Journal of Heuristics (2021) 27:459–496
https://doi.org/10.1007/s10732-021-09466-0

Weighted proximity search

Filipe Rodrigues[1] · **Agostinho Agra**[2] · **Lars Magnus Hvattum**[3] · **Cristina Requejo**[2]

Received: 26 June 2020 / Revised: 28 December 2020 / Accepted: 6 January 2021 /
Published online: 30 January 2021

Abstract
Proximity search is an iterative method to solve complex mathematical programming problems. At each iteration, the objective function of the problem at hand is replaced by the Hamming distance function to a given solution, and a cutoff constraint is added to impose that any new obtained solution improves the objective function value. A mixed integer programming solver is used to find a feasible solution to this modified problem, yielding an improved solution to the original problem. This paper introduces the concept of weighted Hamming distance that allows to design a new method called weighted proximity search. In this new distance function, low weights are associated with the variables whose value in the current solution is promising to change in order to find an improved solution, while high weights are assigned to variables that are expected to remain unchanged. The weights help to distinguish between alternative solutions in the neighborhood of the current solution, and provide guidance to the solver when trying to locate an improved solution. Several strategies to determine weights are presented, including both static and dynamic strategies. The proposed weighted proximity search is compared with the classic proximity search on instances from three optimization problems: the p-median problem, the set covering problem, and the stochastic lot-sizing problem. The obtained results show that a suitable choice of weights allows the weighted proximity search to obtain better solutions, for 75% of the cases, than the ones obtained by using proximity search and for 96% of the cases the solutions are better than the ones obtained by running a commercial solver with a time limit.

Keywords Mixed integer programming · Matheuristic · Local search

1 Introduction

Heuristic strategies are of vital importance to obtain good quality solutions for problems that cannot be solved to optimality within a reasonable time. Several heuristic strategies exploring different lines of thought have been proposed in the literature

Extended author information available on the last page of the article

(Boussaïd et al. 2013; Hvattum and Esbensen 2011). One of those lines consists of searching for better solutions by exploring neighborhoods of given reference solutions. Some examples are large-neighborhood search (Shaw 1998), iterated local search (Agra et al. 2016, 2018a; Fischetti and Lodi 2003), relaxation induced neighborhood search (Danna et al. 2005), feasibility pump (Fischetti et al. 2005), proximity search (Fischetti and Monaci 2014), and other methods based on a large-neighborhood search framework such as Rothberg (2007).

Proximity search was proposed in 2014 by Fischetti and Monaci (2014) as a promising approach for solving 0–1 mixed-integer convex programs. The preliminary tests were performed on set covering, network design, and machine learning classification instances from the literature. Despite being a very promising heuristic, revealing a good performance when employed, the applications of proximity search in the literature are still scarce. A heuristic framework combining ad-hoc heuristics and mixed integer programming (MIP) was proposed in Fischetti and Monaci (2016) to solve the wind farm design problem. In such heuristics, the proximity search is used to refine the current best solution iteratively. Fischetti et al. (2016) proposed a MIP branch-and-cut framework for solving the uncapacitated facility location problem, in which the solutions for a master problem are generated by a local branching heuristic and then improved by the proximity search heuristic.

To the best of our knowledge, only two applications of proximity search to problems involving uncertainty can be found in the literature. Boland et al. (2016) introduced a variant of proximity search that drives a Benders decomposition, specially designed to handle MIP problems arising in stochastic programming. To evaluate the performance of the proposed heuristic, benchmark instances of three different stochastic problems (capacitated facility location, network interdiction, and fixed charge multi-commodity network design) were considered. Alvarez-Miranda et al. (2014) proposed a three-phase heuristic algorithm to solve hard instances of the single-commodity robust network design problem. The proximity search is incorporated into the third phase of the algorithm to enhance the solutions obtained in the second phase by a neighborhood search heuristic.

Proximity search improves a given feasible solution by searching for a solution that has a better objective function value, while being close in Hamming distance to the current solution. As the Hamming distance is used in the objective function of the proximity search, the objective function does not provide guidance to distinguish between the quality of equally distant solutions, nor does it incorporate information already available regarding the values of variables. In this paper we propose a new method called weighted proximity search that replaces the objective function by a weighted Hamming distance function, where different coefficients may be assigned to the variables. By using this distance function, we expect to solve each subproblem faster and thus obtain better quality solutions, in particular when weights are based on exploiting problem-dependent structures.

Our main contributions are the following:

1. Propose a new method, called weighted proximity search, based on replacing the Hamming distance function by a weighted Hamming distance function.

2. Propose several strategies to compute the weights used in the weighted Hamming distance function and discuss advantages and disadvantages of each of them.
3. Show the benefits of using the weights to improve the classic proximity search based on extensive computational results on three different problems: the p-median problem, the set covering problem, and the stochastic lot-sizing problem.

The paper is organized as follows. In Sect. 2 we review the proximity search proposed in Fischetti and Monaci (2014) and introduce the new weighted proximity search. Several approaches to determine the weights are discussed in Sect. 3. The optimization problems used to evaluate the proposed weighted proximity search are presented in Sect. 4, while some implementation details are presented in Sect. 5. Computational results are reported in Sect. 6, and concluding remarks are given in Sect. 7.

2 Search procedures

In this section we review the classic proximity search (PS) and introduce the proposed weighted proximity search (WPS). Regarding the notation used, each vector (either of constants or of decision variables) is denoted by $\mathbf{x}$ and x_i denotes the ith component of that vector; when $\mathbf{x}$ is a vector of decision variables, $\bar{\mathbf{x}}$ denotes the value of the decision variables $\mathbf{x}$ in the current iteration of the procedure.

We consider general MIP problems of the form

$$\begin{aligned} \min \quad & f(\mathbf{x}, \mathbf{y}) \\ s.t. \quad & g_i(\mathbf{x}, \mathbf{y}) \leq 0, \qquad i \in I, \\ & \mathbf{x} \in \{0, 1\}^n, \\ & \mathbf{y} \in \mathbb{R}^p \times \mathbb{Z}^q, \end{aligned} \tag{2.1}$$

where I is a set of indices, and f and g_i, $i \in I$, are real functions (linear or not) defined over $\mathbb{R}^{n+p+q}$. The binary decision variables $\mathbf{x}$ could also be part of the decision vector $\mathbf{y}$ (composed of p continuous variables and q integer variables), however, since the discussed methods specifically affect the binary variables $\mathbf{x}$, they are kept separately. We assume that it is mainly due to the presence of the $\mathbf{x}$-variables that the problem is difficult to solve, and when their values are fixed, the values of the $\mathbf{y}$-variables can be quickly determined. This is the case of optimization problems with either few or no integer variables.

2.1 Proximity search

The most common enumeration methods used to solve MIP problems, such as the branch-and-bound and the branch-and-cut methods (Achterberg et al. 2005; Morrison et al. 2016), are based on exploiting the neighborhood of the solution provided by the linear programming (LP) relaxation. As a result, large search trees are generated around the solution of the LP-relaxation. Since feasible solutions are more likely to be

located in the lower part of the tree, finding such solutions can be very time-consuming, specially when the search tree is large. The PS procedure was proposed to overcome some of these problems by searching a neighborhood of an integer solution instead of searching a neighborhood of the LP-relaxation solution. This can be done by replacing the objective function of the problem by the Hamming distance function centered in the current solution.

Define $N := \{1, \ldots, n\}$. The Hamming distance function centered in a current solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is defined as

$$\Delta(\mathbf{x}, \bar{\mathbf{x}}) := \sum_{j \in N | \bar{x}_j = 0} x_j + \sum_{j \in N | \bar{x}_j = 1} (1 - x_j). \tag{2.2}$$

By using the Hamming distance function, the PS can be described as follows. Start with a feasible solution $(\mathbf{x}^0, \mathbf{y}^0)$. Then, at each iteration the cutoff constraint

$$f(\mathbf{x}, \mathbf{y}) \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - \theta, \tag{2.3}$$

(depending on a given cutoff tolerance value $\theta > 0$) is added to the MIP problem and the objective function is replaced by the Hamming distance function defined in expression (2.2). The obtained solution is then used to recenter the Hamming distance function and to define the new cutoff constraint, and the process is repeated until a given stopping criterion is reached. Algorithm 1 describes the PS procedure proposed in Fischetti and Monaci (2014).

Algorithm 1 Proximity Search

1: Let $(\mathbf{x}^0, \mathbf{y}^0)$ be an initial feasible solution. Set $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) := (\mathbf{x}^0, \mathbf{y}^0)$.
2: **repeat**
3: add the cutoff constraint (2.3) to the MIP problem
4: replace $f(\mathbf{x}, \mathbf{y})$ by the Hamming distance function $\Delta(\mathbf{x}, \bar{\mathbf{x}})$
5: run the MIP solver on the modified program until a termination condition is reached
6: **if** a new feasible solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is found **then**
7: refine $\bar{\mathbf{y}}$ by solving the problem $\bar{\mathbf{y}} = \text{argmin}\{f(\mathbf{x}, \mathbf{y}) : g_i(\mathbf{x}, \mathbf{y}) \leq 0, \ \mathbf{x} = \bar{\mathbf{x}}, \ i \in I\}$
8: **end if**
9: update θ (optional)
10: **until** a given stopping criterion is reached

At each iteration of Algorithm 1, two different problems are solved. The first one is solved at Step 5 and is referred to as the modified problem, since its optimal value is not a bound for the original problem (2.1). The second problem is solved at Step 7 and is called the restricted main problem, since its optimal value is an upper bound for the original problem (2.1) given that the binary variables $\mathbf{x}$ are fixed.

If the stopping criterion used at Step 10 is to prove that the current modified problem is infeasible then the cutoff value θ can be regarded as the algorithm tolerance. In particular, when the objective function of the original problem is integer-valued, the optimality of a given solution can be proved by showing that the modified problem

is infeasible for θ lower than or equal to one. When other stopping criteria are used, such as a running time limit, the PS becomes a heuristic procedure.

The PS used exactly as described in Algorithm 1 is known as PS with recentering. However, two other variants of PS were introduced in Fischetti and Monaci (2014):

- *PS without recentering* The Hamming distance function in Step 4 remains centered on the first solution $(\mathbf{x}^0, \mathbf{y}^0)$ considered, i.e., using $\Delta(\mathbf{x}, \mathbf{x}^0)$ instead of $\Delta(\mathbf{x}, \bar{\mathbf{x}})$.
- *PS with incumbent* The cutoff constraint added in Step 3 is soft, in the sense that the feasible solution found in the previous iteration is also feasible for the current iteration but is highly penalized in the objective function. Hence, the cutoff constraint (2.3) is replaced by the soft cutoff constraint

$$f(\mathbf{x}, \mathbf{y}) \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - \theta + u, \tag{2.4}$$

with a continuous slack variable $u \geq 0$, and the Hamming distance function is replaced by

$$\Delta(\mathbf{x}, \bar{\mathbf{x}}) + Mu, \tag{2.5}$$

where M is a sufficient large constant.

2.2 Weighted proximity search

The PS proposed in Fischetti and Monaci (2014) aims to improve a given feasible solution $\bar{\mathbf{x}}$ by exploring its neighborhood, where the closest neighbors are those that minimize the Hamming distance function. Since all the variables x_j, $j \in N$, have the same coefficient (equal to 1) in the Hamming distance function, no additional information is provided to the MIP solver regarding the quality of the current value of the variables. To guide the PS, we introduce a new distance function, called weighted Hamming distance function, relative to a given feasible solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, defined by

$$\Delta_w(\mathbf{x}, \bar{\mathbf{x}}) := \sum_{j \in N | \bar{x}_j = 0} w_j x_j + \sum_{j \in N | \bar{x}_j = 1} w_j (1 - x_j) \tag{2.6}$$

where $w_j \in \mathbb{R}$ represents the weight of variable x_j, $j \in N$. By setting $w_j = 1$ for all $j \in N$, the weighted Hamming distance function reduces to the classic Hamming distance function.

This new distance function intends to inform the MIP solver about which variables are more promising to change in order to find an improved solution. Hence, given a feasible solution, higher weights should be assigned to the variables that are less promising to change, while lower weights should be associated with the variables that are more promising to change (i.e., the ones that one expects to be different in similar solutions with better objective function values).

Moreover, when using an unweighted Hamming distance function, the MIP solver is unable to differentiate between alternative improving solutions equally close to the current solution. If those solutions are of substantially different quality, the PS may require many iterations where only small improvements of the objective function are

observed. With a weighted Hamming distance function, we may be able to differentiate between these alternative improving solutions, thus making the procedure more efficient. The WPS that we propose is described in Algorithm 2.

Algorithm 2 Weighted Proximity Search

Let $(\mathbf{x}^0, \mathbf{y}^0)$ be an initial feasible solution. Set $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) := (\mathbf{x}^0, \mathbf{y}^0)$.
repeat
  add the cutoff constraint (2.3) to the MIP problem
  for each variable x_j, $j \in N$, compute the associated weight w_j
  replace $f(\mathbf{x}, \mathbf{y})$ by the weighted Hamming distance function $\Delta_w(\mathbf{x}, \bar{\mathbf{x}})$
  run the MIP solver on the modified program until a termination condition is reached
  if a new feasible solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is found **then**
    refine $\bar{\mathbf{y}}$ by solving the problem $\bar{\mathbf{y}} = \text{argmin}\{f(\mathbf{x}, \mathbf{y}) : g_i(\mathbf{x}, \mathbf{y}) \leq 0,\ \mathbf{x} = \bar{\mathbf{x}},\ i \in I\}$
  end if
  update θ (optional)
until a given stopping criterion is reached

Algorithm 2 corresponds to the WPS version with recentering. The other two variants of the PS presented in the previous section (PS without recentering and PS with incumbent) can also be obtained for the WPS by making identical modifications.

3 Strategies for determining weights

In this section we explain strategies to determine weights. First, we introduce several approaches to compute weights. Second, we present different processes of discretizing weights such that only a small number of distinct weights appear in the objective function.

3.1 Computing weights

Here we discuss several ways for computing the weights used in the weighted Hamming distance function and classify them into two classes: static and dynamic. In the static class the weights are obtained through calculations performed only once, while in the dynamic class, calculations performed at each iteration of the WPS are required to obtain the weights.

3.1.1 Static weights

In this class, the weights needed at each iteration use calculations performed only once, at the beginning of the algorithm. The classic PS heuristic is in this class since the weights are unchanged at each iteration and are equal to one for all the variables.

When the problem (2.1) cannot be solved to optimality in a reasonable amount of time, heuristic strategies can sometimes be employed to obtain feasible solutions in a short time. Depending on the heuristic used, the quality of these solutions can vary. However, if a *large enough* number of solutions is generated, it may be possible to

identify binary variables that are consistently taking a specific value. Such consistent variables have been exploited in heuristics such as tabu search (Glover and Laguna 1997) and in an adjustable sample average approximation method (Agra et al. 2018b). The consistent variables are the key issue of the first approach presented here to compute the weights.

A common approach to obtain feasible solutions is to use the information resulting from the LP-relaxation. The optimal solution of the LP-relaxation is frequently infeasible for the original problem. Furthermore, rounding each integer variable with a fractional value in the LP-relaxation to the nearest integer can still lead to infeasible solutions for the original problem. Berthold (2014) proposed an heuristic called relaxed enforced neighborhood search, that explores the set of feasible roundings of an optimal solution of a linear or nonlinear relaxation. There are also cases in which the LP-relaxation solution is *close* to the optimal solution of the original problem. In those cases, the LP-relaxation solution can be used to determine the weights of the variables at each iteration. The second and third approaches presented below use the solutions of LP-relaxations to compute the weights.

Consistent Variables Identification (CVI) The idea of this approach is to improve the current solution $\bar{\mathbf{x}}$ using the observed frequency of each binary variable taking the values zero or one in a set of heuristic solutions. Start by generating m feasible solutions of the problem (2.1), denoted by $(\bar{\mathbf{x}}^1, \bar{\mathbf{y}}^1), \ldots, (\bar{\mathbf{x}}^m, \bar{\mathbf{y}}^m)$. Then, for each variable $x_j,\ j \in N$, calculate and store $r_{j,1} := \sum_{i=1}^{m} \bar{x}_j^i$ and $r_{j,0} := m - r_{j,1}$. Afterwards, at each iteration, the initially stored values are used to define the weight associated with variable $x_j,\ j \in N$ as follows:

$$w_j = \begin{cases} r_{j,1}, & \bar{x}_j = 1, \\ r_{j,0}, & \bar{x}_j = 0. \end{cases}$$

For clarification, let us suppose that we have m feasible solutions in which variable $x_k,\ k \in N$, takes the value one more frequently than the value zero. Then, from a heuristic point of view, it may seem more promising to fix the value of x_k to one, rather than to zero. If the value of variable x_k is one in the current solution of the WPS, its weight will be high to encourage the variable to keep its value. If variable x_k is currently zero, its weight will be low to entice the variable to change.

Notice that m is a predefined static value that does not change from iteration to iteration, and all the m solutions required to compute the weights are determined once at the beginning of the search. Therefore, the values $r_{j,1}$ and $r_{j,0}$ are computed only once, making the CVI a static approach.

Linear Relaxation Proximity (LRP) In this approach, the LP-relaxation of the original problem (2.1) is solved first, yielding optimal values $\bar{\mathbf{x}}^{LP}$. At each iteration, the weight associated with the variable x_j is defined as

$$w_j = 1 - |\bar{x}_j - \bar{x}_j^{LP}|.$$

Hence, the highest weights are associated with the variables x_j for which its value in the current solution is closer to its value in the LP-relaxation. This approach resembles the relaxation induced neighborhood search heuristic (Danna et al. 2005), but the variables with the same value in both the current solution and the LP-relaxation are not fixed when using the LRP.

Linear Relaxation Loss (LRL) In this approach $2N + 1$ linear problems are initially solved and their optimal values $f_{jk},\ j \in N, k \in \{0, 1\}$, and $f_{0,0}$ are stored. Each value f_{jk} is the optimal value of the LP-relaxation of the original problem (2.1) with the binary variable x_j fixed to k (while the remaining variables x_i, for $i \in N \backslash \{j\}$, are free) and $f_{0,0}$ is the value of the LP-relaxation of problem (2.1) without any variable fixed.

By using the initially stored values, at each iteration, the weight associated with the variable $x_j,\ j \in N$, is defined as:

$$w_j = \begin{cases} f_{j,0} - f_{0,0}, & \bar{x}_j = 1, \\ f_{j,1} - f_{0,0}, & \bar{x}_j = 0. \end{cases}$$

In this setting, the weight of each variable is related with the loss corresponding to solving the LP-relaxation of the original problem with the value of that variable fixed to the opposite value in the current solution. When $f_{j,1} > f_{j,0}$, it is more promising for x_j to take the value zero than the value one (by assuming that the LP-relaxation solution is *close* to the optimal solution of the original problem). This approach was inspired in the strong branching technique, see Linderoth and Savelsbergh (1999).

Remark 1 The number $(2N + 1)$ of linear problems to solve in this approach can be reduced by taking into account the solution of the LP-relaxation of problem (2.1). If the variable x_j takes the value zero (one) in the LP-relaxation solution then $f_{j,0}$ ($f_{j,1}$) coincides with the value of the LP-relaxation $f_{0,0}$.

Remark 2 For each non-basic variable x_j in the optimal solution of the LP-relaxation, the weight w_j can be approximated by the absolute value of the corresponding reduced cost.

3.1.2 Dynamic weights

In this class, the major effort to compute the weights is taken at each iteration of Algorithm 2, either taking into account the solutions obtained in the previous iterations or by analyzing the impact of changing the value of each variable in the current solution.

Recent Change Indicator (RCI) Let $(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0) := (\mathbf{x}^0, \mathbf{y}^0)$, the initial feasible solution for the original problem, and let $(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k)$ be the solution obtained at iteration $k,\ k \geq 1$. The first iteration is performed with all the weights equal to one. Then, at every iteration

$k,\ k > 1$ the weight of variable x_j is defined as follows

$$w_j = \begin{cases} w^M, & \bar{x}_j^{k-1} \neq \bar{x}_j^{k-2}, \\ 1, & \bar{x}_j^{k-1} = \bar{x}_j^{k-2}, \end{cases}$$

where w^M is a predefined real number greater than one. This approach assumes that a variable whose value changed in iteration $k-1$ should preferably not be changed again in iteration k. The reason is that since the variable was already changed to improve the solution, it is less likely that changing it back to its previous value will help to improve the solution.

Weighted Frequency (WF) Represent by f^0 the objective function value of the initial feasible solution $(\mathbf{x}^0, \mathbf{y}^0)$. This approach starts by performing the first iteration with all the weights equal to one. Then, at every iteration k, with $k > 1$, and for each variable $x_j,\ j \in N$, compute the values

$$p_{j,1} := \sum_{i=1}^{k-1} (f^0 - f^i)\bar{x}_j^i \quad \text{and} \quad p_{j,0} := \sum_{i=1}^{k-1} (f^0 - f^i)(1 - \bar{x}_j^i)$$

where $\bar{x}_j^i$ and f^i are, respectively, the value of the variable x_j in iteration $i,\ 1 \leq i < k$, and the objective function value of the solution obtained in that iteration.

Values $p_{j,0}$ and $p_{j,1}$ can be seen as the frequency that variable x_j takes the values zero and one, respectively, weighted by the objective function value of the solutions obtained in the previous iterations. Hence, smaller coefficients $(f^0 - f^i),\ 1 \leq i < k$, are associated with the solutions obtained in the earlier iterations (with worse objective function values). Those weighted frequencies are expected to be good indicators for whether a variable x_j will take the values zero or one in near-optimal solutions. Thus, the weight associated with variable x_j can be defined as

$$w_j = \begin{cases} p_{j,1}, & \bar{x}_j = 1, \\ p_{j,0}, & \bar{x}_j = 0. \end{cases}$$

Loss/Saving (LS) In this approach, n independent subproblems are solved to obtain the values $f_j^*,\ j \in N$, at each iteration. Each value f_j^* is the optimal value of problem (2.1) with all variables $x_i,\ i \neq j$, fixed to their value in the current solution (that is, $x_i = \bar{x}_i$) and with variable x_j fixed to the opposite value in the current solution (that is, $x_j = 1 - \bar{x}_j$).

Denoting by $\bar{f}$ the objective function value of the current solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, the weight associated with each variable $x_j,\ j \in N$, is defined as follows:

$$w_j = f_j^* - \bar{f}.$$

The difference $f_j^* - \bar{f}$ is the saving or the loss obtained by fixing variable x_j to its opposite value in the current solution while keeping the remaining variables fixed. Hence, variables with lower weights are more promising to change since such change may result in an improvement of the current solution.

Remark 3 The obtained weights can be negative when changing the variable value leads to a lower cost solution.

Remark 4 In some optimization problems, changing the value of a single binary variable in a feasible solution can lead to infeasible solutions. In those cases, the computation of the weights needs to be adjusted. A simple adjustment strategy consists of attributing a high weight to the variables for which the change of its value leads to infeasibility. More sophisticated strategies based on recourse actions can also be employed. That is, when a change on the value of the variable leads to the infeasibility of the current solution, the feasibility can be restored by allowing some additional changes in the remaining variables.

3.1.3 A deeper look at each approach

In this section we discuss the main advantages and shortcomings of each approach proposed to determine the weights, and identify general features of the problems under which each approach becomes more suitable to use.

Consistent Variables Identification The CVI approach starts by creating a pool of feasible solutions for the problem. Therefore, the performance of this approach strongly depends on the heuristic used to obtain such solutions. This approach is problem dependent and it is particularly suitable for optimization problems for which feasible solutions can easily be determined. The main advantage of this approach is that after determining the pool of feasible solutions, the weights are computed very quickly at each iteration. This is an advantage shared by all the static approaches.

Linear Relaxation Proximity The LRP approach only requires solving the LP-relaxation at the beginning of the algorithm, which makes this general approach applicable to any optimization problem. Moreover, the total computational time required for determining the weights is independent of the number of iterations and corresponds to the time associated with solving the LP-relaxation. Since the LRP approach aims to find solutions *close* to the LP-relaxation solution, a better performance is expected for optimization problems in which the optimal solution is close to the solution of the LP-relaxation.

Linear Relaxation Loss The LRL is a general approach that can be applied to a wide range of problems. However, it starts by solving a large number of linear problems (at most two for each binary variable), thus it can be impractical for optimization problems with a large number of binary variables as well as for optimization problems for which the LP-relaxation solution is not quickly determined. As the LRP, the LRL is more suitable for optimization problems in which the LP-relaxation is a good indicator of the optimal solution.

Recent Change Indicator and Weighted Frequency The RCI and the WF are general approaches that can be applied to any optimization problem since they depend on

the solutions determined by the algorithm, keeping memory of previously obtained solutions. Another advantage of these approaches is that the total time required to compute the weights is very short. These two approaches are suitable for problems where Algorithm 2 will perform a large number of iterations.

Loss/Saving The main advantage of the LS approach is the ability to take into account the impact of changing the value of each variable in the current solution. One of the main drawbacks is that when a single binary variable changes its value, the resulting solution can become infeasible. In those cases, the LS approach needs to be adapted to restore the feasibility of the solution and such an adaptation is problem dependent. Moreover, this approach can be very time-consuming when the number of binary variables is large.

Therefore, the LS is more suitable for problems where feasible solutions can be easily computed and in which the number of binary variables $\mathbf{x}$ is not high. Additionally, it is also desirable to consider optimization problems with a small number of integer variables to ensure that the subproblem associated with each binary variable is quickly solved.

3.2 Weights discretization

The weights as described above can take a wide range of values. For the LS approach they can even be negative. Using such weights directly makes each modified problem hard to solve, as observed in preliminary tests reported in Sect. 6.3. To better deal with the weights generated by each approach, we propose to map them into a small set of discrete values. We consider three different discretization schemes:

- *R-Discretization (RD)* In this scheme the weights are discretized into R different values $\{1, \ldots, R\}$, where R is an integer number greater than one defined *a priori*. Denoting by $\{w_1, \ldots, w_n\}$ the initial weights, the corresponding discretized weights $\{w_1^d, \ldots, w_n^d\}$ are computed as follows:

$$w_j^d := (R+1) - \left\lceil R - \frac{(w_j - w^{min})(R-1)}{w^{max} - w^{min}} \right\rceil$$

  where $w^{min} := \min\{w_j : j \in N\}$ and $w^{max} := \max\{w_j : j \in N\}$. This process of discretizing the weights corresponds to converting the initial weights varying in $[w^{min}, w^{max}]$ into modified weights in the interval $[1, R]$ and then round the obtained values.
- *Two-Value System (2-VS)* In this scheme the weights are initially discretized using the R-discretization process. Then, the resulting weights are converted into a two-value system as follows:

$$\bar{w}_j^d = \begin{cases} 1, & \text{if } 1 \le w_j^d \le t_1, \\ R, & \text{if } t_1 < w_j^d \le R, \end{cases}$$

  where t_1 is an integer threshold value between 1 and R defined *a priori*.

- *Three-Value System (3-VS)* Weights are initially discretized using the R-discretization scheme. Then, the resulting weights are converted into a three-value system as follows:

$$\bar{w}_j^d = \begin{cases} 1, & \text{if } 1 \leq w_j^d \leq t_2, \\ \frac{R}{2}, & \text{if } t_2 < w_j^d \leq t_3, \\ R, & \text{if } t_3 < w_j^d \leq R, \end{cases}$$

  where t_2 and t_3 are integer threshold values between 1 and R defined *a priori* such that $t_2 \leq t_3$.
  The three-value system is an extension of the two-value system with an intermediate level. This system assigns the weight one to the variables that are more promising to change, the weight R to the variables that are unpromising to change and an intermediate value to the variables for which there is not a clear evidence indicating whether they should change its value or not.

Remark 5 All the discretization schemes described in this section are tested in the computational section, including the choice of the parameters R, t_1, t_2 and t_3.

4 Optimization problems

In this section we describe the optimization problems considered to evaluate the performance of the proposed WPS. We consider three different problems: the uncapacitated p-median problem, the set covering problem, and the stochastic lot-sizing problem with setups. These problems were chosen since they are structurally different and exhibit different properties, in terms of feasibility and of the nature of the variables.

In the set covering problem there is only a single group of binary variables and all of them are used in the weighted Hamming distance function. In the p-median problem there are two sets of binary variables, but only one of them is used in the weighted Hamming distance function. In the SLS problem with setups, there is a set of binary variables (used in the weighted Hamming distance function), a set of integer variables, and a set of continuous variables. For this problem, any realization of the binary setup variables leads to a feasible solution. This is not the case of the p-median problem and of the set covering problem for which changing the value of one binary variable can render the solution infeasible.

4.1 The uncapacitated *p*-median problem

Given a set $N = \{1, \ldots, n\}$ of facilities and a set $M = \{1, \ldots, m\}$ of customers, the p-median problem consists of selecting p facilities such that the sum of the distances between the customers and the selected facilities is minimized. Let us denote by d_{ji} the distance between facility $j \in N$ and client $i \in M$ and let us consider the following binary decision variables: x_j indicates whether facility $j \in N$ is selected or not, and y_{ji} indicates whether customer $i \in M$ is served by facility $j \in N$. The uncapacitated

p-median problem (Beasley 1985) can be formulated as follows:

$$\begin{aligned}
\min \quad & \sum_{j\in N}\sum_{i\in M} d_{ji}y_{ji} & \\
s.t. \quad & \sum_{j\in N} y_{ji} = 1, & i \in M,\\
& y_{ji} \le x_j, & j \in N, i \in M,\\
& \sum_{j\in N} x_j = p, & \\
& x_j \in \{0,1\}, & j \in N,\\
& y_{ji} \in \{0,1\}, & j \in N, i \in M.
\end{aligned}$$

The objective function minimizes the total distance between the customers and the selected facilities. The first set of constraints ensure that each customer is served by exactly one facility. The second set of constraints imposes that a customer can only be served by a facility if that facility is selected, while the third constraint ensures that exactly p facilities must be selected.

4.2 The set covering problem

Given a set $M = \{1, \ldots, m\}$ of customers and a set $N = \{1, \ldots, n\}$ of services, where each service has an associated cost, the set covering problem (Chvatal 1979) consists of finding the set of services with the minimum cost that covers all the customers.

Let us consider an $m \times n$ matrix A, where each entry a_{ji} is one if customer $j \in M$ is covered by service $i \in N$ and zero otherwise. Denoting by c_i the cost associated with the service i and considering the binary variables x_i indicating whether service $i \in N$ is chosen or not, the set covering problem is formulated as follows:

$$\begin{aligned}
\min \quad & \sum_{i\in N} c_i x_i & \\
s.t. \quad & \sum_{i\in N} a_{ji} x_i \ge 1, & j \in M,\\
& x_j \in \{0,1\}, & j \in N.
\end{aligned}$$

The objective function minimizes the total cost associated with the choice of the services while the constraints ensure that all the customers are covered.

4.3 The stochastic lot-sizing problem with setups

The stochastic lot-sizing (SLS) problem (Rodrigues et al. 2020) is defined over a finite set of n time periods, $N = \{1, \ldots, n\}$. We assume that the demand in one period can either be met in that period (using production of that period or using stocks from

production in previous periods), or satisfied with delay in a later period. The demand satisfied with delay in a given time period is referred to as the backlog (Rodrigues et al. 2019). Consider for each time period $t \in N$, the unit holding cost h_t, the unit backlog cost b_t, and the unit production cost c_t. The demand in each time period $t \in N$ is assumed to be uncertain and to follow a uniform distribution in the interval $[0.8\bar{d}_t, 1.2\bar{d}_t]$, where $\bar{d}_t$ represents the expected demand value in period t. To deal with the stochastic demands we follow the sample average approximation method. Let Ω represent a finite set of scenarios for the demands, and d_{tw} define the demand on time period t when scenario $w \in \Omega$ occurs. Define I_0 as the initial inventory level, P as the production capacity in each time period, and C as the setup cost. The binary variable x_t indicates whether there is a setup in period t or not, while variable $y_t^p \in \mathbb{N}_0$ represents the quantity to produce in that time period. Variables y_{tw}^a are auxiliary variables used to compute the holding cost or the backlog cost in period t for scenario w.

The SLS problem can be formulated as follows:

$$
\begin{aligned}
\min \quad & \sum_{t \in N} \left(C x_t + c_t y_t^p + \frac{1}{|\Omega|} \sum_{w \in \Omega} y_{tw}^a \right) \\
s.t. \quad & y_{tw}^a \geq h_t \left(I_0 + \sum_{j=1}^{t} (y_j^p - d_{jw}) \right), && t \in N, w \in \Omega, \\
& y_{tw}^a \geq -b_t \left(I_0 + \sum_{j=1}^{t} (y_j^p - d_{jw}) \right), && t \in N, w \in \Omega, \\
& y_t^p \leq P x_t, && t \in N, \\
& x_t \in \{0, 1\}, && t \in N, \\
& y_t^p \in \mathbb{N}_0, && t \in N, \\
& y_{tw}^a \geq 0, && t \in N, w \in \Omega.
\end{aligned}
$$

The objective function minimizes the setup costs plus the total production costs plus the expected value of both the storage and the backlog costs. The first and the second sets of constraints compute the storage and the backlog cost for each time period in each scenario, respectively. The third set of constraints ensures that production can only occur in a given time period if there is a setup in that period.

5 Implementation details of the weighted proximity search

Here we describe implementation details associated with the computation of the weights for the three test problems.

5.1 Heuristic used in the CVI approach

The CVI approach requires the determination of a large number of feasible solutions to determine the weights for the binary variables $\mathbf{x}$. In our experiments we used a randomized construction heuristic to obtain those solutions. The heuristic starts by defining a performance measure for the problem that allows to determine a score s_j for each binary variable x_j, $j \in N$. Defining $Q := \sum_{i \in N} s_i$, selection probabilities for the variables $\mathbf{x}$ can be defined such that higher probabilities are associated with higher scores. Hence, for each variable x_j, the corresponding selection probability is defined as

$$\frac{s_j + \epsilon}{Q + n\epsilon},$$

where ϵ is a small positive value lower than one (0.01 in our experiments) used to ensure that all the variables have a non-zero selection probability. By using such probabilities, the variables $\mathbf{x}$ are successively selected (i.e., assigned with the value 1) until a given stopping criterion is met. The process is then repeated until the total number of required solutions is obtained. The randomized construction heuristic is described in Algorithm 3.

Algorithm 3 Randomized construction heuristic

1: Define a performance measure for the problem
2: **repeat**
3: consider all the variables $\mathbf{x}$ as non-selected, that is, $x_j = 0$ for all $j \in N$
4: **repeat**
5: compute the score and the selection probability for all the non-selected variables $\mathbf{x}$
6: randomly select a variable $\mathbf{x}$ to be fixed to one
7: **until** a given stopping criterion is verified
8: **until** the total number of required solutions is achieved

Next we identify the performance measure used for each one of the three optimization problems considered and explain how the scores are computed in each case.

In the p-median problem, the performance measure is based on the sum of costs between a facility and all the customers. Hence, for each variable x_j, the corresponding score is

$$s_j := d^{max} - \sum_{i \in M} d_{ji}$$

where $d^{max} = max_{j \in N} \left\{ \sum_{i \in M} d_{ji} \right\}$. The internal loop in the algorithm (Steps 4–7) ends when exactly p facilities are selected, meaning that a feasible solution is found.

In the set covering problem, the performance measure is related to the total number of customers covered and the cost associated with the services: For each variable x_j,

the corresponding score is defined as

$$s_j := \frac{\#customers\ to\ cover\ that\ are\ covered\ by\ service\ j}{cost\ of\ the\ service\ j}.$$

The internal loop in the algorithm (Steps 4–7) ends when all the customers are covered, meaning that a feasible solution is found.

In the SLS problem, the LP-relaxation provides a good indicator of the optimal solution (Rodrigues et al. 2020). Therefore, the performance measure is the proximity to the solution of the LP-relaxation, and the score associated with variable x_j is defined as

$$s_j := x_j^{LR}$$

where x_j^{LR} is the optimal value of variable x_j in the LP-relaxation. With this performance measure, the variables whose values in the LP-relaxation are closer to one are more promising to be selected (fix to one in the obtained solution). In the SLS problem, any realization of the binary variables results in a feasible solution, and the stopping criterion for the internal loop in the algorithm (Steps 4–7) is therefore defined using the sum of all the scores associated with the non-selected variables. Denote by Q^t the sum of the scores associated with the non-selected variables in iteration t. That is,

$$Q^t := \sum_{j \in N | x_j \text{ is non-selected at iteration } t} s_j.$$

Then, the internal loop stops at iteration k, where k corresponds to the smallest index for which the relation $Q^k \leq 0.5 \times Q$ holds, where Q is the sum of the score values of all the variables $\mathbf{x}$. This stopping criterion leads to solutions in which the number of production periods (binary variables with value one) is about half of the total number of production periods, n.

For each instance of the three optimization problems considered, 100 solutions are generated by the randomized construction heuristic. However, to obtain better weights for the CVI approach, the weights are computed by using only the 50 best solutions found.

5.2 Dealing with infeasibility in the LS approach

In the SLS problem, the feasibility of a given feasible solution is kept when any binary variable changes its value. Hence, for this problem, the LS approach can be used as described in Sect. 3.1.2. However, in both the p-median and the set covering problems, the feasibility of a given feasible solution can be lost when a binary variable changes its value (see Remark 4).

Since a feasible solution of the p-median problem requires exactly p variables $\mathbf{x}$ with value one, changing one variable from zero to one (one to zero) requires the change of another variable from one to zero (zero to one). The weight associated with

each variable $x_j,\ j \in N$, is determined as follows. Let us denote by $\bar{S} := \{\ell_1, \ldots, \ell_p\}$ the set of the selected facilities in the current solution $\bar{\mathbf{x}}$.

- If facility j belongs to $\bar{S}$ (i.e., $\bar{x}_j = 1$), then it is removed from $\bar{S}$ and one facility $i,\ i \in N\backslash\{j\}$ (such that $\bar{x}_i = 0$) is selected. Hence, a new set $\tilde{S}$ of selected facilities defined as $\tilde{S} := \bar{S}\backslash\{j\} \cup \{i\}$ is obtained. The facility i to be selected is the one such that the total distance of the customers to the new set $\tilde{S}$, denoted by $\tilde{f}_j$, is minimized.
- If facility j does not belong to $\bar{S}$ (i.e., $\bar{x}_j = 0$), then it is added to $\bar{S}$ and one facility $i,\ i \in \bar{S}\backslash\{j\}$ becomes a non-selected facility. Hence, we have that $\tilde{S} := \bar{S}\backslash\{i\}\cup\{j\}$. The facility i that becomes non-selected is the one that minimizes the total distance $\tilde{f}_j$ of the customers to the new set $\tilde{S}$ of open facilities.

After selecting the set of facilities, we can compute by inspection which customers are associated with each facility, and consequently, the cost of the solution. The weight associated with the variable x_j is determined as $w_j := \tilde{f}_j - \bar{f}$, where $\bar{f}$ is the objective function value of the current solution.

In the set covering problem, a feasible solution is a solution in which all customers are covered by the selected services. Hence, for this problem, the weights are determined as follows. Let us denote by $\bar{S} := \{\ell_1, \ldots, \ell_t\}$ the set of the selected services in the current solution $\bar{\mathbf{x}}$ and by $\bar{f}$ the cost of that solution.

- If service j belongs to $\bar{S}$ (i.e., $\bar{x}_j = 1$), then it is removed from $\bar{S}$ and the cost of the resulting solution is updated to $\tilde{f}_j := \bar{f} - c_j$. If the resulting solution is infeasible (meaning that there is at least one customer not covered), another service is added to $\bar{S}$. The service i chosen is the one that covers the largest number of uncovered customers (and the one with the lowest cost, in case of ties). The cost of the current solution is then updated to $\tilde{f}_j := \tilde{f}_j + c_i$. If the current solution is still infeasible the process is repeated until feasibility is achieved.
- If service j does not belong to $\bar{S}$ (i.e., $\bar{x}_j = 0$), then it is added to $\bar{S}$, i.e., $\tilde{S} := \bar{S}\cup\{j\}$, and the cost of the resulting solution is updated to $\tilde{f}_j := \bar{f} + c_j$. The resulting solution is always feasible and in some cases the feasibility can be kept even if some services are removed from $\bar{S}$. Hence, for each service $i \in \tilde{S}\backslash\{j\}$ (starting from the ones having the higher cost) we check if the feasibility of the solution is kept when service i is removed from $\tilde{S}$. If yes, then service i is removed from $\tilde{S}$ and the cost of the current solution is updated to $\tilde{f}_j := \tilde{f}_j - c_i$. The process is repeated until it is not possible to remove any service while keeping the feasibility of the solution.

For both cases, the weight associated with variable x_j is determined as $w_j := \tilde{f}_j - \bar{f}$.

6 Computational results

This section reports the computational experiments carried out to compare the performance of the proposed WPS heuristic (with all the variants: CVI, LRP, LRL, RCI, WF, and LS) against the classic PS heuristic. All tests were run using a computer with

an Intel Core i7-4750HQ 2.00 GHz processor and 8 GB of RAM, and were conducted using the Xpress-Optimizer 8.6.0 solver (Xpress Mosel Version 4.0.3) with the default options using up to 8 threads and 4 cores.

The LRL approach requires the solution of a large number of LP-relaxations. In both the p-median and the set covering problems the total time required to solve a single LP-relaxation for typical instances is greater than 9 s. Thus, the LRL is only tested for the SLS problem. Having solved the LP-relaxation of problem (2.1) to obtain the value $f_{0,0}$, each value $f_{j,0}$ ($f_{j,1}$) can be obtained in two different ways: (i) by imposing the constraint $x_j = 0$ ($x_j = 1$) and solving the LP-relaxation of the original problem (2.1) with this new constraint; or (ii) by imposing the constraint $x_j = 0$ ($x_j = 1$) and applying the dual simplex algorithm to the optimal tableau associated with the solution of the LP-relaxation problem. Preliminary tests showed that it is more efficient to obtain the values $f_{j,0}$ and $f_{j,1}$ when Xpress is used with the Newton–Barrier method (without reoptimization) than when it is used with the dual simplex algorithm (with reoptimization). For example, for the SLS instances with n=90 the computational time required for determining the weights in the first case is 220 s while in the second case it is around 1129 s.

6.1 Training set and test set

In our experiments we consider two sets of instances: a training set and a test set. The test set consists of 16 groups of instances (nine from the p-median problem, four from the set covering problem and three from the SLS problem), with $L = 10$ randomly generated instances each (giving a total of 160 instances). The training set consists of 16 instances (nine from the p-median problem, four from the set covering problem and three from the SLS problem) differing from the ones in the test set, but generated in the same way. The training set is used for parameter tuning. Data for the instances in the test set are available at: http://home.himolde.no/~hvattum/benchmarks/.

For the p-median problem we consider three different values for the number m of customers (500, 750, and 1000) and assume that every customer location is also a facility location, that is, $n = m$ and $N = M$. The exact number p of facilities to select is 5, 15, and 25. The combinations of the different values for the number of customers and facilities lead to nine sets of instances, each with ten instances that differ in terms of the distance matrix. The distances $d_{ji},\ j \in N, i \in M$ are integer, symmetric and randomly generated in the interval [1, 100] for $i \neq j$, and they are set equal to zero for $j = i$.

For the set covering problem we consider randomly generated instances with either $m = 1000$ or $m = 2000$ customers, and either $n = 500$ or $n = 1000$ services. The combinations of different values for m and n give rise to four sets of instances, each with ten instances. These ten instances differs in terms of both the costs $c_i,\ i \in N$, associated with the services (randomly generated in the interval [50, 150]) and the binary coefficients matrix (randomly generated such that the percentage of non-zero elements is 30%).

The computational experiments for the SLS problem use instances generated as in Rodrigues et al. (2020). The total number n of periods considered is 60, 90, or

Table 1 Gaps obtained by Xpress after 1200 s

	Gap_1		Gap_2	
	Average (%)	Standard deviation	Average (%)	Standard deviation
p-median	39	1.8	65	4.6
Set covering	57	1.3	139	6.6
Stochastic lot-sizing	39	0.9	63	2.6

120. For each time period, $t \in N$, the nominal demand $\bar{d}_t$ is randomly generated in [0, 50] and the demand d_{tw} corresponding to scenario $w \in \Omega$ is randomly generated in $[0.8\bar{d}_t, 1.2\bar{d}_t]$. For each instance we consider 100 scenarios, i.e., $|\Omega| = 100$.

The initial stock level at the producer, I_0, is randomly generated between 0 and 30, the production capacity P and the setup cost are constants and equal to $\sum_{t \in T} \bar{d}_t$ and 150, respectively. The production, holding, and backlog costs are $c_t = 1$, $h_t = 4$, $b_t = 6$, respectively, for all $t \in N$. A set of ten random instances is generated for each of the three values of n.

To evaluate the hardness of the instances in the test set we solve each one of the 160 instances using Xpress with a time limit of 1200 s. For ease of presentation we grouped all the instances of each problem and present the average and the standard deviation of the obtained gaps in Table 1. The gaps identified as Gap_1 and Gap_2 are defined by

$$Gap_1 = \frac{UB - LB}{UB} \times 100 \quad \text{and} \quad Gap_2 = \frac{UB - LB}{LB} \times 100$$

where UB and LB are the upper bound and the lower bound obtained after 1200 s, respectively.

The results presented in Table 1 show that the instances considered are hard to solve, since the minimum average gaps are about 39% and 63%. The standard deviations associated with the gaps are small, indicating that for each instance the corresponding gap is close to the average.

6.2 Performance measures

The approaches are compared using two metrics. The first metric is an approximation of the primal integral (Achterberg et al. 2012; Berthold 2013). The primal integral measures the trade-off between the quality of the solutions obtained by a particular method and the running time required in their computation. Denoting by z^* the cost of the optimal solution of the problem and by $z(t)$ the cost of the best feasible solution found until time t, the primal gap function v is defined by

$$v(t) = \begin{cases} 1, & \text{if no feasible solution is found until time t,} \\ \gamma(z(t)), & \text{otherwise,} \end{cases}$$

where function $\gamma(z(t))$ is the primal gap defined as

$$\gamma(z(t)) = \begin{cases} 0, & \text{if } z^* = z(t), \\ 1, & \text{if } z(t)z^* < 0, \\ \frac{z(t)-z^*}{max\{z^*,z(t)\}}, & \text{otherwise.} \end{cases}$$

The primal integral of a running time t^{max} is

$$P(t^{max}) = \int_0^{t^{max}} v(t)dt.$$

This metric allows a global vision of the heuristic performance until time t^{max}, with lower values indicating a better heuristic performance. The computation of the primal integral measure requires the knowledge of the optimal value of the problem. As reported in Table 1, our test set is composed of instances that are difficult to solve and for which the optimal solution is not known, making the use of the primal integral measure impractical. Hence, in our experiments we use an approximation called the primal integral approximation measure, in which the optimal value z^* is replaced by the cost of the best solution found among all the approaches considered. The primal integral measure (and consequently the primal integral approximation measure) is highly affected by the performance of the heuristic in the early search phases, meaning that heuristics that take more time to find feasible solutions tend to have high primal integral values, even when they are able to find good solutions later on.

To circumvent the drawbacks of the primal integral approximation, we create a second metric, called the signal metric, that is computed in relation to the classic PS heuristic. Denoting by $z_i^{PS}(t)$ and by $z_i^{WPS}(t)$ the values of the best solutions found by the classic PS heuristic and by the WPS heuristic, for instance $i \in \{1, \ldots, L\}$, until time t, we compute the value

$$h_i(t) = \begin{cases} 0, & \text{if } z_i^{PS}(t) = z_i^{WPS}(t), \\ -1, & \text{if } z_i^{PS}(t) < z_i^{WPS}(t), \\ 1, & \text{if } z_i^{PS}(t) > z_i^{WPS}(t), \end{cases} \tag{6.1}$$

and finally the signal metric

$$H(t) := \frac{1}{L}\sum_{i=1}^{L} h_i(t)$$

which varies between -100 and 100%. When $H(t) = 100\%$, this metric indicates an absolute better performance of the WPS heuristic, meaning that for all the L instances tested the WPS heuristic always obtains a better solution than the one obtained by the PS heuristic until time t. Analogously, $H(t) = -100\%$ indicates an absolute better performance of the PS heuristic.

6.3 Calibration

In this section we explain the choice of some parameters and the use of some procedures. These experiments are conducted on the 16 instances of the training set.

Among the three variants of the classic PS heuristic (without recentering, with recentering, and with incumbent) the PS with incumbent is known to be the best (Fischetti and Monaci 2014, 2016). Preliminary results conducted on the training set instances corroborated this result as the solutions obtained by the PS with incumbent after 1200 s are around 2.5% better than the ones by the PS with recentering, and around 5% better than the ones obtained by the PS without recentering. Therefore, all the preliminary tests and computational results reported next for both the PS and the WPS correspond to the variant with incumbent.

All the three discretization schemes described in Sect. 3, (RD, 2-VS, 3-VS), require the definition of the maximum possible value for the weights (previously denoted by R). To determine the most suitable value for R, we conducted some preliminary tests (reported in the appendix) from which we chose $R = 10$. After choosing the value R we test several different schemes to define the weights:

(i) Use the initial weights directly, without discretization (denoted by WD).
(ii) Use all the weights equal to one, which corresponds to the classic PS heuristic (denoted by PS).
(iii) The R-discretization scheme (denoted by RD).
(iv) Three variants of the two-value system corresponding to threshold values t_1 equal to 3, 5 and 7 (denoted by 2-$VS_{(3)}$, 2-$VS_{(5)}$ and 2-$VS_{(7)}$, respectively).
(v) Three variants of the three-value system corresponding to threshold values (t_2, t_3) equal to (2, 4), (3, 6) and (4, 8) (respectively denoted by 3-$VS_{(2,4)}$, 3-$VS_{(3,6)}$ and 3-$VS_{(4,8)}$).

Apart from the second scheme (corresponding to the classic PS heuristic) all the other schemes used to define the weights are tested on the WPS heuristic with the Loss/Saving variant. The obtained results are reported in Table 2. The first column identifies the weights scheme used. The second column reports the average gaps obtained by solving all the 16 instances of the training set for 1200 s. Gaps are computed with respect to the best feasible solution found by the set of the nine discretization schemes tested. In the third column, the value of the final solution obtained by each method after 1200 s is divided by the best feasible solution found by all the schemes, and the average value over all the 16 instances is displayed. The last column reports the number of best solutions found by each of the schemes.

The results presented in Table 2 show that it is beneficial to discretize the weights, since the largest gaps are obtained when the weights are not discretized. The two-value system and the three-value system seem to perform well for the lower threshold values tested. The scheme with the best global performance is the three-value system with thresholds $(t_2, t_3) = (2, 4)$. Therefore, in what follows, we use this scheme in all the variants of the WPS heuristic.

Each iteration of the PS and WPS heuristics stops when the first feasible solution is found with the slack variable u equal to zero (Fischetti and Monaci 2014). We also consider a time limit of 1200 s and a cutoff tolerance of $\theta = 1$. For the cutoff tolerance,

Table 2 Comparison of the schemes to define weights

	Avg. Gap	Norm. values	# Best solutions found
WD	12.9	1.13	1
PS	4.9	1.05	0
RD	4.5	1.05	3
2-$VS_{(3)}$	4.3	1.04	1
2-$VS_{(5)}$	4.2	1.04	0
2-$VS_{(7)}$	5.1	1.05	0
3-$VS_{(2,4)}$	1.2	1.01	7
3-$VS_{(3,6)}$	2.2	1.02	3
3-$VS_{(4,8)}$	6.1	1.06	0

$\theta = 1$ was shown to perform better than the alternatives $\theta = 10$ and $\theta = 20$ tested for all three problems. For brevity, we omit these tests here. The value of M associated with the slack variable u is set to 10^5.

6.4 Main results

In this section we report the computational results performed on instances of the p-median problem, the set covering problem, and the stochastic lot-sizing problem. The results are aggregated for each problem and they are presented following the same pattern. We start by presenting a table reporting some information about the number of iterations performed and the computational times. Then, we present two tables comparing variants of the WPS against the PS in terms of the minimum, the average, and the maximum gaps obtained after 1200 s. For a better understanding of the behavior of each approach over time and not only at the end of the time limit, we present plots showing the primal integral approximation measure and the signal metric. To improve the readability of the primal integral approximation plots, we add symbols to each line. These symbols help to identify each line, and their x-coordinates correspond to the time instants $\{0, 200, \ldots, 1200\}$. A similar strategy is used for the signal metric plots, with x-coordinates of the points corresponding to the time instants $\{0, 100, \ldots, 1200\}$.

After presenting and analyzing the plots associated with both the primal integral approximation measure and the signal metric, we compare the PS and the best variant of the WPS with solving the original problem directly using Xpress, by imposing a time limit corresponding to the total time spent by PS and WPS. The comparison is done in terms of the primal integral approximation measure, the signal metric, and also in terms of the final gap obtained after the time limit is reached.

6.4.1 Computational results for the *p*-median problem

As described in Sect. 6.1, we consider nine sets of instances of the p-median problem with ten instances each. These sets result from the combination of the different numbers

Table 3 Minimum, average, and maximum computational times and number of iterations for the p-median instances with a time limit of 1200 s

	Minimum values				Average values				Maximum values			
Appr.	W_{time}	I_{time}	R_{time}	$\#It$	W_{time}	I_{time}	R_{time}	$\#It$	W_{time}	I_{time}	R_{time}	$\#It$
PS	0	45	51	5	0	132	263	9	0	220	630	23
LS	9	17	12	6	342	67	494	13	911	122	968	31
LRP	11	46	96	4	47	125	278	9	92	205	606	24
CVI	29	42	85	4	128	112	311	9	292	180	631	25
RCI	0	47	60	4	0	139	281	9	0	227	684	23
WF	0	48	80	5	0	127	273	9	0	219	684	23

of customers considered (500, 750, and 1000) and the exact number of facilities to select (5, 15, and 25). The initial solution used in both the PS and the WPS heuristics is obtained by solving the original model for 120 s using Xpress.

Table 3 reports minimum, average, and maximum computational times (in seconds) and number of iterations corresponding to the average values obtained for the 9 sets of instances used. The first column identifies the approach. Columns W_{time} show the total time spent on computing weights, while columns I_{time} report the average time spent on solving each modified problem (for each iteration, not including the time spent on computing weights). For all the approaches, a time limit of 1200 s is imposed. Columns R_{time} show the difference between the time limit imposed and the time corresponding to the last solution found improving the objective function value. The number of iterations is reported in columns $\#It$.

The results reported in Table 3 indicate that the LS approach performs more iterations than the other approaches, while the computational time associated with solving the modified problem at each iteration (after the weights are determined) is much lower than the alternative approaches. The total computational time spent on computing the weights in the LS approach increases as the number of customers n and the value of p increase. Moreover, since the number of iterations performed by the LS approach is usually high (specially for the instances with $p = 25$) and the weights need to be computed at each iteration, a large portion of the 1200 s is used to compute such weights.

Next, we compare the final solutions provided by the variants of WPS with the solutions obtained by PS. For each of the 90 instances we compute a gap according to the formula:

$$gap := \frac{z^X - z^{PS}}{z^{PS}}, \tag{6.2}$$

where z^{PS} is the value of the solution obtained by PS heuristic and z^X is the value of the solution found by each variant of the WPS heuristic. Then, we group all the instances by sets and compute the average value of the gaps within each set (giving a total of 9 gaps). Finally, in Table 4 we report the minimum, average, and maximum of those average gaps.

Table 4 Average gaps (%) obtained for the p-median instances after 1200 s

	LS	LRP	CVI	RCI	WF
Minimum	−7.5	−2.4	−2.4	−0.6	−0.8
Average	−4.3	−0.4	−0.4	0.2	1.0
Maximum	−0.7	0.2	0.5	0.9	4.9

Table 5 Minimum, average, and maximum gaps (%) obtained for the p-median instances after 1200 s for the LS approach

	$n = 500$			$n = 750$			$n = 1000$		
p	5	15	25	5	15	25	5	15	25
Minimum	−3.4	−4.7	−8.3	−4.0	−8.0	−12.8	−20.8	−10.2	−12.5
Average	−0.7	−1.6	−3.3	−1.7	−5.1	−7.4	−4.2	−6.5	−7.5
Maximum	0.2	2.3	2.5	1.1	0.6	1.4	−0.7	−0.8	−0.9

Negative gaps indicate a better quality of the solutions obtained by the corresponding variant of the WPS heuristic. Hence, despite the large amount of time spent by LS to compute weights, the approach can still provide better solutions than the ones obtained by the classic PS heuristic. The static approaches (LRP and CVI) also perform slightly better than the PS heuristic. However, the PS heuristic provides, on average, better results than the ones obtained with the dynamic approaches RCI and WF.

Since the best results among all the WPS variants considered were obtained with the LS approach, we report in Table 5 more specific gaps for this approach computed as follows. First we calculate the gap for each of the 90 instances according to formula (6.2). Then we group all the instances by sets and report the minimum, average, and maximum gaps in each one of the sets.

Table 5 shows that for the instances with 1000 customers (the hardest ones), the solutions obtained by the LS approach are always better than the ones obtained by the PS heuristic, since the maximum gaps are negative. Moreover, for some instances, the LS can provide solutions up to 21% better than PS, while PS is at most 2.5% better than LS on the instances considered.

Table 4 gives a general idea of the behavior of the approaches at the end of the 1200 s, while the primal integral approximation measure and the signal metric provide a better vision of such behavior over the time. The primal integral approximation measure and the signal metric for the nine sets of instances of the p-median problem aggregated are reported in Figs. 1 and 2, respectively.

In Figs. 1 and 2 the dominance of the LS approach is evident. The primal integral approximation measure for this approach is much lower than the one corresponding to the PS heuristic. Analyzing the signal metric, we can see that the LS approach not only performs better in the later instants, but also early in the runs. For the other approaches, there is no clear evidence indicating which of them performs better.

To demonstrate the benefits of the LS approach we compare it against the use of the commercial MIP solver Xpress, imposing a time limit of $1200 + 120 = 1320$ s

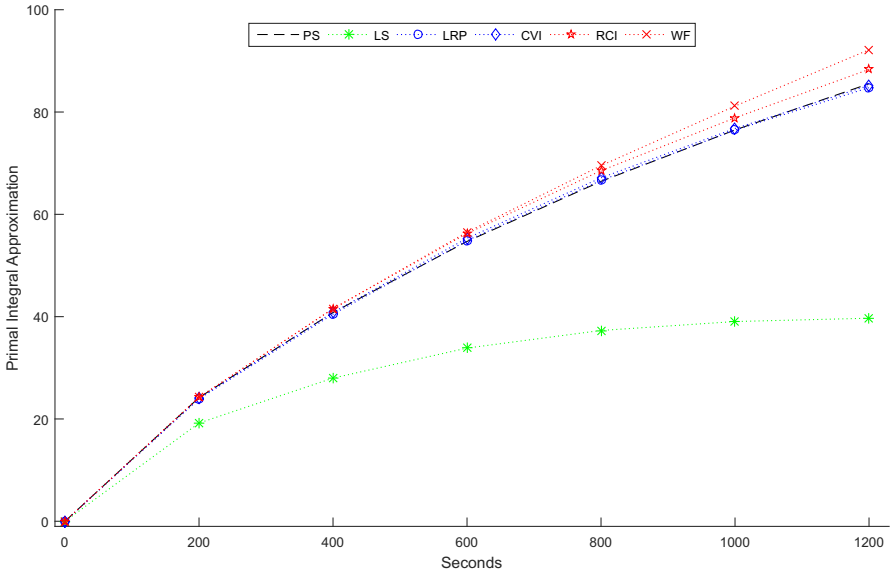

Fig. 1 Average primal integral approximation measure for instances of the p-median problem

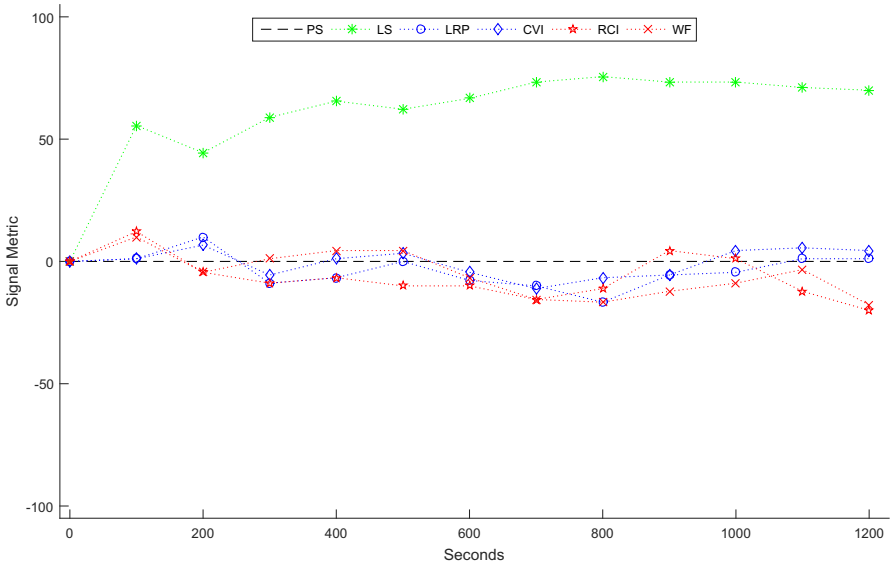

Fig. 2 Signal metric for instances of the p-median problem

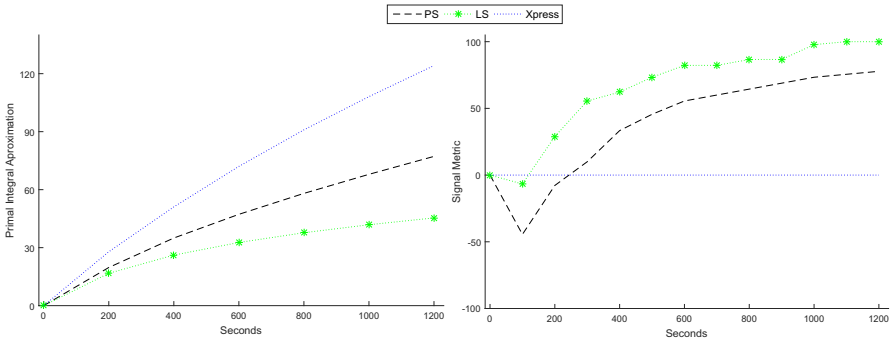

Fig. 3 Average primal integral approximation measure and signal metric for the 9 sets of instances of the p-median problem

Table 6 Minimum, average and maximum gaps for the 9 sets of instances of the p-median problem aggregated after 1200 s

	Minimum	Average	Maximum
PS	−16.1	−5.9	13.3
LS	−18.7	−9.9	−1.1

for the latter. The 120 s correspond to the computational time required for finding the initial solution used in both the PS and the WPS. In the PS and the WPS variants, the 120 s required to find the initial solution are not included in the time limit of 1200 s.

Figure 3 presents the average primal integral approximation measure (on the left) and the signal metric (on the right) for Xpress, for the PS heuristic, and for the LS heuristic (which is the best variant of the WPS for the p-median problem), for the nine sets of instances. The signal metric is computed with respect to Xpress, that is, in formula (6.1) $z_i^{Xpress}(t)$ is used instead of $z_i^{PS}(t)$ and z_i^X instead of z_i^{WPS}, where X denotes the approaches PS and LS, and $z_i^{Xpress}(t)$ denotes the solution obtained by Xpress for instance i at time t.

Figure 3 indicates that both the PS and the LS heuristics outperform the direct use of Xpress. In the beginning of the search, Xpress is able to find better solutions than both the PS and the LS heuristic. However, at later stages of the search, the opposite behavior is observed.

Table 6 reports the average gaps obtained by the PS and by the LS heuristics computed with respect to the solution obtained by Xpress after 1320 s. The gaps are computed as in formula (6.2), but replacing z^{PS} with the value of the best solution found with Xpress. The results indicate that the solutions obtained by LS are better than the ones obtained by Xpress for all the 90 instances of the p-median problem. The maximum gap reported for the PS heuristic (13.3%), is an outlier value in the set of the 90 instances and corresponds to an instance with 1000 costumers and p=5. The second highest gap for this approach is around 6%. On average, the final solutions obtained by the PS and by the LS heuristic are, respectively, 6% and 10% better than the ones obtained by Xpress.

6.4.2 Computational results for the set covering problem

We consider four sets of instances for the set covering problem, with ten instances each. These instances result from the combinations of the number of customers ($m = 1000$ and $m = 2000$) and the number of services ($n = 500$ and $n = 1000$). The initial solution used in both the PS and the WPS heuristics is the same and corresponds to solving the original model for 120 s using Xpress. Minimum, average, and maximum computational times and number of iterations for the four sets of instances aggregated are reported in Table 7.

The results reported in Table 7 indicate that the number of iterations performed is similar for all the approaches. The CVI approach seems to be the one for which the time per iteration is lower. However, this approach also presents, in general, higher R_{time} values, indicating that the early iterations are fast and that the value of the objective function is not improved in the later stages of the search. In terms of the computational time spent to compute the weights, we can see that it is short (in all the approaches) when compared with the time limit of 1200 s.

Table 8 reports the minimum, the average, and the maximum of the average gaps for all the variants of the WPS heuristic, obtained after the time limit is reached. The reported gaps are obtained following the same procedure used for Table 4. The displayed results indicate that, in terms of the final solutions obtained, the LS, the LRP, the CVI and the WF approaches perform, on average, better than the classic PS heuristic. However, PS performs better than the RCI approach for the four sets of instances. Table 9 reports the minimum, average, and maximum gaps associated with the LS approach for the four sets of instances after 1200 s, independently.

Table 9 shows that the LS approach is unable to obtain consistently better solutions than the classic PS heuristic for all the instances. However, when it performs better, the improvement can reach 7%. The primal integral approximation measure and the signal metric for the four sets of instances of the set covering problem are reported in Figs. 4 and 5, respectively.

Figures 4 and 5 confirm the dominance of the LS approach (and also the good performance of the LRP, CVI, and WF approaches) when compared with the classic PS heuristic. The poor performance of the RCI approach is also shown in these figures.

Now we compare the PS and the LS heuristics against Xpress. Figure 6 presents the average primal integer approximation measure (on the left) and the signal metric (on the right) for Xpress, for the PS heuristic, and for the LS heuristic (which is the best variant of the WPS for the set covering problem), for the four sets of instances aggregated. The signal metric is computed with respect to Xpress as in the previous section.

Figure 6 indicates the superiority of both the PS and the LS heuristics in relation to Xpress. The plot associated with the signal metric indicates that after 100 s almost all the solutions found by the LS approach are better than the ones obtained by Xpress. Table 10 reports the minimum, average, and maximum gaps obtained by the PS and by the LS heuristics computed with respect to the solution obtained by Xpress. On average, the final solutions obtained by the PS and by the LS approach are, respectively, up to 4% and 5% better than the ones obtained by the Xpress.

Table 7 Minimum, average, and maximum computational times and number of iterations for the set covering instances with a time limit of 1200 s

	Minimum values				Average values				Maximum values			
Appr.	W_{time}	I_{time}	R_{time}	$\#It$	W_{time}	I_{time}	R_{time}	$\#It$	W_{time}	I_{time}	R_{time}	$\#It$
PS	0	50	428	8	0	60	642	9	0	78	781	11
LS	9	33	299	9	34	57	609	10	67	105	818	11
LRP	3	27	643	8	12	40	784	10	27	51	911	11
CVI	64	5	694	7	153	16	896	10	274	33	1023	11
RCI	0	66	370	7	0	83	527	8	0	126	660	10
WF	0	10	433	9	0	45	740	10	0	72	1111	12

Table 8 Average gaps (%) obtained for the set covering problem instances after 1200 s

	LS	LRP	CVI	RCI	WF
Minimum	− 1.8	− 2.3	− 2.2	1.1	− 1.2
Average	− 1.2	− 0.4	− 0.5	1.8	− 0.7
Maximum	− 0.6	1.0	1.2	2.3	− 0.2

Table 9 Minimum, average, and maximum gaps (%) obtained for the set covering instances after 1200 s for the LS approach

	$n = 500$		$n = 1000$	
m	1000	2000	1000	2000
Minimum	− 3.9	− 6.9	− 4.4	− 6.5
Average	− 1.1	− 1.3	− 0.5	− 2.1
Maximum	1.3	3.7	0.9	5.5

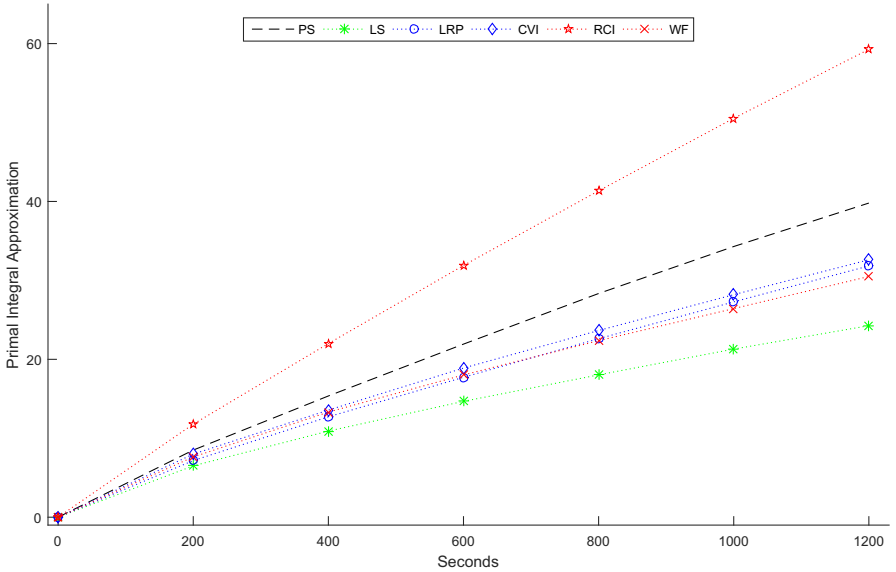

Fig. 4 Average primal integral approximation measure for the instances of the set covering problem

6.4.3 Computational results for the stochastic lot-sizing problem with setups

For the stochastic lot-sizing problem we use three sets of instances corresponding to the number of periods n equal to 60, 90, and 120. Each of the sets contains ten instances. Since the LP-relaxations of the SLS problem are solved very quickly, the instances of this problem can be used to test the LRL approach. The LRL was not applied to the p-median and to the set covering problems, due to the large time required to solve all the LP-relaxations. For both the PS and the WPS heuristics, the initial solution considered is obtained by solving a deterministic lot-sizing problem with the demands fixed to their expected values for 120 s using Xpress.

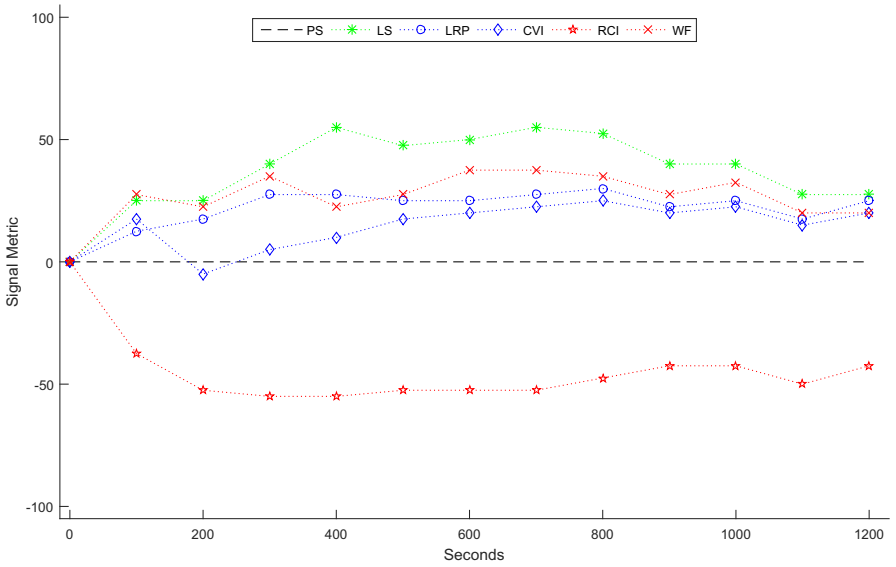

Fig. 5 Signal metric for the instances of the set covering problem

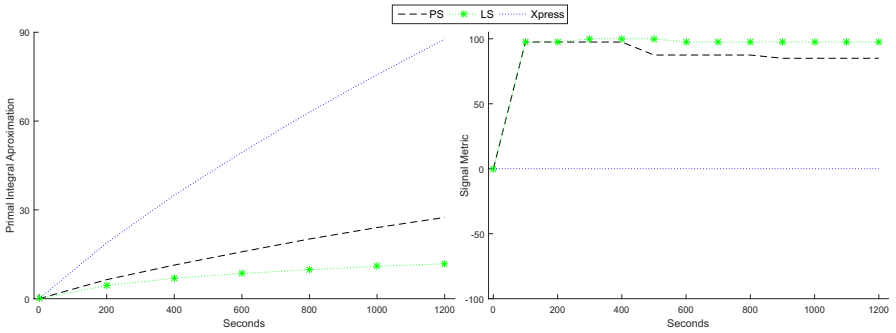

Fig. 6 Average primal integral approximation measure and signal metric for instances of the set covering problem

Table 10 Minimum, average and maximum gaps for the instances of the set covering problem after 1200 s

	Minimum	Average	Maximum
PS	−12.2	−4.4	1.8
LS	−12.5	−5.6	0.0

The minimum, average, and maximum computational times and number of iterations for all the approaches are reported in Table 11. The total computation time required to determine the weights in the *LS* approach can be large when the number of scenarios considered is large. We consider SLS instances with 100 scenarios, but in order to determine the weights in the *LS* approach more efficiently (decreasing the time spent on their computation), we used only 20 of those 100 scenarios (randomly selected at each iteration).

Table 11 shows that the variation of the number of iterations among the approaches is small. The lowest computational times per iteration required to solve the modified problem (after the weights are determined) are provided by the LS approach.

Table 12 reports the minimum, the average, and the maximum of the average gaps for all the variants of the WPS heuristic, obtained after the time limit is reached. The reported gaps are obtained following the same procedure used for Table 4.

These results show that there are no large differences between the average cost of the final solutions obtained. When comparing the WPS against PS, the WPS variants provide final solutions that are, in general, better for instances with 60 and 90 time periods and worse for instances with 120 time periods. However, in all the three cases, the best results are obtained by the LS approach.

In Table 13 we report the minimum, average, and maximum gaps obtained for LS (the best approach in terms of the average cost of the final solutions obtained) for the three sets of instances of the SLS problem after 1200 s. Moreover, we also present the gaps associated with both the LRP and the LRL since these approaches also perform reasonably well.

The results presented in Table 13 reveal that the largest improvements in terms of gaps with respect to the solutions of the PS heuristic are associated to the instances with 90 time periods and are at most 2.3%. Moreover, we can see that in terms of the absolute values, the minimum gaps are higher than the maximum gaps (for $n = 60$ and $n = 90$), which also indicate a better performance of the three WPS variants when compared against the PS heuristic.

Remark 6 A particular feature of the SLS instances tested is that large changes in the structure of the solutions usually do not lead to a large variation in the objective function value, while the objective function value itself is often high. This can help to explain why the reported gaps reported in this section for the SLS problem are small. Indeed, the largest gap obtained among all the approaches computed between the initial solution considered and the final solution obtained (after 1200 s) is only 6%. However, the SLS problem arises in capital intensive markets, thus modest improvements on the solutions may result in considerable profits.

Figures 7 and 8 report the average primal integral approximation measure and the signal metric for the three sets of instances aggregated. Regarding the primal integral approximation measure, the PS heuristic is completely dominated by the LS, LRP and LRL approaches. A similar behavior is observed for the signal metric plot.

Figure 9 presents the average primal integral approximation measure (on the left) and the signal metric (on the right) for Xpress used directly, for the PS, and for the LS approach, for the three sets of instances aggregated. The signal metric is computed with respect to Xpress as in the previous sections.

The behavior of the primal integral approximation measure for the instances of the SLS problem is similar to the one observed for the instances of both the set covering problem and p-median problem. However, the plot associated with the signal metric is very different to the ones presented for those problem. The PS and the WPS heuristics start from an initial solution obtained by solving the deterministic problem (with only one scenario) and such solution is, in general, a good solution for the stochastic problem. Xpress gives the value of the solution obtained with the solver considering

Table 11 Minimum, average, and maximum computational times and number of iterations for the SLS instances with a time limit of 1200 s

	Minimum values				Average values				Maximum values			
Appr.	W_{time}	I_{time}	R_{time}	$\#It$	W_{time}	I_{time}	R_{time}	$\#It$	W_{time}	I_{time}	R_{time}	$\#It$
PS	0	89	175	2	0	197	334	6	0	372	420	9
LS	42	46	142	4	140	105	394	7	257	166	760	9
LRP	1	77	127	2	3	214	367	6	5	426	582	8
LRL	90	55	199	1	374	153	341	6	695	306	616	9
CVI	12	77	151	2	22	213	374	6	12	390	442	8
RCI	0	89	166	2	0	210	330	6	0	390	442	9
WF	0	82	230	2	0	204	305	7	0	396	408	11

Table 12 Average gaps (%) obtained for the SLS instances after 1200 s

	LS	LRP	LRL	CVI	RCI	WF
Minimum	−0.9	−0.3	−0.4	−0.2	−0.1	−0.1
Average	−0.3	−0.1	−0.1	−0.0	0.0	−0.0
Maximum	−0.2	0.2	0.3	0.2	0.0	0.2

Table 13 Minimum, average, and maximum gaps (%) obtained for the SLS instances after 1200 s for the LS, LRP, and LRL approaches

	$n = 60$			$n = 90$			$n = 120$		
	LS	LRP	LRL	LS	LRP	LRL	LS	LRP	LRL
Minimum	−0.8	−0.7	−0.5	−2.3	−1.2	−0.8	−1.2	−0.3	−0.2
Average	−0.2	−0.1	−0.1	−0.9	−0.3	−0.3	−0.2	0.3	0.1
Maximum	0.1	0.5	0.2	0.5	0.9	0.5	0.6	0.8	0.8

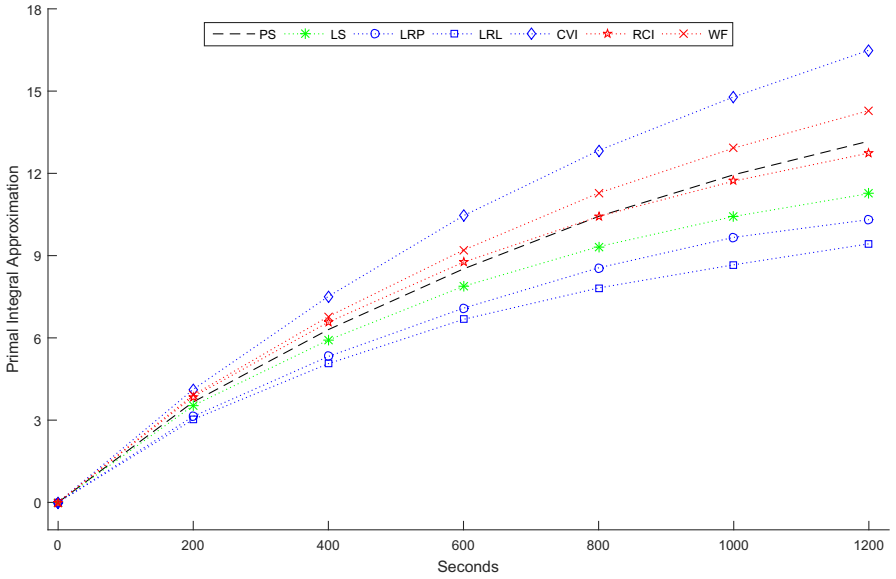

Fig. 7 Average primal integral approximation measure for the instances of the SLS problem

the stochastic model with all the scenarios (which is a large mixed-integer model) after the time limit is reached. This fact explains why there is no intersection between the line associated with Xpress and the other two lines at time zero.

Table 14 reports the minimum, average, and maximum gaps obtained by the PS and by the LS approach, computed with respect to the solutions obtained by Xpress after 1320 s. The final solutions obtained by the PS and by the LS approach are always better than the ones obtained by Xpress, having average gaps around 9%.

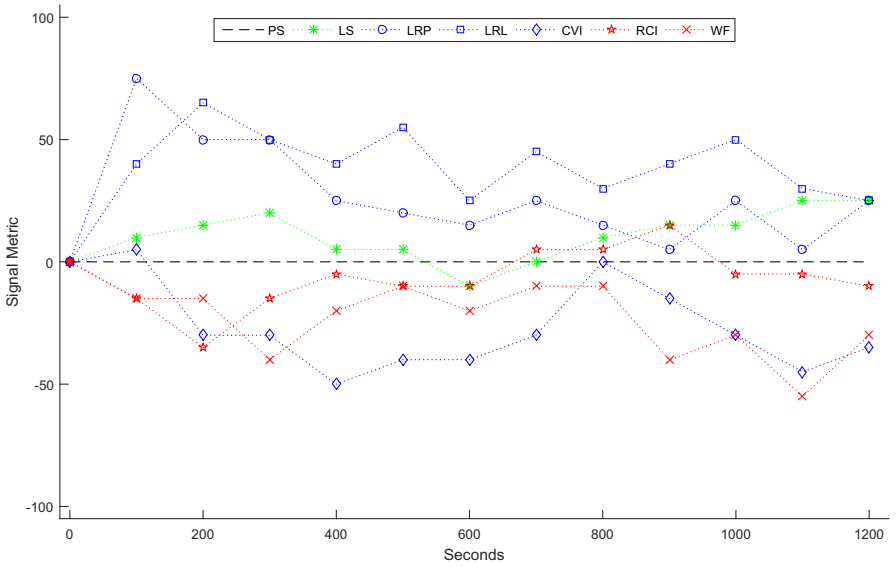

Fig. 8 Signal metric for the instances of the SLS problem

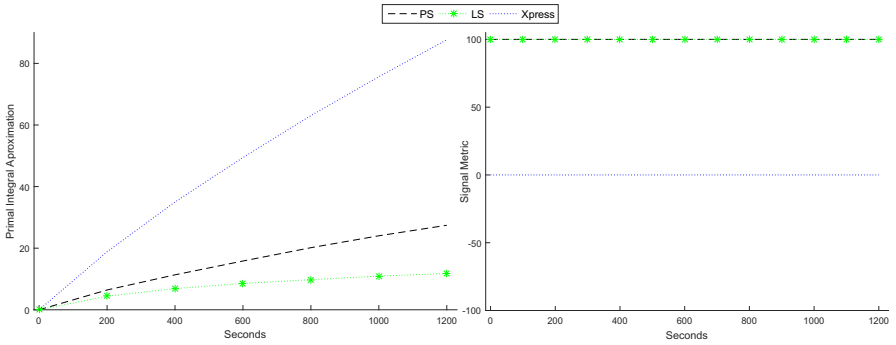

Fig. 9 Average primal integral approximation measure and signal metric for the instances of the SLS problem

Table 14 Minimum, average, and maximum gaps for the instances of the SLS problem after 1200 s

	Minimum	Average	Maximum
PS	−13.0	−9.0	−5.1
LS	−12.6	−9.1	−5.7

7 Conclusion

This paper introduces a new method referred to as weighted proximity search (WPS). In WPS, a weighted version of the Hamming distance function used in PS is introduced. The aim is to guide a MIP solver to identify better feasible solutions faster.

Six strategies to compute weights are proposed and their advantages and disadvantages are discussed. Considering the information used and the moments in which weights are computed, the strategies can be classified into two classes: static and dynamic. Each strategy proposed for computing the weights leads to a variant of the WPS heuristic. Some of the variants of the WPS presented (LRL, LRP, WF and RCI) are very general and easy to implement, meaning that they can directly be applied to a wide range of optimization problems. This is not the case of the CVI and the LS variants since they need to be adapted to each particular problem.

The WPS heuristic is applied to a set of 160 randomly generated hard instances from three optimization problems: the p-median problem, the set covering problem, and the stochastic lot-sizing problem. Among all the variants of the WPS heuristic tested, the dynamic Loss/Saving (LS) approach, in which the weights are computed at each iteration, is the one with the best performance. The most important conclusion that can be drawn from our computational study is that the LS approach provides results that are consistently better than the ones obtained by both the classic PS heuristic and a commercial MIP solver used as reference methods. Besides of being generally better in terms of the quality of the obtained solutions, the LS approach is also better in terms of the computational time required to obtain such solutions.

For future research, it would be interesting to investigate other forms of computing Loss/Savings that could enable the use of WPS as a problem-independent procedure. Another line of future research is to explore the best way to use parallel computing in the context of WPS.

Acknowledgements The authors wish to thank the anonymous reviewers, whose comments and suggestions helped us to improve the manuscript. The research was partially supported by the Center for Research and Development in Mathematics and Applications (CIDMA) through the Portuguese Foundation for Science and Technology (FCT - Fundação para a Ciência e a Tecnologia), references UIDB/04106/2020 and UIDP/04106/2020. The research of the first author was also partially supported by the Project CEMAPRE/REM - UIDB/05069/2020, financed by FCT/MCTES through national funds. The third author was supported by the AXIOM project, partially funded by the Research Council of Norway.

Funding Open Access funding provided by Molde University College - Specialized University in Logistics

Appendix

Here we report some preliminary results used to determine the maximum value R used in the discretization schemes described in Sect. 3.2. Such results can also be used to understand the potential of the proposed WPS heuristic and are obtained as follows:

(i) use Xpress for solving the original model (2.1) until the first feasible solution is found. Let us denote such solution by $(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$;
(ii) run the classic PS heuristic by imposing a time limit of 500 s and save both the initial solution $(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$ and the final solution $(\bar{\mathbf{x}}^f, \bar{\mathbf{y}}^f)$ obtained;
(iii) use these solutions to define static weights for the WPS heuristic as

$$w_j = \begin{cases} 1, & \text{if } \bar{x}_j^0 \neq \bar{x}_j^f, \\ \bar{R}, & \text{otherwise,} \end{cases}$$

where $\bar{R}$ is an integer number greater than one;
(iv) run the WPS heuristic, starting from the solution $(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$, until a solution better than or equal to the one obtained by the classic PS is found;
(v) compare the computational times.

For the parameter $\bar{R}$ we tested 4 different values (2, 5, 10 and 20) on the training set of instances. The obtained average results are reported in Table 15. The first row identifies the problem. The second row reports the average running times associated with the best solutions found by the PS heuristic, while the remaining four rows report the average running times spent by the WPS heuristic for the different values of $\bar{R}$ considered to obtain a solution better than or equal to the one obtained by the PS heuristic.

The results presented in Table 15 clearly indicate that, when both the initial and the final solutions obtained with the PS heuristic are assumed to be known, assigning different weights to the variables may lead to significant improvements by distinguishing between variables that changed their value and those that did not. Moreover, according with Table 15, the lowest computational times correspond to the value $\bar{R} = 10$. Therefore, we consider $R = \bar{R} = 10$.

Remark 7 The main goal of the results presented in Table 15 is not to compare the classic PS heuristic against the WPS heuristic. This table is used to show that by attributing different weights to different variables, a target solution can be reached more quickly.

Table 15 Computational times, in seconds, associated with the PS heuristic and with the WPS for different values of $\bar{R}$

	p-median	Set covering	Stochastic lot-sizing
PS	455	281	379
WPS_2	271	135	251
WPS_5	197	146	218
WPS_{10}	30	58	237
WPS_{20}	65	88	290

References

Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. Oper. Res. Lett. **33**(1), 42–54 (2005)

Achterberg, T., Berthold, T., Hendel, G.: Rouding and propagation heuristics for mixed integer programming. In: Klatte, D., Lüthi, H.-J., Schmedders, K. (eds.) Operations Research Proceedings 2011, pp. 71–76. Springer, Berlin (2012)

Agra, A., Christiansen, M., Hvattum, L.M., Rodrigues, F.: A MIP based local search heuristic for a stochastic maritime inventory routing problem. In: Paias, A., Ruthmair, M., Voß, S. (eds.) Lecture Notes in Computer Science, Computational Logistics, vol. 9855, pp. 18–34. Springer, Berlin (2016)

Agra, A., Christiansen, M., Hvattum, L.M., Rodrigues, F.: Robust optimization for a maritime inventory routing problem. Transp. Sci. **52**(3), 509–525 (2018a)

Agra, A., Requejo, C., Rodrigues, F.: An adjustable sample average approximation algorithm for the production-inventory-routing problem. Networks **72**(1), 5–24 (2018b)

Alvarez-Miranda, E., Cacchiani, V., Lodi, A., Parriani, T., Schmidt, D.: Single-commodity robust network design problem: complexity, instances and heuristic solutions. Eur. J. Oper. Res. **238**(3), 711–723 (2014)

Beasley, J.: A note on solving large p-median problems. Eur. J. Oper. Res. **21**, 270–273 (1985)

Berthold, T.: Measuring the impact of primal heuristics. Oper. Res. Lett. **41**(6), 611–614 (2013)

Berthold, T.: RENS—the optimal rounding. Math. Program. Comput. **6**, 33–54 (2014)

Boland, N., Fischetti, M., Monaci, M., Savelsbergh, M.: Proximity Benders: a decomposition heuristic for stochastic programming. J. Heuristics **22**, 181–198 (2016)

Boussaïd, I., Lepagnot, J., Siarry, P.: A survey on optimization metaheuristics. Inf. Sci. **237**, 82–117 (2013)

Chvatal, V.: A greedy heuristic for the set-covering problem. Math. Oper. Res. **4**(3), 233–235 (1979)

Danna, E., Rothberg, E., Pape, C.L.: Exploring relaxation induced neighborhoods to improve mip solutions. Math. Program. **102**(1), 71–90 (2005)

Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Math. Program. **104**(1), 91–104 (2005)

Fischetti, M., Ljubic, I., Sinnl, M.: Redesigning Benders decomposition for large-scale facility location. Manag. Sci. **63**(7), 2146–2162 (2016)

Fischetti, M., Lodi, A.: Local branching. Math. Program. **98**(1–3), 23–47 (2003)

Fischetti, M., Monaci, M.: Proximity search for 0−1 mixed-integer convex programming. J. Heuristics **20**(6), 709–731 (2014)

Fischetti, M., Monaci, M.: Proximity search heuristics for wind farm optimal layout. J. Heuristics **22**(4), 459–474 (2016)

Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publisher, Boston (1997)

Hvattum, L.M., Esbensen, E.F.: Metaheuristics for stochastic problems. In: Cochran, J., Cox Jr., L., Keskinocak, P., Kharoufeh, J., Smith, J. (eds.) Wiley Encyclopedia of Operations Research and Management Science, pp. 3218–3229. Wiley, New York (2011)

Linderoth, J.T., Savelsbergh, M.W.P.: A computational study of search strategies for mixed integer programming. INFORMS J. Comput. **11**(2), 173–187 (1999)

Morrison, D.R., Jacobson, S.H., Sauppe, J.J., Sewell, E.C.: Branch-and-bound algorithms: a survey of recent advances in searching, branching, and pruning. Discr. Optim. **19**, 79–102 (2016)

Rodrigues, F., Agra, A., Christiansen, M., Hvattum, L.M., Requejo, C.: Comparing techniques for modelling uncertainty in a maritime inventory routing problem. Eur. J. Oper. Res. **277**(3), 831–845 (2019)

Rodrigues, F., Agra, A., Requejo, C., Delage, E.: Lagrangian duality for robust problems with decomposable functions: the case of a robust inventory problem. INFORMS J. Comput. (2020). https://doi.org/10.1287/ijoc.2020.0978

Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS J. Comput. **19**, 534–541 (2007)

Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: Maher, M., Puget, J.-F. (eds.) Principles and Practice of Constraint Programming—CP98, pp. 417–431. Springer, Berlin (1998)

Affiliations

Filipe Rodrigues[1] · Agostinho Agra[2] · Lars Magnus Hvattum[3] · Cristina Requejo[2]

✉ Lars Magnus Hvattum
hvattum@himolde.no

Filipe Rodrigues
frodrigues@iseg.ulisboa.pt

Agostinho Agra
aagra@ua.pt

Cristina Requejo
crequejo@ua.pt

1 Department of Mathematics, ISEG-School of Economics and Management, University of Lisbon, Lisbon, Portugal

2 Department of Mathematics, University of Aveiro, Aveiro, Portugal

3 Faculty of Logistics, Molde University College, Molde, Norway