Scenario Consensus Algorithms for Solving Stochastic and Dynamic Problems

Felipe Lagos

Faculty of Engineering and Sciences

Universidad Adolfo Ibáñez

Santiago, Chile 7941169

felipe.lagos@uai.cl

August, 2021

Abstract

In transportation problems and in many other planning problems, there are important sources of uncertainty that must be addressed to find effective and efficient solutions. A common approach for solving these dynamic and stochastic problems is the Multiple Scenario Approach (MSA), that has been proved effective for transportation problems, but it does not provide flexibility for finding solutions that take into account all the uncertainty of the problem. Alternative approaches for solving problems with finite number of scenarios are the Progressive Hedging Algorithm (PHA) and the Subgradient Algorithm (SA). The similarity between PHA and SA are many, however, there are some differences that lead them to have very different theoretical guarantees and performance. We present a new exact algorithm, the Dynamic Progressive Hedging Algorithm, for which we provide theoretical guarantees that helps to understand both this algorithm and the PHA from a new point of view. In addition, we propose a DPHA-based Heuristic (DPHH), and show optimality guarantees for the solution obtained. Then, we present the MSA and the SA with consensus functions. Our analysis allows us to highlight the advantages and disadvantages of the DPHA, the SA and the MSA, which gives guidance for future research in choosing the proper method for the problem in hand. In a computational study, we study the Stochastic Server Location Problem (SSLP) and the Two-Stage Stochastic Assignment and Team-Orienteering Problem (TSSATOP), and we show the empirical performance of the proposed Scenario Consensus Algorithms (SCA).

Keywords: mixed-integer stochastic programming, progressive hedging, lagrangian duality, consensus functions

1 Introduction

In transportation problems and in many other planning problems, there are important sources of uncertainty that must be addressed to find effective and efficient solutions. In recent years, services for which information is only partially known have grown in importance, such as ride-sharing, food delivery services or e-commerce, where it is expected that the service does not fail and that the delivery is fulfilled in a very short time. For all these problems, there are multiple sources of uncertainty. Some examples are: travel times between two points in a route; delivery service times; customer demands, both in terms of volume and quantity of the delivery; and, the requirements for delivery time, which may even take place during the same day that is being planned.

In transportation and logistics, a common approach for solving these dynamic and stochastic problems is the Multiple Scenario Approach (MSA). This approach, proposed by Bent and Van Hentenryck [5], consists of generating scenarios that represent future states of the current decision time, finding feasible solutions for each of these scenarios and then determing a single plan to be executed by the planner.

One of the main problems with this approach, that can be seen as a solving approach for Two-Stage Stochastic Mixed-Integer Programs (TSMIP), is that there is no room for the different scenarios to coordinate after a solution is found. Each scenario is solved once and then a consensus plan is determined. If this consensus plan is not good enough for some (or all) scenarios, the MSA does not provide flexibility for finding new solutions that can be better considering all scenarios. This motivates the study of other approaches to better coordinate scenario solutions, which leads to Stochastic Programming [6].

Stochastic problems, in which random factors are approximated using a set of future scenarios, have been extensively studied. Since the size and complexity of the problems grows with the number of scenarios, decomposition methods are used. In these approaches, like the MSA, the scenario problems are solved independently. Two of the most well-known approaches are the Progressive Hedging Algorithm (PHA) and dual decomposition or subgradient methods.

1.1 Preliminaries

We consider two-stage stochastic programs of the following form,

$$\begin{aligned} \min \quad & \mathbb{E}_{\tilde{\xi}}(F(x,\tilde{\xi})), && \text{(1a)} \\ \text{s.t.} \quad & x \in \mathcal{X} \subseteq \mathbb{Z}^{N-P} \times \mathbb{R}^{P}, && \text{(1b)} \end{aligned}$$

where $\tilde{\xi}$ is a random vector defined on a probability space $(\Xi, \mathscr{A}, \mathscr{P})$. The objective function is to minimize the expected cost of the second stage problem $F(x,\tilde{\xi})$, which for a particular realization ξ of $\tilde{\xi}$, we consider it as follows,

$$\begin{aligned} F(x,\xi) \coloneqq \min \quad & c(\xi)^\intercal y, && (2a)\\ \text{s.t.} \quad & W(\xi)y \geq h(\xi) - T(\xi)x, && (2b)\\ & y \in \mathbb{Z}^{M-Q} \times \mathbb{R}^{Q}. && (2c) \end{aligned}$$

The matrices $W(\xi)$ and $T(\xi)$, and the vectors $c(\xi)$ and $h(\xi)$, depend on the realization of $\tilde{\xi}$ and have rational components. In the first stage, the decisions x are made before uncertainty is revealed and, in the second stage, recourse actions are taken. The objective in the second stage is given by $c(\xi)$, so it is linear on y and the decisions are restricted by constraints (2b). The first-stage decision variables x constrain y variables through the matrix $T(\xi)$.

The two-stage problem is usually simplified by assuming that the random variable $\tilde{\xi}$ has only finite $K \in \mathbb{Z}_+$ many realizations ξ. The support of $\tilde{\xi}$ is $\Xi = \{\xi_k\}_{k\in[K]}$, with $\mathbb{P}(\tilde{\xi} = \xi_k) = p_k$ and p_k positive probabilities, for all $k \in [K] \coloneqq \{1,\dots,K\}$. In this setting, we redefine the parameters $T(\xi_k) \coloneqq T_k$, $W(\xi_k) \coloneqq W_k$ and $h(\xi_k) \coloneqq h_k$, to simplify notation. Also, note that the expected value can be rewritten as $\sum_{k\in[K]} p_k F(x,\xi_k) = \sum_{k\in[K]} p_k F_k(x)$, with $F(x,\xi_k) \coloneqq F_k(x)$.

A common approach for solving (1) considers reformulating the problem by creating a copy of x for each scenario, x_k, $k \in [K]$. Then,

$$\begin{aligned} \min \quad & \sum_{k\in[K]} p_k F_k(x_k), && && (3a)\\ \text{s.t.} \quad & x_k = x_{k+1} && k \in [K-1], && (3b)\\ & x_K = x_1 && && (3c)\\ & x_k \in \mathcal{X}, && k \in [K]. && (3d) \end{aligned}$$

Constraints (3b) and (3c) are known as non-anticipativity or implementability constraints and they impose that first stage variables are independent of the realized scenario. An easier way of handling these constraints is aggregating them using the scenarios probabilities. The following problem is equivalent to (3),

$$\begin{aligned} \min \quad & \sum_{k\in[K]} p_k F_k(x_k), && && (4a)\\ \text{s.t.} \quad & x_k = \sum_{i\in[K]} p_i x_i, && k \in [K], && (4b)\\ & x_k \in \mathcal{X}, && k \in [K]. && (4c) \end{aligned}$$

Solving the scenario formulation (4) is difficult for most practical problems, because of the large number of scenarios and the mixed-integer nature of the problem. This motivates to dualize the non-anticipativity constraints (4b), and exploit the scenario decomposition structure. The resulting formulation has a well-known block diagonal structure, which permits to manage the problem complexity. Multiplying constraints (4b) by p_k and using the

Lagrange duals $\lambda_k \in \mathbb{R}^N$ for each constraint $k \in [K]$, we have the following Lagrange function L,

$$\begin{aligned} L(X,\Lambda) &:= \sum_{k\in[K]} p_k F_k(x_k) + \sum_{k\in[K]} p_k \lambda_k^\intercal \left(x_k - \sum_{i\in[K]} p_i x_i \right), \\ &= \sum_{k\in[K]} p_k F_k(x_k) + \sum_{k\in[K]} \left(p_k \lambda_k^\intercal - \sum_{i\in[K]} p_i p_k \lambda_i^\intercal \right) x_k, \\ &= \sum_{k\in[K]} p_k F_k(x_k) + \sum_{k\in[K]} p_k \left(\lambda_k - \sum_{i\in[K]} p_i \lambda_i \right)^\intercal x_k, \end{aligned}$$

with $\Lambda = \begin{pmatrix} \lambda_1 \\ \dots \\ \lambda_K \end{pmatrix} \in \mathbb{R}^{KN}$ and $X = \begin{pmatrix} x_1 \\ \dots \\ x_K \end{pmatrix} \in \mathbb{R}^{KN}$.

Note that shifting the multipliers λ_k, $k \in [K]$, by a constant vector does not change the value of the term $\lambda_k - \sum_{i\in[K]} p_i\lambda_i$, so we can assume w.l.o.g. that these multipliers hold,

$$\sum_{i\in[K]} p_i \lambda_i = 0, \tag{5}$$

and, consequently, we can formulate the dual problem as,

$$\begin{aligned} \max \quad & \underline{L}(\Lambda), && \text{(6a)} \\ \text{s.t.} \quad & \Lambda \in \mathcal{D}, && \text{(6b)} \end{aligned}$$

where,

$$\mathcal{D} := \left\{ \Lambda = \begin{pmatrix} \lambda_1 \\ \dots \\ \lambda_K \end{pmatrix} \middle| \begin{array}{ll} \sum_{k\in[K]} p_k \lambda_k = 0, & \\ \lambda_k \in \mathbb{R}^N, & k \in [K]. \end{array} \right\},$$

with the corresponding Lagrangian relaxation $\underline{L}$ given by,

$$\underline{L}(\Lambda) := \min_{x_k\in\mathcal{X}, k\in[K]} \sum_{k\in[K]} p_k \left(F_k(x_k) + \lambda_k^\intercal x_k \right) = \sum_{k\in[K]} p_k \underline{L}_k(\lambda_k), \tag{7}$$

and $\underline{L}_k(\lambda_k) := \min_{x_k\in\mathcal{X}} F_k(x_k) + \lambda_k^\intercal x_k$.

The Subgradient Algorithm (SA) (also known as the dual decomposition or subgradient method) solves the problem (7) using the subgradient of the function. Starting with a solution $\Lambda^0 \in \mathcal{D}$, the optimal value of $\underline{L}$ is successively approximated by using the solution of the scenario subproblems. Let $x_k \in \mathcal{X}$ be an optimal solution to $\underline{L}_k(\lambda_k)$, for some λ_k. The dual updating is given by,

$$\lambda_k' = \lambda_k + \alpha(x_k - \hat{x}), \tag{8}$$

with $\hat{x} = \sum_{k\in[K]} p_k x_k$. The weights α must define a sequence of decreasing values along the iterations. This algorithm guarantees convergence of the Lagrangian function to the optimal value but not to a primal solution.

The PHA also updates the dual multipliers using the equation (8), but in this case, the α value must be constant along iterations. The PHA, unlike the SA, solves a non-linear problem since a quadratic term is added to the subproblem objective function. In addition, it requires the subproblem to be solved by considering a convex set for the variables (so the variables nature cannot be integer), but it guarantees a dual and primal solution will be found. Finally, as the SA, the value of the objective function converges to the Lagrangian $\underline{L}$ optimal value.

1.2 Research Motivations and Contributions

The similarity between the PHA and SA are many, however, there are some differences that lead them to have very different theoretical guarantees and performance. One of the first motivations is to understand why these differences generate two algorithms with different convergence results: the PHA for the primal and dual solutions, while the SA only for the Lagrangian function.

For this it is important to understand the PHA from a more "OR" point of view. The algorithm was introduced by Rockafellar and Wets [30], where a proof based on monotone operators is given [29]. Since then, similar proofs have been offered. These operators do not correspond to the common approach used in OR to understand algorithm's convergence. Even though these operators have a connection with the subgradient of a function, it is important to establish an explanation that uses a more well-known approach. We present a new exact algorithm, the Dynamic Progressive Hedging Algorithm (DPHA), a generalization of PHA, and we provide a proof that allows to understand this algorithm and the PHA from a different angle. This approach offers a new perspective for extensions of the DPHA.

This algorithm, the DPHA, allows to adjust the value of the weight α, so that it is not a constant value during the execution of the algorithm. In Mulvey and Vladimirou [25] a dynamic strategy is used to solve a stochastic network problem, where it is determined that this strategy is superior to the constant value strategy. The authors present this PHA-based algorithm as a heuristic, in our work we prove this is an exact algorithm.

The PHA is commonly used as a heuristic for stochastic problems with integer variables, where the scenario subproblems are solved on a non-convex set. The theoretical guarantees of the PHA are not valid for this case. We provide optimality guarantees for the implementable solution obtained using PHA-based heuristics. These guarantees allow us not only to give clarity of a widely used algorithm, but also to explain a trade-off often reported in the literature: the faster the convergence speed, the lower the quality of the solution. We also show an efficient method to solve the Two-Stage Stochastic Mixed-Integer with Pure First-Stage Binary Variables Problem (TSMIPB) using the DPHA. We refer to this DPHA-based heuristic as the Dynamic Progressive Hedging Heuristic (DPHH).

Consensus functions and MSA have been extensively studied, especially in the transportation field. Therefore, efficient methods have been proposed to solve the symmetry of the scenario solutions and to establish low-cost plans for first-stage decisions. The Subgradient Algorithm, on the other hand, finds lower bounds to the problem and does not provide feasible solutions. We propose to combine these two approaches together and, at each iteration, find feasible solutions using consensus functions. This allows to find high quality solutions from the scenarios coordination made by the SA with no additional computational cost (it does not require solving any problem). This new approach provides the flexibility for coordinating first-stage solutions that the MSA does not have.

We have called Scenario Consensus Algorithms (CSA) to the algorithms that find an unique plan by solving independent subproblems from scenarios representing future states. In a computational study we show the differences of all the CSA we propose, testing on instances of the Stochastic Server Location Problem (SSLP) and the Two-Stage Stochastic Assignment and Team-Orienteering Problem (TSSATOP). For the SSLP, we solve some instances studied in Ntaimo and Sen [27] and also, some instances from Lim et al. [23]. For the last ones, we provide the best solutions found so far. In the TSSATOP, the symmetry of the integer solution is an important factor to address for the solving approach, so it is a relevant setting in which we can study our algorithms. From the results obtained, we show that the DPHH is the best approach in terms of balancing running time and solution quality. This also shows that the DPHH is a promising approach to solve other stochastic and dynamic vehicle routing problems.

This paper is organized as follows. In Section 2 we present the relevant literature. In Section 3 we present the DPHA, the algorithm's convergence theoretical results, the DPHH and the guarantees of this heuristic. In Section 4 we describe the MSA and some of the consensus functions, while in Section 5 we present the SA with consensus functions. At the end of this section, we make a brief comparison of the SA with the DPHA. Finally, in Section 6 is the computational study and in Section 7 we conclude and discuss future work.

2 Literature Review

The methods discussed in this work have been extensively studied. The PHA, presented in Rockafellar and Wets [30], is one of the most widely used algorithms and currently under very active research. Several improvements have been proposed to the original version, such as: calculation of initial weights based on the instance data [36]; lower bound calculations for measuring solution quality [16], or for obtaining initial solutions for other decomposition methods [18]; integration of the PHA with simplicial decomposition methods [7]; or, integration with a branch-and-bound method for finding provable optimal solutions for non-convex problems [3]). It should be noted that only in Rockafellar and Wets [30] the PHA's theoretical guarantees are provided and that the improvements that have been made subsequently rely on them.

A dynamic strategy for weights $\{\alpha^{\nu}\}_{\nu\geq 0}$ for the PHA has been studied in the literature. In Mulvey and Vladimirou [25] the authors study the Two-Stage Generalized Network Problem, a problem defined on convex sets, which is solved using sequences of weights that gradually increase until converging to a finite value. In Crainic et al. [12] a dynamically adjusted strategy, with a monotonically increasing sequence of weights, is also proposed, although, it does not converge to a finite value. In Escudero et al. [14] a PHA-based heuristic is used where the sequence of weights is updated. This update is not monotone: depending on the subgradient direction the current iteration $\nu \geq 0$ weight α^ν is updated by multiplying by a number greater or less than 1. In all these works, computational experiments support that a dynamic strategy is an effective approach.

One of the most important uses of the PHA is as a heuristic, where the set of feasible subproblem solutions is restricted to non-convex sets. These subproblems are usually solved to optimality, but it is the convergence to an implementable solution that makes this approach a heuristic. In general, these heuristics are considered to be an efficient approach to solving combinatorial stochastic optimization models. They have been studied in different contexts, such as the stochastic network design [10, 11] management of hydroelectric reservoir system [9], forestry production planning [34], fisheries management and application [19], hydrothermal systems operations planning application [13] and for hospital surgery planning [17]. There are also cases in which the PHA subproblem is solved heuristically, as in Løkketangen and Woodruff [24], where the algorithm for the subproblems is the tabu search.

There are few applications of the PHA for Vehicle Routing Problems (VRP). It highlights the work in Hvattum et al. [20], where the Dynamic Stochastic Hedging Heuristic (DSHH) is proposed for solving a dynamic and stochastic VRP. This algorithm is inspired by the PHA, but it has many differences with respect to the PHA-based heuristics found in the literature. In the DSHH, using a finite set of randomly generated scenarios, a subset of the deterministic customers is found. Then, all identified customers from the set are included at some vehicle (one by one) by locking pairs customer-vehicle at the final solution. This procedure, unlike the PHA-based heuristic, finds a consensus in a greedy way. In Hvattum and Løkketangen [21], the Stochastic Inventory Routing Problem is solved using a PHA-based heuristic solving the subproblems with a GRASP and a dynamic adjustment strategy for the weights. Recently, in Jalilvand et al. [22] is proposed a PHA-based heuristic for solving Vehicle Routing Problems with stochastic service times.

The MSA has been used in several transportation and logistics contexts, both for dynamic and stochastic problems, such as the Dynamic VRP with Stochastic Customers [5], the Same-Day Delivery Problem [35], the Stochastic Team Orienteering with Consistency Constraints Problem [33], and recently, the menu design problem for peer-to-peer transportation platforms providers [4]. In these works, different consensus functions are proposed to deal with the nature of solutions that usually arise in the context of vehicle routing. For a review of this and other approaches for dynamic routing problems, see Soeffker et al. [31].

The subgradient or dual decomposition method has been intensively studied. In Carøe and Schultz [8] a decomposition method for multistage problems with mixed-integer variables, integrated with a branch-and-bound strategy to find feasible solutions, is proposed. The method of Aravena and Papavasiliou [1] also develops a Lagrangian relaxation for non-anticipativity constraints with an incremental method to solve the dual problem. They integrate this approach with primal solution recovery methods. They extend this method in Aravena and Papavasiliou [2], exploiting the parallelization of the subproblems and proposing different heuristics to recover a primal solution. It is worth noting that none of these heuristics has any resemblance with the consensus functions in the literature. Finally, in Lim et al. [23], an asynchronous dual decomposition method is studied, in which not all subproblems are solved to perform duals updating.

3 Dynamic Progressive Hedging Algorithm

In this section we present the generalization of the Progressive Hedging Algorithm (PHA), the Dynamic Progressive Hedging Algorithm (DPHA). Then, we introduce the DPHA-based heuristic, the DPHH, and provide theoretical guarantees for the solution found. Finally, we show how to exploit the structure of the problem when the first-stage variables are binary using the DPHH.

3.1 Algorithm

The PHA works on convex sets, so we redefine the previous functions for variables on convex sets. For the reader's sake we keep the same notation but we add a (^{c}) superscript to the function when it is defined for convex sets. Let $\mathrm{conv}(\mathcal{Z})$ be the convex hull of a set $\mathcal{Z}$. For a set $\mathcal{Y} \subseteq \mathbb{Z}^{M-Q} \times \mathbb{R}^Q$, a scenario $k \in [K]$ and $x \in \mathrm{conv}(\mathcal{X})$, we define

$$\begin{aligned} F_k^c(x) := \min \quad & c_k^\intercal y, && \text{(9a)}\\ \text{s.t.} \quad & W_k y \geq h_k - T_k x, && \text{(9b)}\\ & y \in \mathrm{conv}(\mathcal{Y}), && \text{(9c)} \end{aligned}$$

which corresponds to the formulation (2) with variables y defined on the convex hull of $\mathcal{Y}$. We have the primal problem,

$$\begin{aligned} \min \quad & \sum_{k\in[K]} p_k F_k^c(x_k), && \text{(10a)}\\ \text{s.t.} \quad & x_k = \sum_{i\in[K]} p_i x_i, \quad k \in [K], && \text{(10b)}\\ & x_k \in \mathrm{conv}(\mathcal{X}), \quad k \in [K], && \text{(10c)} \end{aligned}$$

and the dual problem,

$$\begin{aligned} \max \quad & \underline{L}^c(\Lambda), && \text{(11a)}\\ \text{s.t.} \quad & \Lambda \in \mathcal{D}, && \text{(11b)} \end{aligned}$$

with $\underline{L}^c(\Lambda) := \min_{x_k\in\mathrm{conv}(\mathcal{X})} \sum_{k\in[K]} p_k \left(F_k^c(x_k) + \lambda_k^\intercal x_k\right)$. Similarly, we define $L^c(X, \Lambda) := \sum_{k\in[K]} p_k \left(F^c(x_k) + \lambda_k^\intercal x_k\right)$.

For practical proposes, it is usually assumed that there exists a solution $X^* \in \mathrm{conv}(\mathcal{X})^K$, that is implementable and feasible for all second stage scenarios $k \in [K]$, and also that there exists a $\Lambda^* \in \mathcal{D}$ dual optimal solution.

Assumption 1. *There exist* $X^* \in \text{conv}(\mathcal{X})^K$ *and* $\Lambda^* \in \mathcal{D}$ *optimal solutions for problems* (10) *and* (11)*, respectively.*

For a given $\Lambda \in \mathcal{D}$ vector, the problem $\min_{X \in \text{conv}(\mathcal{X})^K} L^c(X, \Lambda)$ can be solved independently for each scenario. In the PHA, this decomposable structure is maintained, but the subproblem considers has an extra term that penalizes deviations of the scenario $k \in [K]$ solution vector x_k with respect to an estimate $\hat{x}$ (an implementable solution as reference). For a given $\alpha > 0$, the problem that is solved by the PHA, for each $k \in [K]$, is the following,

$$\min_{x_k \in \text{conv}(\mathcal{X})} F_k^c(x_k) + \lambda_k^\intercal x_k + \frac{\alpha}{2}\|x_k - \hat{x}\|^2. \tag{12}$$

The addition of this penalty term is motivated by the augmented Lagrangian approach that, in contrast to ordinary Lagrangian function, is not limited in its numerical usefulness [30]. Note that in this approach, unlike the augmented Lagrangian method, the problem is decomposable by scenario, so each scenario can be solved independently.

The Dynamic Progressive Hedging Algorithm (DPHA) is similar to the PHA, but the parameter α can change from one iteration to another. Let $\{\alpha^\nu\}_{\nu=0}^\infty$ be a non-decreasing sequence ($\alpha^{\nu+1} \geq \alpha^\nu$), with positive ($\alpha^\nu > 0$) and bounded ($\alpha^\nu < \infty$) values, for all $\nu \geq 0$. Note that the definition of α^ν generalizes the case with $\alpha^\nu = \alpha^{\nu+1}$, $\nu \geq 0$, of the PHA. At iteration $\nu \geq 0$, let λ_k^ν be the dual vector for scenario $k \in [K]$ and let $x_k^{\nu+1}$ be an optimal solution to (12) with dual Λ^ν and weight α^ν. The DPHA is detailed in Algorithm 1.

```
1  find a $x_k^0 \in \text{argmin}_{x_k \in \text{conv}(\mathcal{X})} F_k^c(x_k)$, for all $k \in [K]$;
2  let $\lambda_k^0 = 0$ and $\hat{x}^0 = \sum_{k \in [K]} p_k x_k^0$, for all $k \in [K]$;
3  set $\nu = 0$;
4  for each $k \in [K]$ get $x_k^{\nu+1}$ by solving,

       $x_k^{\nu+1} \in \underset{x_k \in \text{conv}(\mathcal{X})}{\text{argmin}} F_k^c(x_k) + \lambda_k^{\nu\intercal} x_k + \frac{\alpha^\nu}{2}\left\|x_k - \hat{x}^\nu\right\|^2;$

5  compute $\hat{x}^{\nu+1} = \sum_{k \in [K]} p_k x_k^{\nu+1}$;
6  if $x_k^{\nu+1} = \hat{x}^{\nu+1}$ for all $k \in [K]$ then
7      stop;
8  else
9      get $\lambda_k^{\nu+1} = \lambda_k^\nu + \alpha^\nu(x_k^{\nu+1} - \hat{x}^{\nu+1})$;
10     $\nu \leftarrow \nu + 1$ and go to line 4;
11 end
```

Algorithm 1: Dynamic Progressive Hedging Algorithm.

The DPHA starts with a solution x_k^0 for each scenario $k \in [K]$, that minimizes the function $F_k^c(x_k)$ with $x_k \in \text{conv}(\mathcal{X})$. Then we compute an estimate of the first stage vector $\hat{x}^0$ as the weighted average of x_k^0 solutions. The initial multiplier λ_k^0 is set to the vector zero, for all scenario $k \in [K]$. At each iteration $\nu \geq 0$, we solve the problem (12) using the λ_k^ν multiplier, the weight α^ν and the estimate $\hat{x}^\nu$ vector for x_k. We get an optimal solution $x_k^{\nu+1}$. If for all $k \in [K]$ it holds $x_k^{\nu+1} = \hat{x}^{\nu+1}$, then we stop, the solution is implementable. Otherwise, we update the dual vectors $\lambda_k^{\nu+1} = \lambda_k^\nu + \alpha^\nu(x_k^{\nu+1} - \hat{x}^{\nu+1})$ and we continue with a new iteration.

We show that the DPHA algorithm converges to an optimal solution. The proof is based on Rockafellar and Wets [30] Theorem 5.2's proof. In our case, we do not use maximal monotone operators, but subgradients of the functions. In order to make the proof easier to follow we have split the proof in different propositions, explaining the connection between them explicitly.

We start by defining the following function for $(\bar{x}, \Lambda) \in \operatorname{conv}(\mathcal{X}) \times \mathcal{D}$,

$$G(\bar{x}, \Lambda) \coloneqq \min \left\{ \sum_{k\in[K]} p_k \left(F_k^c(x_k) + \lambda_k^\intercal x_k \right) \middle| X \in \mathcal{S}(\bar{x}) \right\}, \tag{13}$$

with $\mathcal{S}(\bar{x}) \coloneqq \left\{ X \in \operatorname{conv}(\mathcal{X})^K \middle| \sum_{k\in[K]} p_k x_k = \bar{x} \right\}$.

We have the following two properties of function G, whose proofs are in Appendix A.1.

Proposition 1. *If a solution $(x^*, \Lambda^*) \in \operatorname{conv}(\mathcal{X}) \times \mathcal{D}$ is optimal for G, then $X^* = (x_1, x_2, \dots x_K)$, with $x_k = x^*$ for all $k \in [K]$, and Λ^* are optimal for L^c.*

Proposition 2. *The function $G(\bar{x}, \Lambda)$ is convex in $\bar{x} \in \operatorname{conv}(\mathcal{X})$ and concave in $\Lambda \in \mathcal{D}$.*

We now show that the solution $(X^{\nu+1}, \Lambda^{\nu+1})$, obtained at iteration $\nu \geq 0$, is a saddle point for the following augmented Lagrangian function L_+^ν, (with constant vectors $\hat{x}^\nu$ and λ^ν), for $X \in \operatorname{conv}(\mathcal{X})$ and $\Lambda \in \mathcal{D}$,

$$L_+^\nu(X, \Lambda) \coloneqq \sum_{k\in[K]} p_k \left(F_k^c(x_k) + \lambda_k^\intercal x_k + \frac{\alpha^\nu}{2} \|\hat{x}^\nu - \hat{x}\|^2 - \frac{1}{2\alpha^\nu} \|\lambda_k - \lambda_k^\nu\|^2 \right), \tag{14}$$

with $\hat{x} = \sum_{k\in[K]} p_k x_k$.

Proposition 3. *At iteration $\nu \geq 0$, the solution $(X^{\nu+1}, \Lambda^{\nu+1})$ obtained by Algorithm 1 is a saddle point of the function L_+^ν.*

Proof. We have the following equation,

$$\begin{aligned}
\sum_{k\in K} p_k \|x_k^{\nu+1} - \hat{x}^{\nu+1}\|^2 &= \sum_{k\in K} p_k \left(\|x_k^{\nu+1}\|^2 - 2\hat{x}^{\nu+1\intercal} x_k^{\nu+1} + \|\hat{x}^{\nu+1}\|^2 \right), \\
&= \sum_{k\in K} p_k \left(\|x_k^{\nu+1}\|^2 - \|\hat{x}^{\nu+1}\|^2 \right), \\
&= \sum_{k\in K} p_k \left(\|x_k^{\nu+1}\|^2 - 2x_k^{\nu+1\intercal} \hat{x}^\nu + \|\hat{x}^\nu\|^2 - \|\hat{x}^\nu\|^2 + 2\hat{x}^{\nu+1\intercal} \hat{x}^\nu - \|\hat{x}^{\nu+1}\|^2 \right), \\
&= \sum_{k\in K} p_k \left(\|x_k^{\nu+1} - \hat{x}^\nu\|^2 - \|\hat{x}^\nu - \hat{x}^{\nu+1}\|^2 \right).
\end{aligned}$$

In addition, it holds,

$$\begin{aligned}
\sum_{k\in[K]} p_k\lambda_k^{\nu\intercal}x_k^{\nu+1} &= \sum_{k\in[K]} p_k\lambda^{\nu\intercal}(x_k^{\nu+1}-\hat{x}^{\nu+1}),\\
&= \sum_{k\in[K]} p_k\left(\lambda_k^{\nu+1}-\alpha^\nu(x_k^{\nu+1}-\hat{x}^{\nu+1})\right)^\intercal(x_k^{\nu+1}-\hat{x}^{\nu+1}),\\
&= \sum_{k\in[K]} p_k\lambda_k^{\nu+1\intercal}(x_k^{\nu+1}-\hat{x}^{\nu+1}) - p_k\alpha^\nu\|x_k^{\nu+1}-\hat{x}^{\nu+1}\|^2,\\
&= \sum_{k\in[K]} p_k\lambda_k^{\nu+1\intercal}x_k^{\nu+1} - p_k\alpha^\nu\|x_k^{\nu+1}-\hat{x}^{\nu+1}\|^2.
\end{aligned}$$

Using the previous equations, we have,

$$\begin{aligned}
\min_{x_k\in\operatorname{conv}(\mathcal{X})}\sum_{k\in[K]} p_k\left(F_k^c(x_k)+\lambda_k^\intercal x_k+\frac{\alpha^\nu}{2}\|x_k-\hat{x}^\nu\|^2\right) &= \sum_{k\in[K]} p_k\left(F_k^c(x_k^{\nu+1})+\lambda_k^{\nu\intercal}x_k^{\nu+1}+\frac{\alpha^\nu}{2}\|x_k^{\nu+1}-\hat{x}^\nu\|^2\right),\\
&= \sum_{k\in[K]} p_k\left(F_k^c(x_k^{\nu+1})+\lambda_k^{\nu\intercal}x_k^{\nu+1}+\frac{\alpha^\nu}{2}(\|\hat{x}^\nu-\hat{x}^{\nu+1}\|^2+\|x_k^{\nu+1}-\hat{x}^{\nu+1}\|^2)\right),\\
&= \sum_{k\in[K]} p_k\left(F_k^c(x_k^{\nu+1})+\lambda_k^{\nu+1\intercal}x_k^{\nu+1}+\frac{\alpha^\nu}{2}(\|\hat{x}^\nu-\hat{x}^{\nu+1}\|^2-\|x_k^{\nu+1}-\hat{x}^{\nu+1}\|^2)\right),\\
&= \sum_{k\in[K]} p_k\left(F_k^c(x_k^{\nu+1})+\lambda_k^{\nu+1\intercal}x_k^{\nu+1}+\frac{\alpha^\nu}{2}\|\hat{x}^\nu-\hat{x}^{\nu+1}\|^2-\frac{1}{2\alpha^\nu}\|\lambda_k^{\nu+1}-\lambda_k^\nu\|^2\right),
\end{aligned}$$

and thus, $X^{\nu+1}$ is optimal for $L_+^\nu(X,\Lambda^{\nu+1})$.

For $\Lambda\in\mathcal{D}$, we have that the function $L_+^\nu(X^{\nu+1},\Lambda)$ can be also written as,

$$L_+^\nu(X^{\nu+1},\Lambda) = \sum_{k\in[K]} p_k\left(F_k^c(x_k^{\nu+1})+\lambda_k^\intercal(x_k^{\nu+1}-\hat{x}^{\nu+1})+\frac{\alpha^\nu}{2}\|\hat{x}^\nu-\hat{x}^{\nu+1}\|^2-\frac{1}{2\alpha^\nu}\|\lambda_k-\lambda_k^\nu\|^2\right),$$

which is concave on Λ. Taking the derivative of the function with respect to λ_k, $k\in[k]$, and finding the λ_k^* that makes the equation equal to zero, we have,

$$\begin{aligned}
(x_k^{\nu+1}-\hat{x}^{\nu+1}) &= \frac{1}{\alpha^\nu}(\lambda_k^*-\lambda_k^\nu),\\
\lambda_k^* &= \lambda_k^\nu+\alpha^\nu(x_k^{\nu+1}-\hat{x}^{\nu+1}) = \lambda_k^{\nu+1},
\end{aligned}$$

so the $\Lambda^{\nu+1}$ computed by the algorithm is an optimal solution. □

The previous proposition allows us to get one of the most important results of the DPHA: at every iteration the progress made by the DPHA is guaranteed (i.e., the primal or dual or both solutions are "closer" to an optimal solution). For this result, the following norm definition is useful.

For any $\Lambda, \Lambda' \in \mathcal{D}$, we define,

$$\|\Lambda - \Lambda'\|_p^2 \coloneqq \sum_{k\in[K]} p_k \|\lambda_k - \lambda'_k\|^2. \tag{15}$$

Proposition 4. *Consider an optimal solution $(x^*, \Lambda^*) \in \operatorname{conv}(\mathcal{X}) \times \mathcal{D}$ for G. At every iteration $\nu \geq 0$ of the Algorithm 1, we have that,*

$$\begin{aligned} \|\hat{x}^{\nu+1} - x^*\|^2 + \frac{1}{\alpha^{\nu 2}}\|\Lambda^{\nu+1} - \Lambda^*\|_p^2 + \|\hat{x}^{\nu+1} - \hat{x}^{\nu}\|^2 + \frac{1}{\alpha^{\nu 2}}\|\Lambda^{\nu+1} - \Lambda^{\nu}\|_p^2 \\ \leq \|\hat{x}^{\nu} - x^*\|^2 + \frac{1}{(\alpha^{\nu-1})^2}\|\Lambda^{\nu} - \Lambda^*\|_p^2. \end{aligned} \tag{16}$$

Proof. For an optimal primal and dual solution (x^*, Λ^*), we have that for any $(\hat{x}^{\nu+1}, \Lambda^{\nu+1}) \in \operatorname{conv}(\mathcal{X}) \times \mathcal{D}$, it holds,

$$\begin{aligned} G(\hat{x}^{\nu+1}, \Lambda^*) \geq G(x^*, \Lambda^*), \\ G(x^*, \Lambda^{\nu+1}) \leq G(x^*, \Lambda^*), \end{aligned}$$

so, $G(x^*, \Lambda^{\nu+1}) - G(\hat{x}^{\nu+1}, \Lambda^*) \leq 0$.

Now, note that the function L_+^{ν} evaluated at $(X^{\nu+1}, \Lambda^{\nu+1})$, satisfies,

$$\begin{aligned} L_+^{\nu}(X^{\nu+1}, \Lambda^{\nu+1}) &= \sum_{k\in[K]} p_k \left(F_k^c(x_k^{\nu+1}) + \lambda_k^{\nu+1,\intercal} x_k^{\nu+1} + \frac{\alpha^{\nu}}{2}\|\hat{x}^{\nu} - \hat{x}^{\nu+1}\|^2 - \frac{1}{2\alpha^{\nu}}\|\lambda_k^{\nu+1} - \lambda_k^{\nu}\|^2 \right), \\ &= \min\left\{ \sum_{k\in[K]} p_k \left(F_k^c(x_k) + \lambda_k^{\nu+1\intercal} x_k \right) \middle| \, x_k \in \operatorname{conv}(\mathcal{X}), \sum_{k\in[K]} p_k x_k = \hat{x}^{\nu+1} \right\} \\ &\quad + \sum_{k\in[K]} p_k \left(\frac{\alpha^{\nu}}{2}\|\hat{x}^{\nu} - \hat{x}^{\nu+1}\|^2 - \frac{1}{2\alpha^{\nu}}\|\lambda_k^{\nu+1} - \lambda_k^{\nu}\|^2 \right), \\ &= G(\hat{x}^{\nu+1}, \Lambda^{\nu+1}) + \frac{\alpha^{\nu}}{2}\|\hat{x}^{\nu} - \hat{x}^{\nu+1}\|^2 - \frac{1}{2\alpha^{\nu}}\|\Lambda^{\nu+1} - \Lambda^{\nu}\|_p^2, \end{aligned}$$

and since $(X^{\nu+1}, \Lambda^{\nu+1})$ is a saddle point, we have $(0, 0) \in \partial L_+^{\nu}(X^{\nu+1}, \Lambda^{\nu+1})$, which, by subgradient calculus is equivalent to,

$$\alpha^{\nu}(\hat{x}^{\nu} - \hat{x}^{\nu+1}) \in \partial_{\bar{x}} G(\hat{x}^{\nu+1}, \Lambda^{\nu+1}), \tag{17}$$

$$\frac{1}{\alpha^{\nu}} p_k(\lambda_k^{\nu+1} - \lambda_k^{\nu}) \in \partial_{\lambda_k} G(\hat{x}^{\nu+1}, \Lambda^{\nu+1}), \qquad k \in [K]. \tag{18}$$

The function $G(\bar{x}, \Lambda)$ is convex in $\bar{x}$ and concave on Λ, so, by subgradient definition (and notation abuse), we have,

$$\begin{aligned} \partial_{\bar{x}} G(\hat{x}^{\nu+1}, \Lambda^{\nu+1})^\intercal (x^* - \hat{x}^{\nu+1}) + G(\hat{x}^{\nu+1}, \Lambda^{\nu+1}) \leq G(x^*, \Lambda^{\nu+1}), \\ \partial_{\Lambda} G(\hat{x}^{\nu+1}, \Lambda^{\nu+1})^\intercal (\Lambda^* - \Lambda^{\nu+1}) + G(\hat{x}^{\nu+1}, \Lambda^{\nu+1}) \geq G(\hat{x}^{\nu+1}, \Lambda^*). \end{aligned}$$

Combining the above,

$$\partial_{\bar{x}} G(\hat{x}^{\nu+1}, \Lambda^{\nu+1})^\intercal (x^* - \hat{x}^{\nu+1}) + G(\hat{x}^{\nu+1}, \Lambda^*) - \partial_\Lambda G(\hat{x}^{\nu+1}, \Lambda^{\nu+1})^\intercal (\Lambda^* - \Lambda^{\nu+1}) \le G(x^*, \Lambda^{\nu+1}),$$

and so,

$$\alpha^\nu (\hat{x}^\nu - \hat{x}^{\nu+1})^\intercal (x^* - \hat{x}^{\nu+1}) - \frac{1}{\alpha^\nu} \sum_{k\in[K]} p_k (\lambda_k^{\nu+1} - \lambda_k^\nu)^\intercal (\lambda_k^* - \lambda_k^{\nu+1}) \le G(x^*, \Lambda^{\nu+1}) - G(\hat{x}^{\nu+1}, \Lambda^*) \le 0,$$
$$-(\hat{x}^\nu - \hat{x}^{\nu+1})^\intercal (x^* - \hat{x}^{\nu+1}) + \frac{1}{\alpha^{\nu 2}} \sum_{k\in[K]} p_k (\lambda_k^\nu - \lambda_k^{\nu+1})^\intercal (\lambda_k^{\nu+1} - \lambda_k^*) \ge 0.$$

Finally, note that,

$$\begin{aligned} \|\Lambda^\nu - \Lambda^*\|_p^2 &= \sum_{k\in[K]} p_k \|(\lambda_k^\nu - \lambda_k^{\nu+1}) + (\lambda_k^{\nu+1} - \lambda_k^*)\|^2, \\ &= \sum_{k\in[K]} p_k \left(\|\lambda_k^\nu - \lambda_k^{\nu+1}\|^2 + 2(\lambda_k^\nu - \lambda_k^{\nu+1})^\intercal (\lambda_k^{\nu+1} - \lambda_k^*) + \|\lambda_k^{\nu+1} - \lambda_k^*\|^2 \right), \end{aligned}$$

and also,

$$\begin{aligned} \|\hat{x}^\nu - x^*\|^2 &= \|(\hat{x}^\nu - \hat{x}^{\nu+1}) - (x^* - \hat{x}^{\nu+1})\|^2, \\ &= \|\hat{x}^\nu - \hat{x}^{\nu+1}\|^2 - 2(\hat{x}^\nu - \hat{x}^{\nu+1})^\intercal (x^* - \hat{x}^{\nu+1}) + \|x^* - \hat{x}^{\nu+1}\|^2, \end{aligned}$$

which leads to,

$$\begin{aligned} \|\hat{x}^{\nu+1} - x^*\|^2 + \frac{1}{\alpha^{\nu 2}} \|\Lambda^{\nu+1} - \Lambda^*\|_p^2 + \|\hat{x}^{\nu+1} - x^\nu\|^2 + \frac{1}{\alpha^{\nu 2}} \|\Lambda^{\nu+1} - \Lambda^\nu\|_p^2, & \\ \le \|\hat{x}^\nu - x^*\|^2 + \frac{1}{\alpha^{\nu 2}} \|\Lambda^\nu - \Lambda^*\|_p^2, & \\ \le \|\hat{x}^\nu - x^*\|^2 + \frac{1}{(\alpha^{\nu-1})^2} \|\Lambda^\nu - \Lambda^*\|_p^2. & \end{aligned}$$

The last inequality comes from the fact that the sequence $\{\alpha^\nu\}_{\nu\ge0}$ is non-decreasing. $\square$

Let $z^{\nu+1} = \|\hat{x}^{\nu+1} - x^*\|^2 + \frac{1}{\alpha^{\nu 2}} \|\Lambda^{\nu+1} - \Lambda^*\|_p^2$, for all $\nu \ge 0$. In the following Theorem, we show that the sequence $\{z^\nu\}_{\nu\ge0}$ converges to zero so both $\hat{x}^\nu$ and Λ^ν must converge to an optimal solution.

Theorem 1. *Let x^* and Λ^* be optimal solutions for problems* (10) *and* (11)*, respectively. The sequences $\{\hat{x}^\nu\}_{\nu\ge0}$ and $\{\Lambda^\nu\}_{\nu\ge0}$ of primal and dual solutions generated by Algorithm 1 converge to (x^*, Λ^*).*

Proof. Proposition 4 implies that $z^{\nu+1} \leq z^{\nu}$, with strict inequality when the algorithm has not converged yet. Noticing that $z^{\nu+1} \geq 0$ for all $\nu \geq 0$, then there must exist a $\mu \geq 0$ such that $\lim_{\nu\to\infty} z^{\nu} = \mu$, and

$$\lim_{\nu\to\infty}\left(z^{\nu+1} + \|\hat{x}^{\nu+1} - \hat{x}^{\nu}\|^2 + \frac{1}{\alpha^{\nu 2}}\|\Lambda^{\nu+1} - \Lambda^{\nu}\|_p^2\right) \leq \lim_{\nu\to\infty} z^{\nu},$$

then,

$$\lim_{\nu\to\infty}\left(\|\hat{x}^{\nu+1} - \hat{x}^{\nu}\|^2 + \frac{1}{\alpha^{\nu 2}}\|\Lambda^{\nu+1} - \Lambda^{\nu}\|_p^2\right) = 0,$$

thus, there exists a $\tilde{x} = \lim_{\nu\to\infty} \hat{x}^{\nu}$ and a $\tilde{\Lambda} = \lim_{\nu\to\infty} \Lambda^{\nu}$ (α^{ν} is bounded). This also means that $0 \in \partial_{\tilde{x}} G(\tilde{x}, \tilde{\Lambda})$ and $0 \in \partial_{\Lambda} G(\tilde{x}, \tilde{\Lambda})$ (see equations (17) and (18)) so $(\tilde{x}, \tilde{\Lambda})$ is optimal for G. By Proposition 1, the point $(\tilde{x}, \tilde{\Lambda})$ is optimal for L^c. $\square$

The parameters α^{ν}, $\nu \geq 0$, control the speed at which primal and dual solutions are updated in the algorithm: a sequence $\{\alpha^{\nu}\}_{\nu\geq 0}$ with large values leads to a faster primal solution convergence, while a sequence with small values, leads to a faster dual convergence. This observation is more clear when the augmented Lagrangian function (14) is analyzed.

The DPHA provides a flexible framework in which the $\{\alpha^{\nu}\}_{\nu\geq 0}$, sequence can be adapted so controlling the speed at which solutions converge. Through a computational study, Mulvey and Vladimirou [25] show that a dynamic adjustment of the penalty parameter α^{ν} is a strategy that improves the overall convergence behavior. They propose a sequence, that starts with a relatively low value, and then it is gradually raised to a limiting value. According to their analysis, this choice avoids the algorithm gets stalled at a suboptimal solution. The initial iterations prioritize the updating of dual variables, while the last ones, the updating of the primal variables. The DPHA in the work of Mulvey and Vladimirou [25] is used as a heuristic, even though, as we show in Theorem 1, it is an exact method.

3.2 DPHA-based Heuristic

One approach for solving the Two-Stage Stochastic Mixed-Integer Problem (TSMIP) using DPHA is to solve the deterministic quadratic subproblems over a convex relaxation of the feasible set and then, once the algorithm converges, if the solution does not satisfy the integrally constraints, continue with a branch-and-bound strategy. This method is proposed and computational study in Atakan and Sen [3]. However, in practice, the subproblem is usually solved over the non-convex set of mixed-integer solutions rather the convex set, and the algorithm is used as a heuristic (*e.g.*, Gade et al. [16]). In this section, we present the DPHA-based Heuristic (DPHH) for the TSMIP and we provide optimality guarantees for the DPHH and comment extensions that can be studied for this approach.

The DPHH is the same as Algorithm 1 but it solves subproblems (line 4) with solution set $\mathcal{X}$ (instead of $\text{conv}(\mathcal{X})$) and $x_k^0 \in \text{argmin}_{x\in\mathcal{X}} F_k(x_k)$, $k \in [K]$ in the first iteration.

The last section in Rockafellar and Wets [30] is dedicated to the case when the primal problem is non-convex. The authors suggest that the PHA is an effective strategy for solving this class of problems, as it has been confirmed in practice. In Theorem 6.1, they show that the solution obtained is a stationary point that is not necessarily optimal to the original problem but that it is an optimal solution for a slightly different problem. For this, they assume that the algorithm "converges to something". We also assume the DPHH converges to a feasible solution when the subproblem is solved on a non-convex set for the following proposition (this result is not covered in Rockafellar and Wets [30]).

Proposition 5. *At each iteration $\nu \geq 0$ the solution vector $x_k^{\nu+1}$, $k \in [K]$, is optimal for the following problem,*

$$\min_{x_k \in \mathcal{X}} \left\{ F_k(x_k) + \lambda_k^{\nu \intercal} x_k + \frac{\alpha^\nu}{2} \|x_k - \hat{x}^\nu\|^2 \right\}.$$

If the sequence $\{\hat{x}^\nu\}_{\nu \geq 0}$ generated by the DPHH does converge to an implementable solution x^* then,

$$\sum_{k \in [K]} p_k F_k(\bar{x}) + \frac{\alpha^\infty}{2} \|\bar{x} - x^*\|^2 \geq \sum_{k \in [K]} p_k F_k(x^*), \tag{19}$$

with $\bar{x} \in \mathcal{X}$ an optimal solution for problem (4) and α^∞ the corresponding weight associated to solution x^*.

Proof. At each iteration $\nu \geq 0$ and $k \in [K]$, we have that,

$$\begin{aligned} F_k(\bar{x}) + \frac{\alpha^\nu}{2} \|\bar{x} - \hat{x}^\nu\|^2 &\geq \min_{x_k \in \mathcal{X}} \left\{ F_k(x_k) + \lambda_k^{\nu \intercal} x_k + \frac{\alpha^\nu}{2} \|x_k - \hat{x}^\nu\|^2 \right\}, \\ &= F_k(x_k^{\nu+1}) + \lambda_k^{\nu \intercal} x_k^{\nu+1} + \frac{\alpha^\nu}{2} \|x_k^{\nu+1} - \hat{x}^\nu\|^2, \end{aligned}$$

with $\bar{x} \in \mathcal{X}$ an optimal solution. Thus, taking limit $\nu \to \infty$, we have,

$$F_k(\bar{x}_k) + \frac{\alpha^\infty}{2} \|\bar{x} - x^*\|^2 \geq F_k(x^*).$$

□

When the first-stage variables are binary, *i.e.*, when $\mathcal{X} \subseteq \{0,1\}^N$, we can extend the previous proposition. Noticing for binary variables it holds $\|\bar{x} - x^*\|^2 \leq N$, we get the following corollary.

Corollary 1. *For $\mathcal{X} \subseteq \{0,1\}^N$, if the generated sequence $\{\hat{x}^\nu\}_{\nu \geq 0}$ does converge to a solution x^* then,*

$$\sum_{k \in [K]} p_k F_k(\bar{x}) + \frac{\alpha^\infty}{2} N \geq \sum_{k \in [K]} p_k F_k(x^*). \tag{20}$$

Note that both equation (20) and the augmented Lagrangian function in (14) help to explain the classical trade-off seeing in practice when using PHA-based heuristics: convergence speed of the solution versus the quality of the solution found. For large values of $\{\alpha^\nu\}_{\nu\geq 0}$, the primal convergence is prioritized over dual, but the cost of that prioritization is paid in the solution quality bound in (20). Small values for the sequence provide high quality solutions but it might require several iterations to converge.

Many alternative cases can be considered for Proposition 5, not only quadratic regularization term as we have for the DPHH. The result is valid for any function that measures the distance between the scenario solution x_k, $k \in [K]$, and the estimate $\hat{x}$. For example, we can consider a different norm (*e.g.*, $\|x_k - \hat{x}\|^1$) as proposed in Escudero et al. [15], but also functions like $\log(\|x_k - \hat{x}\|^2)$ or $\exp(\|x_k - \hat{x}\|^2)$, and the result in Proposition 5 is essentially the same. Also, as propose in Rockafellar and Wets [30], the subproblem can be solved heuristically (*i.e.*, not finding an optimal solution) and the bound adapted with an additional term.

3.3 DPHH for Solving the TSMIPB

In the transportation field, many stochastic problems have binary variables for decision making, specially when it involves vehicle routing planning. This means that an important class of problems requires a solution approach that can handle the coordination of binary variables along different scenarios. We consider the problem (4) with the first-stage-variables set restricted to binary values, $\mathcal{X} \subseteq \{0,1\}^N$. This problem corresponds to a Two-Stage Stochastic Mixed-Integer with Pure First-Stage Binary Variables Problem (TSMIPB).

If we restrict the algorithm to solve the subproblems over a mixed-integer set with first-stage variables defined in $\mathcal{X} \subseteq \{0,1\}^N$, we can solve a simpler problem: instead of a quadratic objective function, we have a linear one, so the subproblem can be solved as a MIP. Given $x_k \in \mathcal{X}$, we have $\|x_k\|^2 = \sum_{i\in[N]} x_{ki}^2 = \sum_{i\in[N]} x_{ki}$, so,

$$\begin{aligned}
\|x_k - \hat{x}\|^2 &= \|x_k\|^2 - 2x_k^\intercal \hat{x} + \|\hat{x}\|^2, \\
&= \mathbb{1}^T x_k - 2x_k^\intercal \hat{x} + \|\hat{x}\|^2, \\
&= (\mathbb{1} - 2\hat{x})^\intercal x_k + \|\hat{x}\|^2.
\end{aligned}$$

For λ_k and $\hat{x}$ vectors given, and omitting the constant term $\|\hat{x}\|^2$ from the objective function, we can reformulate the problem (12) as a deterministic MIP,

$$\begin{aligned}
\min \quad & c_k^\intercal y + \left(\lambda_k + \frac{\alpha}{2}\mathbb{1} - \alpha\hat{x}\right)^\intercal x_k, && \text{(21a)} \\
\text{s.t.} \quad & W_k y \geq h_k - T_k x_k, && \text{(21b)} \\
& y \in \mathbb{Z}^Q \times \mathbb{R}^{M-Q}, && \text{(21c)} \\
& x_k \in \mathcal{X}. && \text{(21d)}
\end{aligned}$$

The linear structure of problem (21) is not only algorithmically attractive (solve a linear problem is usually more efficient in practice than a quadratic problem for the same instance size), but also it allows to integrate this method with other approaches. In particular, a decomposition approach, such as branch-and-price (column generation), can be easily included for solving the subproblem. In this case, the columns generated from previous iterations or even for different subproblems can be used for solving the current subproblem. The column generation method advantages (*e.g.*, better problem linear relaxations) are then inherited by the DPHH.

4 Multiple Scenario Approach and Consensus Functions

The Multiple Scenario Approach (MSA), a common approach for solving stochastic and dynamic problems in transportation, which was introduced by Bent and Van Hentenryck [5] for the Vehicle Routing Problem with Stochastic Customer. As with the DPHA approach, a finite set of scenarios are generated from known probability distributions that represent the random variables associated to the problem. Then, a solution is found for each scenario, independently, and a consensus plan is determined.

We consider the TSMIP for presenting the MSA. The solution obtained for each scenario is projected to a set of solutions $\mathcal{P}$ in which there is only deterministic information. Then, a consensus plan $\rho \in \mathcal{P}$ for the first-stage variables is generated. In order to determine this plan, a similarity function $\Psi : \mathcal{P}^2 \to \{0, 1\}$ and a scoring function $\Phi : \mathcal{P} \to \mathbb{N}$ are defined. The similarity function measures whether two plans $\rho_1, \rho_2 \in \mathcal{P}$ are identical ($\Psi(\rho_1, \rho_2) = 1$) or not ($\Psi(\rho_1, \rho_2) = 0$). This allows us to obtain a score $\Phi(\rho)$ for each plan ρ, as follows,

$$\Phi(\rho) := \sum_{\rho' \neq \rho} \Psi(\rho, \rho'). \tag{22}$$

This score reflects how many plans ρ' (from different scenarios) are identical to ρ. The consensus function identifies the plan ρ^* that has the highest score and proposes that plan for decision making,

$$\rho^* = \operatorname*{argmax}_{\rho} \{\Phi(\rho)\}. \tag{23}$$

The resulting distinguished plan is not too different to other plans, in a kind of least commitment strategy.

An alternative consensus function is proposed in Song et al. [33], which addresses the symmetry of scenario solutions. This function is based on the Hamming distance, function that measures the difference between vectors or matrices by comparing pairs of entries. Given solutions for scenarios $k, k' \in [K]$, the Hamming distance for solutions $x_k, x_{k'}$ is defined as,

$$H(x_k, x_{k'}) := \sum_{i \in [N]} |x_{ki} - x_{k'i}|. \tag{24}$$

The scenario solution that is chosen is the one with the smallest distance value,

$$x_{k^*} = \operatorname{argmin} \left\{ \sum_{k' \in [K], k' \neq k} H(x_k, x_{k'}) \middle| x_k, k \in [K] \right\}. \tag{25}$$

Many combinatorial problems have equivalent solutions but have different vectors of first stage variables associated with them. For example, for a vehicle routing problem, where the first stage solution represents the customers that are assigned to the vehicles, different vectors can be obtained by simply permuting the order in which the vehicles are represented. This is why the symmetry of the solution must be identified and addressed, allowing the consensus function to be used in an efficient way. In Song et al. [33] a strategy for dealing with symmetry is proposed. Understanding that trying all possible combinations to permute the obtained solutions is not practical computationally, the authors propose a lexicographic heuristic.

The lexicographic heuristic can be applied to any TSMIPB. For a given scenario, the obtained solution is ordered by first considering the sequences that have a higher number of ones (the first-stage is binary so it only has zeros and ones). In case there are two sequences with the same number of ones, then we continue with a lexicographic rule: we order the sequences considering the indices of entries with ones, starting with the smallest index.

In our computational study in Section 6, we implement these consensus functions, presenting a vehicle routing problem variant for which the lexicographic rule is very effective.

5 Subgradient Algorithm

In this section, we present a Subgradient Algorithm (SA) to solve problem (6), in which lower bound values are found by solving a MIP for each scenario and upper bound values (and feasible solutions) are found using the consensus functions presented in Section 4. We present in detail this algorithm and provide some theoretical guarantees.

The SA works in a similar way DPHA works: we solve simple independent subproblems for each scenario, and sequentially update the multipliers associated to non-anticipativity constraints by approximating the Lagrangian dual function from above. However, for the SA, we initially need make an additional assumption: all scenarios have the same realization probability, $p_k = 1/K$ for all $k \in [K]$. After presenting the SA and its properties, we show how we can relax this assumption and consider instead any rational $p_k > 0$.

Consider the TSMIP. For each scenario $k \in [K]$, we solve $\underline{L}_k(\lambda_k)$,

$$\min_{x_k \in \mathcal{X}} F_k(x_k) + \lambda_k^\intercal x_k, \tag{26}$$

with $\Lambda \in \mathcal{D}$.

For a given optimal solution to (26), we can easily compute a lower bound for the optimal value of the primal formulation (4).

Proposition 6. *Let $\Lambda \in \mathcal{D}$ be a dual multiplier. For an optimal solution x^* for problem* (4)*, with $x_k = x^*$, $k \in [K]$. We have $\frac{1}{K}\sum_{k\in[K]} \underline{L}_k(\lambda_k) \leq \frac{1}{K}\sum_{k\in[K]} F_k(x^*)$.*

Proof. Details in Appendix A.1. □

For a given solution x_k, $k \in [K]$, of problem (26), we update the dual values in the direction in which the Lagrangian function increases the most. This update helps to coordinate the x_k solutions be the same by adding penalties or rewards associated to the first-stage variables in the objective function. The approach, similar to the Gradient Descent, requires the subgradient of the Lagrangian function and the step size for each iteration.

Let x_k be an optimal solution of formulation (26) when using multipliers Λ. A subgradient of $\underline{L}(\Lambda)$, $g(\Lambda) \in \partial\underline{L}(\Lambda)$, at point $\Lambda \in \mathcal{D}$, is given by, $g_k(\Lambda) = \frac{1}{K}x_k$. Expressing g as a vector, we have $g(\Lambda) = \frac{1}{K}(x_1, \dots, x_K) = \frac{1}{K}X$.

Lemma 1. *For a given $\Lambda^0 \in \mathcal{D}$, the vector $g(\Lambda^0)$ is a subgradient of $\underline{L}(\Lambda)$ at Λ^0.*

Proof. Details in Appendix A.1. □

Given a $x_k \in \mathcal{X}$, $k \in [K]$, an optimal solution to (26) when multipliers $\Lambda \in \mathcal{D}$ are used, we update the dual multipliers using the subgradient presented above. A new λ'_k multiplier is computed as follows,

$$\begin{aligned}
\lambda'_k &= \Pi\left(\lambda_k + \alpha g_k(\lambda)\right), \\
&= \Pi\left(\lambda_k + \alpha\frac{1}{K}x_k\right), \\
&= \lambda_k + \alpha\frac{1}{K}x_k - \frac{1}{K}\sum_{i\in[K]}\left(\lambda_i + \alpha\frac{1}{K}x_i\right), \\
&= \lambda_k + \frac{\alpha}{K}\left(x_k - \hat{x}\right),
\end{aligned}$$

where $\alpha > 0$ is an step size, and the operator $\Pi(\cdot)$ is the projection function onto the set $\mathcal{D}$. Note that w.l.o.g, we can assume that α absorbs the constant term $\frac{1}{K}$, so we just refer to α^ν as the step size for iteration $\nu \geq 0$.

The Subgradient Algorithm is the following.

```
1  let $\lambda_k^0 = 0$, $k \in [K]$;
2  set $\nu = 0$;
3  for each $k \in [K]$ get $x_k^{\nu+1}$ by solving,
       $$x_k^{\nu+1} \in \operatorname*{argmin}_{x_k \in \mathcal{X}} F_k(x_k) + \lambda_k^{\nu\intercal} x_k;$$
4  compute $\hat{x}^{\nu+1} = \frac{1}{K}\sum_{k\in[K]} x_k^{\nu+1}$;
5  find a feasible solution $\tilde{x}^{\nu+1}$ using *consensus function*;
6  update $L^{\mathrm{best}(\nu)}$ and $U^{\mathrm{best}(\nu)}$;
7  **if** $L^{\mathrm{best}(\nu)} = U^{\mathrm{best}(\nu)}$ **then**
8      stop, $\tilde{x}^{\nu+1}$ is optimal;
9  **else**
10     get $\lambda_k^{\nu+1} = \lambda_k^{\nu} + \alpha^{\nu}(x_k^{\nu+1} - \hat{x}^{\nu+1})$;
11     $\nu \leftarrow \nu + 1$ and go to line 3;
12 **end**
```
Algorithm 2: Subgradient Algorithm.

We start with multipliers $\lambda_k^0 = 0$, for all $k \in [K]$, and iteration index $\nu = 0$. We solve problem (26), getting an optimal solution $x_k^{\nu+1}$. With this solution, we compute the average vector $\hat{x}^{\nu+1} = \frac{1}{K}\sum_{k\in[K]} x_k^{\nu+1}$ (an implementable solution) and a feasible solution $\tilde{x}^{\nu+1}$ using the consensus function (an admissible solution). We then update the best lower bound $L^{\mathrm{best}(\nu)}$ and the best upper bound $U^{\mathrm{best}(\nu)}$ at iteration $\nu \geq 0$, $L^{\mathrm{best}(\nu)} = \max\{L^{\mathrm{best}(\nu-1)}, \underline{L}(\lambda^{\nu})\}$ and $U^{\mathrm{best}(\nu)} = \min\left\{U^{\mathrm{best}(\nu-1)}, \frac{1}{K}\sum_{k\in[K]} F_k(\tilde{x}_k^{\nu+1})\right\}$. We check the condition $L^{\mathrm{best}(\nu)} = U^{\mathrm{best}(\nu)}$. If it holds, we stop, the first-stage solution $\tilde{x}^{\nu+1}$ is optimal; otherwise, we update the multipliers $\lambda_k^{\nu+1} = \lambda_k^{\nu} + \alpha^{\nu}(x_k^{\nu+1} - \hat{x}^{\nu+1})$, $k \in [K]$, we increase the iteration index $\nu \leftarrow \nu+1$ and continue with a new iteration in line 3. The sequence of step size $\{\alpha^{\nu}\}_{\nu=0}^{\infty}$ is any sequence with $\alpha^{\nu} > 0$, for all $\nu \geq 0$, such that $\frac{\sum_{\nu=0}^{\eta} \alpha^{\nu 2}}{\sum_{\nu=0}^{\eta} \alpha^{\nu}} \xrightarrow{\eta\to\infty} 0$.

The binary case, when $\mathcal{X} \subseteq \{0,1\}^N$, has an useful interpretation of the algorithm duals updating. For a given feasible solution x_k, $k \in [K]$, with $x_{ik} = 1$ for a component $i \in [N]$, the multiplier λ_{ik} is updated adding the value $\alpha(1 - \hat{x}_{ik}) \in [0, \alpha]$. If for many scenarios the i-th component of $\bar{x}$ is zero, then a penalty (whose value is close to α) is included for that component. This penalty motivates that the next iteration subproblem (26) is solved, the resulting x_{ki} is zero. When $x_{ik} = 0$ for some $i \in [N]$, the i-th component for λ_k is updated adding $-\alpha\hat{x}_{ik} \in [-\alpha, 0]$. Again, if for many scenarios the i-th component of the first-stage solution is one, then a reward is added, which promotes this variable to be equal to one the next time we solve (26). This is the way this approach coordinates first-stage variables, by adding penalties and rewards according to the scenario solution values majority.

In the following Theorem, we show that the Lagrangian function of the SA converges to an optimal solution value. This result is provided by Polyak [28], where it is proved that the SA converges when the sequence of α^{ν} satisfies the conditions we stated previously. However, unlike the DPHA, the convergence is not to an optimal primal and dual solutions but to a Lagrangian solution value.

Theorem 2. *Let Λ^* be an optimal solution for problem* (6)*, and let $\{L^{\text{best}(\nu)}\}_{\nu \geq 0}$ be the sequences of best Lagrangian function values that the Algorithm 2 generates. The sequence $\{L^{\text{best}(\nu)}\}_{\nu \geq 0}$ converges to $\underline{L}(\Lambda^*)$.*

Proof. Details in Appendix A.1. □

The optimal value of the Lagrangian function $\underline{L}(\Lambda^*)$ is equal to the problem (4) solved on the convex hull.

Proposition 7. *Let Λ^* be an optimal solution for problem* (6)*. The optimal value $\underline{L}(\Lambda^*)$ equals the optimal value of the problem* (10)*.*

Proof. Details in Appendix A.1.

□

In this algorithm we compute upper bound values using the consensus function presented in Section 4. This allows us to have feasible solutions that improve over iterations (so does the upper bound value). This approach takes advantage of all the benefits that the consensus function has. First, it allows us to find a plan (first-stage variables values) that is the most similar to scenarios' plans; second, in the case of full-recourse, this solution is also feasible and implementable; and finally, it addresses the symmetry that the first stage solutions may have. The algorithm we propose is an extension of the well-known subgradient method by finding feasible solutions through efficient methods, the consensus functions.

We initially assume that $p_k = \frac{1}{K}$, $k \in [K]$. Even though in many practical cases this simplification is not a problem, we show how we can recover a setting with probabilities that can be any positive rational number. In Algorithm 2, we change how the implementable solution $\hat{x}^{\nu+1}$ is computed: as in the DPHA, we use $\hat{x}^{\nu+1} = \sum_{k \in [K]} p_k x_k^{\nu+1}$. The following proposition shows that this modification does not change the SA convergence guaranty.

Proposition 8. *The convergence of Algorithm 2 to an optimal solution value (stated in Theorem 2), holds for formulation* (4) *with rational $p_k > 0$, $k \in [K]$, and $\sum_{k \in [K]} p_k = 1$.*

Proof. Details in Appendix A.1. □

5.1 Connection with the DPHA

After presenting the SA and the DPHA, we can show the similarities and differences between them. There are several points to highlight:

1. Both algorithms converge to the optimal value of problem (10), but the DPHA converges to a dual and a primal solution, while the SA only to the value of the objective function. The main reason for this is that in the DPHA an augmented Lagrangian is solved (indirectly we solve function (14)), while in SA only the Lagrangian function.

This shows one of the major advantages of using DPHA: the scenario subproblems are independent each other but the algorithm has guarantees like those of the augmented Lagrangian method.

2. In both algorithms the duals updating is performed using the distance of the scenario solutions with respect to the average multiplied by the weights α^ν, $\nu \geq 0$. However, in the SA the sequence of these weights is decreasing while in DPHA the sequence has to be bounded and non-decreasing. The reason for this is that, in the DPHA case, the weights simultaneously affect both the dual and the primal solutions, so they must not have extreme values (*i.e.*, $\alpha^\nu \to 0$ or $\alpha^\nu \to \infty$), while in the SA, they only update dual variables.

3. This last point also means that the DPHA guarantees improvements at each iteration, as stated in Proposition 4, while the SA can have a convergence that oscillates (a lower bound value that is not monotone), because the α^ν, $\nu \geq 0$, only affects the dual variables.

4. An advantage of the SA over the DPHA is that at each iteration lower bounds are obtained for the problem. Since we also propose an efficient way to find feasible solutions, for a given optimality tolerance, the SA allows earlier stopping.

5. The SA solves a MIP for each scenario subproblem while the DPHA solves a quadratic problem on a convex set. In general, the convex hull of the problem is not available, so the DPHA requires additional work. For example, in Boland et al. [7] it is proposed to use a SDM. When the DPHA is restricted to solve a mixed-integer quadratic problem for subproblems, a heuristic is obtained; for the SA, solving a MIP is an exact method.

6 Computational Experiments

We test our algorithms for two different problems, the canonical Stochastic Server Location Problem (SSLP) and the Two-Stage Stochastic Assignment and Team-Orienteering Problem (TSSATOP). The SSLP is one of the most popular problems considered for studying the PHA or subgradient methods, so it is a natural setting for our experiments. The TSSATOP, introduced in Song et al. [33], is studied using the MSA with a lexicographic rule consensus function, which we consider a relevant problem setting with an interesting approach for comparing our consensus algorithms.

All experiments are implemented using the Java programming language, with CPLEX 12.4 as the optimization solver. The computer servers employed for computation use CentOS Linux with four eight-core Intel Xeon E5-2670 processor and 128 GB of RAM. Both SSLP and TSSATOP instances are run for up to two hours, with up to 5 subproblems running in parallel for each instance. Since most of the algorithms run for several iterations with scenario subproblems that differ only in the objective function from the previous iteration, we initialize these subproblems with the last solution found for each of them.

In our computational experiments we do not study the DPHA since it is already proven in Mulvey and Vladimirou [25] to be an efficient strategy for solving stochastic convex problems. Here we focus on comparing different consensus algorithms for problems with integer variables, so we can indicate which are the best algorithms for problems that are usually studied in the literature.

In this section, we first show the sequences we use for our consensus algorithms and then, we present each problem, the SSLP and the TSSATOP, reporting the results for each of them.

6.1 Sequences

For the SA it is known that the best strategy is to use a sequence $\alpha^\nu = \frac{L}{\sqrt{\nu+1}}$ for $\nu \geq 0$, with L a constant. This sequence guarantees the converge of the algorithm and also it is the one with the fastest rate (see Nesterov [26] equation (3.2.10)). Our experiments consider this sequence with different values for L.

For the DPHH we consider a step size drawn from the step size we use for the SA. For the DPHH we need a non-decreasing sequence, so we propose,

$$\alpha^\nu = L\left(1 - \frac{1}{\sqrt{1+\nu}}\right), \qquad \nu \geq 0, \tag{27}$$

with $L > 0$, a constant. In our experiments, we also consider different values for L.

In Figure 1, we plot the sequence given by equation (27) for $L = 10, 50, 100$. In this class of sequences, in the first few iterations the sequence reaches an important fraction of the value L, having then iterations with small marginal increments.

Inspired by Mulvey and Vladimirou [25]'s adjustment sequences, we consider a sequence with multiplicative updates, with a multiplier $\theta \geq 1$ and an exponent determined by the Riemann zeta function $\zeta(f)$, $f > 1$. We define the following function,

$$\zeta^\nu(f) := \sum_{i=1}^{\nu} \frac{1}{i^f}. \tag{28}$$

Thus, we have $\zeta(f) = \lim_{\nu\to\infty} \zeta^\nu(f)$. Note that when $f = 2$, we get the known limit $\zeta(2) = \frac{\pi^2}{6} \approx 1.64$. For a given $f > 1$ and $\theta \geq 1$, the sequence of $\{\alpha^\nu\}_{\nu\geq 0}$ is,

$$\alpha^\nu = \begin{cases} 1, & \nu = 0, \\ \theta^{\zeta^\nu(f)}, & \nu \geq 1, \end{cases} \tag{29}$$

so, in the limit, $\alpha^\infty = \theta^{\zeta(f)}$.

We can rewrite the value of α^ν recursively as follows,

$$\begin{aligned} \alpha^\nu &= \theta^{\sum_{i=1}^{\nu} \frac{1}{i^f}}, \\ &= \theta^{\frac{1}{\nu^f} + \sum_{i=1}^{\nu-1} \frac{1}{i^f}}, \\ &= \alpha^{\nu-1}\theta^{\frac{1}{\nu^f}}, \end{aligned}$$

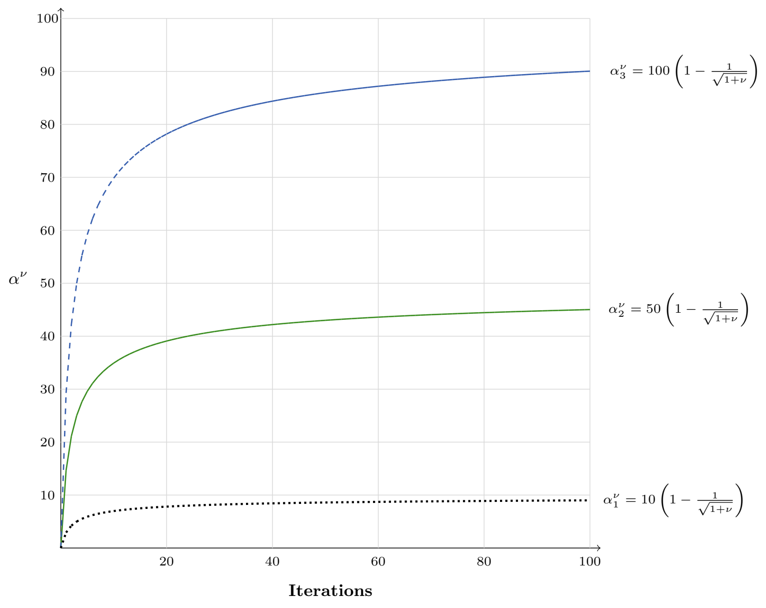


Figure 1: Non-decreasing sequence $\alpha^\nu = \frac{L}{\sqrt{\nu+1}}$, $\nu \geq 0$, for the DPHH with values $L = 10, 50, 100$.

expression that shows that the sequences we propose differed from the one suggested in Mulvey and Vladimirou [25]. They have a strategy where both $\alpha^{\nu-1}$ and θ are updated by the exponent f. Out strategy provides more control and flexibility on the sequence growth.

In Figure 2 we plot sequences defined in (29), all with limiting value of 50, and with factors $f = 1.2, 1.5, 1.8$. The multipliers θ are adjusted accordingly, so we have $\theta = 2.01$ for $f = 1.2$, $\theta = 4.47$ for $f = 1.5$, and $\theta = 8.01$ for $f = 1.8$. In these sequences we can see the effect of factor f in the speed of converge, with $f = 1.2$ the slowest growth and $f = 1.8$ the fastest.

6.2 Stochastic Server Location Problem

We consider the Stochastic Server Location Problem (SSLP) studied in Ntaimo and Sen [27]. This problem is motivated by the servers location problem a telecommunication service provider faces, where it has to be determined the most profitable plan for serving potential customers. By using future scenarios that are deemed possible, a plan is chosen, where a scenario corresponds to a set of potential clients that do materialize. In Appendix A.2 we describe in detail this problem and formulate it as a TSMIPB.

We consider 9 instances studied in Ntaimo and Sen [27][1], with locations ranging from 5 to 15, and 8 instances studied in Lim et al. [23][2] that have a larger number of locations, ranging from 20 to 90. Details can be found in Table 1. In the first column of the table is the source of the instance; in the second column is the instance name; in the third column is the

[1]available in `https://www2.isye.gatech.edu/ sahmed/siplib/sslp/sslp.html`

[2]available in `https://limconghan.github.io/smip/`

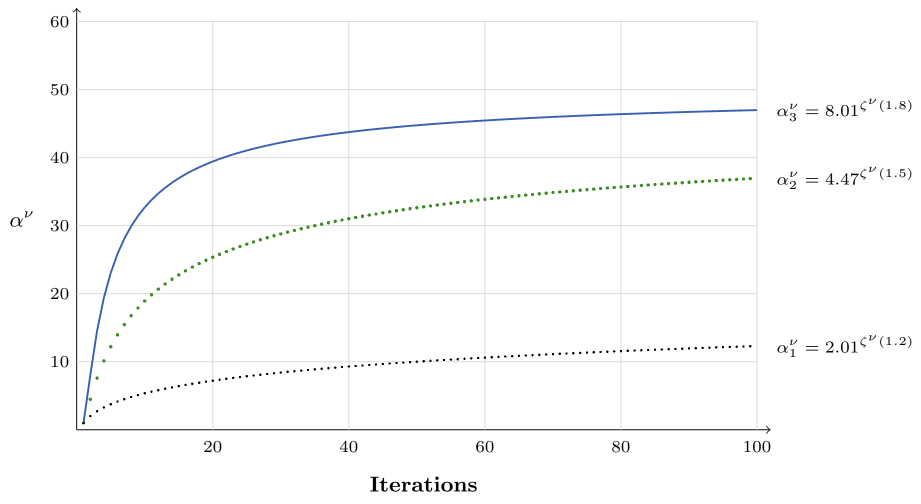


Figure 2: Non-decreasing sequence in (29) for the DPHH, with values $f = 1.2, 1.5, 1.8$. All sequences converge to 50.

number of locations; and, in the fourth and fifth columns are the number of clients and the number of scenarios, respectively. In the "Best Known" column is the best known objective function value for the instance. For Ntaimo and Sen [27] instances, these values are optimal, while for Lim et al. [23] no best values are known, since the methodology developed here is for finding lower bounds for the instances and not feasible solutions. In the "Best Found" column, we report the best value found by any of the implemented algorithms. Note that for the Ntaimo and Sen [27] instances our proposed methods are able to find the best known solution for each instance. Finally, in the "Gap (%)" column, we report the optimality gap obtained by our proposed methods (percentage computed using the best lower bound and the best upper bound found). When the gap is zero, we guarantee optimality for the instance.

Source	Instance	Locations	Clients	Scenarios	Best Known	Best Found	Gap (%)
Ntaimo and Sen [27]	SSLP-L5C25S50	5	25	50	-121.6	-121.6	optimal
	SSLP-L5C25S100	5	25	100	-127.37	-127.37	optimal
	SSLP-L10C50S50	10	50	50	-364.64	-364.64	optimal
	SSLP-L10C50S100	10	50	100	-354.19	-354.19	0.05
	SSLP-L10C50S500	10	50	500	-349.14	-349.14	0.53
	SSLP-L10C50S1000	10	50	1000	-351.71	-351.71	0.61
	SSLP-L15C45S5	15	45	5	-262.4	-262.4	optimal
	SSLP-L15C45S10	15	45	10	-260.5	-260.5	optimal
	SSLP-L15C45S15	15	45	15	-253.6	-253.6	optimal
Lim et al. [23]	SSLP-L20C100S50	20	100	50	-	-846.26	0.25
	SSLP-L20C100S200	20	100	200	-	-839.33	0.73
	SSLP-L30C100S50	30	100	50	-	-855.84	0.21
	SSLP-L30C100S200	30	100	200	-	-845.02	0.68
	SSLP-L60C60S50	60	60	50	-	-489.96	optimal
	SSLP-L60C60S200	60	60	200	-	-476.99	0.47
	SSLP-L90C45S50	90	45	50	-	-339.58	0.76
	SSLP-L90C45S200	90	45	200	-	-331.63	1.64

Table 1: SSLP instances. For each instance we report the source, details about the instance, the best known solution value, the solution value and the optimality gap we found in our computational study.

In this work we are able to find feasible solutions for all Lim et al. [23] instances, prove optimality for one of the instances (SSLP-L60C60S50), and report optimality gaps mostly less than 1 %.

We consider the MSA, the PHA-based Heuristic (PHH), the DPHH and the SA for solving the instances. For the DPHH, different parameters were tested for the sequences $\{\alpha^\nu\}_{\nu\geq 0}$. The parameters of these sequences are determined so that the resulting limits are 10, 50 and 100. In the case of DPHH with square-root sequences (as in equation (27)), the values of L are defined according to these limits. For the sequences defined in (29), three values are taken for f, $f = 1.2$ (slow growth); $f = 1.5$ (moderate growth); and $f = 1.8$ (fast growth). In all cases, the value of the multipliers are adjusted so that the sequences converge to 10, 50, and 100. For the PHH with fixed parameter, four values are considered for α, 1, 10, 50, and 100. Finally, for the SA, four values are taken for L, $L = 1, 10, 50, 100$, generating the sequences $\alpha^\nu = \frac{L}{\sqrt{\nu+1}}$, $\nu \geq 0$.

The Table 2 contains a summary of the results produced by the algorithms. In the first column is the algorithm. For the DPHH we have two types of sequences, one generated according the equation (29), which we refer to as DynamicMult(θ, f), and another one using equation (27), which we refer to as DynamicSquare(L). The PHH is represented by PHH(α), with α the fixed parameter of the algorithm, while the SA corresponds to Subgradient(L). The “Parameters” column contains the parameters that have been used for each algorithm. The “Optimal” column has the number of instances in which the algorithm proves optimality, while the “Feasible” column contains the number of instances in which the algorithm finds a feasible solution, within the two hours time limit. The “Gap Opt” column corresponds to the average optimality gap, while “Gap Alg” shows the average gap guaranteed by the algorithm, *i.e.*, the best lower bound that the method can guarantee with respect to the best feasible solution found. The “Gap Best” column shows the average of the gap of the best solution found by any of the algorithms with respect to the solution of the method for all instances. Finally, the "Iterations" and "Runtime" columns have the average number of iterations and the average time (in seconds), respectively.

Algorithm	**Parameters**		**Solution**		**Gap**				
			Optimal	**Feasible**	**Opt**	**Alg**	**Best**	**Iterations**	**Runtime (s)**
DynamicMult(θ, f)	1.51	1.20	0	12	0.15	3.77	0.04	58.53	3674
	2.01	1.20	0	13	0.24	4.19	0.04	31.18	2839
	2.28	1.20	0	16	0.42	4.86	0.10	26.12	2499
	2.41	1.50	0	13	0.24	4.20	0.04	34.18	2816
	4.47	1.50	0	17	0.60	5.24	0.25	15.65	1497
	5.83	1.50	0	17	0.75	5.38	0.40	11.35	1030
	3.40	1.80	0	14	0.27	4.15	0.08	29.59	2570
	8.01	1.80	0	17	0.81	5.44	0.46	10.71	1001
	11.58	1.80	0	17	0.76	5.40	0.42	7.71	665
DynamicSquare(L)	10		0	13	0.26	3.99	0.09	34.06	2975
	50		0	17	0.71	5.35	0.36	9.29	814
	100		0	17	0.80	5.44	0.45	8.18	400
MSA			0	17	7.60	11.92	7.28	1.00	116
PHH(α)	1		0	6	0.00	1.99	0.00	152.71	5646
	10		0	15	0.39	4.60	0.11	23.94	1888
	50		0	17	0.67	5.32	0.33	7.47	380
	100		0	17	1.83	6.42	1.48	5.76	238
Subgradient(L)	1		3	17	0.85	2.38	0.51	439.12	6140
	10		4	17	0.72	0.84	0.37	144.53	6006
	50		6	17	0.59	1.01	0.25	135.35	5377
	100		3	17	0.43	1.62	0.08	299.35	6017

Table 2: Summary results for SSLP instances.

From the table, we see that only the SA proves optimality. For all other algorithms, the best lower bound is obtained in the first iteration, when the duals are $\lambda_k = 0$, $k \in [K]$. This shows that the lower bound can be improved for some instances using the SA, and that many of these instances have no optimality gap. Note also that the algorithms are only executed for 2 hours, so there may be other instances to which optimality can be guaranteed. In the "Feasible" column we see that for several instances both PHH and DPHH cannot find feasible solutions within 2 hours. In particular, all cases with $\{\alpha^\nu\}_{\nu\geq 0}$ sequences converging to 10 or less have problems (*e.g.*, $\theta = 3.4$ and $f = 1.8$). Also, the DPHH configurations with slow growth ($f = 1.2$), are not able to find feasible solutions for all instances, even when convergence is to 100 ($\theta = 2.28$). However, for the approaches that do not find feasible solutions for all instances, the solution quality is high, as shown in the "Gap Opt" column. This supports studies in which this trade-off between running times and solution quality is reported [25, 36]. Finally, a correlation between the number of iterations and the execution time is observed: the higher the number of iterations, the higher the average running times. The only exception is the SA, where the number of iterations has no predictable pattern.

To perform a further analysis of the algorithms that find feasible solutions for all instances (17 in total), we generate a plot of the average running time and optimality gap for each algorithm in Figure 3 (we exclude MSA). We observe that all approaches are competitive (except the MSA). For the majority, the average optimality gap are around 0.7%. The PHA-based algorithms have low average execution times, less than 1500 seconds, with DynamicMult$(4.47, 1.5)$ being the algorithm with the longest running times but the one with the highest average solution quality. We can also see that PHH(100) converges in a very

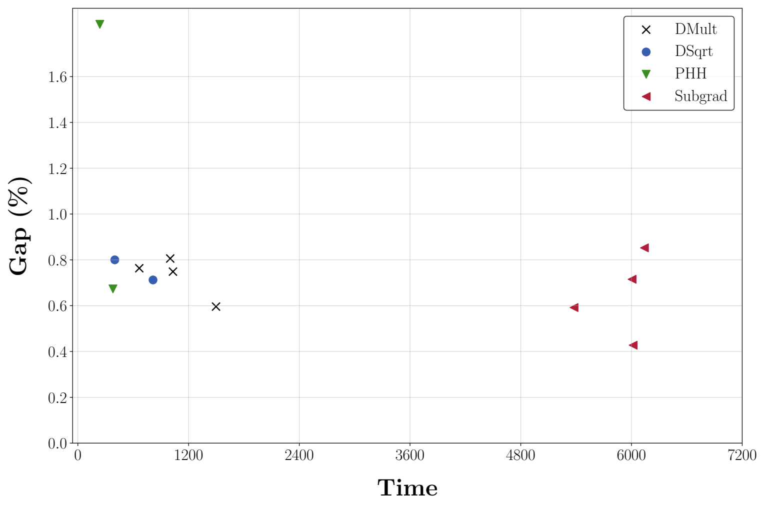


Figure 3: Average running time and optimality gap for algorithms that find feasible solutions for all instances (excluding MSA). [Labels: "DMult" = DynamicMult; "DSqrt" = DynamicSquare; "Subgrad" = SA.]

short time, but it is the algorithm with the lowest solution quality. The SA methods have longer running times, but it is one of them, the Subgradient(100), that finds the best feasible solutions (the algorithm that has the smallest average optimality gap). Moreover, all SA methods find feasible solutions for all instances.

The equation (19) shows that the number of scenarios does not affect the quality of the solution obtained by the PHA-based heuristic, but rather the number of variables in the first stage of the problem. For further analysis of this point, we consider the best of the SSLP algorithms (in terms of running time and solution quality), DynamicMult(4.47,1.5), and the instances for which the optimality of the solution has been proved. The instances considered are those proposed by Ntaimo and Sen [27] and the instance of Lim et al. [23] for which we were able to prove optimality, SSLP-L60C60S50. In Table 3 we show details for each instance. The first column has the value found by the DynamicMult(4.47,1.5) algorithm; the second, the optimal value for the instance; and, the last one, the percentual difference between the first two columns.

Instance	**DynamicMult**	**Optimal**	**Gap**
SSLP-L5C25S50	-118.98	-121.60	2.15
SSLP-L5C25S100	-127.37	-127.37	0.00
SSLP-L10C50S50	-364.64	-364.64	0.00
SSLP-L10C50S100	-354.19	-354.19	0.00
SSLP-L10C50S500	-349.14	-349.14	0.00
SSLP-L10C50S1000	-351.71	-351.71	0.00
SSLP-L15C45S5	-259.20	-262.40	1.22
SSLP-L15C45S10	-260.50	-260.50	0.00
SSLP-L15C45S15	-251.60	-253.60	0.79
SSLP-L60C60S50	-489.96	-489.96	0.00

Table 3: Optimal results and the best solution value found by algorithm DynamicMult(4.47, 1.5) for instances SSLP.

From the table it can be noted that for instances with 10 locations, the algorithm finds optimal solutions for all of them. This is independent of the number of scenarios, which for instances with 10 locations (L10), ranges from 50 to 1000. For instances with 5 locations (L5) and those with 15 (L15), we see that the number of scenarios is not a decisive factor of the quality of the solution found by the algorithm. However, the solution quality dependency on the number of first stage variables of the instance (determined by the number of potential locations) is not clear for this problem.

Several conclusions can be drawn from the SSLP instances. In this problem, we can see the typical trade-off between convergence speed and solution quality, shown in the performance of the PHA-based algorithms. We can also see that DPHH is a competitive approach, having high quality solutions in short running times. Of these DPHH, there are some that show better performances than others. The version of DPHH with multiplicative sequences has more promising results than the version with square-root sequences. This square-root sequence, that is optimal for the subgradient method, it is not optimal for the DPHH, at least for the SSLP. The best performing PHA-based algorithms converge to the value 50, which may be a number that is associated to the parameters of the instances studied in this work. We also highlight that algorithms whose sequences converge to the same values (*e.g.*, 50), have different performances, being an important factor to consider at the moment of algorithm's tuning. Finally, the SA achieves to improve the lower bound for the SSLP instances, showing that many of them do not have an optimality gap. For 7 instances out of 17, optimality can be proved within 2 hours of algorithm execution. It is possible that for other instances optimality can be proven if the SA is run longer.

6.3 Two-Stage Stochastic Assignment and Team-Orienteering Problem

The Two-Stage Stochastic Assignment and Team-Orienteering Problem (TSSATOP) is introduced in Song et al. [33] and it is mainly motivated by the delivery operations of a large European grocery chain.

A company, that manages customer orders, has two type of customers, subscription and on-demand customers. The subscription customers are served daily and, each time the customer is visited, the same driver must deliver the order. Customers require the same driver delivers the order, since building trust between the customer and the driver is an important component of the service: the delivery drive enters the customers' home to place groceries into the customers' refrigerators. The on-demand customers require the service the same day so the company does not know in advance the information associated to these customers. The company might or might not serve them but if it does, a reward is gained and also these customers might subscribe the service in the future, so it is important to serve them.

A finite fleet of vehicles is available to serve customers. Also a finite number of sampled scenarios represents realizations of on-demand customers. A routing and assignment plan is needed, that includes all subscription customers and a subset of on-demand customers for each scenario, so the difference between the expected routing cost and revenue is minimized. This problem is a two stage problem. In the first stage, subscription customers are assigned to drivers, while in the second stage a set of on-demand customer are selected and route together with subscription customers. Details of the TSSATOP and the mathematical formulation can be found in Appendix A.3.

For the experiments carried out in this work, new instances were generated. We follow the approach taken in Song et al. [33], in which the instances of Solomon [32] are used as a basis, and adapted to generate subscription and on-demand customers. The only difference with the Song et al. [33]'s' approach is how we generate the distribution of on-demand customers: instead of generating a set of clients once and then sample scenarios based on that fixed set customers, we generate different sets of customers, so that for the same instance, the scenarios might have completely different customers. We generate instances with 10, 15 and 20 subscription customers, and with clustered and random locations distributions. For each of these combinations, 5 instances were generated, so the total number of instances is 30. Each instance has 10 scenarios. Our instances are available in `https://github.com/felipelagos/tssatop`.

For this problem, only one DPHH (DynamicMult) is considered, where parameters for the sequence with good performance are detected for this problem: we use $\theta = 2.41$ and $f = 1.5$ (so the limit converges to 10). For the SA, we take $L = 50$. To compare the performance of these algorithms, the MSA and the PHH with fixed parameter $\alpha = 1$ and $\alpha = 10$ are considered. All algorithms executions run for up to 2 hours under the same computing conditions as the SSLP instances.

A summary of the results obtained by each algorithm for the 30 TSSATOP instances can be found in the Table 4. In the first column, it is the solving algorithm. The second column, the "Feasible" column, contains the number of instances for which a feasible solution was found. The "Opt Gap" and "Best Gap" columns show the optimality gap and the gap with respect to the best solution found by any of the approaches, respectively. Finally, the "Iterations" column contains the average number of iterations and the "Runtime" column shows the average time (in seconds).

Algorithm	**Feasible**	**Opt Gap**	**Best Gap**	**Iterations**	**Runtime (s)**
MSA	30	6.88	2.44	1.00	70
DynamicMult(2.41, 1.50)	30	4.44	0.14	4.67	947
PHH(1)	25	2.60	0.02	7.17	771
PHH(10)	29	5.40	1.25	3.93	648
Subgradient(50)	30	4.28	0.00	318.83	6982

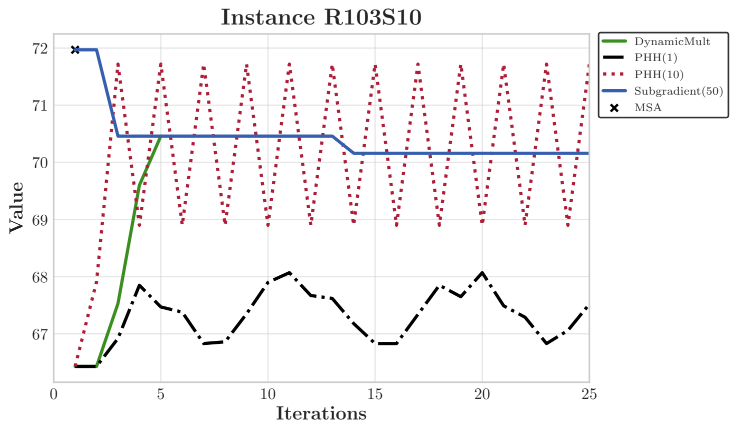
Table 4: Summary results for TSSATOP instances.

The PHH cannot solve all instances, the PHH(1) only manages to solve 25, while PHH(10) solves 29 (out of 30). The DPHH, as well as the SA and the MSA, solve all instances. The SA is the best solving approach in terms of solution quality, finding the best solution for all instances (Best Gap equal to zero). However, the number of iterations and running times are very large. It is worth noting that for this problem, unlike the SSLP, the optimality gap is not closed: for no instance the lower bound value is better than the one found in the first iteration (when $\lambda_k = 0$, $k \in [K]$, *i.e.*, when the non-anticipativity constraint is relaxed). The iteration numbers of DPHH are greater than those of PHH(10), but less than those of PHH(1), while the average running time of DPHH is greater than that of both PHH's. Again, the DPHH shows that it is able to find high quality solutions in short running times, with better convergence stability than the PHH.

To illustrate the convergence of these algorithms, we have two plots in Figure 4. We have the instances R103S10 and C101S10, both with 10 subscription customers, where C101S10 has a clustered distribution for the locations and R103S10 has a random distribution. On the y-axis, it is the objective value of the primal solution, while on the x-axis, it is the iterations number. For instance R103S10, both the PHH(1) and the PHH(10), do not converge to a feasible solution within 2 hours. While the pattern of PHH(10) shows some regularity (which could support the development of some cycle detection strategy), the PHH(1) has no predictable convergence. In instance C101S10, the PHH(1) does also not converge, showing, again, not very predictable convergence. In this case, the PHH(10) does converge, but the convergence is not as monotonic as it is in the DPHH. In both examples the DPHH has a stable convergence which helps to explain the high performance of this algorithm. Finally, note that the SA always finds better solutions than those of the MSA: in the first iteration both approaches are equal, but the SA has more iterations after that to find better solutions.

		PHH(1)			**PHH(10)**		
		Feasible	**Gap**	**Runtime (s)**	**Feasible**	**Gap**	**Runtime (s)**
Customers	10	5	0.00	636.44	9	0.56	274.90
	15	10	0.00	237.15	10	0.35	108.72
	20	10	0.03	134.59	10	0.49	61.61
Distribution	C	14	0.00	242.96	15	0.37	43.61
	R	11	0.03	765.22	14	1.03	401.62

Table 5: Analysis results for TSSATOP instances using PHH(1) and PHH(10). Two instances dimensions are considered: number of subscription customers and customer distributions.



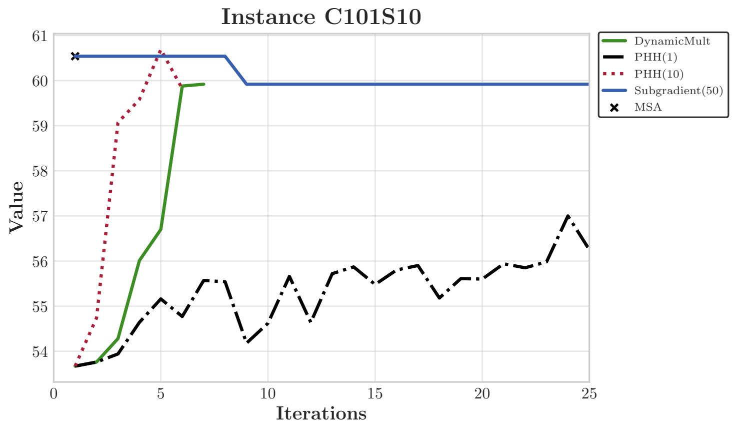


Figure 4: Algorithms convergences for two instances, R103S10 and C1010S10. In the x-axis is the iterations, while in the y-axis is the value of the primal solution.

To analyze the impact of the instances design with respect to the performance of the PHH(1) and PHH(10), we have Table 5. We decide to analyze the performance of the PHH instead of the DPHH, since the later one shows a high performance for this problem, making the analysis difficult (almost for all instances the DPHH finds the known best solution). Here two dimensions are analyzed for the instances: the number of subscription customers and the instance distribution. The analysis of the number of feasible solutions (column "Feasible"), the average gap with respect to the best solution found ("Gap"), and the average time ("Runtime") are carried out for both algorithms. Note that for both algorithms clustered instances are easier to solve than random instances. Clustered instances have fewer "attractive" solutions for the scenarios, so it is easier to coordinate the different solutions. In addition, the distance between the solution found by the algorithm and an optimal one can be large in random instances, so the guarantees that PHA-based heuristics have are of lower quality (equation (20)). With respect to the number of subscription customers, instances with 10 customers show greater complexity than those with 15 or 20. While the number of first stage variables is higher for these instances with 15 or 20 customers, it is more difficult to coordinate solutions along the scenarios when the number of customers is 10.

The TSSATOP problem, unlike the SSLP, exhibits an optimality gap that is not closed by the SA, which can be inherent of the Stochastic Vehicle Routing Problem nature of the TSSATOP. The DPHH has the best convergence performance, while the PHH (with different values for α) is not the most efficient approach for finding feasible solutions. Finally, the number of first stage variables is not always the only factor to take into account to understand the complexity of the problem, in some cases the number of potential solutions for the scenarios may be more decisive. Thus, factors such as the locations distribution and number of subscription customers are relevant components for understanding the performance of the algorithms.

7 Conclusions and Future Work

In this paper we study Scenario Consensus Algorithms: considering a set of scenarios that represent future outcomes, these algorithms solve subproblems independently to find a feasible solution. We present a new exact algorithm, the DPHA, for which we provide theoretical guarantees that helps to understand both this algorithm and the PHA from a new point of view. In addition, we propose a DPHA-based heuristic, the DPHH, and show optimality guarantees for the solution obtained. Then, we show the MSA and the SA with consensus functions. Our analysis allows us to highlight the advantages and disadvantages of the DPHA, the SA, and the MSA, which gives guidance for future research in choosing the proper method for the problem in hand. In a computational study, we study the SSLP and the TSSATOP, and we show the empirical performance of the proposed algorithms.

There are many next steps that can be studied for future research.

We present the DPHA for two-stage stochastic problems, mainly because the analysis and notation is simpler and easier to follow. The approach to be used for the multi-stage case is the same, but requires a clear presentation. A future research direction includes extending the DPHA for multi-stage stochastic problems, not only for the exact version of the algorithm, but also for the heuristic we propose, the DPHH.

In this paper we present a generalization of the PHA. For the PHA several improvements have been proposed. A future direction is to incorporate and study those improvements to the DPHA. In particular, the improvements presented in Boland et al. [7], that allow solving the PHA subproblem efficiently, are interesting extensions to the DPHA. Also, the exact algorithm of Atakan and Sen [3], which integrates the PHA with a branch-and-bound method, can be studied for the DPHA.

The guarantee we propose for the DPHH assumes that the algorithm converges to an implementable solution. Empirically, in most of the cases, this is confirmed. There are exceptions. For example, in our computational study, when the $\{\alpha^\nu\}_{\nu\geq 0}$ sequence is constant, the heuristic does not always converge. In contrast, when we use the DPHH, the dynamic strategy of the sequence leads to higher stability and therefore convergences in few iterations. An interesting research direction is to determine under what conditions the DPHH has a provable convergence. In particular, problems with binary first-stage variables is a setting that usually arises in practice, so theoretical guarantees for this case are valuable to the community.

The DPHH we propose can be studied in more detail. We consider the case with a quadratic regularization term, but other functions can be studied to incorporate into the objective function. As we mentioned in the optimality guarantee in Proposition 5, the function used determines how close the solution is to the optimal solution. Another extension that can be carried out is to integrate the DPHH (and the resulting MIP) with a column generation approach. An PHA-based approach with such a decomposition approach has never been studied.

We propose a new subgradient method with consensus functions. For many problems the lower bound and the upper bound provided by this algorithm do not converge to the same value, so it is important to understand under what conditions the optimality of the feasible solution found can be guaranteed. Can the SA converge to the Lagrangian function optimal value without finding the optimal (implementable and admissible) solution of the problem along the way?

Finally, an important motivation of this work is to solve stochastic and dynamic problems that arise in the transportation field. One direction of future work is to study consensus algorithms for dynamic problems, such as the Same-Day Delivery Problem studied in Voccia et al. [35]. That is, to propose an online approach for solving these problems, so handling the uncertainty of the problem and finding efficient plans. This approach could not only be efficient for real life problems, but could also be used to train offline methods, so an efficient policy can be learned for the problem.

References

[1] Ignacio Aravena and Anthony Papavasiliou. A distributed asynchronous algorithm for the two-stage stochastic unit commitment problem. In *2015 IEEE Power & Energy Society General Meeting*, pages 1–5. IEEE, 2015.

[2] Ignacio Aravena and Anthony Papavasiliou. Asynchronous lagrangian scenario decomposition. *Mathematical Programming Computation*, pages 1–50, 2020.

[3] Semih Atakan and Suvrajeet Sen. A progressive hedging based branch-and-bound algorithm for mixed-integer stochastic programs. *Computational Management Science*, 15(3-4):501–540, 2018.

[4] Rosemonde Ausseil, Jennifer Pazour, and Marlin Ulmer. Supplier menus for dynamic matching in peer-to-peer transportation platforms. *Researchgate*, 03 2021.

[5] Russell W Bent and Pascal Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987, 2004.

[6] John R Birge and Francois Louveaux. *Introduction to stochastic programming.* Springer Science & Business Media, 2011.

[7] Natashia Boland, Jeffrey Christiansen, Brian Dandurand, Andrew Eberhard, Jeff Linderoth, James Luedtke, and Fabricio Oliveira. Combining progressive hedging with a frank–wolfe method to compute lagrangian dual bounds in stochastic mixed-integer programming. *SIAM Journal on Optimization*, 28(2):1312–1336, 2018.

[8] Claus C Carøe and Rüdiger Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45, 1999.

[9] P-L Carpentier, Michel Gendreau, and Fabian Bastin. Long-term management of a hydroelectric multireservoir system under uncertainty using the progressive hedging algorithm. *Water Resources Research*, 49(5):2812–2827, 2013.

[10] Teodor Gabriel Crainic, Xiaorui Fu, Michel Gendreau, Walter Rei, and Stein W Wallace. Progressive hedging-based metaheuristics for stochastic network design. *Networks*, 58(2):114–124, 2011.

[11] Teodor Gabriel Crainic, Mike Hewitt, and Walter Rei. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers & Operations Research*, 43:90–99, 2014.

[12] Teodor Gabriel Crainic, Luca Gobbato, Guido Perboli, and Walter Rei. Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging meta-heuristic. *European Journal of Operational Research*, 253(2):404–417, 2016.

[13] Marcelo LL dos Santos, Edson Luiz da Silva, Erlon Cristian Finardi, and Raphael EC Gonçalves. Practical aspects in solving the medium-term operation planning problem of hydrothermal power systems by using the progressive hedging method. *International Journal of Electrical Power & Energy Systems*, 31(9):546–552, 2009.

[14] Laureano F Escudero, M Araceli Garín, Gloria Pérez, and Aitziber Unzueta. Scenario cluster decomposition of the lagrangian dual in two-stage stochastic mixed 0–1 optimization. *Computers & Operations Research*, 40(1):362–377, 2013.

[15] Laureano F Escudero, M Araceli Garín, Juan F Monge, and Aitziber Unzueta. Some matheuristic algorithms for multistage stochastic optimization models with endogenous un-

certainty and risk management. *European Journal of Operational Research*, 285(3):988–1001, 2020.

[16] Dinakar Gade, Gabriel Hackebeil, Sarah M Ryan, Jean-Paul Watson, Roger J-B Wets, and David L Woodruff. Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. *Mathematical Programming*, 157(1):47–67, 2016.

[17] Serhat Gul, Brian T Denton, and John W Fowler. A progressive hedging approach for surgery planning under uncertainty. *INFORMS Journal on Computing*, 27(4):755–772, 2015.

[18] Ge Guo, Gabriel Hackebeil, Sarah M Ryan, Jean-Paul Watson, and David L Woodruff. Integration of progressive hedging and dual decomposition in stochastic integer programs. *Operations Research Letters*, 43(3):311–316, 2015.

[19] Thorkell Helgason and Stein W Wallace. Approximate scenario solutions in the progressive hedging algorithm. *annals of Operations Research*, 31(1):425–444, 1991.

[20] Lars M Hvattum, Arne Løkketangen, and Gilbert Laporte. Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40 (4):421–438, 2006.

[21] Lars Magnus Hvattum and Arne Løkketangen. Using scenario trees and progressive hedging for stochastic inventory routing problems. *Journal of Heuristics*, 15(6):527, 2009.

[22] Mahdi Jalilvand, Mahdi Bashiri, and Erfaneh Nikzad. An effective progressive hedging algorithm for the two-layers time window assignment vehicle routing problem in a stochastic environment. *Expert Systems with Applications*, 165:113877, 2021.

[23] Cong Han Lim, Jeffrey T Linderoth, James R Luedtke, and Stephen J Wright. Parallelizing subgradient methods for the lagrangian dual in stochastic mixed-integer programming. *Informs Journal on Optimization*, 3(1):1–22, 2021.

[24] Arne Løkketangen and David L Woodruff. Progressive hedging and tabu search applied to mixed integer (0, 1) multistage stochastic programming. *Journal of Heuristics*, 2(2):111–128, 1996.

[25] John M Mulvey and Hercules Vladimirou. Applying the progressive hedging algorithm to stochastic generalized networks. *Annals of Operations Research*, 31(1):399–424, 1991.

[26] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[27] Lewis Ntaimo and Suvrajeet Sen. The million-variable "march" for stochastic combinatorial optimization. *Journal of Global Optimization*, 32(3):385–400, 2005.

[28] Boris Teodorovich Polyak. A general method for solving extremal problems. In *Doklady Akademii Nauk*, volume 174, pages 33–36. Russian Academy of Sciences, 1967.

[29] R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

[30] R Tyrrell Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

[31] Ninja Soeffker, Marlin W Ulmer, and Dirk C Mattfeld. Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research*, 2021.

[32] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[33] Yongjia Song, Marlin W Ulmer, Barrett W Thomas, and Stein W Wallace. Building trust in home services—stochastic team-orienteering with consistency constraints. *Transportation Science*, 54(3):823–838, 2020.

[34] Fernando Badilla Veliz, Jean-Paul Watson, Andres Weintraub, Roger J-B Wets, and David L Woodruff. Stochastic optimization models in forest planning: a progressive hedging solution approach. *Annals of Operations Research*, 232(1):259–274, 2015.

[35] Stacy A Voccia, Ann Melissa Campbell, and Barrett W Thomas. The same-day delivery problem for online purchases. *Transportation Science*, 53(1):167–184, 2019.

[36] Jean-Paul Watson and David L Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8 (4):355–370, 2011.

A Appendix

A.1 Mathematical Proofs

Proof of Proposition 1. Note that the vector x^* satisfies the implementable condition (*i.e.*, it satisfies the non-anticipativity constraints). Indeed, for contradiction suppose for some $k_1, k_2 \in [K]$ $\tilde{x}_{k_1} \neq \tilde{x}_{k_2}$ with $(\tilde{x}_1, \ldots, \tilde{x}_K) \in \operatorname{argmin} G(x^*, \Lambda)$. W.l.o.g we assume that $\|\tilde{x}_{k_1}\|^2 \geq \|\tilde{x}_{k_2}\|^2$, so $\|\tilde{x}_{k_1}\|^2 - \tilde{x}_{k_1}^\intercal \tilde{x}_{k_2} > 0$. Consider the vector $D = (d_1, \ldots, d_K)$ with $d_{k_1} = \frac{\tilde{x}_{k_1}}{p_{k_1}}$ and $d_{k_2} = -\frac{\tilde{x}_{k_1}}{p_{k_2}}$, and $d_k = 0$ for all $k \neq k_1$ and $k \neq k_2$. It holds $D \in \mathcal{D}$.

Note that $\Lambda^* + \gamma D \in \mathcal{D}$, for any $\gamma \geq 0$, so $(x^*, \Lambda^* + \gamma D)$ is a feasible solution for G, then,

$$\begin{aligned} G(x^*, \Lambda^* + \gamma D) &= \sum_{k \in [K]} p_k \left(F_k^c(\tilde{x}_k) + (\lambda_k + \gamma d_k)^\intercal \tilde{x}_k \right), \\ &= \sum_{k \in [K]} p_k \left(F_k^c(\tilde{x}_k) + \lambda_k^{*\intercal} \tilde{x}_k + \gamma d_k^\intercal \tilde{x}_k \right), \\ &= G(x^*, \Lambda^*) + \gamma(\|\tilde{x}_{k_1}\|^2 - \tilde{x}_{k_1}^\intercal \tilde{x}_{k_2}). \end{aligned}$$

Making $\gamma \to \infty$ we have that $G(x^*, \Lambda^* + \gamma D) \to \infty$ contradicting the fact (x^*, Λ^*) is a saddle point.

Thus, we can write the function G evaluated at (x^*, Λ), with $\Lambda \in \mathcal{D}$, as follows,

$$G(x^*, \Lambda) = \left\{ \sum_{k \in [K]} p_k F_k^c(x_k) \middle| x_k = x^*, k \in [K] \right\} = \sum_{k \in [K]} p_k F_k^c(x^*) = L^c(X^*, \Lambda).$$

For all $(\bar{x}, \Lambda) \in \operatorname{conv}(\mathcal{X}) \times \mathcal{D}$ we have the following equivalence,

$$\begin{aligned} G(\bar{x}, \Lambda) &= \min_{x \in \operatorname{conv}(\mathcal{X})^K} \left\{ \sum_{k \in [K]} p_k \left(F_k^c(x_k) + \lambda_k^\intercal x_k \right) \middle| \sum_{k \in [K]} p_k x_k = \bar{x} \right\}, \\ &\leq \sum_{k \in [K]} p_k \left(F_k^c(x_k) + \lambda_k^\intercal x_k \right), \qquad \forall X \in \mathcal{S}(\bar{x}), \end{aligned}$$

so we have,

$$\begin{aligned} & G(x^*, \Lambda^*) \leq G(\bar{x}, \Lambda^*), && \forall \bar{x} \in \operatorname{conv}(\mathcal{X}), \\ \implies \quad & G(x^*, \Lambda^*) \leq \sum_{k \in [K]} p_k \left(F_k^c(x_k) + \lambda_k^\intercal x_k \right), && \forall X \in \mathcal{S}(\bar{x}), \forall \bar{x} \in \operatorname{conv}(\mathcal{X}), \\ \implies \quad & G(x^*, \Lambda^*) \leq L^c(X, \Lambda^*), && \forall X \in \operatorname{conv}(\mathcal{X})^K. \end{aligned}$$

Finally, since (x^*, Λ^*) is a saddle point of G, we have,

$$\begin{aligned} G(x^*, \Lambda^*) &\leq G(\bar{x}, \Lambda^*), && \forall \bar{x} \in \operatorname{conv}(\mathcal{X}), \\ G(x^*, \Lambda^*) &\geq G(x^*, \Lambda), && \forall \Lambda \in \mathcal{D}, \end{aligned}$$

so using the previous inequalities, we have that (X^*, Λ^*) is also a saddle point for L^c. □

Proof of Proposition 2. We first prove that the function $G(\bar{x}, \Lambda)$ is convex. Take any $\bar{x}_1, \bar{x}_2 \in \operatorname{conv}(\mathcal{X})$, $\Lambda \in \mathcal{D}$ and $\alpha \in [0,1]$. Let $x^1 \in \operatorname{argmin}_{x \in \mathcal{S}(\bar{x}_1)}\{G(\bar{x}_1, \Lambda)\}$ and let $x^2 \in \operatorname{argmin}_{x \in \mathcal{S}(\bar{x}_2)}\{G(\bar{x}_2, \Lambda)\}$. Using that the function $F_k^c(x)$ is convex in $x \in \operatorname{conv}(\mathcal{X})$, we have,

$$\begin{aligned}
G(\alpha\bar{x}_1 + (1-\alpha)\bar{x}_2, \Lambda) &= \min\left\{\sum_{k\in[K]} p_k\left(F_k^c(x_k) + \lambda_k^\intercal x_k\right) \middle| x \in \mathcal{S}(\alpha\bar{x}_1 + (1-\alpha)\bar{x}_2)\right\}, \\
&\leq \sum_{k\in[K]} p_k\left(F_k^c(\alpha x_k^1 + (1-\alpha)x_k^2) + \lambda_k^\intercal(\alpha x_k^1 + (1-\alpha)x_k^2)\right), \\
&\leq \sum_{k\in[K]} p_k\left(\alpha F_k^c(x_k^1) + (1-\alpha)F_k^c(x_k^2) + \alpha\lambda_k^\intercal x_k^1 + (1-\alpha)\lambda_k^\intercal x_k^2\right), \\
&= \alpha G(\bar{x}_1, \Lambda) + (1-\alpha)G(\bar{x}_2, \Lambda).
\end{aligned}$$

Similarly, in order to show that the function $G(\bar{x}, \Lambda)$ is concave in $\Lambda \in \mathcal{D}$, we take any $\Lambda_1, \Lambda_2 \in \mathcal{D}$, $\bar{x} \in \operatorname{conv}(\mathcal{X})$ and $\alpha \in [0,1]$. Let $x^* \in \operatorname{argmin}_{x \in \mathcal{S}(\bar{x})} G(\bar{x}, \alpha\Lambda_1 + (1-\alpha)\Lambda_2)$, we have,

$$\begin{aligned}
G(\bar{x}, \alpha\Lambda_1 + (1-\alpha)\Lambda_2) &= \sum_{k\in[K]} p_k\left(F_k^c(x_k^*) + (\alpha\lambda_{1k} + (1-\alpha)\lambda_{2k})^\intercal x_k^*\right), \\
&= \sum_{k\in[K]} p_k\left(\alpha(F_k^c(x_k^*) + \lambda_{1k}^\intercal x_k^*) + (1-\alpha)(F_k^c(x_k^*) + \lambda_{2k}^\intercal x_k^*)\right), \\
&\geq \alpha G(\bar{x}, \Lambda_1) + (1-\alpha)G(\bar{x}, \Lambda_2).
\end{aligned}$$

□

Proof of Proposition 6. Take any optimal solution x^* for (4). We have,

$$\begin{aligned}
\frac{1}{K}\sum_{k\in[K]} \underline{L}_k(\lambda_k) &= \frac{1}{K}\sum_{k\in[K]} \min_{x_k\in\mathcal{X}}\left\{F_k(x_k) + \lambda_k^\intercal x_k\right\}, \\
&\leq \frac{1}{K}\sum_{k\in[K]} F_k(x^*) + \lambda_k^\intercal x^*, \\
&= \frac{1}{K}\sum_{k\in[K]} F_k(x^*).
\end{aligned}$$

□

Proof of Lemma 1. Note that, for any $\Lambda \in \mathcal{D}$, we have,

$$\begin{aligned}
\underline{L}(\Lambda) &= \min_{x_k \in \mathcal{X}} \frac{1}{K} \sum_{k\in[K]} \left(F_k(x_k) + \lambda_k^\intercal x_k\right), \\
&= \min_{x_k \in \mathcal{X}} \frac{1}{K} \sum_{k\in[K]} \left(F_k(x_k) + \lambda_k^{0\intercal} x_k + \lambda_k^\intercal x_k - \lambda_k^{0\intercal} x_k\right), \\
&\leq \min_{x_k \in \mathcal{X}} \frac{1}{K} \sum_{k\in[K]} \left(F_k(x_k) + \lambda_k^{0\intercal} x_k\right) + \frac{1}{K} \sum_{k\in[K]} x_k^{0\intercal}(\lambda_k - \lambda_k^0), \\
&= L(\Lambda^0) + \sum_{k\in[K]} g_k(\Lambda^0)^\intercal(\lambda_k - \lambda_k^0), \\
&= \underline{L}(\Lambda^0) + g(\Lambda^0)^\intercal(\Lambda - \Lambda^0).
\end{aligned}$$

□

Proof of Theorem 2. We start by computing the distance between Λ^ν and Λ^*, which allows us to find an upper bound for $L^* - L^{\text{best}(\nu)}$, with $L^* = \underline{L}(\Lambda^*)$.

At any iteration $\nu \geq 0$, we have,

$$\begin{aligned}
\frac{1}{K} \sum_{k\in[K]} \|\lambda_k^{\nu+1} - \lambda_k^*\|^2 &= \frac{1}{K} \sum_{k\in[K]} \|\lambda_k^\nu + \alpha^\nu(x_k^\nu - \hat{x}^\nu) - \lambda_k^*\|^2 \\
&= \frac{1}{K} \sum_{k\in[K]} \|\lambda_k^\nu - \lambda_k^*\|^2 + \frac{1}{K} \sum_{k\in[K]} 2\alpha^\nu (x_k^\nu - \hat{x}^\nu)^\intercal(\lambda_k^\nu - \lambda_k^*) + \frac{1}{K} \sum_{k\in[K]} \alpha^{\nu 2} \|x_k^\nu - \hat{x}^\nu\|^2, \\
&\leq \frac{1}{K} \sum_{k\in[K]} \|\lambda_k^\nu - \lambda_k^*\|^2 + 2\alpha^\nu(\underline{L}(\Lambda^\nu) - L^*) + \frac{1}{K} \sum_{k\in[K]} \alpha^{\nu 2} \|x_k^\nu - \hat{x}^\nu\|^2,
\end{aligned}$$

Summing inequalities up for $\nu = 0, \dots, \eta$, we have,

$$\begin{aligned}
\frac{1}{K} \sum_{k\in[K]} \|\lambda_k^{\eta+1} - \lambda_k^*\|^2 &\leq \frac{1}{K} \sum_{k\in[K]} \|\lambda_k^*\|^2 + 2\sum_{\nu=0}^{\eta} \alpha^\nu(\underline{L}(\Lambda^\nu) - L^*) + \frac{1}{K} \sum_{k\in[K]} \sum_{\nu=0}^{\eta} \alpha^{\nu 2} \|x_k^\nu - \hat{x}^\nu\|^2, \\
&\leq \frac{1}{K} \sum_{k\in[K]} \|\lambda_k^*\|^2 + 2\sum_{\nu=0}^{\eta} \alpha^\nu(\underline{L}(\Lambda^\nu) - L^*) + N\sum_{\nu=0}^{\eta} \alpha^{\nu 2}, \\
&\leq \frac{1}{K} \sum_{k\in[K]} \|\lambda_k^*\|^2 + 2(L^{\text{best}(\eta)} - L^*)\sum_{\nu=0}^{\eta} \alpha^\nu + N\sum_{\nu=0}^{\eta} \alpha^{\nu 2},
\end{aligned}$$

It follows,

$$0 \leq L^* - L^{\text{best}(\eta)} \leq \frac{\frac{1}{K}\sum_{k\in[K]}\|\lambda_k^*\|^2 + N\sum_{\nu=0}^{\eta}\alpha^{\nu 2}}{2\sum_{\nu=0}^{\eta}\alpha^\nu} \xrightarrow{\eta\to\infty} 0,$$

and since for all $\Lambda \in \mathcal{D}$, $\underline{L}(\Lambda) \leq L^*$, we conclude.

□

Proof of Proposition 7. For proving this result we just need note that solving a mixed-integer problem is equivalent to solve it on the convex hull of the variables set and that for convex sets we can swap the min for the max.

$$\begin{aligned}
\underline{L}(\Lambda^*) &= \max_{\Lambda\in\mathcal{D}} \min_{x_k\in\mathcal{X}} \frac{1}{K}\sum_{k\in[K]} F(x_k) + \lambda_k^\intercal x_k,\\
&= \max_{\Lambda\in\mathcal{D}} \min_{x_k\in\text{conv}(\mathcal{X})} \frac{1}{K}\sum_{k\in[K]} F(x_k) + \lambda_k^\intercal x_k,\\
&= \min_{x_k\in\text{conv}(\mathcal{X})} \max_{\Lambda\in\mathcal{D}} \frac{1}{K}\sum_{k\in[K]} F(x_k) + \lambda_k^\intercal x_k,\\
&= \min\left\{ \frac{1}{K}\sum_{k\in[K]} F(x_k) \,\middle|\, \begin{array}{ll} x_k = \frac{1}{K}\sum_{i\in[K]} x_i, & k\in[K],\\ x_k\in\text{conv}(\mathcal{X}), & k\in[K]. \end{array} \right\}.
\end{aligned}$$

□

Proof of Proposition 8. Consider any $p_k > 0$ rational, $k\in[K]$, with $\sum_{k\in[K]} p_k = 1$, and take K' the minimum integer such that $p_kK'\in\mathbb{N}$. Note that the rational assumption of scenarios probabilities guarantees this K' exists. For each $k\in[K]$, we create identical $E_k := p_kK'$ copies of scenario k. We now use an induction argument for showing the result.

At iteration $\nu = 0$ for the SA and for each scenario $j\in E_k$, $k\in[K]$, we solve the subproblem and get an optimal solution $x_j^1 \in \text{argmin}_{x_j\in\mathcal{X}} F_j(x_j)$, and since the subproblem is the same for all $j\in E_k$, we can pick the same solution for each j. We have that, for all $k_1,k_2\in E_k$, $x_{k_1}^1 = x_{k_2}^1$, for all $k\in[K]$.

For the implementable solution we have,

$$\hat{x}^1 = \frac{1}{K'}\sum_{k\in[K']} x_k^1 = \frac{1}{K'}\sum_{k\in[K]}\sum_{j\in E_k} x_j^1 = \frac{1}{K'}\sum_{k\in[K]} |E_k| x_k^1 = \sum_{k\in[K]} p_k x_k^1,$$

and for each λ_j^1,

$$\lambda_j^1 = \lambda_j^0 + \alpha^1(x_j^1 - \hat{x}^1) = \alpha^1(x_j^1 - \hat{x}^1),$$

so $\lambda_{k_1}^1 = \lambda_{k_2}^1$, for all $k_1,k_2\in E_k$.

At iteration $\nu\geq 0$, using a similar argument, we show that the subproblem copies, $j\in E_k$, $k\in[K]$, have the same dual multipliers $\lambda_j^{\nu+1}$, so we can pick the same optimal solution for all of them. Indeed, we solve $x_j^{\nu+1}\in\text{argmin}_{x_j\in\mathcal{X}} F_j(x_j) + \lambda_j^{\nu\intercal}x_j$, so for all $k_1,k_2\in E_k$, $x_{k_1}^{\nu+1} = x_{k_2}^{\nu+1}$. The implementable solution $\hat{x}^{\nu+1}$ holds that $\hat{x}^{\nu+1} = \sum_{k\in[K]} p_k x_k^{\nu+1}$ and the new dual multipliers,

$$\lambda_j^{\nu+1} = \lambda_j^\nu + \alpha^{\nu+1}(x_j^{\nu+1} - \hat{x}^{\nu+1}),$$

so for all $k_1,k_2\in E_k$, $\lambda_{k_1}^{\nu+1} = \lambda_{k_2}^{\nu+1}$.

We conclude the Lagrangian function of the SA converges an optimal $\underline{L}(\Lambda^*)$ for any formulation with $p_k > 0$ rational, $k\in[K]$, and $\sum_{k\in[K]} p_k = 1$. □

A.2 SSLP Formulation

The SSLP considers a I number of potential locations for servers, a J number of clients and a S number of zones. Each server can serve up to u amount of resource to clients and the location cost it in $j \in [J]$ is c_j. The revenue generated by serving client $j \in [J]$ from location $i \in [I]$ is r_{ij} and the total demand for the pair location-client (i, j) is d_{ij}. If from a server at j more units of resource has to be supplied, a shortage cost q_i (penalty) is paid. A maximum number of v servers can be installed. The original formulation of Ntaimo and Sen [27] considers zones, geographical areas that have locations assigned, however, in the benchmark instances there are no zones. The locations for zone $s \in [S]$ are indexed with I_s and a minimum of ℓ_s servers has to be installed in s.

The formulation considers K scenarios that represent possible clients outcomes. The parameter γ_j^k, $j \in [J]$ and $k \in [K]$, is equal to one if the customer j is present in scenario k; zero, otherwise.

Let x_i be a binary variable that is equal to one if a server is installed in $i \in [I]$; zero, otherwise; and let y_{ij}^k be a binary variable that is equal to one if location $i \in [I]$ serves client $j \in [J]$ in scenario $k \in [K]$; zero, otherwise. The variable z_i^k corresponds to the demand overflow due to server capacity limitations at $i \in [I]$ in scenario $k \in [K]$.

$$
\begin{aligned}
\min \quad & \sum_{i\in[I]} c_i x_i + \frac{1}{K}\sum_{k\in[K]}\left(q_i z_i^k - \sum_{j\in[J]} r_{ij} y_{ij}^k\right), && && \text{(30a)}\\
\text{s.t.} \quad & \sum_{i\in[I]} x_i \le v, && && \text{(30b)}\\
& \sum_{j\in[J]} d_{ij} y_{ij}^k \le u x_i + z_i^k, && i \in [I], k \in [K], && \text{(30c)}\\
& \sum_{i\in[I]} y_{ij}^k = \gamma_j^k, && j \in [J], k \in [K], && \text{(30d)}\\
& \sum_{i\in I_s} x_i \ge \ell_s, && s \in [S], && \text{(30e)}\\
& x_i \in \{0,1\}, && i \in [I], &&\\
& y_{ij}^k \in \{0,1\}, && i \in [I], j \in [J], k \in [K], &&\\
& z_i^k \ge 0, && i \in [I], k \in [K]. &&
\end{aligned}
$$

The objective function in (30a) is to minimize the total expected cost of installing servers and satisfying client demands. Constraint (30b) enforces that no more than v servers are installed. Constraints (30c) impose the client demands are served, while constraints (30d) ensure all realized clients in a given scenario have a server associated. Finally, constraints (30e) impose the minimum number of servers for each zone.

A.3 TSSATOP Formulation

A two-stage formulation of this problem is provided by Song et al. [33]. Subscription customers are indexed by V_S and all on-demand customers (each potential customer requiring the service on any day) are represented by V_D, with $N = |V_S \cup V_D|$. The depot is indexed by 0 and $N+1$. Sampled scenarios represent realizations of V_D. We consider K the number of scenarios and for each of them, a known set V_D is associated, V_D^k, $k \in [K]$. Each scenario has the same realization probability of $\frac{1}{K}$. The set of all locations for a given scenario k is $V^k = V_s \cup V_D^k \cup \{0, N+1\}$.

To serve customers, a fleet of M vehicles is available. Each vehicle must start at the depot and return before a time t_{max}. Not all vehicles have to be used during a day but each of them can be assigned to at most one route. Each $i \in V_D \cup V_s$ has a fixed time window $[\ell_i, u_i]$ and a location associated, so the travel cost c_{ij} and time t_{ij} between $i \in V_D \cup V_s$ and any $j \in V_D \cup V_s$ is known. Travel times and cost are assumed to be symmetric and positive. Also, for any customer a service time $\tau_i > 0$ is known. It is also assumed the time windows are hard, so a vehicle that arrives early must wait until ℓ_i and it cannot arrive after u_i. The subscription customers must be served by the same driver and their locations and time window are known in advance. Each served on-demand customer $i \in V_D$ earns a reward $\rho_i > 0$ and it is not required to server all on-demand customers.

The binary variable that assigns subscription customers to vehicles is x, which corresponds to the first-stage variable of the formulation. For a customer $i \in V_S$ and a vehicle $m \in [M]$, x_{im} is equal to one if the customer i is assigned to m, zero otherwise. For the second-stage, we consider the binary variable y_{ijm}^k, which is equal to one if the vehicle m goes from location $i \in V^k$ to $j \in V^k$ in scenario k; zero otherwise; the binary variable z_i^k which is equal to one if the location $i \in V^k$ is visited in scenario $k \in [K]$; zero otherwise; and the non-negative variable t_i^k that corresponds to the arrival time to location $i \in V^k$ in scenario k. The zero value for t_i^k means there is no arrival at location i.

$$
\begin{aligned}
\min \quad & \sum_{k\in[K]} \frac{1}{K}\left(\sum_{m\in[M]}\sum_{i\in V^k, j\in V^k} c_{ij} y^k_{ijm} - \sum_{i\in V^k_D} \rho_i z^k_i\right), && && \text{(31a)}\\
\text{s.t.} \quad & \sum_{m\in[M]} x_{im} = 1, && i\in V_S && \text{(31b)}\\
& \sum_{j\in V^k} y^k_{ijm} = x_{im}, && i\in V_S, m\in[M], k\in[K] && \text{(31c)}\\
& \sum_{j\in V^k} y^k_{jim} = \sum_{j\in V^k} y^k_{ijm}, && i\in V^k\setminus\{0,N+1\}, m\in[M], k\in[K] && \text{(31d)}\\
& \sum_{i\in V^k} y_{0jm} \le 1 && m\in[M], k\in[K] && \text{(31e)}\\
& z^k_i = \sum_{m\in[M]}\sum_{j\in V^k} y^k_{ijm}, && i\in V^k_D, k\in[K], && \text{(31f)}\\
& t^k_j \ge t^k_i + (t_{ij}+\tau_i+M)\sum_{m\in[M]} y^k_{ijm} - M, && k\in[K], i\in V^k\setminus\{N+1\}, j\in V^k\setminus\{0\}, && \text{(31g)}\\
& \ell_i \le t^k_i \le u_i, && i\in V^k, k\in[K], && \text{(31h)}\\
& x_{im}\in\{0,1\}, && i\in V_S, m\in[M], &&\\
& y^k_{ijm}\in\{0,1\}, && i\in V^k, j\in V^k, m\in[M], k\in[K], &&\\
& z^k_i\in\{0,1\}, && i\in V^k_D, k\in[K]. &&
\end{aligned}
$$

Constrains (31b) impose all subscription customers are served and constrains (31c) forces that the assignment of vehicles to subscription customers is the same for all scenarios. The vehicle flow conservation is enforced by constrains (31d). The z variable, that represent if a customer is served in a given scenario, is defined by constrains (31f). Visiting time constrains are represented by (31g) and the time windows conditions are imposed by constrains (31h). The objective function (31a) minimizes the expected routing cost minus the expected revenue gained by serving on-demand customers.

A.4 Results Details for TSSATOP

Instance	Best	DynamicMult(2.41, 1.50)			
		Lower	Found	Runtime (s)	Iterations
C101S10	59.61	53.67	59.92	236.71	6
C101S15	125.12	121.58	125.12	89.51	5
C101S20	176.46	171.44	176.46	824.00	3
C102S10	90.20	87.03	90.20	30.29	3
C102S15	111.22	108.57	111.22	206.03	2
C102S20	181.36	178.93	181.36	117.43	3
C103S10	74.41	72.09	74.41	50.10	4
C103S15	110.13	105.84	112.09	234.09	5
C103S20	176.46	171.44	176.46	56.88	3
C104S10	51.63	48.23	51.63	41.42	3
C104S15	102.55	97.59	102.55	561.14	5
C104S20	181.87	180.23	181.87	72.02	2
C105S10	59.03	55.47	59.03	55.78	5
C105S15	119.48	115.66	119.48	825.26	5
C105S20	135.14	133.80	135.14	385.04	2
R101S10	55.03	51.59	55.03	64.20	4
R101S15	60.32	60.32	60.32	41.48	1
R101S20	136.51	134.69	136.51	250.83	3
R102S10	76.94	70.59	77.26	40.37	7
R102S15	67.84	65.03	67.84	2087.52	4
R102S20	136.14	133.00	136.14	806.05	3
R103S10	70.16	66.43	70.46	1159.91	4
R103S15	88.19	86.15	88.19	1310.64	4
R103S20	143.60	142.53	143.60	1076.69	3
R104S10	57.22	50.94	57.22	7134.18	5
R104S15	97.12	93.78	97.12	934.45	3
R104S20	140.31	137.39	141.24	1504.41	3
R105S10	45.97	41.78	47.56	6252.09	4
R105S15	95.68	92.87	95.68	1627.33	4
R105S20	145.49	140.67	145.49	361.93	2

Table 6: Result details for instances TSSATOP. The column "Best" contains the best solution found by any of the executed algorithms. The remaining columns have details for the DPHA configuration with the best performance.