but without the exact mechanism to maintain optimality. Combined with a metaheuristic, their arc-selection method resulted in a 70% reduction of the computation time with a cost deterioration of only 1.4%.

Finally, machine learning can also be used as a replacement when classical algorithms become inefficient. Reynal et al. (2026a) introduce a hybrid pricing heuristic for the VRPSD with fixed-fleet constraint. In this context, the dynamic programming algorithm classically used is unable to generate relevant columns in a reasonable amount of time. Their heuristic combines reinforcement learning with several heuristic mechanisms to compute negative reduced cost columns to add to the RMP. Embedded in a restricted master heuristic framework, their approach outperforms classical heuristics and reaches solutions with an average optimality gap under 1%.

These works show that machine learning can improve CG performance by acquiring problem-specific knowledge that would otherwise be inaccessible to simpler or faster methods.

2.3 Learning dual information in optimization

Building on the strength of machine learning to acquire problem-specific knowledge, machine learning architectures can then be used to compute otherwise inaccessible data to guide existing optimization methods such as CG or Lagrangian relaxation-based algorithms. In the latter, optimization relies on Lagrangian multipliers that quantify the value of relaxing problem constraints and play a similar role to dual variables in CG.

Several studies have shown that Lagrangian multipliers exhibit structural patterns that can be exploited by graph-based neural architectures. For example, Parjadis et al. (2024) use a GNN to predict Held-Karp multipliers for the traveling salesman problem; Demelas et al. (2024) implement a problem-agnostic algorithm representing a mixed-integer linear program as a bipartite graph to predict multipliers directly from the problem structure; and Bessa et al. (2025) proposed a self-supervised learning framework leveraging a GNN for predicting dual bounds within a Lagrangian decomposition scheme for constraint programming. In all these works, learned multiplier estimates improve the optimization process, leading to faster convergence and reduced computational effort. Although they focus on Lagrangian multipliers rather than linear programming dual variables, they demonstrate that graph-based learning architectures can capture structural relationships between variables and constraints to predict high-quality dual information.

Regarding CG, one recent attempt to incorporate learned dual information into CG is found in Kraul et al. (2023). These authors develop a regression model to predict optimal dual values for the cutting stock problem which are used as a reference point in the method of Du Merle et al. (1999). This accurate estimate of an optimal dual solution yields an improved performance of the method, showing the potential of computing a good stability center.

Another method was later proposed by Sugishita et al. (2024). The authors investigate the use of machine learning models to warm-start CG for the unit commitment problem. Their approach relies on predicting dual variables before the CG procedure begins and using these predictions to initialize the RMP. Computational experiments show that accurate dual predictions can substantially reduce the number of CG iterations required to reach optimality. This work provides early evidence that dual information can be learned and successfully exploited within a CG framework.

Rather than only providing an initial estimate, Shen et al. (2024) integrated machine learning directly into the stabilization process. Their method predicts an optimal dual solution and continuously exploits this information to adapt the stabilization parameters during optimization, achieving large computation time reductions compared with classical stabilization techniques.

Finally, Fang et al. (2025) formulated stabilization itself as a learning problem. Instead of relying on a predefined stabilization mechanism, they trained a neural network to generate stabilized dual solutions directly from the current state of the CG algorithm using a combination of supervised and reinforcement learning. Their learned stabilization policy significantly accelerates convergence on cutting stock and vehicle routing problems while preserving solution quality.

These works show that machine learning can effectively exploit dual information to accelerate CG. While Sugishita et al. (2024) focus on warm-starting the optimization process through dual predictions, Shen et al. (2024) and Fang et al. (2025) integrate learned information directly into the stabilization mechanism itself. This makes machine learning a highly versatile tool to include in a stabilization mechanism. In our work, we develop an architecture adapted to the VRPSD. We choose to use a GNN-based architecture (Wang et al. 2021) able to capture high-level information regarding customer relations, especially when combined with edge features and an attention mechanism. This architecture is able to accurately predict dual values without having to implement a complex and possibly hard-to-tune reinforcement learning framework as in Fang et al. (2025).

3 Problem statement and mathematical formulation

We consider the VRPSD as defined in Christiansen and Lysgaard (2007), Gauvin et al. (2014), Reynal et al. (2026a). Let $\mathcal{N}$ be a set of customers to be serviced by a homogeneous and unlimited fleet of vehicles of capacity Q . Each customer $j \in \mathcal{N}$ is associated with a demand modeled as a random variable following a Poisson distribution of known parameter μ_j , the realization of which is only revealed upon visitation. The demands, considered independent, are to be satisfied by visiting each customer with a single vehicle. The route of each vehicle used must start and end at a common depot, denoted by 0. Let us consider a vehicle already covering a cumulated expected demand of k ; one can then compute the probability $P(CD_j > nQ \mid k)$ that CD_j , the cumulated expected demand served by this vehicle after serving a new customer j , exceeds n times the capacity Q as follows:

$$P(CD_j > nQ \mid k) = 1 - \sum_{i=0}^{nQ-k} \frac{(\mu_j)^i}{i!} e^{-\mu_j}. \quad (1)$$

A failure occurs when the realized demand of a customer cannot be fully satisfied by the inventory left in the vehicle when reaching this customer. Failures are handled using the simple detour-to-depot policy. Routes are designed *a priori*, and whenever a vehicle experiences a failure, it performs a round trip to the depot to replenish and completes the route as planned.

A feasible route $r = (j_0^r, j_1^r, j_2^r, \dots, j_{n_r}^r)$ is a sequence of n_r visited customers, beginning and ending at the depot (i.e., $j_0^r = j_{n_r}^r = 0$) that respects the following constraints:

$$\sum_{l=1}^{n_r-1} \mu_{j_l^r} \leq Q, \quad (2)$$

$$j_{i_1}^r \neq j_{i_2}^r, \quad \forall i_1, i_2 \in \{1, \dots, n_r - 1\}, i_1 < i_2, \quad (3)$$

where the capacity constraint (2) forces the cumulated expected demand served along route r to not exceed the vehicle capacity to avoid multiple expensive round trips to the depot, and the elementarity constraints (3) require that each customer be visited at most once in a route. The VRPSD aims to find a set of feasible routes serving each customer exactly once while minimizing the total expected costs (total distance traveled along the selected routes). Figure 1 illustrates a toy VRPSD example.

We now express the expected cost c_r of traveling along a route r . Let $\mathcal{N}_0 = \mathcal{N} \cup \{0\}$ be the set of all locations and $d_{i,j}$ the Euclidean distance between two locations $i, j \in \mathcal{N}_0$. The cumulated expected demand served by a vehicle following route r upon reaching its l^{th} customer is equal to $\kappa_l^r = \sum_{i=1}^{l-1} \mu_{j_i^r}$ (with the convention $\kappa_1^r = 0$). Furthermore, the failure penalty $p_{j_l^r, \kappa_l^r, j_{l-1}^r}$ is equal to the round trip cost times the probability of experiencing a failure at customer j_l^r , provided it did not already occur at the previous customer j_{l-1}^r :

$$p_{j_l^r, \kappa_l^r, j_{l-1}^r} = 2 d_{0, j_l^r} \sum_{n=1}^{\infty} (P(CD_{j_l^r} > nQ \mid \kappa_l^r) - P(CD_{j_{l-1}^r} > nQ \mid \kappa_{l-1}^r)). \quad (4)$$

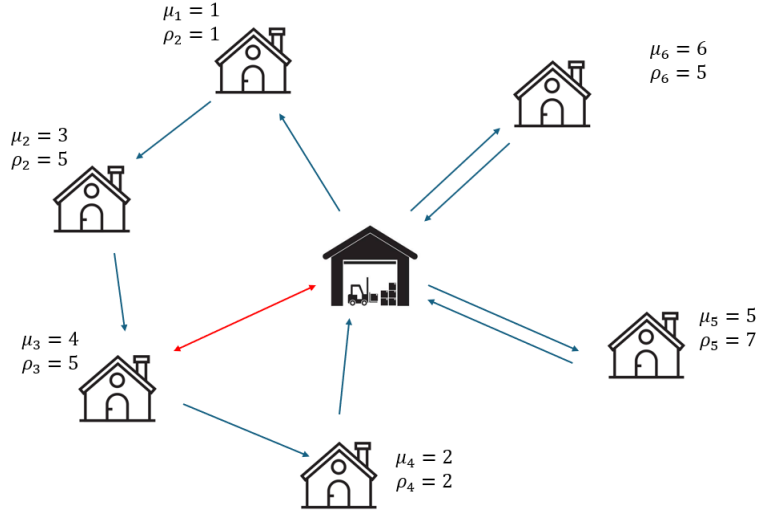


Figure 1: A toy VRPSD example with a feasible solution (blue). At least $K = 3$ vehicles of capacity $Q = 10$ must serve customers of expected demand μ_j . Realization ρ_j is revealed upon visitation. A failure occurs at customer 3, triggering a round trip to the depot (red).

In practice, only the probabilities $P(CD_{j_l^r} > nQ \mid \kappa_l^r)$ with a value greater than 10^{-4} are considered. The resulting expected cost c_r of route r is then:

$$c_r = \sum_{l=1}^{n_r-1} \left(d_{j_{l-1}^r, j_l^r} + p_{j_l^r, \kappa_l^r, j_{l-1}^r} \right) + d_{j_{n_r-1}^r, j_{n_r}^r}. \quad (5)$$

As in previous works, we formulate the VRPSD using a path-flow formulation. Let $\mathcal{R}$ be the set of feasible routes and $\nu_{j,r}$ a binary parameter taking value 1 if route $r \in \mathcal{R}$ serves customer $j \in \mathcal{N}$. Next, let x_r be a binary selection variable associated with route $r \in \mathcal{R}$. Using this notation, the VRPSD is modeled as the following set-partitioning model:

$$\min_x \sum_{r \in \mathcal{R}} c_r x_r \quad (6)$$

$$\text{s.t.} \quad \sum_{r \in \mathcal{R}} \nu_{j,r} x_r = 1, \quad \forall j \in \mathcal{N} \quad (7)$$

$$\sum_{r \in \mathcal{R}} x_r \geq K \quad (8)$$

$$x_r \in \{0, 1\}, \quad \forall r \in \mathcal{R} \quad (9)$$

The objective function (6) minimizes the total expected cost of the selected routes. Constraints (7) ensure that each customer is visited exactly once. Constraint (8) imposes the trivial lower bound $K = \lceil \sum_{j \in \mathcal{N}} \mu_j / Q \rceil$ on the number of vehicles required to service all customers. Finally, constraints (9) ensure that the route variables are binary. In this work, we focus on the linear relaxation of this model, which means that these binary requirements are relaxed.

To solve this linear relaxation, i.e., the master problem, Gauvin et al. (2014) have designed a CG algorithm that will serve as a benchmark algorithm for our tests and that will also be embedded in the dual variable stabilization procedure described in the next section. For the sake of conciseness, we briefly discuss this algorithm, only highlighting what is essential to understand the rest of the paper. For complete details, see Gauvin et al. (2014).

As mentioned in Section 1, a CG algorithm solves at each iteration a RMP and a pricing problem, which aims at finding promising columns to add to the RMP. Let $\pi_j, j \in \mathcal{N}$ and π_0 be the dual

variables associated with constraints (7) and (8), respectively. Then, the reduced cost $\bar{c}_r$ of a variable x_r is given by:

$$\bar{c}_r = c_r - \sum_{j \in \mathcal{N}} \nu_{j,r} \pi_j - \pi_0. \quad (10)$$

The pricing problem searches for routes r with a negative reduced cost $\bar{c}_r$. If no such routes are found, then the current RMP solution is also optimal for the master problem and the algorithm stops. Otherwise, variables representing the routes found are added to the RMP before starting a new iteration.

The pricing problem can be formulated as an elementary shortest path problem with resource constraints (see, e.g., Costa et al. 2019). As proposed by Christiansen and Lysgaard (2007), the pricing network is acyclic and contains duplicated nodes for each customer, namely, one for each possible expected cumulative demand. This network structure allows us to take into account the capacity constraint (2) and to include the failure penalties (4) in the arc costs. Elementary requirements are needed to ensure that no more than one node copy of each customer is visited in a path. The pricing problem is solved using the labeling algorithm of Gauvin et al. (2014).

4 Dual variable stabilization approach

The efficiency of the CG algorithm heavily relies on the quality of the dual information provided by the RMP, which can be improved by a stabilization mechanism. We implement such a method that is similar to that of Benchimol et al. (2012). It consists of penalizing dual values lying outside an interval centered around a so-called *stability center* $\tilde{\pi}$ in order to prevent large dual variations. In the rest of this work, we adopt the convention of denoting vectors and matrices using bold symbols.

In this section, we start by defining the stabilized version of the RMP, including the choice of the penalty function (Section 4.1). We then detail our learning-based algorithm to predict the initial stability center (Section 4.2) before presenting the stabilization parameters' update strategy (Section 4.3) and the stopping criteria (Section 4.4). Finally, a pseudo-code of our stabilized CG algorithm is provided (Section 4.5).

4.1 Stabilized restricted master problem - SolveStabRMP(.)

Dual stabilization aims to penalize a large portion of the dual space to reduce the number of CG iterations needed to find the optimal value of the master problem. As in Oukil et al. (2007), Ben Amor et al. (2009), and Benchimol et al. (2012), we use 3-piecewise linear penalty functions $f_j(\pi_j)$, one for each dual variable π_j , $j \in \mathcal{N}_0$, with $\mathcal{N}_0 = \mathcal{N} \cup \{0\}$. Each function features a trust region $[\delta_j^-, \delta_j^+]$ of non-penalized dual values containing the corresponding component $\tilde{\pi}_j$ of the stability center and two linear pieces of slopes $-\varepsilon_j^-$ and ε_j^+ , respectively, inducing a dual penalty proportional to the distance to the trust region. The penalty functions are thus given by:

$$f_j(\pi_j) = \begin{cases} 0 & \text{if } \pi_j \in [\delta_j^-, \delta_j^+] \\ \varepsilon_j^-(\delta_j^- - \pi_j) & \text{if } \pi_j < \delta_j^- \\ \varepsilon_j^+(\pi_j - \delta_j^+) & \text{if } \pi_j > \delta_j^+ \end{cases} \quad \forall j \in \mathcal{N}_0. \quad (11)$$

To impose these penalty functions, the primal version of the master problem is modified by introducing vectors of bounded slack and surplus variables, denoted $\mathbf{y}^- \in [\mathbf{0}, \boldsymbol{\varepsilon}^-]$ and $\mathbf{y}^+ \in [\mathbf{0}, \boldsymbol{\varepsilon}^+]$, respectively. The stabilized master problem becomes:

$$\min_{\mathbf{x}, \mathbf{y}} \sum_{r \in \mathcal{R}} c_r x_r - \boldsymbol{\delta}^{-\top} \mathbf{y}^- + \boldsymbol{\delta}^{+\top} \mathbf{y}^+ \quad (12)$$

$$\text{s.t.} \quad \sum_{r \in \mathcal{R}} \nu_{j,r} x_r - y_j^- + y_j^+ = 1, \quad \forall j \in \mathcal{N} \quad (13)$$

$$\sum_{r \in \mathcal{R}} x_r - y_0^- + y_0^+ \geq K \quad (14)$$

$$x_r \geq 0, \quad \forall r \in \mathcal{R} \quad (15)$$

$$0 \leq y_j^- \leq \varepsilon_j^- \quad \forall j \in \mathcal{N}_0 \quad (16)$$

$$0 \leq y_j^+ \leq \varepsilon_j^+ \quad \forall j \in \mathcal{N}_0. \quad (17)$$

Denoting by γ^- and γ^+ the vectors of nonnegative dual variables associated with the upper-bound constraints (16) and (17), respectively, the dual stabilized master problem is:

$$\max_{\pi, \gamma} \quad K\pi_0 + \sum_{j \in \mathcal{N}} \pi_j - \varepsilon^{-\top} \gamma^- - \varepsilon^{+\top} \gamma^+ \quad (18)$$

$$\text{s.t.} \quad \pi_0 + \sum_{j \in \mathcal{N}} \nu_{j,r} \pi_j \leq c_r \quad \forall r \in \mathcal{R} \quad (19)$$

$$\delta_j^- - \gamma_j^- \leq \pi_j \leq \delta_j^+ + \gamma_j^+ \quad \forall j \in \mathcal{N}_0 \quad (20)$$

$$\gamma^-, \gamma^+ \geq 0. \quad (21)$$

This dual version clearly highlights how the penalty functions steer the dual values π_j towards their trust-region interval $[\delta_j^-, \delta_j^+]$. The CG algorithm summarized at the end of Section 3 can still be used to solve the stabilized master problem (12)–(17). In particular, it solves at each iteration a stabilized RMP defined as problem (12)–(17) restricted to a subset of its variables that we denote $\hat{\mathcal{R}}$.

Note that adding the $\mathbf{y}^-$ and $\mathbf{y}^+$ variables in the master problem yields a relaxation that may provide a weaker lower bound depending on the values of the stabilization parameters $\delta^-, \delta^+, \varepsilon^-, \varepsilon^+$. To obtain a lower bound equal to the optimal value of the non-stabilized master problem, these parameter values can be adjusted during the solution process. This process stops when an optimal solution to the master problem is obtained and proven optimal according to the following sufficient conditions. A solution to the stabilized RMP is optimal for the non-stabilized master problem if no negative reduced cost column is returned by the pricing problem, and $\mathbf{y}^- = \mathbf{y}^+ = \mathbf{0}$.

4.2 Stability center prediction - Predict(.)

Solving the stabilized master problem (12)–(17) requires choosing the values of parameters $\delta^-, \delta^+, \varepsilon^-, \varepsilon^+$ beforehand. If the size of the trust region and the penalties for lying outside of it have been thoroughly documented (see e.g., Benchimol et al. 2012), the initialization of the stability center has received much less attention, the only general consensus being that an estimate of an optimal dual solution is preferred, the more precise, the better (Ben Amor et al. 2009).

Since no efficient algorithm exists to provide such an estimate for the VRPSD, we predict an optimal dual solution with a supervised machine learning model trained on historical data. Our architecture is based on the edge-featured graph attention network (EGAT, Wang et al. 2021). In the following, we first describe how we collected the training data before describing the features and the architecture itself. Finally, we detail the training phase.

4.2.1 Data collection

Our training set is built from the VRPSD instances solved in Gauvin et al. (2014). Following the procedure used by Reynal et al. (2026b), for each instance, we build several new instances by reducing the fleet size K and proportionally increasing the capacity Q to keep the KQ product constant. Excluding those used in Reynal et al. (2026b) that we keep for testing, this procedure yields 639 VRPSD instances. We solve their linear relaxation to optimality using a barrier algorithm with no crossover for the RMPs, providing an optimal dual solution π^* . We then associate each node in $\mathcal{N}_0$ with a label to predict: π_j^* for each vertex $j \in \mathcal{N}$ and π_0^* for the depot 0. We keep 80% of these dual solutions for training and 20% for validation.

Table 1: Definition of input graph $G = (\mathcal{N}_0, A)$.

Description	Value
$\mathbf{h}_j^{(0)}$: Features on node $j \in \mathcal{N}_0$	
Proportion of the minimal fleet capacity required to serve this customer	$\frac{\mu_j}{KQ}$
Distance to the depot	$d_{0,j}$
Average distance from the 5 nearest customers ($N_5(j)$: set of the 5 nearest customers from j)	$\frac{\sum_{j' \in N_5(j)} d_{j',j}}{5}$
Demand density (σ : standard deviation of the instance's distance distribution)	$\sum_{i \in \mathcal{N} \setminus \{j\}} \mu_i e^{-d_{i,j}/\sigma}$
$\mathbf{e}_{j_1 j_2}^{(0)}$: Features on arc $(j_1, j_2) \in A$	
Length of (j_1, j_2)	d_{j_1, j_2}
Units of demand served per unit of distance traveled	$\frac{\mu_{j_2}}{d_{j_1, j_2}}$

4.2.2 Description of the features

Let $G = (\mathcal{N}_0, A)$ be a directed graph representing a VRPSD instance. An arc exists from node j_1 to node j_2 if they can belong to the same feasible route, i.e., if $\mu_{j_1} + \mu_{j_2} \leq Q$, assuming a demand of 0 for the depot. Each node and arc is embedded using a set of features summarized in Table 1. They contain instance information (normalized demand on the nodes, arc length on the arcs), impact on the objective value of serving the customer (distance needed to return to the depot on the nodes, units of demand served by distance unit traveled on the arcs), and also a representation of the interaction between the customers by computing if a customer is isolated (average distance to the 5 nearest customers on the nodes) or in a high-density demand cluster that requires several vehicles to serve it (demand density on the nodes).

4.2.3 Neural network architecture

Our architecture takes the feature vectors $(\mathbf{h}_j^{(0)})_{j \in \mathcal{N}_0}$ and $(\mathbf{e}_{ij}^{(0)})_{(i,j) \in A}$ representing the nodes and arcs, and transforms them into a prediction $\tilde{\pi}$ of an optimal dual solution. It consists of several EGAT layers with N_H attention heads (as introduced by Wang et al. 2021), followed by a standard multilayer perceptron.

The EGAT layers proceed as follows. Let $(\mathbf{h}_j^{(l)})_{j \in \mathcal{N}_0}$ and $(\mathbf{e}_{ij}^{(l)})_{(i,j) \in A}$ be the node and arc features vectors before the l^{th} layer, respectively. For each attention head $k \in \{1, \dots, N_H\}$, the layer first updates the arc features:

$$\mathbf{f}_{ij}'^{(k)} = \text{LeakyReLU}(\mathbf{W}_i^{(k)} \mathbf{h}_i^{(l)} + \mathbf{W}_j^{(k)} \mathbf{h}_j^{(l)} + \mathbf{W}_{ij}^{(k)} \mathbf{e}_{ij}^{(l)}), \quad \forall (i, j) \in A, \quad (22)$$

where $\mathbf{W}_i^{(k)}$, $\mathbf{W}_j^{(k)}$ and $\mathbf{W}_{ij}^{(k)}$ are learnable matrices, and $\text{LeakyReLU}(x) = \max(0.01x, x)$ is a standard non-linear activation function. It then computes the attention coefficients $\alpha_{ij}^{(k)}$;

$$\alpha_{ij}^{(k)} = \frac{\exp(\mathbf{a}^{(k)\top} \mathbf{f}_{ij}'^{(k)})}{\sum_{p \in \mathcal{V}(j)} \exp(\mathbf{a}^{(k)\top} \mathbf{f}_{pj}'^{(k)})}, \quad \forall (i, j) \in A, \quad (23)$$

where $\mathbf{a}^{(k)}$ is a learnable vector and $\mathcal{V}(j) = \{i \in \mathcal{N}_0 \mid (i, j) \in A\}$ is the set of neighbors of node j . The node information is then aggregated:

$$\mathbf{h}_j^{(k)} = \sum_{p \in \mathcal{V}(j)} \alpha_{pj}^{(k)} \mathbf{W}_p^{(k)} \mathbf{h}_p^{(l)}, \quad \forall j \in \mathcal{N}_0, \quad (24)$$

where $\mathbf{W}^{(k)}$ is a learnable matrix. The node features are then updated by applying **LeakyReLU** as non-linearity and concatenating the aggregated information over all attention heads:

$$\mathbf{h}_j^{(l+1)} = \parallel_{k=1}^{N_H} \text{LeakyReLU}(\mathbf{h}_j'^{(k)}), \quad \forall j \in \mathcal{N}_0. \quad (25)$$

For the arc features, only the aggregation operation is performed since the non-linearity is already applied to compute $\mathbf{f}_{ij}'$:

$$\mathbf{e}_{ij}^{(l+1)} = \parallel_{k=1}^{N_H} \mathbf{f}_{ij}'^{(k)}, \quad \forall (i, j) \in A. \quad (26)$$

After the L^{th} layer, the node feature vectors $(\mathbf{h}_j^{(L)})_{j \in \mathcal{N}_0}$ are passed through M standard multilayer perceptron layers. For each layer $m \in \{1, \dots, M-1\}$ receiving vector $\mathbf{h}_j^{(L+m-1)}$, the update is:

$$\mathbf{h}_j^{(L+m)} = \text{ReLU}(\mathbf{W}_m \mathbf{h}_j^{(L+m-1)} + \mathbf{b}_m), \quad \forall j \in \mathcal{N}_0, \quad (27)$$

where $\mathbf{W}_m$ and $\mathbf{b}_m$ are a learnable coefficient matrix and a bias vector, respectively. Function $\text{ReLU}(x) = \max(0, x)$ is again a standard activation function. Finally, the last linear layer provides the output $\tilde{\pi} = (\tilde{\pi}_j)_{j \in \mathcal{N}_0}$:

$$\tilde{\pi}_j = \mathbf{W}_M \mathbf{h}_j^{(L+M-1)} + b_M, \quad \forall j \in \mathcal{N}_0. \quad (28)$$

4.2.4 Training phase

The core of our loss function is the classic mean squared error between the predicted dual values $\tilde{\pi}$ and the optimal ones π^* . We also exploit the fact that the predicted objective dual value must match the optimal objective dual value to provide a proper bound and thus a relevant stability center. We thus include a regularization term in the loss function by also penalizing predicted dual values, resulting in an objective value $z_{\tilde{\pi}}$ far from the optimum z_{π^*} . The complete loss function is:

$$\mathcal{L}(\tilde{\pi}, \pi^*) = \frac{1}{|\mathcal{N}_0|} \sum_{j \in \mathcal{N}_0} (\tilde{\pi}_j - \pi_j^*)^2 + \alpha |z_{\tilde{\pi}} - z_{\pi^*}|. \quad (29)$$

After testing several values for α , we choose to set it to 0.01, which resulted in the lowest mean absolute prediction error. Our architecture features 4 EGAT layers with respective size of (32, 64, 64, 96) monitored by 3 attention heads, followed by a multilayer perceptron (3 layers of (32, 64, 32) neurons, respectively), resulting in an average inference time of 0.05s. We use the Adam (Kingma 2014) optimizer with default parameters and assume convergence if the loss reduction over 5 iterations is less than 10^{-6} . The output is, for each node $j \in \mathcal{N}_0$, the prediction $\tilde{\pi}_j$ of its associated value in an optimal dual solution. Our architecture is implemented using PyTorch and DGL libraries (Wang et al. 2019, Paszke et al. 2017) for a total training time of 3 hours.

4.3 Stabilization parameters' update strategy - Update(.)

As said at the end of Section 4.1, the stabilization parameters may need to be updated during the solution process to reach the lower bound provided by the non-stabilized master problem. We apply the following expansion-contraction update strategy, as presented in Benchimol et al. (2012), which independently adjusts the parameters for each slack and surplus variable, y_j^+ and y_j^- .

Let $\pi_j^t, j \in \mathcal{N}_0$, be the dual values at a CG iteration t , where an update is performed. Let $(\delta_j^-)^t$ and $(\delta_j^+)^t$ be the corresponding bounds on the trust region, and $(\varepsilon_j^-)^t$ and $(\varepsilon_j^+)^t$ be the associated linear penalties for lying outside the trust region for dual variable π_j^t at the same iteration. The stability center is updated to the current dual solution:

$$\tilde{\pi}_j^{t+1} = \pi_j^t, \quad \forall j \in \mathcal{N}_0. \quad (30)$$

The trust region is updated using this updated stability center. The trust interval of a dual variable π_j is reduced if its current value π_j^t lies strictly inside its current trust region $[(\delta_j^-)^t, (\delta_j^+)^t]$, and increased otherwise:

$$(\delta_j^-)^{t+1} = \begin{cases} \tilde{\pi}_j^{t+1} - 0.5\lambda_j^t & \text{if } (\delta_j^-)^t < \pi_j^t < (\delta_j^+)^t, \\ \tilde{\pi}_j^{t+1} - 4\lambda_j^t & \text{if } \pi_j^t \leq (\delta_j^-)^t \text{ or } (\delta_j^+)^t \leq \pi_j^t, \end{cases} \quad \forall j \in \mathcal{N}_0, \quad (31)$$

$$(\delta_j^+)^{t+1} = \begin{cases} \tilde{\pi}_j^{t+1} + 0.5\lambda_j^t & \text{if } (\delta_j^-)^t < \pi_j^t < (\delta_j^+)^t, \\ \tilde{\pi}_j^{t+1} + 4\lambda_j^t & \text{if } \pi_j^t \leq (\delta_j^-)^t \text{ or } (\delta_j^+)^t \leq \pi_j^t, \end{cases} \quad \forall j \in \mathcal{N}_0, \quad (32)$$

where $\lambda_j^t = ((\delta_j^+)^t - (\delta_j^-)^t)/2$ is half of the trust interval length for $j \in \mathcal{N}_0$. Finally, the penalties are increased if the current dual value strictly lies in its current trust region, and decreased otherwise:

$$(\varepsilon_j^-)^{t+1} = \begin{cases} \min\{2(\varepsilon_j^-)^t, 1\} & \text{if } (\delta_j^-)^t < \pi_j^t < (\delta_j^+)^t, \\ \max\{0.5(\varepsilon_j^-)^t, 10^{-4}\} & \text{if } \pi_j^t \leq (\delta_j^-)^t \text{ or } (\delta_j^+)^t \leq \pi_j^t, \end{cases} \quad \forall j \in \mathcal{N}_0, \quad (33)$$

$$(\varepsilon_j^+)^{t+1} = \begin{cases} \min\{2(\varepsilon_j^+)^t, 1\} & \text{if } (\delta_j^-)^t < \pi_j^t < (\delta_j^+)^t, \\ \max\{0.5(\varepsilon_j^+)^t, 10^{-4}\} & \text{if } \pi_j^t \leq (\delta_j^-)^t \text{ or } (\delta_j^+)^t \leq \pi_j^t, \end{cases} \quad \forall j \in \mathcal{N}_0. \quad (34)$$

In a classical framework (see, e.g., Oukil et al. 2007, Benchimol et al. 2012), the update is performed after a *major iteration*, defined as an iteration that did not return at least one negative reduced cost column. However, with this update strategy, the CG tailing-off effect occurs multiple times, namely, before each major iteration. This can result in a stabilized CG approach performing worse than a non-stabilized one (Ben Amor et al. 2009). We address this issue by updating the stabilization parameters when no negative reduced cost columns are generated or when a given number of iterations have been performed since the last update, thereby mitigating the tailing-off effect during stabilized CG. After testing several values in $[1, 25]$, we chose to update the parameters at most at every 10 iterations.

4.4 Stopping criteria

As mentioned in Section 4.1, the pricing problem not returning a negative reduced cost column is no longer a valid stopping criterion: the slack and surplus variables $\mathbf{y}^+$ and $\mathbf{y}^-$ must also be equal to 0. Thus, the algorithm stops when these conditions are met.

We also apply the following stopping criterion. Let ϕ be an iteration for which no negative reduced cost columns are returned by the pricing problem and at least one variable y_j^+ or y_j^- takes a positive value. Let ϕ^+ be the following iteration when this occurs. Denote by z_t^* and $(\mathbf{x}_t^*, \mathbf{y}_t^{+*}, \mathbf{y}_t^{-*})$ the optimal value and solution of the stabilized RMP at iteration t . Given our parameter update strategy and according to Proposition 1 of Benchimol et al. (2012), the algorithm can stop if $z_\phi^* = z_{\phi^+}^*$ and no parameter update has been performed after iteration ϕ . In this case, $\mathbf{x}_\phi^*$ (with cost z_ϕ^*) is an optimal solution of the non-stabilized master problem.

4.5 Pseudo-code of the algorithm

We provide in Algorithm 1 a pseudo-code of our dual variable stabilization algorithm. At line 1, we start by initializing the optimal value z^* to ∞ , the optimal solution $\mathbf{x}^*$ to $\{\}$, the iteration counter t to 0, and the last iteration at which the pricing problem did not find a negative reduced cost column. We then predict at line 2 the initial stability center $\tilde{\pi}$ according to the description made in Section 4.2 through procedure PREDICT(.), and initialize the penalty function parameters at line 3. We enter the CG loop: we first update the iteration counter at line 5 and solve the stabilized RMP using procedure SOLVESTABRMP(.) to get the objective value z , and the primal and dual solutions $(\mathbf{x}, \mathbf{y}^+, \mathbf{y}^-)$ and $(\pi, \gamma^+, \gamma^-)$ at line 6. The pricing problem is then solved at line 7, returning a pool $\mathcal{P}$ of negative reduced cost columns if some are found. If this pool is not empty, we update the set of routes in the

RMP accordingly (line 9) and, if no parameter update has been performed in the last 10 iterations (line 10), we also update the stability center, trust region, and penalties through procedure UPDATE(.) as described in Section 4.3 (line 11). Otherwise ($\mathcal{P} = \emptyset$), we check at lines 13–15 if one of the stopping criteria described in Section 4.4 is satisfied: if this the case, we return the optimal value and the optimal solution that depend on the conditions satisfied. Otherwise, we update z^* , $\mathbf{x}^*$, the stability center, the trust region, and the penalties at lines 18–19, before starting a new iteration.

Algorithm 1 DualStabilization(.)

Require: $\hat{\mathcal{R}}$: subset of routes in the initial RMP
Require: K, Q : minimal number of vehicles and their capacity
Require: μ : expected customer demands
Require: d : traveling distances
Require: l : initial trust region size
Require: ε_0 : initial penalty
Output: $z^*, \mathbf{x}^*$: optimal objective value and corresponding solution

```

1:  $z^* \leftarrow \infty, \mathbf{x}^* \leftarrow \{\}, t \leftarrow 0, t_m \leftarrow 0$ 
2:  $\tilde{\pi} \leftarrow \text{PREDICT}(\mu, d, Q, K)$ 
3:  $\delta^- \leftarrow \tilde{\pi} - l/2, \delta^+ \leftarrow \tilde{\pi} + l/2, \varepsilon^- \leftarrow \varepsilon_0, \varepsilon^+ \leftarrow \varepsilon_0$ 
4: while true do
5:    $t \leftarrow t + 1$ 
6:    $z, (\mathbf{x}, \mathbf{y}^+, \mathbf{y}^-), (\pi, \gamma^+, \gamma^-) \leftarrow \text{SOLVESTABRMP}(\hat{\mathcal{R}}, \delta^-, \delta^+, \varepsilon^-, \varepsilon^+)$ 
7:    $\mathcal{P} \leftarrow \text{PRICE}(\pi)$ 
8:   if  $\mathcal{P} \neq \emptyset$  then
9:      $\hat{\mathcal{R}} \leftarrow \hat{\mathcal{R}} \cup \mathcal{P}$ 
10:    if  $t - t_m \equiv 0 \pmod{10}$  then
11:       $\tilde{\pi}, \delta^-, \delta^+, \varepsilon^-, \varepsilon^+ \leftarrow \text{UPDATE}(\pi, \delta^-, \delta^+, \varepsilon^-, \varepsilon^+)$ 
12:    else
13:      if  $(\mathbf{y}^+ = \mathbf{0}) \wedge (\mathbf{y}^- = \mathbf{0})$  then
14:        return  $z, \mathbf{x}$ 
15:      else if  $(z = z^*) \wedge (t - t_m < 10)$  then
16:        return  $z^*, \mathbf{x}^*$ 
17:      else
18:         $z^*, \mathbf{x}^* \leftarrow z, \mathbf{x}$ 
19:         $\tilde{\pi}, \delta^-, \delta^+, \varepsilon^-, \varepsilon^+ \leftarrow \text{UPDATE}(\pi, \delta^-, \delta^+, \varepsilon^-, \varepsilon^+)$ 
20:         $t_m \leftarrow t$ 

```

5 Experimental results

To assess the performance of our dual stabilization framework, we perform a series of experiments on 40 instances of the VRPSD featuring 34 to 60 customers, introduced in Reynal et al. (2026b). These instances are absent from the training and validation datasets used to train the prediction model. We set the initial size of the trust region $[\delta_j^-, \delta_j^+]$ to 0.1, and the initial penalties ε_j^- and ε_j^+ to 0.1 for all $j \in \mathcal{N}_0$, as in Benchimol et al. (2012). The CG algorithm is implemented using the Gencol library, version 4.5, relying on CPLEX 22.1.1 to solve the RMPs using a single thread. All tests were conducted on an Intel(R) Core(TM) i7 – 8700 3.20GHz CPU.

We compare the following methods:

BASE: The standard CG algorithm without a stabilization mechanism.

STABPRED: Our dual stabilization framework using the predicted stability center as the starting point of the stabilization and periodic updates of the stabilization parameters.

STABPRED_NOPER: Same as STABPRED but without periodic stabilization parameter updates (i.e., only when no negative reduced cost columns are found).

DELSTABT: We delay stabilization by first solving the unstabilized RMP for T CG iterations ($T = 1, 5, 10, 20, 30, 40, 50$) before applying the stabilization strategy of STABPRED using the dual variables returned by the T^{th} RMP resolution as the first stability center. This method constitutes a simple yet efficient baseline one could implement in the absence of a relevant algorithm to pro-

vide a relevant initial stability center. Method DELSTAB1 with a non-periodic update strategy is similar to the method implemented in Du Merle et al. (1999).

DELSTABT_NOPER: Same as DELSTABT but without periodic stabilization parameter updates.

NOISEP: Same as STABPRED but we use optimal dual variables to which we apply a randomly uniform multiplicative (positive or negative) perturbation of $P\%$ ($P = 1, 2, 5, 10, 15, 20$), different for each variable. Similarly to Ben Amor et al. (2009), this method is a standard baseline for evaluating the quality of the dual variable prediction made in STABPRED but cannot be used to assess the competitiveness of other methods since they require an optimal dual solution as input. To account for stochasticity, we average the performance over 5 seeds.

We evaluate our method against an oracle that we consider an idealized benchmark (with perfect prediction and perfect confidence in our model) without fundamentally changing the framework. It performs stabilized CG with an initial point corresponding to an optimal dual solution to the problem, and waits until no negative reduced cost columns are found to update the stability center. Since the trust region contains an optimal dual point, the method is guaranteed to reach the optimum of the linear relaxation without having to update the stability center.

Since computation time varies heavily between two instances, we express the results as a computation time increase factor compared to the oracle. On any given instance i , let $t_m(i)$ and $t_o(i)$ be the times needed by a method m and the oracle, respectively, to solve the instance. The time increase factor $F(m, i)$ of method m for instance i is then:

$$F(m, i) = \frac{t_m(i)}{t_o(i)}. \quad (35)$$

In the following subsections, all results are reported through cumulative distribution profiles, representing the proportion of instances solved with a time increase factor less than a specified value. We impose a timeout of 600 seconds and use 600 seconds to compute the time increase factor for a method that cannot solve an instance within this time limit.

In Table 2, we report the computation times for each instance obtained by the most relevant methods. For each method, the two bottom rows provide the averages of the computation times and of the time increase factors over all instances. We observe that, for both statistics, STABPRED yields the best averages followed closely by DELSTAB10 for the average time. Nevertheless, the much better average time increase factor indicates that STABPRED is more robust than DELSTAB10. To complement these results, Figure 2 also provides boxplots of the number of CG iterations performed by each method for all methods. Like for the computation time, this data is normalized with respect to the number of iterations required by the oracle. These results highlight again the robustness of STABPRED compared to DELSTAB10.

Below, we further study these results. We first focus on analyzing the sensitivity of the baseline method DELSTABT to its parameter T in Section 5.1. We then evaluate the impact of the periodic update strategy in Section 5.2, before assessing the performance of our predicted stability center in Section 5.3. Finally, we discuss the performance of our model in terms of dual variables' prediction through a comparison with NOISEP in Section 5.4

5.1 Baseline tuning

The algorithms used in Ben Amor et al. (2009) and Benchimol et al. (2012) to compute an accurate initial stability center cannot be applied to the VRPSD, thus calling for another method to compare to. In this section, we tune method DELSTABT, which serves as this comparison baseline.

We exploit the behavior of CG: during the first iterations, the dual variables are far from their optimal value, but they tend to stabilize after some time. The algorithm then enters the tailing-off phase in which the dual variables oscillate without resulting in much change in the objective value.

Table 2: Computation times (in seconds) for relevant methods. Asterisks indicate a timeout (> 600 seconds).

Instance	ORACLE	BASE	STABPRED	DELSTAB1	DELSTAB5	DELSTAB10	DELSTAB1	STABPRED
							_NoPER	_NoPER
A-n39-k2-300	27.6	39.7	33.5	37.6	45.3	44.8	43.5	34.5
A-n44-k2-300	39.3	61.9	49.9	53.0	49.7	48.2	62.0	56.1
A-n44-k4-150	27.9	35.7	30.9	33.0	30.9	36.6	39.3	33.3
A-n44-k5-120	20.8	26.0	25.6	27.2	25.6	28.8	30.4	25.6
A-n45-k2-300	45.4	69.1	46.1	60.7	50.6	54.2	70.8	52.2
A-n45-k3-200	29.4	45.1	30.4	31.6	41.1	32.6	42.3	36.9
A-n48-k2-350	93.0	161.9	111.4	135.2	118.8	118.3	110.9	108.1
A-n48-k4-175	19.4	26.8	21.7	19.6	21.4	23.3	27.3	29.5
A-n53-k2-350	180.6	340.7	295.7	287.5	273.8	208.7	318.3	224.4
A-n53-k4-175	30.8	38.7	31.5	35.1	39.3	34.5	39.5	33.1
A-n53-k5-140	27.0	35.0	29.8	32.3	32.0	29.8	44.5	32.1
A-n55-k2-450	163.9	275.4	178.0	199.1	188.2	174.0	269.2	203.5
A-n55-k3-300	55.9	74.5	55.9	59.1	60.7	65.8	69.4	70.2
A-n60-k3-300	128.9	148.6	154.4	171.2	178.4	133.9	151.6	138.9
B-n34-k2-250	49.7	74.6	56.4	68.6	51.7	50.9	75.1	68.3
B-n34-k4-125	24.0	31.8	27.9	30.6	26.4	28.2	33.3	28.2
B-n34-k5-100	22.8	30.0	26.4	28.8	27.0	32.4	33.0	28.2
B-n35-k2-250	55.6	114.9	59.2	133.8	113.1	93.2	99.3	51.7
B-n39-k2-250	75.5	(*)	80.9	109.2	87.8	76.6	(*)	74.0
B-n41-k3-200	27.2	36.9	29.9	33.2	28.1	39.8	37.4	31.1
B-n43-k3-200	31.8	39.2	32.5	43.7	42.9	34.3	38.7	29.9
B-n43-k4-150	27.8	34.8	31.3	33.8	33.3	30.3	39.8	41.5
B-n43-k6-100	23.4	30.6	25.8	28.2	27.0	25.2	34.2	34.8
B-n44-k4-175	32.8	44.0	36.5	35.8	35.8	46.5	53.3	51.8
B-n44-k5-140	22.8	32.1	24.6	26.1	28.8	23.1	30.9	29.4
B-n44-k7-100	21.7	35.7	27.3	32.9	29.4	24.5	33.6	31.5
B-n45-k2-300	92.6	113.3	92.7	94.1	94.7	93.0	123.7	95.8
B-n45-k3-200	22.2	30.1	27.7	32.4	34.4	26.8	32.6	28.0
B-n45-k4-125	40.2	44.2	40.7	42.2	42.2	42.3	45.9	42.8
B-n45-k4-150	36.5	39.3	40.8	39.8	47.8	38.8	38.5	38.5
B-n45-k5-120	29.6	36.4	33.2	35.2	37.6	34.4	42.4	35.2
B-n50-k4-175	28.0	47.7	30.1	33.7	29.8	30.5	43.4	41.6
B-n50-k5-140	29.9	38.9	34.3	33.0	36.7	56.8	40.1	44.4
B-n51-k2-350	208.2	(*)	259.3	305.6	303.1	276.4	(*)	(*)
B-n51-k4-175	27.1	37.0	30.4	30.8	30.9	28.4	38.2	32.0
B-n51-k5-140	25.7	35.5	31.1	39.3	36.7	28.4	33.5	37.2
B-n51-k7-100	24.0	24.3	26.1	28.2	28.5	24.3	30.3	26.4
B-n52-k2-350	348.2	(*)	363.7	410.0	514.2	413.2	536.8	343.7
B-n52-k4-175	32.8	47.8	41.0	45.7	44.3	42.4	41.8	36.3
B-n52-k5-140	20.5	22.9	23.5	24.8	29.2	24.3	26.7	26.3
Average time (s)	56.8	> 105.0	65.7	74.5	74.9	67.5	> 102.5	> 75.2
Average time increase factor		> 1.60	1.14	1.28	1.27	1.20	> 1.62	> 1.27

We thus perform a given number T of unstabilized CG iterations, before applying the stabilization framework described in Section 4.5, using the last dual values as stability center. If the stabilization is applied right before the beginning of the tailing-off phase, one can keep the benefits of the algorithm, namely, mitigating the tailing-off effect (Ben Amor et al. 2009, Benchimol et al. 2012), while ensuring that the stability center is rather accurate.

We experiment with several values for T ranging from 1 to 50. We did not explore larger values because half of our instances take less than 50 iterations to solve, meaning that with larger values, it would be equivalent to an unstabilized CG for half of the dataset. The cumulative distribution profiles are provided in Figure 3, and computation times for DELSTAB1, DELSTAB5 and DELSTAB10 can be found in Table 2. As can be seen from this figure, a value of $T = 10$ iterations seems to be the sweet spot for our specific benchmark, which might depend on the problem size, but provides the best results for our dataset. With fewer iterations, the dual values are not yet sufficiently informative, which mitigates the benefits of stabilization. Nevertheless, fewer iterations are preferable to a long, unstabilized CG phase. Since dual stabilization is an efficient method in itself, even without a good

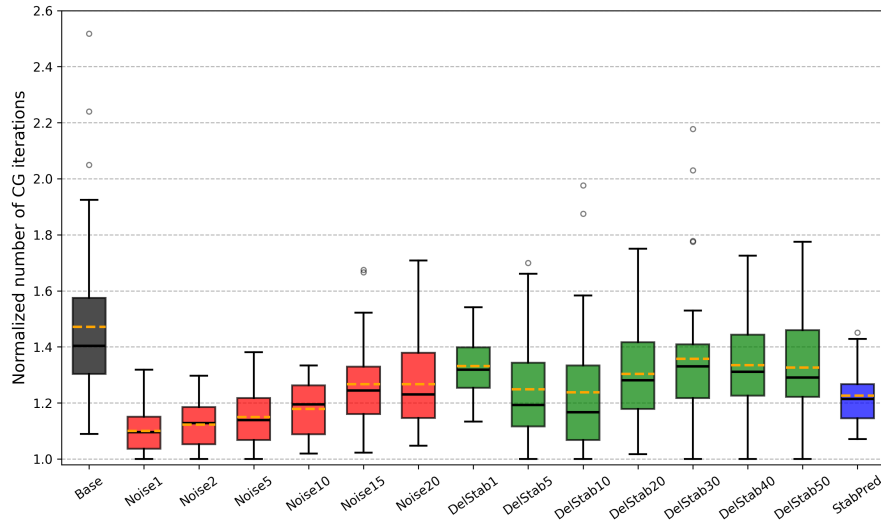


Figure 2: Boxplot of the normalized number of CG iterations required by each method. Medians appear as a plain black line while means appear as an orange dotted line. Boxes represent the first and third quartiles, whereas circles are outliers.

initial stability center (see, e.g., Ben Amor et al. 2009), it is better to benefit from it as soon as possible rather than trying to obtain a good estimate of the optimal dual values at any cost. This conclusion still holds when looking at the number of required CG iterations, even if DELSTAB10 needs a larger number of iterations for two instances among the 40. However, it still presents the lowest average and median of all considered DELSTABT variants.

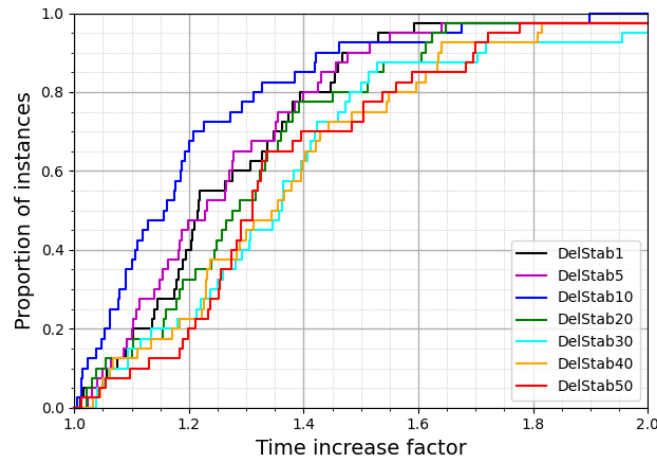


Figure 3: Cumulative distribution profiles of the time increase factor for DelStabT with several values for T .

5.2 Impact of the periodic update strategy

We now present the impact of our periodic stabilization parameter update strategy. To do so, we compare methods STABPRED and DELSTAB1 with periodic updates to their counterpart versions STABPRED_NOPER and DELSTAB1_NOPER without periodic updates, respectively. The latter method is briefly studied in Benchimol et al. (2012). We also include the unstabilized method BASE in the comparison. The cumulative distribution profiles related to this experiment are provided in Figure 4, and computation times for all methods studied in this section can be found in Table 2.

The profiles show that the periodic update produces a major improvement for both methods: for STABPRED, introducing the periodic update results in 77.5% of the instances taking at most 20% more time to solve than the oracle. Removing the periodic update drops the proportion of instances under a 20% time increase to 47.5% and the 80% threshold is only reached with a time increase of 45%. Similar conclusions can be drawn for DELSTAB1, thus underlining that the benefits of a regular update are independent of the quality of the initial stability center. An explanation (mentioned in Ben Amor et al. 2009) is that variants without a periodic update suffer from the tailing-off effect between updates. This explains why DELSTAB1 without a periodic update performs worse than an unstabilized CG algorithm.

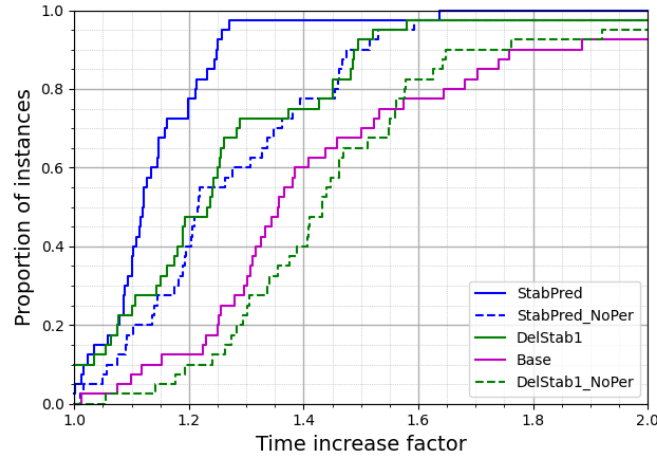


Figure 4: Cumulative distribution profiles of the time increase factor for BoxPred, DelStab1, and Base. Solid lines feature periodic stabilization parameters' updates while dash lines do not.

5.3 Impact of the initial stability center

We now focus on the impact of our prediction model on the method's overall speed. We compare STABPRED to BASE, DELSTAB5 and DELSTAB10 (which seemed to be the two best candidates for method DELSTABT). The cumulative distribution profiles related to this experiment are provided in Figure 5. The computation times for these methods are also provided in Table 2.

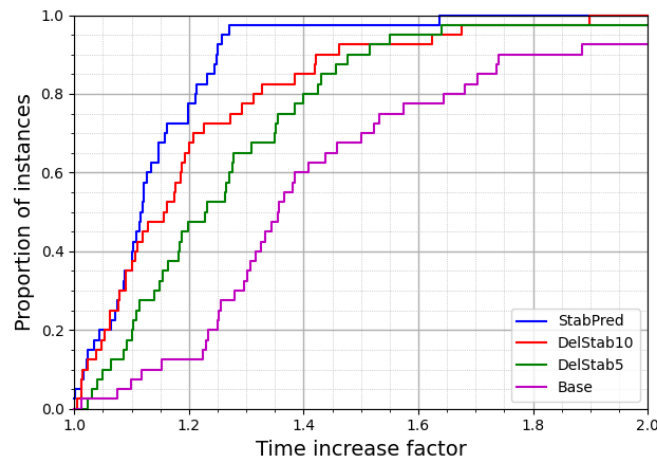


Figure 5: Cumulative distribution profiles of the time increase factor for StabPred, DelStab5, DelStab10 and Base

As the profiles show, STABPRED is the best alternative. Even if both methods STABPRED and DELSTAB10 give similar time increase factors of around 10% for half of the instances, the difference is

made on the harder ones: STABPRED achieves a 20% time increase factor for 77.5% of the instances, while DELSTAB10 only keeps 65% instances within a 20% time increase factor and reaches the 90% threshold with a 42% time increase factor. This confirms that starting stabilization with an accurate stability center is more efficient than choosing it arbitrarily and shows that even in the absence of an algorithm to compute it, machine learning is a viable option for finding one, providing robustness and better worst-case scenarios. This conclusion still holds considering the boxplots in Figure 2, STABPRED presents a similar average number of iterations but a much better worst-case scenario than DELSTAB10.

5.4 Performance analysis of predicted dual variables

Our last experiment aims to determine the gap still to be filled to improve our method. We compare our method STABPRED to methods DELSTAB5 and DELSTAB10, and to method NOISEP with several values of P ranging from 1 to 20%. The related cumulative distribution profiles are provided in Figure 6. In the same figure, we also present a cumulative distribution of the percentage error in predicting the duals. We chose not to represent the absolute error, which would be misleading here since the dual variables range from a few units to more than a hundred.

Our method STABPRED positions itself between NOISE10 and NOISE15, both in terms of number of iterations and computation time. The prediction itself has an error of less than 20% for almost 80% of the duals. Some predictions are made with a much larger error. However, in most cases, these large errors result from a misprediction on a dual with a small value, where a small absolute error can correspond to a large relative error and have a marginal impact on the algorithm's convergence.

Finally, NOISE20 yields performance similar to that of DELSTAB5 in terms of time increase factor. This allows gauging the accuracy required to compute dual estimates so that they meaningfully impact the method's computation time. With a 20% error, the benefits of the heavy prediction architecture are limited compared to simpler stabilization strategies. One should then aim for an error of at most 15%, 10% appearing as an achievable bound outperforming all other reasonable variants and thus appearing as an ideal compromise.

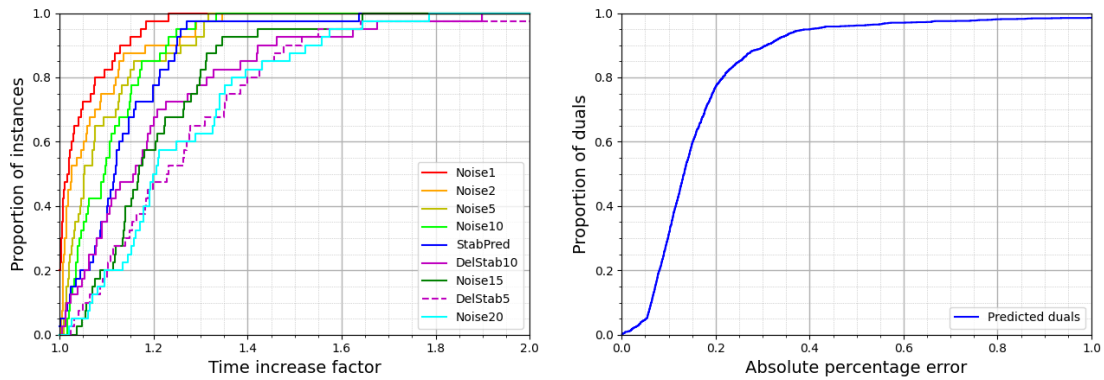


Figure 6: Cumulative distribution profiles of the time increase factor for StabPred, DelStab5, DelStab10 and NoiseP with several P values (left) and cumulative distribution profile of the absolute percentage error in prediction (right).

6 Conclusion

Dual variable stabilization methods have been proposed to mitigate high degeneracy and the tailing-effect observed in standard CG algorithms. Such a stabilization mechanism penalizes dual variables lying outside a trust region, limiting dual variable oscillations from one CG iteration to the next. The performance of this kind of algorithm depends on the choice of the initial stability center that is used to define the initial trust region. Our results showed that a machine learning architecture

relying on GNNs to predict this stability center can substantially improve the performance of existing stabilization techniques. On the considered benchmark instances, our approach outperforms variants that do not benefit from the predicted stability center. We also demonstrated empirically that the periodic update represents a valuable and easy-to-implement asset.

Finally, our experiments showed that exact estimates of optimal dual values are not necessary for the stability center to have an impact on the overall method's efficiency. Moderate prediction errors can still lead to substantial acceleration, highlighting the robustness of stabilization with respect to the quality of the initial stability center. Our study on the impact of the initial stability center shows that one should aim for an error of less than 15% when trying to predict the optimal dual values. Otherwise, a faster or more intuitive method should be preferred since the choice of initial stability center does not seem to matter much under these conditions.

Future work could aim for a more accurate prediction model. However, the fact that this paper only tackles the linear relaxation leaves an unaddressed gap: in many practical contexts, an integer solution is often wished for. We suggest trying to integrate the dual stabilization mechanism in an exact branch-and-price or in a restricted master heuristic (see, e.g., Sadykov et al. 2019), which would certainly improve the performance of these integer methods and open the door to many applications.

References

- Hatem Ben Amor, Jacques Desrosiers, and José Manuel Valério de Carvalho. Dual-optimal inequalities for stabilized column generation. *Operations Research*, 54(3):454–463, 2006.
- Hatem MT Ben Amor, Jacques Desrosiers, and Antonio Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167–1184, 2009.
- 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.
- Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d’horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.
- Swann Bessa, Darius Dabert, Max Bourgeat, Louis-Martin Rousseau, and Quentin Cappart. Learning valid dual bounds in constraint programming: boosted Lagrangian decomposition with self-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 11113–11121, 2025.
- Olivier Briant, Claude Lemaréchal, Ph Meurdesoif, Sophie Michel, Nancy Perrot, and François Vanderbeck. Comparison of bundle and classical column generation. *Mathematical Programming*, 113(2):299–344, 2008.
- Quentin Cappart, Didier Chételat, Elias B Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.
- Cheng Chi, Amine Aboussalah, Elias Khalil, Juyoung Wang, and Zoha Sherkat-Masoumi. A deep reinforcement learning framework for column generation. *Advances in Neural Information Processing Systems*, 35:9633–9644, 2022.
- Christian H Christiansen and Jens Lysgaard. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 35(6):773–781, 2007.
- Luciano Costa, Claudio Contardo, and Guy Desaulniers. Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985, 2019.
- Luciano Costa, Claudio Contardo, Guy Desaulniers, and Julian Yarkony. Stabilized column generation via the dynamic separation of aggregated rows. *INFORMS Journal on Computing*, 34(2):1141–1156, 2022.
- Francesco Demelas, Joseph Le Roux, Mathieu Lacroix, and Axel Parmentier. Predicting Lagrangian multipliers for mixed integer linear programs. In *International Conference on Machine Learning*, pages 10368–10384, 2024.
- Guy Desaulniers, François Lessard, and Ahmed Hadjar. Tabu search, partial elementarity, and generalized k -path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008.
- Martin Desrochers and François Soumis. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13, 1989.

- Jacques Desrosiers, Marco Lübbecke, Guy Desaulniers, and Jean Bertrand Gauthier. Branch-and-price. Springer, Cham, 2026.
- Olivier Du Merle, Daniel Villeneuve, Jacques Desrosiers, and Pierre Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.
- Issmail El Hallaoui, Daniel Villeneuve, François Soumis, and Guy Desaulniers. Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research*, 53(4):632–645, 2005.
- Issmail El Hallaoui, Abdelmoutalib Mentrane, François Soumis, and Guy Desaulniers. Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123(2):345–370, 2010.
- Lichang Fang, Haofeng Yuan, Shiji Song, and Bokui Chen. Learning to stabilize column generation. In 2025 International Joint Conference on Neural Networks (IJCNN), pages 1–8, 2025.
- Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), pages 15580–15592, 2019.
- Charles Gauvin, Guy Desaulniers, and Michel Gendreau. A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 50:141–153, 2014.
- Juliette Gerbaux, Guy Desaulniers, and Quentin Cappart. A machine-learning-based column generation heuristic for electric bus scheduling. *Computers & Operations Research*, 173:106848, 2025.
- Timo Gschwind and Stefan Irnich. Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, 28(1):175–194, 2016.
- Naveed Haghani, Claudio Contardo, and Julian Yarkony. Smooth and flexible dual optimal inequalities. *INFORMS Journal on Optimization*, 4(1):29–44, 2022.
- Atoosa Kasirzadeh, Mohammed Saddoune, and François Soumis. Airline crew scheduling: models, algorithms, and data sets. *EURO Journal on Transportation and Logistics*, 6(2):111–137, 2017.
- Diederik P Kingma. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
- Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In International Conference on Learning Representations, 2019.
- Sebastian Kraul, Markus Seizinger, and Jens O Brunner. Machine learning-supported prediction of dual variables for the cutting stock problem with an application in stabilized column generation. *INFORMS Journal on Computing*, 35(3):692–709, 2023.
- Marco E Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.
- Roy E Marsten, William Walter Hogan, and Jacob Watson Blankenship. The boxstep method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.
- Mouad Morabit, Guy Desaulniers, and Andrea Lodi. Machine-learning-based column selection for column generation. *Transportation Science*, 55(4):815–831, 2021.
- Mouad Morabit, Guy Desaulniers, and Andrea Lodi. Machine-learning-based arc selection for constrained shortest path problems in column generation. *INFORMS Journal on Optimization*, 5(2):191–210, 2023.
- 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.
- Augustin Parjadis, Quentin Cappart, Bistra Dilkina, Aaron Ferber, and Louis-Martin Rousseau. Learning Lagrangian multipliers for the travelling salesman problem. In 30th International Conference on Principles and Practice of Constraint Programming (CP 2024), pages 1–22, 2024.
- Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, Adam Lerer, et al. Automatic differentiation in PyTorch. In 31st Conference on Neural Information Processing Systems (NeurIPS 2017), page vol 8, 2017.
- Max B Paulus, Giulia Zarpellon, Andreas Krause, Laurent Charlin, and Chris Maddison. Learning to cut by looking ahead: Cutting plane selection via imitation learning. In 39th International Conference on Machine Learning (ICML 2022), pages 17584–17600, 2022.
- Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018.

- Frédéric Quesnel, Alice Wu, Guy Desaulniers, and François Soumis. Deep-learning-based partial pricing in a branch-and-price algorithm for personalized crew rostering. *Computers & Operations Research*, 138: 105554, 2022.
- Gaël Reynal, Quentin Cappart, Guy Desaulniers, and Louis-Martin Rousseau. A hybrid learning-based matheuristic to solve the vehicle routing problem with stochastic demands. In *23rd International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2026)*, pages 453–469, 2026a.
- Gaël Reynal, Quentin Cappart, Guy Desaulniers, and Louis-Martin Rousseau. Improving column complementarity in a restricted master heuristic with a grasp-guided completion: Application to the vehicle routing problem with stochastic demands. *Computers & Operations Research*, page 107569, 2026b.
- Louis-Martin Rousseau, Michel Gendreau, and Dominique Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668, 2007.
- 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.
- Lara Scavuzzo, Karen Aardal, Andrea Lodi, and Neil Yorke-Smith. Machine learning augmented branch and bound for mixed integer linear programming. *Mathematical Programming*, pages 1–44, 2024.
- Yunzhuang Shen, Yuan Sun, Xiaodong Li, Zhiguang Cao, Andrew Eberhard, and Guangquan Zhang. Adaptive stabilization based on machine learning for column generation. In *41st International Conference on Machine Learning (ICML 2024)*, pages 44741–44758, 2024.
- Nagisa Sugishita, Andreas Grothey, and Ken McKinnon. Use of machine learning models to warmstart column generation for unit commitment. *INFORMS Journal on Computing*, 36(4):1129–1146, 2024.
- Frank A Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3(3):192–204, 1969.
- José M Valério de Carvalho. Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, 17(2):175–182, 2005.
- Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.
- Ziming Wang, Jun Chen, and Haopeng Chen. EGAT: Edge-featured graph attention network. In *International Conference on Artificial Neural Networks*, pages 253–264, 2021.
- Paul Wentges. Weighted Dantzig-Wolfe decomposition for linear mixed-integer programming. *International Transactions in Operational Research*, 4(2):151–162, 1997.
- Julian Yarkony, Yossiri Adulyasak, Maneesh Singh, and Guy Desaulniers. Data association via set packing for computer vision applications. *INFORMS Journal on Optimization*, 2(3):167–191, 2020.