

Solving the Coarseness Problem by ILP Using Column Generation

Allan Sapucaia(✉), Pedro J. de Rezende, and Cid C. de Souza

Institute of Computing, University of Campinas, Campinas, Brazil
allansapucaia@students.ic.unicamp.br, {rezende,cid}@ic.unicamp.br

Abstract. A core problem in machine learning is the classification of points in high dimensions. In an attempt to minimize the resources required and facilitate data visualization, a recent effort has been made to project the points non-linearly onto the plane and apply classification methods. The quality of any such projection can be measured by evaluating how the resulting points of different classes are set apart. In this paper, we study integer linear programming (ILP) models to determine the coarseness of sets of bicolored points in the plane, which measures how well-separated these two classes are. The complexity of computing the coarseness of a point set is unknown, but conjectured to be NP-hard. We present a base ILP model for the problem with an exponential number of variables and constraints, followed by a series of improvements in the quality of its relaxation and close with an efficient column generation implementation. These modifications allow us to solve instances with three times as many points as the base model, within the same time and memory limits. By combining of our preprocessing techniques and a heuristic from the literature, we are even able to solve to proved optimality many instances of our benchmark in polynomial time. A comprehensive experimental study is presented to evaluate the impact of each improvement.

Keywords: Computational geometry · Combinatorial optimization · Coarseness

1 Introduction

In this paper, we study integer linear programming (ILP) models to determine the coarseness of sets of bicolored points in the plane. Given a bipartite planar point set $P = R \cup B$, with $R \cap B = \emptyset$, we call R (red points) and B (blue points) *classes of points* and say that P is a *bicolored set*. Intuitively, the *coarseness* of P is a measure of how blended the two classes of points of P are.

This work was supported in part by grants from: *Brazilian National Council for Scientific and Technological Development* (CNPq) #313329/2020-6, #309627/2017-6, #304727/2014-8; *São Paulo Research Foundation* (FAPESP) #2020/09691-0, #2018/26434-0, #2018/14883-5, #2014/12236-1.

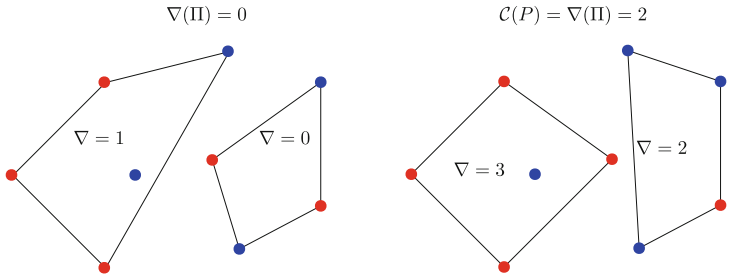
© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12953, pp. 30–45, 2021.
https://doi.org/10.1007/978-3-030-86976-2_3

A core problem in machine learning is the classification of points in high dimensions, where two of the main issues are the necessary resources, time or memory, and the difficulty in visually representing the data. Non-linear projection techniques, such as t-SNE [12], are capable of projecting points from a high dimension onto the plane while attempting to preserve neighborhood relationships. These projections allow for the development of algorithms that first map the points to the plane and then work on classifying and presenting them. Successful applications of this approach can be found in [3,13] and [4].

In this context, non-linear projections can be evaluated by how well separated the classes of points are in the plane after projection. This evaluation is particularly useful for parameter tuning for the projection algorithms. The most popular metric for evaluating projections for classification problems is Normalized Cuts [17], which is based on graph theory. A geometric-based alternative metric is given by the *coarseness* of a point set that is discussed below.

As before, let $P = R \cup B$, with $R \cap B = \emptyset$, be a bicolored planar point set. From here on, unless stated otherwise, we assume that the points in P are in general position, i.e., no three points in P are colinear.

A subset $Q \subseteq P$ is an *island* if there is a convex region C such that $Q = C \cap P$. The set of all islands of P is denoted by $\mathcal{I}(P)$. We assume that the sequence of vertices of the convex hull of island Q, denoted $\mathrm{CH}(Q)$, is always given in counter-clockwise (CCW) order with respect to (w.r.t.) a point in its interior. Also, $\mathrm{INT}(Q)$ denotes the subset of points of Q that do not lie on the boundary of $\mathrm{CH}(Q)$. As island Q can be uniquely characterized by its convex hull, we may sometimes refer to Q simply by the CCW sequence of vertices of $\mathrm{CH}(Q)$. Two islands are called *disjoint* if their convex hulls are disjoint.



(a) Suboptimal Partition (b) Optimal Partition

Fig. 1. Point set P and two partitions of its points into disjoint islands. (Color figure online)

We associate a *discrepancy* with each island $Q \in \mathcal{I}(P)$, given by $\nabla_Q = ||R \cap Q| - |B \cap Q||$. Moreover, the discrepancy of a partition $\Pi = \{Q_1, Q_2, \ldots Q_k\} \subseteq \mathcal{I}(P)$ of P into k pairwise disjoint islands, denoted by ∇_Π, is given by $\nabla_\Pi = \min_{Q \in \Pi} \nabla_Q$. The *coarseness* of P, denoted by $\mathcal{C}(P)$, is the maximum discrepancy among all possible partitions of P into pairwise disjoint islands. Figure 1 illustrates the concepts of discrepancy and coarseness.

Problem Statement. Given a bicolored planar point set, the *Coarseness Problem* consists in determining its coarseness.

This problem was first introduced in [5]. In that paper, the authors conjectured that the problem is NP-hard and studied theoretical bounds for the coarseness of point sets. They also proposed exact polynomial-time algorithms for two special cases: when the points are in convex position and when the partition is restricted to exactly two islands. The algorithm for the later case is reviewed in Sect. 4. Also, a work leading to polynomial algorithms for two other special cases is presented in [6], where the islands are restricted to lie within strips or within boxes. We are not aware of any studies in the literature that report experimental results regarding this problem.

Our Contributions. In this paper, we present an ILP model to compute the coarseness of bicolored planar point sets. We also investigate a number of improvements to the initial model, including a column generation approach, followed by a comprehensive experimental study that shows the impact of each refinement in terms of computational efficiency. The final model is able to solve instances with three times as many points when restricted to the same computational resources. Many of those instances are solved to proved optimality in polynomial time during preprocessing due to tight upper and lowerbound estimates. The benchmark used to assess the performance of the model is made publicly available to allow for future comparisons to our results.

This paper is organized as follows. In Sect. 2, we present a base ILP model for the Coarseness Problem. In Sect. 3, we improve the base model. Section 4 introduces a column generation algorithm. Lastly, Sect. 5 shows the empirical results comparing the improvements previously introduced.

2 ILP Model for the Coarseness Problem

In this section, we present the base ILP model for the Coarseness Problem.

Given a set P of n points, let $L(P)$ denote the set of $\Theta(n^2)$ line segments $\overline{jk}$ where j and k are distinct points in P. The *complete (geometric) graph* induced by P is $G(P) = (P, E(P))$, where $E(P) = \{\{j,k\} : \overline{jk} \in L(P)\}$. In this text, we refer to a segment $\overline{jk} \in L(P)$ and the corresponding edge in $\{j,k\} \in E(P)$ interchangeably. We denote by $S^C(P)$ the set of edge pairs whose corresponding line segments share an interior point, i.e., they cross each other.

Let LB and UB represent lower and upper bounds, respectively, for the coarseness of a given bicolored point set P. Associating a positive variable u_Q to each island $Q \in \mathcal{I}(P)$, a binary variable x_e for edge edge $e \in E(P)$ and a binary variable z_d for each possible discrepancy value d, we can express the Coarseness Problem as Model (1).

Constraint (1b) ensures that only one discrepancy value is selected, while Constraints (1c) ascertain that if an island is selected, its discrepancy is an upper bound on the discrepancy of the solution. Constraints (1d) guarantee that the points in P are partitioned into islands. Constraints (1e) enforce that if an island

is being used, its edges are part of the solution. Constraints (1f), together with Constraints (1d), require the islands to be disjoint.

Notice that there is potentially an exponential number of islands.

From Constraints (1d) and (1e), it follows that having integer values for edge variables, (1h), implies that the island variables necessarily assume integer values. If edge variables are integral, their corresponding line segments define a planar subdivision. Now, assume that Q_1 and Q_2 are two different islands that share an edge and are fractional, i.e., $0 < u_{Q_1} \leq 1$ and $0 < u_{Q_2} < 1$, and $u_{Q_1} + u_{Q_1} < 1$. Since one of those islands must have and edge that is not shared with the other, at least one of the edge variables must assume a fractional value, which is a contradiction.

$$\max \sum_{LB \leq d \leq UB} dz_d \tag{1a}$$

$$\text{s.t.} \sum_{LB \leq d \leq UB} z_d = 1 \tag{1b}$$

$$u_Q \leq \sum_{LB \leq d \leq \nabla_Q} z_d \qquad \forall Q \in \mathcal{I}(P) \tag{1c}$$

$$\sum_{Q \in \mathcal{I}: p \in Q} u_Q = 1 \qquad \forall p \in P \tag{1d}$$

$$\sum_{Q \in \mathcal{I}: e \in \mathrm{CH}(Q)} u_Q = x_e \qquad \forall e \in E(P) \tag{1e}$$

$$x_e + x_f \leq 1 \qquad \forall \{e, f\} \in S^C(P) \tag{1f}$$

$$u_Q \in \mathbb{R}^+ \qquad \forall Q \in \mathcal{I}(P) \tag{1g}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E(P) \tag{1h}$$

$$z_d \in \{0, 1\} \qquad \forall d \in \{LB, LB + 1, \ldots, UB\} \tag{1i}$$

Prior to investigating the geometrical structure of P, we can set $LB = 1$ and $UB = \max_{Q \in \mathcal{I}(P)} \nabla_Q$ and solve the model by standard branch-and-bound (BnB) techniques, which rely on solving its linear relaxation obtained by dropping the integrality constraints on the variables. However, the relaxation is very weak, leading to a dual bound very close to UB. Moreover, another feature of this model that makes it hard to solve in practice is the exponential number of variables and constraints.

To overcome some of the deficiencies noted above, in the next section, we improve the quality of the relaxation by finding, in polynomial time, tighter values for LB and UB, and by adding a new class of constraints. Besides, in a subsequent section, a column generation approach to address the exponential size of the model is presented.

3 Improving the Base Model

In this section, we propose methods to find better values for the bounds, LB and UB, for the coarseness, as well as new constraints to strengthen Model (1).

As discussed in the previous section, the relaxation of our Model leads to a very weak dual bound. For many instances, this poor bound is attained by assigning to z_{UB} a value close to 1 and setting $z_{LB} + z_{UB} = 1$. Notice that the value of z_{LB} is an upper bound on all island variables due to Constraints (1c).

To illustrate this point, let us make reference to a benchmark of 30 randomly generated planar bicolored point sets of sizes $\{10, 15, \ldots, 60\}$ that we built for testing purposes. For each point, both x and y coordinates are randomly chosen in the interval $[0, 1]$ using uniform distribution and the colors are also assigned at random, with equal probability. This benchmark is available at [15]. Firstly, we observed that instances of size 20 are the largest that could be solved within one hour on a fairly standard computer (see Sect. 5) using the base model. Figure 2 shows the optimal value, the relaxation dual bound and the upper bound $UB = \max_{Q \in \mathcal{I}(P)} \nabla_Q$ for all thirty instances of size 20. For those instances, the difference between UB and the relaxation value, given as avg±stddev, is 0.017 ± 0.004.

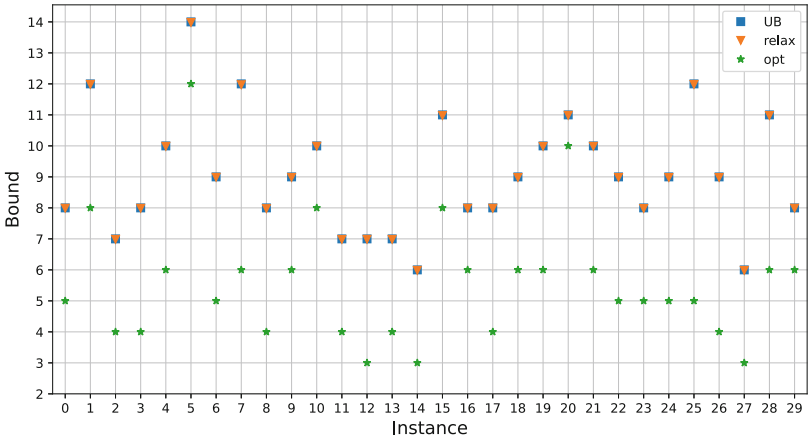


Fig. 2. Coarseness, UB and value of the relaxation given by the base model for instances with 20 points.

Based on these observations, we can infer that the quality of the relaxation could be improved by tightening the LB and UB bounds.

3.1 Lower Bound (LB)

By finding a tighter value for LB, we not only get a stronger bound, but can also eliminate from the model any island whose discrepancy is smaller than LB, effectively increasing solving speed and reducing memory requirements.

Clearly, the discrepancy of any partition of P into islands, i.e., a primal solution for Model (1), can be used to provide a lower bound. For example, a trivial solution would simply be to regard P as comprised of a single island.

Now, when the two classes of points that comprise P are linearly separable, there exists a partition that splits the points into two monochromatic islands, which obviously is an optimal solution to the coarseness problem for P.

As an educated guess, corroborated by the aforementioned extreme case, we suspect that partitions into at most two islands might lead to good quality solutions.

As discussed in the literature review (see the latter part of Sect. 1), there are results regarding bounds for the coarseness of a point set that may serve our purpose. In particular, we decided to use a polynomial time algorithm for the special case where the partitions are restricted to two islands, as it is efficient and straightforward to be implemented. That algorithm consists in finding a line that divides the plane into two islands with the largest minimum discrepancy. In spite of the $O(n^2)$-time algorithm from [5], for solving this special case, we opted for a trivial $O(n^3)$-time procedure since, for the instances at hand, $n = |P|$ is quite small. This algorithm inspects the partitions obtained by using the lines supported by the $\Theta(n^2)$ segments in $L(P)$, corresponding to edges in $E(P)$ and returns the best value found.

3.2 Upper Bound (UB)

Denote by z^*_{UB} the optimal solution to the relaxation of Model (1) for an instance P using LB as described previously and an upper bound UB. Note that since z^*_{UB} is an upper bound on the coarseness of P, we can take $UB = \lfloor z^*_{UB} \rfloor$ as a new upper bound, since coarseness is an integral number. This leads to an iterative procedure that successively finds decreasing upper bounds UB by solving the linear relaxation until this process converges, i.e., $UB = z^*_{UB}$. As the coarseness of P is bounded by $|P| = n$, there are a linear number of iterations which consist on solving the relaxation of Model (1).

In the next section, we show how to modify Model (1) so its relaxation can be solved in polynomial time through column generation. This will allow the iterative procedure we just discussed to be performed in polynomial time as well.

3.3 Least Discrepancy Constraints

We now improve Model (1) by observing that any optimal partition of discrepancy d must include at least one island of discrepancy d. This requirement can be expressed by the following linear constraints

$$\sum_{Q:\nabla_Q = d} u_Q \geq z_d \qquad \forall d \in \{LB, LB+1, \ldots, UB\}, \tag{2}$$

which complements Constraints (1c).

Adding Constraints (2) prevents the solver from setting the value of z_{UB} close to 1 without actually including an island of discrepancy UB. Indeed, we observed that Constraints (2) improve the quality of the relaxation, as discussed in Sect. 5.

4 A Column Generation Algorithm

The number of variables and constraints in Model (1) makes it hard to solve instances in practice due not only to the high memory requirements, but also to the time spent enumerating them and building the model. In this section, we show how to use column generation to solve the linear relaxation of this model in polynomial time. For this, we need to improve a transient weakened version of the original model.

To create the set of all islands, we use an iterative procedure that generates islands by the size of their convex hull. Initially, we generate all single points, segments and islands whose convex hull is a triangle. Then, new islands are generated by adding a new point to the islands of the preceding convex hull size in such a way that the resulting island has all vertices from the previous island plus the new point as vertices. The algorithm stops when no new island can be expanded in this fashion.

Instead of enumerating all variables, we can solve the relaxation of Model (1) by starting with a small subset of variables and generating new ones when necessary. This approach, known as *Column Generation* [2], consists of iterating between two problems: the Restricted Master Problem (RMP) and the Pricing problem. The RMP solves the original model restricted to a subset of variables. On the other hand, given an optimal dual solution to the RMP, the Pricing problem seeks variables with positive (in the case of maximization problems) reduced cost that, when added to the RMP, will potentially increase the value of the objective function or, alternatively, certifies that no such variables exist, in which case the procedure stops. The solution of the last RMP is optimal for the relaxation of the complete model and, if the pricing algorithm runs in polynomial time, then the relaxation also is solved in polynomial time [10].

By associating dual variables $\alpha_Q \geq 0$ to Constraints (1c), β_p to Constraints (1d), γ_e to Constraints (1e) and $\delta_d \geq 0$ to Constraints (2), given a dual solution $(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\delta})$ we can write the pricing problem, obtained from the dual problem, as:

$$\overline{c} = \max_{Q \in \mathcal{I}} \left\{ -\overline{\alpha}_Q - \sum_{p \in Q} \overline{\beta}_p - \sum_{e \in \mathrm{CH}(Q)} \overline{\gamma}_e + \overline{\delta}_{\nabla_Q} \right\}. \tag{3}$$

If $\overline{c} > 0$, then Q has a positive reduced cost and must be added to the RMP.

Notice that before island $Q \in \mathcal{I}$ is included in the RMP, $\alpha_Q = 0$, and after it is included, its reduced cost is non-positive in any optimal solution of the RMP.

This being the case, to solve the pricing problem in polynomial time, we can use dynamic programming. A similar approach proved successful for solving the pricing problem in [14].

To express the pricing problem as a recurrence suitable for an efficient dynamic programming implementation, it suffices to weaken the model by combining discrepancy constraints (1c) in such a way that each dual variable is associated with a discrepancy value, similar to the dual variable associated with Constraints (2), instead of to individual islands. With this change, we can solve

a recurrence that finds an island with the maximum reduced cost that has k as its leftmost and first point, ℓ and m as the second-to-last and last points and discrepancy d. Notice that, as is, we lack a necessary property to devise divide-and-conquer algorithms for the pricing problem (3), since knowing whether we have $\alpha_Q > 0$ is only possible after island Q is constructed by combining solutions to sub-problems, changing the reduced cost of Q.

This is done by adding together all Constraints (1c) with the same discrepancy d and noticing that, since we are partitioning n points of P into islands, we can have at most $\frac{n}{d}$ islands of discrepancy d in a solution. By doing this, we obtain the following constraints:

$$\sum_{Q:\nabla_Q = d} u_Q \leq \frac{n}{d} \sum_{LB \leq s \leq d} z_s \qquad \forall d \in \{LB, LB+1, \ldots, UB\}. \tag{4}$$

We can lift the new constraints by including all islands with discrepancy lower than d, leading to these constraints:

$$\sum_{Q:\nabla_Q \leq d} \nabla_Q u_Q \leq n \sum_{LB \leq s \leq d} z_s \qquad \forall d \in \{LB, LB+1, \ldots, UB\}. \tag{5}$$

By replacing Constraints (1c) with Constraints (5) and associating to them a dual variable η, we obtain the following final pricing problem:

$$\overline{c} = \max_{Q \in \mathcal{I}} \left\{ -\sum_{p \in Q} \overline{\beta}_p - \sum_{e \in \mathrm{CH}(Q)} \overline{\gamma}_e + \nabla_Q \sum_{s \geq \nabla_Q} \overline{\eta}_s - \overline{\delta}_{\nabla_Q} \right\}. \tag{6}$$

We now solve this pricing problem using dynamic programming. We first provide an $O(n^5)$ algorithm and show how to improve the time complexity to $O(n^4)$.

To this end, we extend the concept of discrepancy so that we know the exact value of the resulting discrepancy when two islands are combined. For that, we introduce the notation $\nabla^{\pm}$ as the signed discrepancy. By assigning $\nabla^{\pm}(r) = +1$ for $r \in R$ and $\nabla^{\pm}(b) = -1$ for $b \in B$, the *signed discrepancy* of island $Q \in \mathcal{I}(P)$ is given by $\nabla^{\pm}(Q) = |R \cap Q| - |B \cap Q| = \sum_{p \in Q} \nabla^{\pm}(p)$.

Let $B(k, \ell, m, d)$ denote the maximum reduced cost from among all the islands with k as the leftmost point (and first in CCW order), ℓ as the second-to-last, m as the last point, and d as the signed discrepancy (w.r.t. the dual solution $(\overline{\beta}, \overline{\gamma}, \overline{\delta}, \overline{\eta})$). We say that (k, ℓ, m) is the *last triangle* of such islands. Figure 3 illustrates these concepts.

To simplify notation, let $D(d) = |d|(\sum_{|d| \leq s \leq UB} \overline{\eta}_s) - \overline{\delta}_{|d|}$ denote the sum of the dual variables associated with (signed) discrepancy d in (6) and $\Delta(k, \ell, m) = -\overline{\gamma}_{k,\ell} - \overline{\gamma}_{\ell,m} - \overline{\gamma}_{m,k} - \sum_{p \in (k,\ell,m)} \overline{\beta}_p + D(\nabla(k, \ell, m))$ denote the reduced cost of triangle (k, ℓ, m).

Given a triangle (k, ℓ, m) and a discrepancy d, we wish to determine whether there exists an island with maximum reduced cost that, once merged with triangle (k, ℓ, m), leads to discrepancy d. We can compute the discrepancy necessary for such an island to be mergeable with triangle (k, ℓ, m), and denote it by

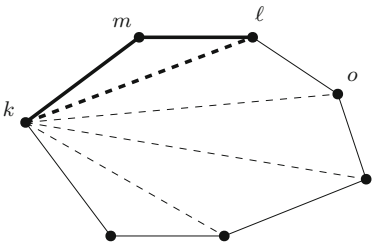


Fig. 3. An island whose last triangle is (k, ℓ, m). Dashed lines show how it is decomposed into triangles with k as the leftmost vertex. The edges of (k, ℓ, m) are emphasized.

$d'(k, \ell, m, d) = d - \nabla^{\pm}((k, \ell, m)) + \nabla^{\pm}(k) + \nabla^{\pm}(\ell)$. Note that since the points are in general position and k and ℓ are the only points shared by both the island and the triangle, the last two additive terms are necessary. To compute the resulting reduced cost of combining the triangle and an island, we need to consider that both share edge $\{k, \ell\}$, and also account for the reduced cost associated to their own discrepancy. Lastly, since a triangle can be regarded as an island, when its discrepancy is equal to the target discrepancy, island merging becomes optional.

Let P_k denote the list of points to the right of k, in increasing order of abscissae. Given a sequence of three points (k, ℓ, m) in the plane, we say that they are *colinear*, *positively oriented*, or *negatively oriented* depending on the value of the cross product $\overrightarrow{k\ell} \times \overrightarrow{\ell m}$ being 0, positive or negative. For simplicity, we say that $\mathrm{CVX}(k, \ell, m) = \text{true}$ (or simply $\mathrm{CVX}(k, \ell, m)$) if (k, ℓ, m) are positively oriented. We can combine an island whose last triangle is (k, o, ℓ) and triangle (k, ℓ, m) to form an island, as long as $\mathrm{CVX}(k, o, \ell)$, so that (k, o, ℓ) is a triangle given in CCW order, and $\mathrm{CVX}(o, \ell, m)$, to ensure convexity. Let $O_{k\ell m}$ denote the set $\{o \in P_k : \mathrm{CVX}(k, o, \ell) \wedge \mathrm{CVX}(o, \ell, m)\}$.

To simplify notation, we introduce an auxiliary recurrence $B_u(k, \ell, m)$, which computes a maximum reduced cost island that has (k, l, m) as its last triangle and discrepancy d by merging this triangle with another island. Using the aforementioned notation, these observations lead the following recurrence:

$$B_u(k, \ell, m, d) = \left\{ \begin{array}{ll} -\infty, & \text{if } O_{k\ell m} = \emptyset \\ \left. \begin{array}{l} \max\limits_{o \in O_{k\ell m}} \{B(k, o, \ell)\} + \Delta(k, \ell, m) + D(d) \\ \qquad + 2\overline{\gamma}_{k,\ell} + \overline{\beta}(k) + \overline{\beta}(\ell) \\ - D(\nabla^{\pm}(k, \ell, m)) - D(\nabla^{\pm}(d'(k, \ell, m, d) \end{array} \right] & \text{otherwise} \end{array} \right\} \quad (7)$$

$$B(k, \ell, m, d) = \left\{ \begin{array}{ll} \max\{B_u(k, \ell, m, d), \Delta(k, \ell, m)\} & \text{if } d = \nabla^{\pm}((k, \ell, m) \quad (8a) \\ B_u(k, \ell, m, d) & \text{otherwise} \quad (8b) \end{array} \right.$$

A naive implementation of this algorithm would have a time complexity of $O(n^5)$. This is the result of having $O(n^4)$ states that require finding a maximum

over $O(n)$ candidates plus the sum of costs associated with a triangle for each of them.

To calculate the costs associated with a triangle in constant time, we need to calculate $\sum_{p\in(k,\ell,m)} \overline{\beta}_p$ and $\nabla^{\pm}(k,\ell,m) = \sum_{p\in(k,\ell,m)} \nabla^{\pm}(p)$ in $O(1)$. Notice that the triangles are not necessarily empty. This can be realized using a technique presented in [8] for: given a set P of n weighted points, after an $O(n^2)$-time preprocessing, compute, in $O(1)$ time, the sum of the weights of all points within a query triangle with vertices in P. The basic idea is to precompute the sum above each segment in $L(P)$ and use an inclusion-exclusion principle to calculate the sum inside any triangle, based on its edges. To determine the signed discrepancy of a triangle, we assign weights $+1$ and -1 to points in R and B, respectively, and to compute the sum of the β variables of points within a triangle we associate to each point its β value.

To reach our proposed time complexity target, we adapt an angular sweeping technique first introduced to find the Maximum Cardinality Empty Convex Polygon [1], and recently employed in [14] to solve the pricing problem of an ILP model for the Minimum Convex Partition Problem. This technique allows us to compute the maximum within Eq. (7), which corresponds to finding a third-to-last point for the island being constructed, in amortized constant time.

While referring to Fig. 4, notice that, for fixed k and ℓ, if we visit the possible last points m in counter-clockwise order around ℓ, the list of candidates third-to-last points o can also be visited in counter-clockwise order in such a way that each candidate is only considered once. For further details, we refer the reader to [1] and [14].

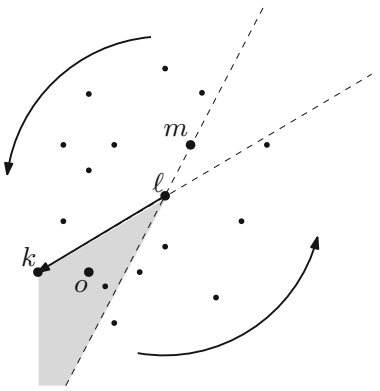


Fig. 4. Illustration of the angular sweep technique.

Finally, we use a technique of early stop to accelerate the convergence of the RMP. Instead of halting the column generation algorithm when the maximum reduced cost $\overline{c}$ is zero, we stop when it is small enough so that the floor of the current optimal of the RMP, z^*_{RMP}, cannot change. We can calculate a

dual bound for the complete model based on the maximum reduced cost as $z^* \geq z^*_{\text{RMP}} + \overline{c}\kappa$, where κ is an upper bound on the number of islands in an optimal solution. Since we know that $z^* \geq LB$, we must have $\kappa \leq \lfloor \frac{n}{LB} \rfloor$. Finding a better LB reduces significantly the tailing-off effect, which is a frequent issue for column generation, when the value of z^*_{RMP} is very close to z^* and yet many iterations are needed to achieve a reduced cost of zero [11].

Now, we can solve to integrality the model that results from replacing Constraints (4) with (5) and adding Constraints (2), by using Branch-and-Price (BnP) [2]. We start the model with the islands corresponding to each individual point and use the pricing as described. When solving multiple relaxation, as in the case of the iterative procedure to find UB, we start the new relaxation with all islands generated to solve the previous one. Branching is done on edge and discrepancy variables.

5 Experimental Results

In this section, we show the positive impact of the modifications to the basic model described in the previous sections. All experiments were run on an Intel Xeon E5-2420 at 1.9 GHz and 32 GB of RAM, running Ubuntu 18.04. Models and algorithms were implemented in C++ v.11 and compiled with `gcc` 7.5. Geometric algorithms and data structures were implemented using CGAL 5.2 [18], using Gmpq for exact number representation. ILP models were implemented using SCIP 7.0.2 [9] with CPLEX 12.10 as the LP solver. In each experiment, we use the instances from the benchmark described in Sect. 3. A time limit of one hour was set for the ILP solver for each instance.

When running times are presented, we consider both the time to generate and to solve the model. Most of the data are presented in a standard boxplot, grouped by size. All data used to generate the figures is available at [15].

We begin by showing the impact of tightening the LB and UB bounds in the relaxation. In Figs. 5 and 6, the modifier +heuLB indicates that LB was found by a heuristic solution, +itUB means that the iterative procedure was used to find UB and the addition of the least discrepancy constraints is denoted by the +cut. The calculation LB and UB are considered the *preprocessing phase*.

Figure 5 shows the relative gap, computed as $\frac{\text{RELAX}-\text{OPT}}{\text{RELAX}}$, where OPT and RELAX are the values of an optimal integer solution and optimal relaxation solution, respectively, for instances of different sizes. The Base model could not solve instances with 25 or more points within one hour, while instances with at least 35 points did not fit in 32 GB of memory without column generation.

As expected, improving LB by finding the best partition into 1 or 2 islands has a positive impact in reducing the relative gap. The benefit of the least discrepancy constraints can also be noted. The iterative algorithm to find a better UB yields a tight bound for most instances.

Similarly, the total solving time for the same configurations is shown in Fig. 6. In order not to distort the scale and hinder details in the graph, two outlier instances with 30 points are not shown for the Base +heuLB +itUB +cut graph

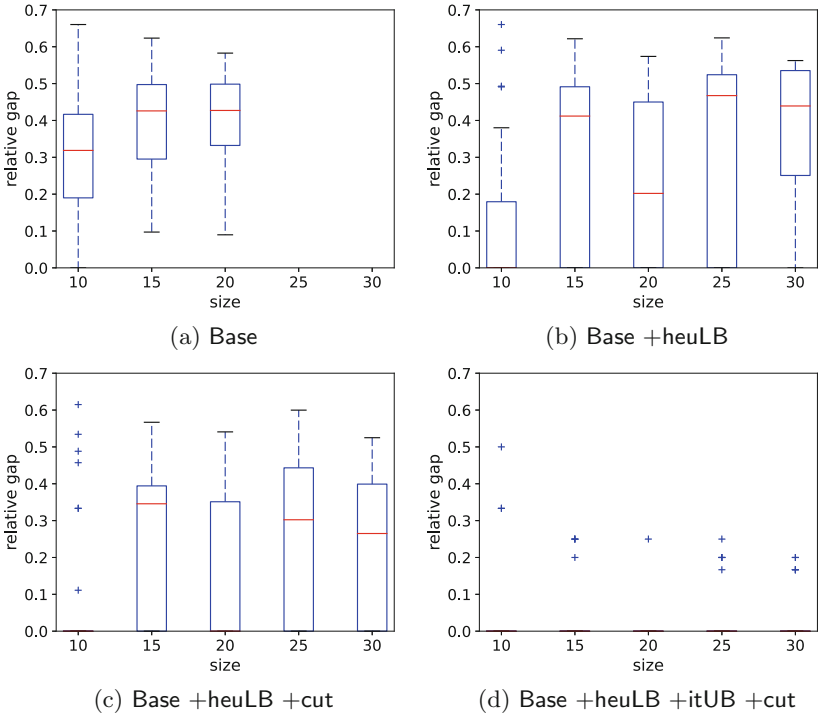


Fig. 5. Relative gap for different configurations of the Base model.

as they took 2359 and 3980 s, respectively, to be solved. This experiment shows the impact in performance of the improved LB values, as that allows for the removal of a large number of variables, drastically reducing both solving time and memory consumption, which, in turn, raises the size of instances being solved from 20 to 30. Notably, the high quality bounds obtained by using the iterative algorithms to find UB come with a cost in overall time. This extra time is greatly reduced, along with the expected memory requirement, when solving the relaxations using column generation, as we will see.

Next, we show, in Fig. 7, the solving time to find exact solutions with and without the tighter UB, while still using the least discrepancy constraints and the LB found by the heuristic. As before, so as not to distort the scale, two outlier instances are omitted in Fig. 7b as they took 2594 and 4369 s, respectively, to be solved. Instances of more than 30 points could not fit in 32 GB of memory. Here, for most instances, we observe the advantage of using the iterative algorithm to find UB when it comes to solving the ILP. In such cases, the stronger bound compensates the extra time during preprocessing leading to an overall performance gain.

Lastly, as our paramount result, we focus now on the solving times using the same bounds as before, but employing column generation for both the iterative

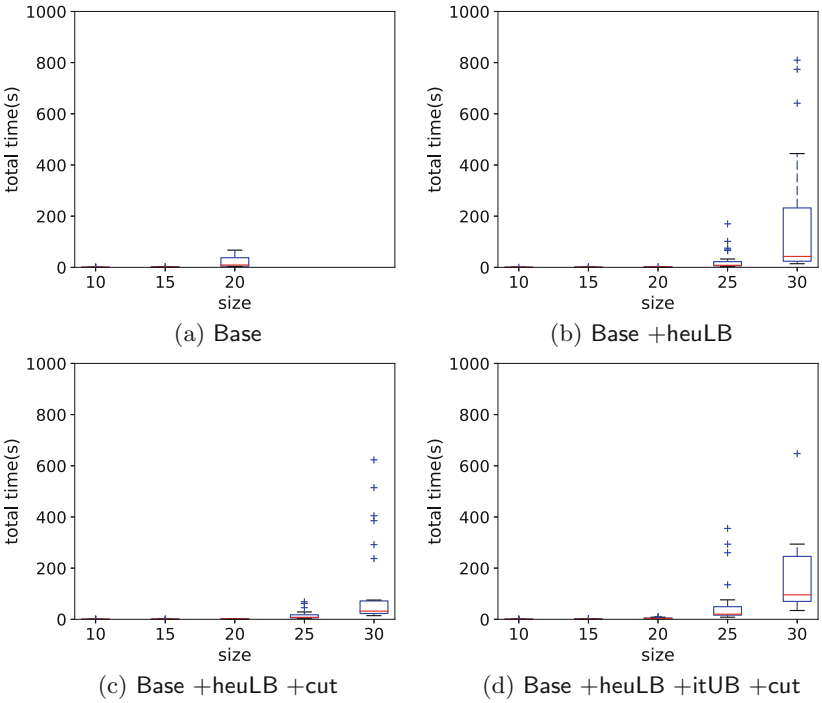


Fig. 6. Total solving time for the relaxation of different configurations of the model.

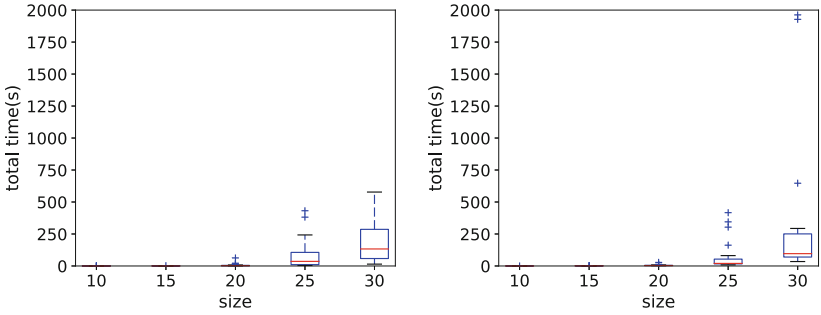
procedure to find *UB* and for the actual ILP, as depicted in Fig. 8a. Having thus lessened the memory limitation, larger instances could now be solved and we set aside a time limit of one extra hour just for the iterative procedure to find a tight *UB* value.

For the actual memory consumption, we refer the reader to Fig. 8b. Notice that the most remarkable development is the great reduction in memory usage, which allowed us to solve instances with three times the number of points as before, using one tenth of the total memory limit, on average. Moreover, the use of column generation sped up the procedure to find *UB*. For more details, see Table 1.

Notably, the combination of itUB and heuLB was able to solve to proved optimality a high number of randomly generated instances in polynomial time. This table also shows that even when limited to a total of one hour for preprocessing and solving, the usage of itUB increased the number of instances solved.

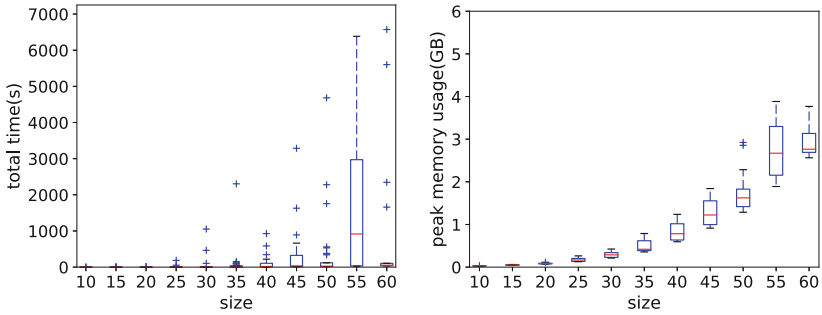
Table 1. Number of instances solved by different methods using column generation for different instance sizes. Solved in preprocessing indicates instances where $UB = LB$.

Method ↓ | **Size** →	10	15	20	25	30	35	40	45	50	55	60
# solved in preprocessing	27	25	28	26	26	25	23	21	27	19	18
# solved with itUB	30	30	30	30	30	29	28	23	28	21	20
# solved in 1h with itUB	30	30	30	30	30	29	28	23	27	19	18
# solved without itUB	30	30	30	30	30	28	27	20	26	13	17



(a) Without itUB (b) With itUB

Fig. 7. Total time for Base +heuLB +cut with and without itUB.



(a) Time (s) (b) Peak Memory Usage (GB)

Fig. 8. Total solving time and peak memory usage for Base +heuLB +itUB +cut using Column Generation. Only instances that were solved to optimality are included.eps

Final Remarks

In conclusion, we showed that by using column generation, in conjunction with the improvements to the bounds on coarseness, we were able to solve to optimality instances of triple the size (up to 60 points) from our benchmark that the initial model could solve, with remarkably small memory requirement.

As to future work, we intend to investigate how to speed up the column generation procedure since its convergence is central for solving large instances quickly. Improving the heuristic is another venue of research, which might not only reduce the solving time for the relaxation but also lead to a larger number of instances that can be solved during preprocessing due to establishing a tighter lower bound.

Despite the fact that randomly generated point sets have a high probability of being in general position, it is important to consider how our model might be adapted for instances where this property may not be guaranteed. Besides the obviously intricate approach of considering all different cases of collinearity that may appear during island construction, it would be worth considering the alternative of applying symbolic perturbation schemes such as [7] or [16], notwithstanding their well known impact in time complexity, in practice.

References

1. Avis, D., Rappaport, D.: Computing the largest empty convex subset of a set of points. In: Proceedings of the First Annual Symposium on Computational Geometry, SCG 1985, pp. 161–167. ACM, New York (1985). https://doi.org/10.1145/323233.323255
2. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H.: Branch-and-Price: column generation for solving huge integer programs. Oper. Res. **46**(3), 316–329 (1998). https://doi.org/10.1287/opre.46.3.316
3. Benato, B.C., Telea, A.C., Falcão, A.X.: Semi-supervised learning with interactive label propagation guided by feature space projections. In: Proceedings of the 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 392–399, October 2018. https://doi.org/10.1109/SIBGRAPI.2018.00057
4. Benato, B.C., Gomes, J.F., Telea, A.C., Falcão, A.X.: Semi-supervised deep learning based on label propagation in a 2D embedded space. CoRR abs/2008.00558 (2020). https://arxiv.org/abs/2008.00558
5. Bereg, S., Díaz-Báñez, J.M., Lara, D., Pérez-Lantero, P., Seara, C., Urrutia, J.: On the coarseness of bicolored point sets. Comput. Geom. **46**(1), 65–77 (2013). https://doi.org/10.1016/j.comgeo.2012.04.003
6. Díaz-Báñez, J.M., López, M.A., Ochoa, C., Pérez-Lantero, P.: Computing the coarseness with strips or boxes. Discret. Appl. Math. **224**, 80–90 (2017). https://doi.org/10.1016/j.dam.2017.02.022
7. Edelsbrunner, H., Mücke, E.P.: Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. ACM Trans. Graph. **9**(1), 66–104 (1990). https://doi.org/10.1145/77635.77639
8. Eppstein, D., Overmars, M.H., Rote, G., Woeginger, G.J.: Finding minimum area k-gons. Discret. Comput. Geom. **7**, 45–58 (1992). https://doi.org/10.1007/BF02187823
9. Gamrath, G., et al.: The SCIP Optimization Suite 7.0. Technical report, Optimization Online, March 2020. http://www.optimization-online.org/DB_HTML/2020/03/7705.html
10. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1**(2), 169–197 (1981). https://doi.org/10.1007/BF02579273

11. Lübbecke, M.E., Desrosiers, J.: Selected topics in column generation. Oper. Res. **53**(6), 1007–1023 (2005). https://doi.org/10.1287/opre.1050.0234
12. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(86), 2579–2605 (2008). http://jmlr.org/papers/v9/vandermaaten08a.html
13. Peixinho, A.Z., Benato, B.C., Nonato, L.G., Falcão, A.X.: Delaunay triangulation data augmentation guided by visual analytics for deep learning. In: Proceedings of the 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 384–391, October 2018. https://doi.org/10.1109/SIBGRAPI.2018.00056
14. Sapucaia, A., de Rezende, P.J., de Souza, C.C.: Solving the minimum convex partition of point sets with integer programming. Comput. Geom. **99**, 101794 (2021). https://doi.org/10.1016/j.comgeo.2021.101794
15. Sapucaia, A., de Souza, C.C., de Rezende, P.J.: Coarseness of point sets - Benchmark instances and solutions (2021). http://www.ic.unicamp.br/~cid/Problem-instances/Coarseness
16. Seidel, R.: The nature and meaning of perturbations in geometric computing. Discret. Comput. Geom. **19**(1), 1–17 (1998). https://doi.org/10.1007/PL00009330
17. Shi, J., Malik, J.: Normalized cuts and image segmentation. In: 1997 Conference on Computer Vision and Pattern Recognition (CVPR 1997), Puerto Rico, 17–19 June 1997, pp. 731–737. IEEE Computer Society (1997). https://doi.org/10.1109/CVPR.1997.609407
18. The CGAL Project: CGAL User and Reference Manual. CGAL Editorial Board, 5.2 edn. (2020). https://doc.cgal.org/5.2/Manual/packages.html