Computing diverse solutions to optimization problems

Enrico Malaguti, Michele Monaci, Paolo Paronuzzi

DEI "Guglielmo Marconi", Alma Mater Studiorum – Università di Bologna, Bologna, Italy

Abstract

Classical optimization methods determine a single optimal or near-optimal solution for a decision problem. In many applications, however, the decision maker is interested in evaluating a pool of high-quality solutions, to encode fairness-oriented criteria or to obtain a portfolio of alternatives to use in case of unexpected scenarios. In this paper, we consider the problem of computing a pool of near-optimal solutions to a given optimization problem while maximizing their total pairwise distance. In particular, we focus on Mixed-Integer Linear Programming (MILP) problems where a subset of variables are binary and consider the Hamming distance over these variables. We introduce mathematical formulations for the problem, including one model having an exponential number of variables, for which we develop a column-generation based algorithm. Computational experiments show that, for general MILPs, our modelling framework and solution approaches allow to obtain more diverse pools of solutions compared with the existing literature. Furthermore, for a specific application to the assignment problem, we are able to attack instances of large size in a reasonable amount of time.

Keywords: Diversity of solutions; Integer Programming; Mathematical formulations; Computational experiments

1 Introduction

Optimization is traditionally designed to identify a single optimal solution. In many practical applications, however, decision makers prefer to examine multiple alternatives before committing to a final decision. Indeed, a mathematical model may not fully capture all relevant operational aspects, as some performance indicators may be difficult to quantify precisely, and additional constraints or preferences often emerge only after candidate solutions are inspected. For these reasons, providing a portfolio of feasible and high-quality solutions can significantly improve the robustness and practical usefulness of optimization-based decision support systems. A key requirement for such a portfolio is *diversity*: the proposed alternatives should not merely be small perturbations of the same solution, but should exhibit meaningful structural differences.

Beyond classical decision support systems, the availability of diverse solutions can be useful in several contexts. In fairness-oriented decision making, fairness criteria are often difficult to encode completely in optimization models and may involve trade-offs with efficiency [26]. Providing a set of structurally different high-quality solutions enables decision makers to evaluate different alternatives and select the one that best balances the interests of all relevant stakeholders. In some robustness-oriented frameworks, such as K-adaptability, the decision maker precomputes a limited number of candidate policies and selects the most suitable one after uncertainty is revealed [12, 18]. In such

settings, diversity among the precomputed policies can contribute to improving the flexibility of the selection phase. Similarly, in recommendation systems diversity has been increasingly recognized as an important aspect of recommendation quality [3]. Finally, in population-based metaheuristics such as scatter search and evolutionary algorithms, generating an initial set of high-quality yet diverse solutions is often regarded as beneficial for exploring different regions of the solution space (see, e.g., [16]).

Driven by these practical reasons, we study the problem of generating a set of feasible and near-optimal solutions to an optimization problem, while maximizing their diversity. The resulting problem, denoted as *Diversity Optimization Problem (DOP)*, is formally defined in the next section.

1.1 Problem definition and complexity

Let us consider an *underlying* optimization problem formulated as a Mixed-Integer Linear Progam (MILP) as follows

$$\begin{aligned} \min\ & c'x + d'y \\ & (x, y) \in \mathcal{X} \end{aligned}$$

where x denotes a subset of *core binary variables*, that encode the main structural decisions of the problem and determine the structural diversity of solutions, and y denotes the vector of remaining variables. We denote by $\{1, \ldots, n\}$ and by $\{n+1, \ldots, m\}$ the indices of the core binary variables and of the remaining variables, respectively, and by $\mathcal{X} \subseteq \{0,1\}^n \times \mathbb{R}^{m-n}$ the feasible set of the underlying problem. Integrality constraints for some of the y variables, if any, are embedded in the definition of set $\mathcal{X}$. Finally, we assume that the problem has an optimal solution and denote by z^* its (finite) value.

Structural binary variables, to be selected as core variables, appear in most optimization problems. For instance, in the 0-1 knapsack problem one may expect all variables to belong to this set. In contrast, in a facility location problem only the variables deciding which facilities to open typically belong to the core set, and variables determining the quantity served to each customer, although part of the solution, are less relevant for diversity. In this paper, we measure the diversity of two solutions in terms of their Hamming distance restricted to the core binary variables. We remind the reader that, given two vectors $x^k, x^l \in \{0,1\}^n$, their Hamming distance is defined as $d_H(x^k, x^l) = \sum_{i=1}^{n} |x_i^k - x_i^l|$.

Given a positive integer p and a value $\alpha \geq 0$, the *Diversity Optimization Problem (DOP)* consists in computing p feasible solutions, i.e., p pairs of vectors (x, y), whose objective value is not larger than $(1+\alpha)z^*$, while maximizing the total pairwise Hamming distance over the core binary variables. In other words, DOP consists in selecting p pairs of vectors (x, y) such that

$$(x^k, y^k) \in \mathcal{X}, \qquad \forall k = 1, \ldots, p, \tag{1}$$

$$c'x^k + d'y^k \leq z^*(1+\alpha), \qquad \forall k = 1, \ldots, p, \tag{2}$$

and the total pairwise Hamming distance $\sum_{1 \leq k < l \leq p} d_H(x^k, x^l)$ is maximized. From now on, we refer to (1) and (2) as to the feasibility and quality constraints, respectively.

Complexity. It is clear that DOP cannot be easier than the underlying problem, as the former reduces to the latter when $\alpha = 0$ and $p = 1$. Thus, DOP is in general NP-complete. The following result shows that the complexity of DOP is not trivially determined by the underlying problem.

Proposition 1.1. *DOP remains NP-complete even when the underlying problem is solvable by inspection.*

Proof. We trivially extend the NP-completeness proof for MDP with Squared Euclidean Distance given in [8]. Consider a r-regular graph $G = (V, E)$ and the question whether it contains a stable set of cardinality p, which is known to be NP-complete [23]. This can be answered by solving an instance of DOP where the underlying problem is a MIP in which core binary variables x_e are associated with edges $e \in E$ and binary variables y_v are associated with vertices $v \in V$. The underlying problem reads

$$\max \quad \sum_{e \in e} x_e \tag{3a}$$

$$\text{s.t.} \quad \sum_{v \in V} y_v = 1 \tag{3b}$$

$$x_e \leq y_v + y_w, \quad \forall\, e = (v, w) \in E, \tag{3c}$$

$$x_e \in \{0, 1\}, \quad \forall\, e \in E, \tag{3d}$$

$$y_v \in \{0, 1\}, \quad \forall\, v \in V. \tag{3e}$$

The model maximizes the number of selected edges, where an edge can be selected only if one of its endpoint vertex is activated by the y_v variable. As only one vertex can be activated, in any optimal solution we get the profit associated with the vertices incident on the activated vertex, which is r. Consider the associated DOP instance with $\alpha = 0$, i.e., p optimal solutions with maximum diversity have to be provided. If two solutions activate non adjacent vertices, their distance on the x variables is $2r$ (each solution has r components equal to 1, and they all differ). If two solutions activate adjacent vertices, their distance is $2r - 2$ (each solution has r components equal to 1, and one is the same). Thus, if DOP solution value equals $2r\binom{p}{2}$, it means that all activated vertices are disjoint, i.e., the graph contains a stable set of cardinality p.

Finally, observe that the considered underlying problem could can be solved by inspection, instead of using MIP formulation (3). □

2 Related literature

The idea of exploiting diversity in optimization has been studied in several, partly disconnected, streams of literature.

A classical line of research considers the *Maximum Diversity Problem (MDP)*, in which a finite set of elements and a symmetric distance (or similarity) matrix are given, and the goal is to select a subset of elements of prescribed cardinality with maximum total pairwise distance. Kuo et al. [15] analyze the computational complexity of MDP, model it via 0–1 linear and quadratic formulations, and study their polyhedral structure.

Recent progress has been achieved on MDP by Lei et al. [17], who introduce two exact approaches for its solution: a purely combinatorial algorithm and a method based on a relaxed formulation. These approaches significantly outperform previously known exact algorithms for MDP on standard benchmark instances. According to the survey by Martí et al. [19], the best-performing heuristic for MDP is the memetic algorithm proposed by Zhou et al. [27], which combines population-based search with local improvement and has demonstrated strong performance across a wide range of instances.

A second, more recent, stream of research concerns the generation of *multiple* near-optimal and structurally different solutions to a combinatorial optimization model. Given a Mixed Integer Programming (MIP) model, Danna and Woodruff [6] define MIPDIV(p, q, D) as the problem of finding p different feasible solutions within $q\%$ of the optimal value, while maximizing a given diversity measure D. In their approach, they consider a large set of near-optimal MIP solutions produced by an "oracle" (CPLEX 11), and study how to select a subset of p different solutions that is as diverse as possible. They propose a classical MDP formulation for the resulting problem, and develop local search and screening heuristics.

Trapp and Konrad [25] specifically address the case of pure binary integer programs and propose methods to find diverse optimal and near-optimal solutions, with the objective of balancing solution quality and diversity. Starting from an optimal solution, the method sequentially generates alternative solutions that maximize the ratio between a Hamming-type diversity measure and the deterioration in objective value. This approach is extended by Petit and Trapp [24], who introduce a general framework in which additional quality notions are explicitly modeled via auxiliary variables and constraints. Also in this case, solutions are generated sequentially by repeatedly solving modified optimization problems that maximize an objective function balancing (i) diversity with respect to previously generated solutions, (ii) deterioration in the original objective value, and (iii) a selected quality notion.

Finally, an increasing amount of research in the theoretical computer science community refers to the generation of diverse solution sets in explicitly structured combinatorial spaces. A representative contribution is due to Baste et al. [5], who study the problem of selecting multiple solutions that maximize the sum of pairwise Hamming distances. Their main result shows that, for a broad class of combinatorial problems on graphs, dynamic programming algorithms based on tree decompositions (i.e., decompositions of the input graph into a tree-like structure) can be systematically extended to compute diverse sets of solutions. In particular, when the underlying graph has bounded treewidth, the diverse variant of several NP-hard problems becomes tractable, with algorithms whose complexity depends exponentially only on the treewidth and the number of requested solutions. Additionally, several papers by Hanaka and coauthors (e.g., [9, 10, 11]) study diverse variants of specific graph problems such as s–t paths, spanning trees, and shortest paths. These works present polynomial-time algorithms or approximation methods that exploit the specific structure of the underlying problem, and experimentally show (mainly for shortest path variants) that diverse solutions can be computed efficiently on graphs of practical interest.

While methodologically relevant, these works remain mostly theoretical and do not include formulations applicable to general MIP models nor extensive computational experiments. Their scope is orthogonal to the setting considered here, where we propose a general framework to integrate quality and diversity requirements within mathematical programming formulations which allow for systematic computational assessment.

2.1 Problem positioning

As DOP does not explicitly require the p selected solutions to be pairwise distinct, it may happen that the same solution is selected multiple times. This is always the case when set $\mathcal{X}$ includes less than p solutions that satisfy the quality constraint. Since we assume that the underlying problem has an optimal solution, DOP is always feasible.

On the other hand, MIPDIV(p, q, D) [6] maximizes the diversity among p distinct solutions and therefore requires the existence of at least p feasible solutions lying within $q\%$ of the optimal value,

where D is an arbitrary distance function. When this function is the Hamming distance restricted to binary variables, the resulting problem is denoted by MIPDIV(p, q, D_{bin}). If the underlying MILP is a pure binary problem, MIPDIV(p, q, D_{bin}) considerably differs from DOP as it requires the p solutions to be distinct, resulting in a positive distance for each pair of solutions. When, instead, the underlying problem also includes continuous variables, MIPDIV(p, q, D_{bin}) considers as distinct two solutions that differ in a single continuous variable only, though they have the same values for binary variables and their Hamming distance is zero. Thus, in the latter case, MIPDIV(p, q, D_{bin}) differs from DOP only marginally.

The problem proposed by Trapp and Konrad [25] also requires p distinct feasible solutions, while optimizing a ratio between solution quality and diversity, without explicitly enforcing a quality threshold. Consequently, the problem is infeasible whenever less than p feasible solutions exist.

From an algorithmic viewpoint, our framework aims at computing the entire set of p solutions simultaneously while maximizing the total pairwise Hamming distance. This differs from the two-phase approach of Danna and Woodruff [6], where a large pool of near-optimal solutions is first generated and a subset of size p is then selected, as well as from the sequential strategy of Trapp and Konrad [25], which iteratively generates one solution at a time by maximizing its distance from previously identified solutions.

Contributions In this work we deliberately restrict the notion of diversity to the core binary variables, as they capture the structural decisions of the underlying optimization problem; e.g., which facilities are opened, which arcs are selected in a network, etc. This ensures that the Hamming distance provides a clear and interpretable measure of how two solutions differ in terms of their main decision patterns. Extending the distance to integer or continuous variables would require normalization across heterogeneous domains, potentially mixing scales of different nature and compromising interpretability.

The main contributions of this paper are as follows. First, we formalize the problem at hand, providing compact and extended formulations together with valid dual bounds on their optimal values. Second, we develop a frequency-based reformulation that naturally leads to a column generation scheme, integrating feasibility, near-optimality, and diversity within a unified optimization framework. Third, we validate the framework on MIPLIB benchmark instances, demonstrating that the proposed method produces diversity values that are competitive with, and often superior to, those obtained by existing approaches in the literature. Finally, we consider the case of the assignment problem and we design a specialized pricing strategy that exploits the structure of the underlying problem.

3 Compact formulations

In this section we introduce two mathematical formulations for DOP, both having a polynomial number of variables and constraints. To compare these formulations, we make use of combinatorial upper bounds on the optimal solution value, that are introduced hereafter.

3.1 Upper Bounds

A straightforward upper bound on the optimal solution value can be derived by considering that the Hamming distance between any two solutions is at most n. Since the objective function of DOP

consists of the sum of the distances over all $\binom{p}{2}$ possible solution pairs, it follows that:

$$\mathrm{UB}_1 \;=\; n\binom{p}{2}. \tag{4}$$

This bound assumes that every pair of solutions can simultaneously be at maximum distance to any other solution.

However, a more refined analysis can be conducted, resulting in a tighter bound. Let i be the index of an arbitrary core binary variable and denote by $t = \sum_{k=1}^{p} x_i^k$ its *frequency*, i.e., the number of solutions in which the variable takes value one. Variable i gives a unitary contribution to the objective function for every pair of solutions (k, l) such that $x_i^k \neq x_i^l$. By definition of t, this occurs exactly $t(p-t)$ times and, hence, the contribution of the variable to the objective function is given by

$$f(t) = t\,(p-t), \quad t = 0, \ldots, p. \tag{5}$$

This concave quadratic function is symmetric ($f(t) = f(p-t)$), vanishes at $t = 0$ and $t = p$, and is maximized at $t = p/2$ if p is even or at $t \in \{\lfloor p/2 \rfloor, \lceil p/2 \rceil\}$ if p is odd. Hence, the maximum contribution of each core binary variable is:

$$\max_{t \in \{0,\ldots,p\}} f(t) \;=\; \left\lfloor \frac{p^2}{4} \right\rfloor. \tag{6}$$

By considering all core binary variables, we can derive the following upper bound:

$$\mathrm{UB}_2 \;=\; n \left\lfloor \frac{p^2}{4} \right\rfloor, \tag{7}$$

which is reached when there exists a set of p feasible solutions (x^k, y^k) satisfying (1) and (2), such that each core binary variable attains an optimal frequency across the selected solutions. Note that $\mathrm{UB}_2 \leq \mathrm{UB}_1$ for all $p \geq 2$, with equality holding only for $p = 2$.

We remark that both upper bounds are purely combinatorial, as they only rely on the fact that the DOP objective is defined over p binary vectors of length n, and do not exploit any structural property of the underlying optimization problem.

3.2 Direct Pairwise Formulation

A natural MIP formulation explicitly models pairwise distances between solutions. To linearize the absolute value in the Hamming distance, we introduce, for each variable i and pair of distinct solutions k and l, an auxiliary binary variable $d_i^{k,l} \in \{0, 1\}$ indicating whether solutions k and l differ on the i-th component or not. The resulting linear MIP, maximizing the total Hamming distance, reads:

$$\max \quad \sum_{1 \leq k < l \leq p} \sum_{i=1}^{n} d_i^{k,l} \tag{8a}$$

$$\text{s.t.} \quad (1), (2),$$

$$d_i^{k,l} \leq x_i^k + x_i^l, \qquad \forall\, 1 \leq k < l \leq p, \quad i = 1, \ldots, n, \tag{8b}$$

$$d_i^{k,l} \leq 2 - (x_i^k + x_i^l), \quad \forall\, 1 \leq k < l \leq p, \quad i = 1, \ldots, n, \tag{8c}$$

$$d_i^{k,l} \in \{0, 1\}, \qquad \forall\, 1 \leq k < l \leq p, \quad i = 1, \ldots, n. \tag{8d}$$

We recall that (1) ensures the feasibility of each selected solution with respect to the underlying problem, whereas (2) enforces the quality requirement. Inequalities (8b) and (8c) define the correct value of the auxiliary variables, whereas constraints (8d) can be omitted as they are redundant.

Despite the conceptual simplicity, formulation (8) suffers from significant practical limitations. First, the number of variables $d_i^{k,l}$ grows as $\mathcal{O}(np^2)$, leading to a rapid increase in problem size even for moderate values of n and p. Second, the linear relaxation of the model is inherently weak, as the next proposition shows.

Proposition 3.1. *The upper bound provided by the LP relaxation of* (8) *can be as weak as* UB_1.

Proof. Consider an instance for which the LP relaxation of $\mathcal{X}$ admits a feasible solution (x, y) with $x_i = 1/2$ for all i. Then, setting $x_i^k = 1/2$ for all $i = 1, \dots, n$ and $k = 1, \dots, p$ yields a feasible solution for the LP relaxation of (8). Since $x_i^k + x_i^l = 1$ and $2 - (x_i^k + x_i^l) = 1$, setting $d_i^{k,l} = 1$ for all i, k, l yields a feasible solution with objective value that equals $n\binom{p}{2} = \mathrm{UB}_1$. □

From the proof of Proposition 3.1 it follows that the upper bound provided by the LP relaxation of (8) cannot be larger than UB_1.

3.3 Frequency-Based Formulation

An alternative formulation for DOP focuses on the frequency of selection of each core binary variable across the p solutions, exploiting the aggregated structure of its objective function.

As already observed in Section 3.1, the total contribution to the objective function of a core binary variable depends solely on the number of solutions in which it is selected. Thus, the total pairwise distance between solutions can be expressed through the frequency function (5). This structural property allows to derive a compact reformulation that substantially reduces the number of auxiliary variables.

We introduce binary assignment variables $b_{i,t} \in \{0, 1\}$, equal to 1 if core binary variable i is selected t times across the p solutions, and 0 otherwise. The resulting model reads:

$$\max \quad \sum_{i=1}^{n} \sum_{t=0}^{p} f(t)\, b_{i,t} \tag{9a}$$

$$\text{s.t.} \quad (1), (2),$$

$$\sum_{t=0}^{p} b_{i,t} = 1 \qquad \forall i = 1, \dots n, \tag{9b}$$

$$\sum_{k=1}^{p} x_i^k = \sum_{t=0}^{p} t\, b_{i,t} \quad \forall i = 1, \dots n, \tag{9c}$$

$$b_{i,t} \in \{0, 1\} \qquad \forall i = 1, \dots, n,\ t = 0, \dots, p. \tag{9d}$$

Objective function (9a) maximizes the total contribution, expressed as sum of the frequency functions of the variables. Constraints (9b) impose that each core variable has a unique frequency, whose value is set by (9c). Finally, binary requirements on the assignment variables are redundant and could be omitted.

Proposition 3.2. *The upper bound provided by the LP relaxation of* (9) *can be as weak as* UB_2.

Proof. Consider an instance for which the LP relaxation of $\mathcal{X}$ admits a feasible solution (x, y) with $x_i = 1/2$ for all i. Then, setting $x_i^k = 1/2$ for all $i = 1, \ldots, n$ and for all $k = 1, \ldots, p$ yields a feasible solution for the LP relaxation of (9). In this solution, each core binary variable is activated $p/2$ times across the p solutions and gives a contribution to the objective function equal to $p^2/4$. If $p^2/4$ is not integer, a valid upper bound on this contribution is $\lfloor p^2/4 \rfloor$, resulting in a total objective value equal to upper bound UB_2. □

From the proof of Proposition 3.2 it follows that the upper bound provided by the LP relaxation of (9) cannot be larger than UB_2.

3.4 Relationship between Linear Relaxations

Propositions 3.1 and 3.2 highlight that, in the worst case, the continuous relaxation of formulation (9) is tighter than that of formulation (8). The following proposition further establishes that the LP relaxation of (9) dominates that of (8).

Proposition 3.3. *Let* z_{LP}^H *and* z_{LP}^F *denote the optimal values of the LP relaxations of the direct pairwise formulation* (8) *and of the frequency-based formulation* (9)*, respectively. Then,*

$$z_{LP}^F \leq z_{LP}^H.$$

Proof. Both objective functions are separable over the core binary variables $i = 1, \ldots, n$. Thus, it is enough to compare the contribution to the objective function associated with a single core binary variable. To simplify notation, we omit index i. Given a set of p feasible and near-optimal solutions (x^k, y^k) $(k = 1, \ldots, p)$ for the underlying problem, let $r := \sum_{k=1}^{p} x^k$ be the frequency of the considered core binary variable across the p selected solutions. Let us rewrite $r = m + f$, where $m = \lfloor r \rfloor$ and $f = r - m$ are the integer and fractional parts of r, respectively.

We show that, for any value of m and f, the contribution to the objective function associated with a core binary variable in the frequency-based formulation coincides with the minimum contribution attainable in the direct pairwise formulation. This implies that the LP relaxation of (9) always provides an objective value at most as large as the one obtained by the LP relaxation of (8).

In the frequency-based formulation, the contribution of the variable to the objective function is obtained by solving

$$\phi^* = \max \left\{ \sum_{t=0}^{p} t(p-t) b_t : \sum_{t=0}^{p} b_t = 1,\ \sum_{t=0}^{p} t\, b_t = m + f,\ b_t \geq 0\ \forall t = 0, \ldots, p \right\},$$

where upper bounds $b_t \leq 1$ are omitted as they are redundant. Since the objective function is strictly concave, the optimal distribution concentrates its weight as close to r as possible. Specifically, if $f = 0$ (i.e., $r = m$ is integer), the maximum is uniquely achieved by setting $b_m = 1$. Otherwise, the optimal solution is obtained by setting $b_m = 1 - f$ and $b_{m+1} = f$. In both cases, the contribution to the objective function is

$$\phi^* = m\,(p - m)\,(1 - f) + (m + 1)\,(p - m - 1)\,f.$$

Observe that this value depends only on the total frequency $m + f$, and not on the specific values of the variable in solutions $x^1, \ldots, x^p$.

As to the direct pairwise formulation, the contribution to the objective function associated with the variable in the LP relaxation is

$$\psi(x) = \sum_{1 \le k < \ell \le p} \min\{x^k + x^\ell,\ 2 - x^k - x^\ell\}.$$

Observe that this figure explicitly depends on the values of the variable in solutions $x^1, \ldots, x^p$. The smallest possible value of $\psi(x)$, among all sets of solutions with the same sum $r = m + f$, is obtained by solving

$$\psi^* = \min \left\{ \psi(x) : \sum_{k=1}^{p} x^k = m + f,\ 0 \le x^k \le 1\ \forall k = 1, \ldots, p \right\}.$$

Function $\psi(x)$ is concave, as it is a sum of concave terms, each defined as the minimum of two affine functions. Therefore, its minimum over a polytope is attained at an extreme point of the feasible region, which is defined by a single equality constraint and box constraints. This implies that any optimal solution has at most one fractional component, all the other components of vector x being equal to 0 or 1. Specifically, there are m components that take value 1 and $p - m - 1$ components that take value 0, i.e., up to a permutation, an optimal solution has the form

$$\underbrace{1, \ldots, 1}_{m \text{ times}}, f, \quad \underbrace{0, \ldots, 0}_{p-m-1 \text{ times}}$$

For such a solution, each pair of components equal to 1 and each pair of components equal to 0 give null contribution to the objective function. In addition, there are $m(p - m - 1)$ pairs of solutions with one component equal to 1 and one component equal to 0; each such pair contributes 1 to the objective function. There are m pairs of solutions where the first component is 1 and the second one has value f, each providing a contribution equal to $1 - f$. Finally, there are $p - m - 1$ pairs of solutions where the first component is f and the second one is 0; each pair contributes f to the objective fucntion.

Summing up, the objective function value is

$$\psi^* = m(p - m - 1) + m(1 - f) + (p - m - 1)f,$$

which reduces to ϕ^*. Therefore, for every vector $x \in [0, 1]^p$ with sum r,

$$\psi(x) \ge \psi^* = \phi^*.$$

This shows that, for each core binary variable and for any feasible values of the x-variables, the LP relaxation of the direct pairwise formulation can assign an objective contribution at least as large as that of the frequency formulation. Summing over all core binary variables, we obtain

$$z^F_{LP} \le z^H_{LP}.$$

□

For example, let $p = 3$. Consider a core variable i for which $x^1_i = 0.8$, $x^2_i = 0.4$, $x^3_i = 0.1$, resulting in a total frequency equal to $r = 1.3$. The frequency-based formulation yields $\phi^* = 2$, while the direct pairwise formulation gives

$$\psi(x) = \min\{1.2, 0.8\} + \min\{0.9, 1.1\} + \min\{0.5, 1.5\} = 0.8 + 0.9 + 0.5 = 2.2.$$

Hence, the direct pairwise relaxation can produce a strictly larger contribution.

On the other hand, if the frequency $r = 1.3$ is split as $x_i^1 = 1$, $x_i^2 = 0.3$, $x_i^3 = 0$, then

$$\psi(x) = \min\{1.3, 0.7\} + \min\{1, 1\} + \min\{0.3, 1.7\} = 0.7 + 1 + 0.3 = 2 = \phi^*.$$

4 Exponential-size Formulation

In this section, we reformulate model (9) by introducing an exponential number of variables, whose associated columns encode feasible solutions of the underlying problem.

Let $\mathcal{X}_q \subseteq \mathcal{X}$ denote the set of pairs of vectors (x, y) that are feasible and satisfy the quality constraint (2). Each such pair of vectors will be referred to as solution or column. For each solution $s \in \mathcal{X}_q$, let $a_i^s \in \{0, 1\}$ be the value of the i-th core binary variable in s (i.e. $a_i^s = x_i$ for that solution), and introduce an integer variable $v_s \in \{0, \dots, p\}$ that counts how many times solution s is selected. Finally, binary variables $b_{i,t}$ are maintained to express the objective function in a linear form. Hence, DOP can be formulated as

$$\max \quad \sum_{i=1}^{n} \sum_{t=0}^{p} f(t)\, b_{i,t} \tag{10a}$$

$$\text{s.t.} \quad (9b), (9d),$$

$$\sum_{s \in \mathcal{X}_q} v_s = p \tag{10b}$$

$$\sum_{s \in \mathcal{X}_q} a_i^s\, v_s = \sum_{t=0}^{p} t\, b_{i,t} \quad \forall i = 1, \dots, n \tag{10c}$$

$$v_s \in \{0, \dots, p\} \qquad \forall s \in \mathcal{X}_q. \tag{10d}$$

Constraints (10b) enforce that exactly p (possibly replicated) solutions are selected from the set $\mathcal{X}_q$. Constraints (10c) link the column-selection variables v_s with the frequency variables $b_{i,t}$, ensuring that the total number of times each core binary variable is activated across the selected solutions is consistent with its assigned frequency.

4.1 Column generation algorithm

Formulation (10) requires to explicitly define all solutions in set $\mathcal{X}_q$, whose size can be exponential with respect to input data. For this reason, even solving the LP relaxation of this formulation can be computationally impractical for large-size instances, and one can resort to column generation techniques. Accordingly, we define a *master problem* that includes a subset of the v variables only. Observe that, in the resulting LP relaxation, constraints $b_{it} \leq 1$ and $v_s \leq p$ are redundant, and let ρ and λ_i $(i = 1, \dots, n)$ denote the dual multipliers of constraints (10b) and (10c), respectively. The reduced profit of a column s is

$$\text{rp}(s) \;=\; 0 - \rho - \sum_{i=1}^{n} \lambda_i a_i^s \;=\; -\Big(\rho + \sum_{i=1}^{n} \lambda_i a_i^s\Big).$$

Thus, finding a column with the largest reduced profit corresponds to solving the following *pricing subproblem*, where each variable x_i defines the value of the corresponding parameter a_i^s in the

column:

$$\min_{(x,y)\in\mathcal{X}_q} \sum_{i=1}^{n} \lambda_i x_i.$$

If the optimal solution value of the pricing problem is less than $-\rho$, the identified column has positive reduced profit and must be added to the master. Otherwise the current master LP is optimal.

The computational complexity of the pricing problem is discussed in the next proposition:

Proposition 4.1. *Solving the pricing problem is at least as difficult as solving the underlying problem.*

Proof. The pricing problem has the same feasible region as the underlying problem, intersected with the additional quality constraint (2). Its objective function only includes the core binary variables x_i, whose coefficients are given by the dual multipliers λ_i. Assuming one has an algorithm for the pricing problem, the same algorithm can be used to solve the underlying problem by setting $q = 0$ and $\lambda_i = 0,\ i = 1, \ldots, n$. If a feasible solution is returned by the algorithm, such a solution is optimal for the underlying problem. □

The Lagrangian relaxation of the pricing problem, obtained by dualizing the quality constraint, reads as

$$\max_{(x,y)\in\mathcal{X}} \Big\{ \sum_{i=1}^{n} \lambda_i x_i - \mu\big(c'x + d'y - z^*(1+\alpha)\big)\Big\}, \tag{11}$$

where $\mu \geq 0$ is the Lagrange multiplier associated with the quality constraint. As well known from Lagrangian duality theory, the Lagrangian relaxation (11) provides a valid dual bound on the optimal value of the pricing problem. Moreover, if its optimal solution (x^μ, y^μ) is feasible for the pricing problem and the corresponding quality constraint is tight, namely $c'x^\mu + d'y^\mu = z^*(1+\alpha)$, then (x^μ, y^μ) is also optimal for the pricing problem. If this is not the case, one can still exploit the Lagrangian relaxation to identify an optimal solution for the pricing problem, especially when $\mathcal{X}$ has a combinatorial structure and the underlying problem can be efficiently solved.

4.2 Heuristic Solution of the Formulation

The column generation scheme described above provides an optimal solution for the LP relaxation of formulation (10). However, it can also be used to derive a heuristic solution as follows.

Once column generation terminates — either by convergence to linear relaxation optimality or due to the time limit — the generated columns define an explicit pool of feasible near-optimal solutions to the underlying problem, i.e., solutions (x, y) satisfying conditions (1) and (2). These columns define an MDP instance, which can be solved by using a specialized algorithm, e.g., the memetic algorithm proposed in [27]. To allow repeated selection of the same solution, each generated column is replicated p times in the MDP instance.

When column generation converges, the resulting LP value provides a valid dual bound, allowing to compute a meaningful optimality gap for the heuristic solution returned by solving the induced MDP instance, and possibly to prove its optimality. Numerical experiments in Section 6 show the computational performance of this solution approach.

5 Application to the Assignment Problem

For an illustration of the column generation approach, we consider the assignment problem as the underlying optimization model.

This choice is motivated by the following two reasons. First, the assignment problem is solvable in polynomial time. This property is particularly advantageous in the context of the pricing subproblem arising in the column generation scheme. When the quality constraint is dualized, the resulting Lagrangian relaxation reduces to an assignment problem that can be solved efficiently. This allows us to explicitly exploit the combinatorial structure of the underlying problem and to design a specialized pricing strategy, thereby illustrating the potential benefits of structure-aware decomposition.

Second, the standard assignment formulation does not exhibit trivial symmetries among feasible solutions: each feasible assignment corresponds to a distinct permutation, and there are no interchangeable decision variables. This makes it particularly suitable for studying diversity generation, as differences in the selected solutions reflect genuine structural variations rather than artifacts of model symmetry.

Let $G = (U, W, E)$ be a complete bipartite graph, where c_{uw} denotes the cost of assigning $u \in U$ to $w \in W$. Binary variables x_{uw} indicate whether edge (u, w) is selected. The assignment problem reads

$$\begin{aligned}
\min \quad & \sum_{u\in U}\sum_{w\in W} c_{uw}x_{uw} \\
\text{s.t.} \quad & \sum_{w\in W} x_{uw} = 1, && \forall u \in U, && \text{(12a)}\\
& \sum_{u\in U} x_{uw} = 1, && \forall w \in W, && \text{(12b)}\\
& x_{uw} \in \{0,1\}, && \forall (u,w) \in E. && \text{(12c)}
\end{aligned}$$

Let z^* denote its optimal value. In the DOP setting, we aim at selecting p assignments whose objective values do not exceed $z^*(1+\alpha)$. Let x^k_{uw} denote the assignment variable for solution $k = 1, \ldots, p$. Each solution must satisfy constraints (12), and the quality constraint

$$\sum_{u\in U}\sum_{w\in W} c_{uw}x^k_{uw} \le z^*(1+\alpha).$$

As discussed earlier, the resulting compact formulation is typically too large and computationally inefficient, motivating a column–generation approach.

5.1 Column–generation formulation

Let $\mathcal{X}_q$ be the set of all feasible assignments satisfying the quality constraint. Each column $s \in \mathcal{X}_q$ corresponds to a feasible assignment and is characterized by coefficients

$$a^s_{uw} = \begin{cases} 1 & \text{if edge } (u,w) \text{ is used in assignment } s,\\ 0 & \text{otherwise.}\end{cases}$$

We introduce integer variables v_s counting how many times assignment s is selected, and keep the frequency variables $s_{uw,t}$ to linearize the objective. The master problem is a direct specialization of (10), with indices i replaced by pairs uw.

5.1.1 Pricing problem

Let ρ be the dual multiplier associated with constraint (10b) imposing to select p solutions, and let λ_{uw} be the dual multipliers associated with the frequency constraints (10c) for edges (u, w). The reduced profit of a column s is

$$\mathrm{rp}(s) = -\Big(\rho + \sum_{u\in U}\sum_{w\in W} \lambda_{uw} a^s_{uw}\Big).$$

Hence, the pricing subproblem consists in finding a feasible assignment that minimizes the weighted sum of its selected edges, while respecting the quality constraint:

$$\begin{aligned}
\min \quad & \sum_{u\in U}\sum_{w\in W} \lambda_{uw} x_{uw} && \\
\text{s.t.} \quad & \sum_{w\in W} x_{uw} = 1, && \forall u \in U, \\
& \sum_{u\in U} x_{uw} = 1, && \forall w \in W, \\
& \sum_{u\in U}\sum_{w\in W} c_{uw} x_{uw} \le z^*(1+\alpha), && \\
& x_{uw} \in \{0,1\}, && \forall (u,w) \in E.
\end{aligned} \tag{13}$$

5.1.2 Solving the Pricing Problem

Problem (13) is known as the Constrained Assignment Problem (CAP). This problem has been extensively studied in the literature, and several exact approaches have been proposed, including branch-and-bound and branch-and-cut algorithms tailored to handle the additional knapsack-type constraint (see, e.g., [1, 20]).

In the context of column generation, the pricing problem must be solved repeatedly and efficiently. Rather than relying on a general-purpose MIP solver at each iteration, we exploit the combinatorial structure of the assignment problem and adopt the approach proposed by Aggarwal [4], which reduces the constrained problem to a sequence of instances of the assignment problem, and can be implemented independently of a commercial optimization software.

We further tailor Aggarwal's scheme to the column generation setting by introducing problem-specific stopping and bounding strategies, detailed in the following.

This scheme starts by defining the Lagrangian objective function, obtained by dualizing the quality constraint with a given lagrangian multiplier $\mu \ge 0$

$$L(\mu, x) = \sum_{u\in U}\sum_{w\in W} (\lambda_{uw} + \mu c_{uw}) x_{uw} - \mu\, z^*(1+\alpha). \tag{14}$$

Thus, the Lagrangian relaxation of (13) takes the form:

$$\min_{x\in\mathcal{X}} L(\mu, x). \tag{15}$$

For any fixed value of μ, (15) is an assignment problem, that can be solved in polynomial time and provides a lower bound on the optimal solution value.

The first stage of the Aggarwal's method (Procedure PLAKP in [4]) iteratively solves instances of (15), updating multiplier μ. It terminates in a finite (and polynomial) number of iterations and returns the optimal dual multiplier μ^*, i.e., the one providing the largest lower bound. If the assignment $\hat{x}$ obtained when using multiplier μ^* satisfies the quality constraint at equality, it is also optimal for (13) and the algorithm terminates. If $\hat{x}$ is not feasible, the procedure returns the best feasible solution found, say $\bar{x}$ with value $\bar{z}$, for CAP; a feasible solution always exists, e.g., an optimal solution for the underlying assignment problem. For further details on this first stage, we refer the reader to [4].

The second stage of Aggarwal's method (Procedure $P[AP(\lambda^*)_k]$ receives in input the lower bound LB and upper bound UB computed in the first stage, and is based on Murty's ranking algorithm [21, 22], which enumerates all feasible assignments in non-decreasing order of cost. In Aggarwal's method, Lagrangian costs $\lambda_{uw} + \mu^* c_{uw}$ are used, so that each assignment generated provides a lower bound on the cost of any feasible solution that may be found in later iterations, possibly improving LB. In addition, if the generated assignment satisfies the quality constraint, it also yields a new upper bound, possibly improving UB. Murty's algorithm terminates as soon as LB equals or exceeds UB.

Preliminary experiments showed that, in many cases, CAP is solved to optimally in the first stage and that a few iterations are needed otherwise. However, in rare but critical cases, the number of iterations needed to identify an optimal solution of CAP becomes very large, leading to a severe computational bottleneck. To mitigate this, we introduce two important mechanisms:

1. **Primal cutoff heuristic.** After σ iterations of Murty's algorithm, we prematurely stop the procedure to avoid excessive enumeration. If $UB \leq -\rho$, then we return the corresponding variable with positive reduced profit, that can be added to the current master problem. Otherwise, we solve the pricing problem as a MIP and return the corresponding solution.

2. **Early termination via dual bound.** We stop the algorithm as soon as $LB \geq -\rho$, meaning that no variable with positive reduced profit exists, and the column generation terminates.

Algorithm 1 summarizes the complete method used to solve the pricing problem.

6 Computational Results

This section presents a computational assessment of the proposed approaches for DOP. The goals of the experiments are twofold:

(i) to validate the general modeling approach by comparing the results obtained by solving DOP with the closest existing methods in the literature, namely [6] and [25];

(ii) to demonstrate that, when the underlying optimization problem exhibits exploitable combinatorial structure, the proposed column–generation approach can be strengthened through specialized pricing algorithms, as illustrated on the assignment problem.

Our algorithms are implemented in C++ and use the Gurobi Optimizer version 13 to solve all LP and MIP models. Unless otherwise specified, Gurobi parameters are left at their default values. All comparisons between floating-point values are performed using a relative tolerance of

Algorithm 1 Pricing Solver(ρ, σ)

Run Procedure PLAKP by [4] and get $\mu^*, \hat{x}, \bar{x}, \bar{z}$;
if $\hat{x}$ satisfies the quality constraint at equality **then**
    Return $\hat{x}$;
Initialize $LB \leftarrow L(\mu^*, \hat{x})$, $UB \leftarrow \bar{z}$, $it \leftarrow 1$ and set $\bar{x}$ as incumbent solution;
while $LB < UB$ **do**
    if $(it > \sigma)$ **then**
        if $(UB \leq -\rho)$ **then**
            Return incumbent solution;
        else solve CAP as a MIP and **Return** MIP solution;
    Define next assignment $\hat{x}$ according to Murty's algorithm;
    $LB \leftarrow L(\mu^*, \hat{x})$;
    if $\hat{x}$ satisfies the quality constraint and improves upon UB **then**
        Update UB and set $\hat{x}$ as incumbent solution;
    if $LB \geq -\rho$ **then**
        Return (no variable with positive reduce profit exists);
    $it \leftarrow it + 1$;
Return incumbent solution;

10^{-4}. Experiments are performed on a computer equipped with an AMD Ryzen 3960X processor clocked at 3.8 GHz, with 128 GB RAM, running Linux. Time limits are set to 3600 seconds unless otherwise specified.

Preliminary computational experiments showed that the direct application of a solver to the pairwise formulation (8) is not competitive with other solution methods for the instances considered in our tests. This is due to the large number of auxiliary variables and the weakness of the LP relaxation of this formulation. For this reason, we do not report computational results for this formulation, and only consider the frequency-based formulation (9) and the column-generation based approach. As to the latter, the column generation procedure is initialized with a single variable corresponding to the optimal solution of the underlying problem, and a single column is added at each iteration. When solving the master problem we use Gurobi's barrier method without crossover. Finally, a time limit equal to 60 seconds is imposed when solving the MDP instance induced by the generated columns.

For the experiments aimed at comparing the proposed framework with existing approaches, all optimization runs are performed in single-threaded mode. For the assignment problem experiments, we also report results obtained using multi-threaded computational settings.

6.1 Comparison with existing methods

Our first set of experiments is aimed at comparing our approaches and solutions with their counterparts in the literature for closely related diversity problems. More into details, we consider Danna and Woodruff [6], who generate a subset of near-optimal distinct solutions from a precomputed solution pool, and Trapp and Konrad [25], who iteratively construct a set of distinct feasible solutions by balancing quality and diversity.

To ensure a meaningful comparison, we set $p = 10$, following the experimental setup adopted

in both papers. For the quality requirement, we use $\alpha = 0.01$, as in [6]; this requirement is not explicitly imposed in [25].

6.1.1 Benchmark Instances

For this set of experiments we consider a benchmark of instances taken from those considered in Danna and Woodruff [6] and in Trapp and Konrad [25].

Danna and Woodruff [6] considered 44 instances from MIPLIB 2003 [2] that (i) were solved to proven optimality within 30 minutes using CPLEX 11 on a 3.40 GHz dual-core Pentium processor with 2 GB of memory; and (ii) have at least 10 distinct solutions with an objective value within 1% of optimality.

Trapp and Konrad [25] set their algorithm to retrieve 10 distinct solutions and reported results for 40 binary integer programming instances drawn from MIPLIB 2003 [2] and MIPLIB 2010 [14]. They selected those instances that are classified as *easy* and for which their method was able to generate at least two distinct solutions. From this subset, we further restrict our analysis to the instances where their method generated at least 10 solutions. In total, this produced a testbed with 64 instances.

6.1.2 Results and Analysis

Table 1 reports the detailed computational results on the MIPLIB benchmark instances described in Section 6.1.1. Instances are divided into two groups according to the structure of the underlying problem. The first block contains purely binary instances, while the second block includes instances containing integer and/or continuous variables as well.

Columns under **FQ** report the lower bound (LB) obtained by the frequency-based formulation (9), the corresponding upper bound (UB), the optimality gap (calculated as $100 \cdot (\text{UB} - \text{LB})/\text{UB}$), and the total running time. Similarly, columns under **CG** report the lower bound obtained by the column generation approach, together with the corresponding upper bound, optimality gap, and running time. All lower and upper bounds reported for **FQ** and **CG** are multiplied by $\frac{2}{np(p-1)}$, to make the values directly comparable with those reported in [6, 25]. Whenever the time limit is reached, the table reports "TL" in the corresponding time column.

To account for the differences highlighted in Section 2.1, an asterisk (*) preceding a lower bound value indicates that less than 10 distinct feasible solutions (i.e., solutions that differ each other by at least one binary core variable) satisfying the quality constraint were identified for the corresponding instance, implying that some solutions are selected multiple times in the corresponding DOP solution. The last two columns, labeled **DW** and **TK**, report the best diversity values obtained by the approaches of Danna and Woodruff [6] and Trapp and Konrad [25], respectively. For clarity and conciseness, we report, for each instance and for both competing methods, the best diversity value obtained among the variants that were proposed. The differences among variants within the same study are typically modest and do not affect the overall comparison. In [6], a time limit of one hour was imposed to generate a pool of at most 10,000 feasible solutions using the one-tree strategy implemented in CPLEX 11 [7]. The authors do not provide detailed computational times for their algorithms. However, they report that the exact selection method reached a 10-day time limit in 28 cases, whereas the heuristic methods required at most a few minutes. As to the computing times of the algorithm proposed in [25], they exhibit high variability, ranging from tens of seconds to several tens of thousands of seconds, depending on the instance and version of the algorithm. Since experiments were conducted on substantially different hardware and solver versions, a direct

runtime comparison would be neither reliable nor informative. Hence, detailed timing analyses of the competing approaches are omitted.

For each instance, the best value among the lower bounds produced by **FQ**, **CG**, and **DW** is highlighted in boldface. We exclude **TK** from this comparison since this method may potentially achieve larger diversity values by including solutions that do not satisfy the quality constraint. However, the computational study in [25] indicates that the majority of generated solutions lie within 1% of optimality. For this reason, we still consider the comparison meaningful.

Table 1: Comparison with Danna and Woodruff (**DW**) [6] and Trapp and Konrad (**TK**) [25].

	FQ				**CG**				**DW**	**TK**
Instance	LB	UB	Gap	Time	LB	UB	Gap	Time		
acc-tight5	0.0227	0.1617	85.96	TL	**0.1455**	0.1467	0.77	2059.04		0.1431
acc-tight6	*0.0228	0.1654	86.23	TL	**0.1536**	0.1542	0.38	2342.48		0.1512
air03	*0.0030	0.0031	4.66	TL	**0.0030**	0.0030	0.07	104.40		0.0009
air04	0.0053	0.0167	68.39	TL	**0.0125**	-	-	TL		0.0029
air05	0.0058	0.0150	61.14	TL	**0.0128**	0.0129	0.38	3360.71		0.0070
bley_xl1	*0.0000	0.8569	-	TL	***0.0066**	0.0066	0.00	39.04		0.0064
cap6000	**0.0331**	0.0332	0.27	TL	*0.0326	-	-	TL	0.0088	0.0022
disctom	0.0000	3.6634	-	TL	***0.5271**	-	-	TL	0.4685	
eil33-2	***0.0000**	0.0022	-	TL	***0.0000**	0.0000	0.00	0.02		0.0020
eilB101	0.0008	0.0121	93.50	TL	***0.0048**	0.0048	0.00	4.69		0.0096
l152lav	*0.0117	0.0243	52.07	TL	**0.0229**	0.0231	0.87	399.22	0.0225	0.0121
lseu	***0.1273**	0.1273	0.00	25.87	***0.1273**	0.1273	0.00	2.57		0.1498
mine-166-5	0.0574	0.1453	60.52	TL	**0.0643**	0.0646	0.33	669.72		0.0056
mitre	**0.0494**	0.0494	0.00	46.59	*0.0479	-	-	TL		0.0037
mod008	**0.0223**	0.0223	0.00	55.37	***0.0223**	0.0223	0.00	11.38	0.0220	0.0170
mod010	0.0243	0.0266	8.83	TL	**0.0256**	0.0259	1.16	798.63	0.0230	0.0059
neos-1440225	0.0107	0.0549	80.48	TL	**0.0537**	-	-	TL		0.0538
neos-777800	0.0023	0.0250	90.97	TL	***0.0250**	0.0250	0.00	13.55		0.0250
neos-957389	**0.0250**	0.0250	0.00	490.43	**0.0250**	0.0250	0.32	145.63		0.0250
ns1688347	**0.0329**	0.0329	0.00	209.43	0.0327	0.0329	0.60	114.26		0.0320
nw04	0.0000	0.0001	66.60	TL	**0.0001**	0.0001	0.00	125.22	**0.0001**	0.0001
opm2-z7-s2	0.0108	0.3205	96.64	TL	***0.0993**	-	-	TL		0.0797
p0033	***0.2707**	0.2707	0.00	0.26	***0.2707**	0.2707	0.00	3.67	**0.2707**	0.1778
p0201	**0.1705**	0.1709	0.26	TL	**0.1705**	0.1705	0.00	11.85	**0.1705**	0.1650
p0282	**0.1882**	0.1882	0.00	449.26	0.1854	0.1882	1.47	188.44	0.1118	0.0072
p0548	**0.0931**	0.0931	0.00	8.58	0.0923	0.0932	1.00	82.29	0.0587	0.0371
p2756	**0.0957**	0.0957	0.01	1475.80	0.0848	-	-	TL		0.0627
reblock67	0.0884	0.3087	71.35	TL	**0.1417**	-	-	TL		0.0110
stein27	**0.4889**	0.5547	11.87	TL	*0.4872	0.4889	0.34	5.93	**0.4889**	0.4889
stein45	0.3704	0.5551	33.27	TL	***0.4889**	0.4889	0.00	17.89	0.4849	0.4840
tanglegram2	0.0120	0.2127	94.34	TL	**0.0467**	0.0469	0.39	1384.23		0.0318
10teams	0.0233	0.0444	47.61	TL	***0.0439**	0.0444	1.22	621.10	0.0420	
aflow30a	0.0496	0.0837	40.69	TL	***0.0644**	0.0644	0.00	38.72	0.0640	
arki001	0.4315	0.5032	14.25	TL	**0.4726**	-	-	TL	0.2518	
bell3a	***0.0000**	0.0000	0.00	0.13	***0.0000**	0.0000	0.00	0.00	**0.0000**	
bell5	**0.3911**	0.4007	2.40	TL	**0.3911**	0.3911	0.00	20.49	0.3089	
dcmulti	0.2836	0.3677	22.88	TL	**0.3001**	0.3007	0.20	135.82	0.2859	
dsbmip	0.2658	0.2692	1.24	TL	**0.2664**	0.2675	0.42	105.67	0.1836	
fiber	***0.0222**	0.0222	0.00	2734.77	***0.0222**	0.0222	0.00	27.37	0.0221	
fixnet6	**0.0539**	0.0542	0.65	TL	***0.0539**	0.0539	0.00	56.87	0.0531	
gen	**0.3693**	0.3702	0.25	TL	**0.3693**	0.3694	0.04	76.42	0.3369	
gesa2	0.1767	0.1823	3.10	TL	***0.1792**	0.1795	0.21	418.71	0.0837	
gesa2-o	**0.1906**	0.1909	0.12	TL	0.1803	0.1829	1.42	624.72	0.0612	
gesa3	0.1056	0.1159	8.96	TL	***0.1082**	0.1082	0.00	372.28	0.0395	
gt2	**0.0833**	0.0833	0.00	0.12	**0.0833**	0.0833	0.00	6.56	**0.0833**	
khb05250	0.2361	0.2565	7.94	TL	***0.2370**	0.2370	0.00	6.99	0.2324	
mas76	0.0240	0.1498	83.98	TL	**0.1209**	0.1212	0.24	1552.31	**0.1209**	
misc03	***0.0872**	0.0872	0.00	1370.32	**0.0872**	0.0872	0.00	6.81	**0.0872**	
misc06	0.3857	0.3919	1.57	TL	**0.3861**	0.3903	1.07	2887.97	0.2520	
misc07	**0.0707**	0.1237	43.83	TL	**0.0707**	0.0707	0.00	77.13	**0.0707**	
mod011	0.0553	0.1697	67.39	TL	***0.1250**	0.1250	0.00	293.02	0.1113	
modglob	0.4866	0.4991	2.50	TL	**0.4889**	-	-	TL	0.3946	
mzzv42	0.0101	0.0631	83.99	TL	**0.0336**	-	-	TL	0.0168	
pp08a	0.1146	0.3313	65.41	TL	***0.2406**	0.2406	0.00	25.81	0.2385	
pp08aCUTS	0.1285	0.3146	59.16	TL	***0.2406**	0.2406	0.00	25.14	0.2385	
qiu	0.1093	0.5306	79.41	TL	***0.4111**	0.4111	0.00	68.24	**0.4111**	
qnet1	0.0247	0.0394	37.49	TL	***0.0307**	0.0307	0.22	95.81	0.0298	
qnet1_0	0.0304	0.0321	5.42	TL	**0.0306**	0.0307	0.34	86.30	0.0299	
rgn	**0.0800**	0.0800	0.00	0.77	0.0796	0.0800	0.56	12.06	0.0796	
rout	0.0801	0.1168	31.39	TL	0.1079	0.1081	0.27	329.32	**0.1080**	
set1ch	0.1635	0.1701	3.87	TL	**0.1687**	0.1700	0.76	653.08	0.1413	
tanglegram1	0.0052	0.2873	98.20	TL	***0.0767**	-	-	TL	0.0114	
vpm1	**0.1365**	0.1365	0.00	3.86	0.1357	0.1365	0.58	13.70	0.1360	
vpm2	0.0712	0.1189	40.16	TL	***0.0765**	0.0765	0.00	9.14	0.0742	

The computational results reported in Table 1 show a clear advantage of the proposed DOP approaches over the existing methods from the literature.

From a computational perspective, **CG** appears substantially more effective than **FQ**. While **FQ** reaches the one-hour time limit in 49 instances, this occurs for **CG** in 12 cases only. Moreover, **CG** is able to prove optimality in 25 instances, compared to 14 instances for **FQ**. These results indicate that column generation provides significantly tighter bounds and the application of the column-generation heuristic is a viable strategy for tackling hard instances.

A similar trend emerges when comparing the achieved diversity values on the 45 instances that have been considered by Danna and Woodruff [6]. Considering the comparison among **FQ**, **CG**, and **DW**, the numbers of instances for which each method attains the best value are 18, 37 and 11, respectively (including ties). Overall, these results suggest that the approaches described in this manuscript consistently leads to stronger diversity values than those produced by **DW**.

Finally, although **TK** addresses a related but not directly comparable optimization problem, the obtained diversity values are in most cases still dominated by those produced by our algorithms. This provides additional empirical evidence of the effectiveness of the proposed framework in generating highly diverse near-optimal solutions.

Computational results also confirm that, in many cases, problems DOP and MIPDIV(p, q, D_{bin}) can be directly compared each other, although the former allows the same solution to be selected multiple times. For pure binary instances, corresponding to the first group in the table, algorithm **CG** selects 10 distinct solutions in 18 out of 31 cases. Among these 18 instances, 6 belong to the benchmark set considered by Danna and Woodruff [6], allowing for a direct comparison between the solutions obtained by solving DOP and those returned by **DW**. In four of these cases, **CG** achieves higher diversity than **DW**, while the remaining two instances result in a tie. Algorithm **FQ** selects 10 distinct solutions in 24 out of the 31 pure binary instances. Ten of these instances were also considered by Danna and Woodruff [6]: in five cases, **FQ** returns a solution with higher diversity, in three cases its solution has lower diversity, and in the remaining two cases both methods attain the same diversity. A similar comparison is less straightforward for the 33 instances that include continuous and/or general integer variables. On those instances, **FQ** and **CG** select 10 distinct solutions in 30 and 18 cases, respectively. However, while **FQ** and **CG** solutions are distinguished by their different core binary variables, distinct solution in **DW** may differ in the continuous variables only.

6.2 Application to the Assignment Problem

We now evaluate the benefits of exploiting problem-specific structure in the pricing phase when specializing the framework to the assignment problem, as discussed in Section 5.

We solve the underlying assignment problems by using the algorithm of Jonker and Volgenant [13], implemented via an open-source library[1]. This method is based on shortest augmenting paths and has cubic worst-case time complexity.

Within Murty's framework, multiple assignment subproblems must be solved. However, as noted by Miller et al. [21], after the first optimal solution is computed, each subsequent subproblem can be processed with a single augmentation step. Since each augmentation can be performed in a time that is quadratic in the number of rows and columns, this substantially reduces the cost of the successive assignment computations within Algorithm 1.

[1] `https://github.com/gatagat/lap`

6.2.1 Benchmark Instances

We generate instances of the assignment problem with two cost structures:

1. **Correlated.** Costs are generated according to

$$c_{uw} = \max\{1, \lfloor a_u + b_w + \varepsilon_{uw} \rfloor\},$$

   where a_u and b_v are integers uniformly drawn in $[1, 10]$, and ε_{uw} is a Gaussian perturbation with zero mean and standard deviation equal to 2.

2. **Uniform.** Costs c_{uw} are independently drawn as integers in $[1, 100]$;

We consider bipartite graphs with number of rows and columns $N = |U| = |V|$ taking values from 100 to 800, with step size 100. For each size and cost class, 10 instances are randomly generated. As done in the previous experiments, We set $p = 10$ and $\alpha = 0.01$. Results for correlated instances with $N \geq 600$ are not reported, since all methods hit the time limit on every such instance. For this reason, we report results for the remaining 130 instances only.

6.2.2 Results and Analysis of Column Generation

We first analyze the behavior of the column generation algorithm. In particular, we compare two implementations of the pricing subproblem: a baseline version, denoted **CG-G**, in which pricing is solved entirely through a general-purpose MIP solver (Gurobi), and a problem-specific version, denoted **CG-S**, which applies the combinatorial pricing procedure described in Section 5.1.2, possibly resorting to Gurobi after σ iterations. Preliminary tests indicated that the value $\sigma = 20$ provides the best trade-off between pricing effort and fallback frequency; larger values led to similar results, while smaller values caused the solver to be invoked too frequently, resulting in worse performance.

Since the test machine provides 48 hardware threads, we also evaluate a parallel implementation of both approaches, exploiting all available threads. In the case of **CG-G**, parallelism is entirely delegated to Gurobi by allowing the solver to use all available threads when solving an istance of the pricing problem. For **CG-S**, instead, parallelism is introduced within Algorithm 1. Specifically, Murty's algorithm constructs a ranking tree whose nodes correspond to assignment problems with additional fixing constraints. Since these assignment problems are independent, they can be processed simultaneously and distributed among the available threads. The master problem is always solved in single-threaded mode, as preliminary experiments showed that exploiting multiple threads at the master level did not provide any computational advantage.

Table 2 reports aggregated results for the assignment problem instances. Results are presented separately for the two classes, Correlated (*Corr*) and Uniform (*Unif*), and for each problem size N. For each configuration, we report the number of instances for which column generation was able to solve to optimality the linear relaxation of (10) within the time limit (column *Solved*) and the average running time in seconds (column *Time*), computed over the solved instances only. Results are reported for both variants of the proposed framework, **CG-G** and **CG-S**, executed using either one or 48 threads for the pricing phase. The last row of each block reports the total number of solved instances and the average running times across all tested sizes.

For the *Corr* instances, **CG-G** solves 40 out of 50 instances with one thread and 39 instances with 48 threads, while **CG-S** solves 40 and 41 instances, respectively. For $N = 500$, only **CG-S** with 48 threads is able to solve one instance within the time limit. The average running times show that parallelization has essentially no effect on **CG-G**, whose performance remains almost

Table 2: Solution of the linear relaxation of (10) for the assignment problem.

		1 Thread				48 Threads			
		CG-G		**CG-S**		**CG-G**		**CG-S**	
Type	*N*	*Solved*	*Time*	*Solved*	*Time*	*Solved*	*Time*	*Solved*	*Time*
Corr	100	10	15.45	10	8.73	10	15.87	10	8.36
	200	10	209.71	10	142.07	10	218.90	10	119.09
	300	10	781.44	10	591.66	10	795.59	10	486.00
	400	10	2379.72	10	2099.46	9	2333.02	10	1698.03
	500	–	–	–	–	–	–	1	3306.77
		40	846.58	40	710.48	39	802.58	41	644.43
Unif	100	10	4.78	10	2.37	10	4.65	10	2.10
	200	10	46.58	10	39.15	10	43.78	10	26.36
	300	10	175.76	10	145.33	10	165.25	10	104.43
	400	10	390.45	10	331.04	10	373.59	10	235.42
	500	10	910.59	10	800.76	10	936.80	10	539.65
	600	10	1860.49	10	1612.07	10	1782.73	10	1060.89
	700	5	3161.93	6	3089.94	5	2908.28	10	2216.01
	800	–	–	–	–	–	–	3	3380.63
		65	764.56	66	724.95	65	732.45	73	712.20

unchanged when increasing the number of threads from 1 to 48. In contrast, **CG-S** benefits more noticeably from parallel execution, solving one additional instance and reducing the average running time from 710.48 to 644.43 seconds.

For the *Unif* instances, both algorithms solve all instances up to $N = 600$, independently on the number of threads. For $N = 700$, **CG-G** solves 5 instances regardless of the number of threads, while **CG-S** improves from 6 solved instances with one thread to all 10 instances with 48 threads. For $N = 800$, neither single-threaded method solves any instance within the time limit, whereas **CG-S** with 48 threads solves 3 instances. Also in this case, the effect of parallelization is limited for **CG-G**, whose average running time changes only marginally (764.56 vs. 724.95 seconds). On the other hand, **CG-S** increases the number of solved instances from 66 to 73 when moving from one to 48 threads, while maintaining a comparable average running time on the solved instances (724.95 vs. 712.20 seconds).

6.2.3 Results and Analysis when computing Integer Solutions

Here, we compare two methods for producing integer solutions for the assignment problem instances. The first one, denoted by **FQ**, is the frequency-based formulation (9), solved directly as a MIP. The second one, denoted by **MDP**, corresponds to the second phase of the solution strategy described in Section 4.2: given the columns generated by column generation, define an MDP instance and solve it heuristically by means of the memetic algorithm proposed in [27].

Also in this set of experiments, we evaluate the performance of the algorithms both in single-thread configuration and for the case where 48 threads are available. For **MDP**, the set of columns include near-optimal solutions generated by the corresponding **CG-S** variant: in the single-threaded configuration, we use the columns produced by **CG-S** with one thread, whereas in the 48-thread

configuration we use the columns produced by **CG-S** with 48 threads. In the latter configuration, parallelization is obtained by independently executing 48 times the memetic algorithm, initialized with different random seeds, and returning the best solution found. Conversely, for **FQ**, parallelization is exploited by allowing Gurobi to use all available threads during the branch-and-bound search.

Table 3 reports aggregated computational results for the integer solution phase of the assignment problem instances. Results are presented separately for the two instance classes, Correlated (*Corr*) and Uniform (*Unif*), and for each problem size N.

For **FQ**, column *Solved* reports the number of instances solved to proven optimality within the time limit. Column *Gap* reports the average percentage optimality gap. Column *Time* reports the average running time in seconds, computed only over the instances solved to proven optimality. Gaps for **FQ** with one thread are not reported, as all instances are either solved to optimality (in which case the gap is zero) or terminate with $LB = 0$ (resulting in a gap equal to 100) but for one *Unif* instance with $N = 200$, for which the final percentage gap is 0.08.

For **MDP**, column *Gap* reports the average percentage gap of the heuristic solution with respect to the upper bound provided by **FQ** with 48 threads. This choice makes the reported gaps directly comparable with those of **FQ**. Column *Time* reports the average running time in seconds of the MDP heuristic phase. Both gap and time values for **MDP** are computed over all instances.

Table 3: Computational results for the integer solution phase of the assignment problem instances.

		1 Thread				48 Threads				
		FQ		**MDP**		**FQ**			**MDP**	
Type	N	*Solved*	*Time*	*Gap*	*Time*	*Solved*	*Gap*	*Time*	*Gap*	*Time*
Corr	100	10	109.53	1.03	31.63	10	0.00	28.26	0.97	58.28
	200	6	2515.42	1.65	60.15	10	0.00	1094.55	1.22	62.10
	300	–	–	1.99	60.37	9	0.01	1856.25	1.25	64.69
	400	–	–	3.11	60.94	1	23.63	3566.67	1.54	70.42
	500	–	–	3.34	60.96	–	26.58	–	1.63	74.94
		16	1011.74	2.22	54.81	30	10.04	1050.03	1.32	66.09
Unif	100	10	37.51	0.09	7.38	10	0.00	18.59	0.09	15.04
	200	9	943.97	0.47	25.68	10	0.00	210.59	0.42	53.93
	300	–	–	0.77	52.30	10	0.00	308.54	0.73	60.75
	400	–	–	1.24	59.42	9	3.51	1018.14	1.19	61.08
	500	–	–	1.47	60.09	8	7.32	1314.72	1.47	61.77
	600	–	–	1.69	60.17	7	11.37	2168.86	1.52	62.69
	700	–	–	2.24	60.33	3	23.51	2928.02	1.78	64.60
	800	–	–	1.66	60.22	1	34.33	3586.36	2.10	66.35
		19	466.89	1.20	48.20	58	10.00	907.08	1.16	55.78

For the *Corr* instances, **FQ** solves 16 out of 50 instances with one thread and 30 instances with 48 threads, showing that the branch-and-bound search clearly benefits from parallelization. However, the exact formulation still struggles on the larger instances: with 48 threads, only one instance is solved for $N = 400$, and no instance is solved for $N = 500$. In addition, the average gap of **FQ** with 48 threads increases substantially on the hardest sizes, reaching 26.58 for $N = 500$.

On the same instances, **MDP** provides much more stable solution quality. With one thread, it obtains an average gap of 2.22 in 54.81 seconds, while the 48-thread multi-start version reduces the average gap to 1.32. The improvement is consistent across all problem sizes, and the percentage gap remains below 2.00 even for $N = 500$, where **FQ** is unable to solve any instance and leaves a much larger residual gap. This indicates that, although **MDP** does not provide an optimality certificate, it is able to produce high-quality integer solutions.

For the *Unif* instances, **FQ** with one thread solves only 19 out of 80 instances, essentially limited to the smallest sizes. Using 48 threads consistently improves the performance of this method, increasing the number of solved instances to 58. Nevertheless, the method becomes progressively less efficient as N grows: for $N = 700$ only three instances are solved, and for $N = 800$ only one instance is solved, with average gaps of 23.51 and 34.33, respectively.

In contrast, **MDP** remains very robust across all uniform instances. The average gap is 1.20 with one thread and 1.16 with 48 threads, with running times of 48.20 and 55.78 seconds, respectively.

Finally, the running times reported for **MDP** with 48 threads may exceed the nominal 60-second time limit of the heuristic. This is due to the overhead required to launch the independent runs, synchronize the threads, collect the results, and select the best solution.

7 Conclusions and Future Work

In this paper, we addressed the problem of identifying a pool of near-optimal solutions for a given optimization problem while maximizing their pairwise diversity. We assume that the structural properties of the solutions are encoded by a set of core binary variables, and we measure diversity in terms of the Hamming distance between the corresponding binary vectors.

To tackle this problem, we proposed alternative mathematical formulations, including a model with an exponential number of variables, for which we developed a column-generation-based solution algorithm.

The proposed framework is particularly effective when the decision-maker aims to maximize the average diversity within the solution pool. In other application settings, however, it may be more appropriate to maximize the minimum distance between any pair of solutions, leading to a max-min optimization problem. The formulations and algorithmic framework presented in this paper do not extend directly to this setting, which therefore represents an interesting and challenging direction for future research.

Acknowledgments

This research was funded by the Air Force Office of Scientific Research under award number FA8655-25-1-7013.

References

[1] R. Aboudi and K. Jørnsten. Resource constrained assignment problems. *Discrete Applied Mathematics*, 26(2-3):175–191, 1990.

[2] T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4): 361–372, 2006.

[3] G. Adomavicius and Y. Kwon. Optimization-based approaches for maximizing aggregate recommendation diversity. *INFORMS Journal on Computing*, 26(2):351–369, 2014.

[4] V. Aggarwal. A lagrangean-relaxation method for the constrained assignment problem. *Computers & Operations Research*, 12(1):97–106, 1985.

[5] J. Baste, M. R. Fellows, L. Jaffke, T. Masařík, M. de Oliveira Oliveira, G. Philip, and F. A. Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303:103644, 2022.

[6] E. Danna and D. L. Woodruff. How to select a small set of diverse solutions to mixed integer programming problems. *Operations Research Letters*, 37(4):255–260, 2009.

[7] E. Danna, M. Fenelon, Z. Gu, and R. Wunderling. Generating multiple solutions for mixed integer programming problems. In *International conference on integer programming and combinatorial optimization*, pages 280–294. Springer, 2007.

[8] A. V. Eremeev, A. V. Kel'manov, M. Y. Kovalyov, and A. V. Pyatkin. Selecting a subset of diverse points based on the squared euclidean distance. *Ann. Math. Artif. Intell.*, 90(7-9): 965–977, 2022.

[9] T. Hanaka, Y. Kobayashi, K. Kurita, and Y. Otachi. Finding diverse trees, paths, and more. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3778–3786, 2021.

[10] T. Hanaka, Y. Kobayashi, K. Kurita, S. W. Lee, and Y. Otachi. Computing diverse shortest paths efficiently: A theoretical and experimental study. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3758–3766, 2022.

[11] T. Hanaka, M. Kiyomi, Y. Kobayashi, Y. Kobayashi, K. Kurita, and Y. Otachi. A framework to design approximation algorithms for finding diverse solutions in combinatorial problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 3968–3976, 2023.

[12] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

[13] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[14] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, et al. MIPLIB 2010: mixed integer programming library version 5. *Mathematical Programming Computation*, 3(2):103–163, 2011.

[15] C.-C. Kuo, F. Glover, and K. S. Dhir. Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, 24(6):1171–1185, 1993.

[16] M. Laguna and R. Marti. Scatter search. In *Metaheuristic procedures for training neutral networks*, pages 139–152. Springer, 2006.

[17] J. Lei, L. Liu, M. Xiao, and Y. Zhou. Fast exact algorithms for the maximum diversity problem. *European Journal of Operational Research*, 2025.

[18] E. Malaguti, M. Monaci, and J. Pruente. K-adaptability in stochastic optimization. *Mathematical Programming*, 196(1):567–595, 2022.

[19] R. Martí, A. Martínez-Gavara, S. Pérez-Peló, and J. Sánchez-Oro. A review on discrete diversity and dispersion maximization from an or perspective. *European Journal of Operational Research*, 299(3):795–813, 2022.

[20] J. B. Mazzola and A. W. Neebe. Resource-constrained assignment scheduling. *Operations Research*, 34(4):560–572, 1986.

[21] M. L. Miller, H. S. Stone, and I. J. Cox. Optimizing murty's ranked assignment method. *IEEE Transactions on Aerospace and Electronic Systems*, 33(3):851–862, 1997.

[22] K. G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations research*, 16(3):682–687, 1968.

[23] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, New York, 1994.

[24] T. Petit and A. C. Trapp. Enriching solutions to combinatorial problems via solution engineering. *INFORMS Journal on Computing*, 31(3):429–444, 2019.

[25] A. C. Trapp and R. A. Konrad. Finding diverse optima and near-optima to binary integer programs. *IIE Transactions*, 47(11):1300–1312, 2015.

[26] V. Xinying Chen and J. N. Hooker. A guide to formulating fairness in an optimization model. *Annals of Operations Research*, 326(1):581–619, 2023.

[27] Y. Zhou, J.-K. Hao, and B. Duval. Opposition-based memetic search for the maximum diversity problem. *IEEE Transactions on Evolutionary Computation*, 21(5):731–745, 2017.