European Journal of Operational Research 311 (2023) 444–454

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

An exact cutting plane method for the Euclidean max-sum diversity problem

Sandy Spiers[a,b,*], Hoa T. Bui[a,b], Ryan Loxton[a,b]

[a] ARC Centre for Transforming Maintenance through Data Science, Curtin University, Perth, Australia
[b] Curtin Centre for Optimisation and Decision Science, Curtin University, Perth, Australia

ARTICLE INFO

Article history:
Received 21 July 2022
Accepted 8 May 2023
Available online 12 May 2023

Keywords:
Combinatorial optimization
Maximum diversity
Cutting planes
Euclidean distance
Branch and cut

ABSTRACT

This paper aims to answer an open question recently posed in the literature, that is to find a fast exact method for solving the max-sum diversity problem, a nonconcave quadratic binary maximization problem. We show that, for Euclidean max-sum diversity problems (EMSDP), the distance matrix defining the quadratic term is always conditionally negative definite. This interesting property ensures that the cutting plane method is exact for (EMSDP), even in the absence of concavity. As such, the cutting plane method, which is primarily designed for concave maximisation problems, converges to the optimal solution of (EMDSP). The method was evaluated on several standard benchmark test sets, where it was shown to outperform other exact solution methods for (EMSDP), and is capable of solving two-coordinate problems of up to eighty-five thousand variables.

1. Introduction

The problem of maximizing diversity and dispersion arises in many practical settings. It involves selecting a subset of elements from a larger set to maximize some distance metric. Since the conception of the maximum diversity problem by Kuby (1987) (sometimes referred to as the maximum dispersion problem), the interpretation of diversity has taken many practical and theoretical forms. The topic has now reached a level of maturity where a multitude of problem variations, solution algorithms, and practical applications exist. Over the last thirty years, a significant quantity of research has focused on the max-sum diversity problem (Kuby, 1987), which is to maximize the sum of distances between selected elements, and the max-min diversity problem (Erkut, 1990), which is to maximize the minimum distance among selected points. For this paper, we focus our attention on the Euclidean max-sum diversity problem (EMSDP).

Given a set of n predefined locations $v_1, \dots, v_n$ in a vector space $\mathbb{R}^s$ ($s \geq 1$), the (EMSDP) aims to find a subset of p locations such that the sum of the distances between the p points is maximized. Here, we consider q_{ij} to be the distance between locations i and j defined by $q_{ij} = \|v_i - v_j\|$ where $\|\cdot\|$ is the standard Euclidean distance in $\mathbb{R}^s$. Let $Q = [q_{ij}]$ denote the full distance matrix where $i = 1, \dots, n$ and $j = 1, \dots, n$. The (EMSDP) is then given as

$$\max \quad f(x) = \tfrac{1}{2}\langle Qx, x\rangle, \tag{EMSDP}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_i = p, \tag{1}$$

$$x_i \in \{0, 1\}, \qquad 1 \leq i \leq n.$$

The (EMSDP) is known to be strongly NP-hard (Eremeev et al., 2019; Kuo et al., 1993; Ravi et al., 1994).

The practical applications of the maximum diversity problem are vast. One of the first examples presented in the literature is locating unwanted facilities on a network (Church & Garfinkel, 1978). Since then, many other researchers have relaxed the notion of distance to more general settings. One example is in genetics, where breeders attempt to maximize the diversity of traits among a breeding stock (Porter et al., 1975). Furthermore, social diversity such as gender, cultural and ethnic diversity has become highly desirable in many communities, especially in a workplace setting (Roberge & van Dick, 2010). More recently, maximum diversity has been used to find the optimal locations of chairs for COVID-19 social distancing (Ferrero-Guillén et al., 2022).

While research into heuristic and meta-heuristic approaches to the max-sum diversity problem has gathered significant interest (see Martí et al. (2022) for a recent review), the development

* Corresponding author at: Curtin Centre for Optimisation and Decision Science, Curtin University, Perth, Australia.
E-mail addresses: sandy.spiers@postgrad.curtin.edu.au (S. Spiers), hoa.bui@curtin.edu.au (H.T. Bui), r.loxton@curtin.edu.au (R. Loxton).

https://doi.org/10.1016/j.ejor.2023.05.014

of exact algorithms has fallen behind. One of the first exact approaches was presented in Kuo et al. (1993) and used linear reformulation techniques to transform the problem into an integer linear form. This was done in two ways. The first used a linearization technique presented in Glover & Woolsey (1974), whereby the quadratic $x_i x_j$ terms are replaced by a new auxiliary variable y_{ij}. The linear formulation of (EMSDP) is then given as

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} q_{ij} y_{ij}, && (2)\\
\text{s.t.} \quad & \sum_{i=1}^{n} x_i = p,\\
& y_{ij} \geq x_i + x_j - 1, \quad 1 \leq i < j \leq n, && (3)\\
& y_{ij} \leq x_i, \quad 1 \leq i < j \leq n, && (4)\\
& y_{ij} \leq x_j, \quad 1 \leq i < j \leq n, && (5)\\
& y_{ij} \geq 0, \quad 1 \leq i < j \leq n,\\
& x_i \in \{0,1\}, \quad 1 \leq i \leq n,
\end{aligned}$$

where constraints (3)–(5) enforce $y_{ij} = x_i x_j$. A second reformulation that uses inequalities and real variables to handle quadratic terms, a technique first outlined in Glover (1975), is given as

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n-1} w_i,\\
\text{s.t.} \quad & \sum_{i=1}^{n} x_i = p,\\
& w_i \leq x_i \sum_{j=i+1}^{n} q_{ij}, \quad 1 \leq i \leq n-1,\\
& w_i \leq \sum_{j=i+1}^{n} q_{ij} x_j, \quad 1 \leq i \leq n-1,\\
& w_i \geq 0, \quad 1 \leq i \leq n-1,\\
& x_i \in \{0,1\}, \quad 1 \leq i \leq n. && (6)
\end{aligned}$$

This formulation was shown in Martí et al. (2010) to be far more efficient than (2). It was later used as the exact solver for the comprehensive empirical analyses presented in Parreño et al. (2021) and Martí et al. (2022).

The first significant advancement in exact methods for the (EMSDP) came in Pisinger (2006). This paper presented several upper bounds based on Lagrangean relaxation, semidefinite programming and reformulation techniques. The upper bounds are computationally cheap and can therefore be implemented in a branch and bound procedure. Numerical results show that for Euclidean distance problems, the procedure is capable of solving problems with $n = 80$ with an average solve time of 60 seconds, but it struggles for sizes $n \geq 100$. Martí et al. (2010) presented a branch and bound algorithm based on partial solutions, where a partial solution is a set of k elements where $k < p$. Upper bounds are then calculated based on all other solutions that contain these k elements. The objective function is split into three parts, and an upper bound for each is calculated. These bounds are then integrated into a branch and bound search tree. While the algorithm is faster than the linear formulation (6), the numerical results show that it struggles to solve instances of $n = 150$ in under an hour of computation time.

This paper answers an open question posed in the recent review paper Martí et al. (2022). That is, while progress in exact methods for variants of the maximum diversity problem have advanced significantly (such as Sayyady & Fathi, 2016 for the max-min problem and Garraffa et al., 2017 for the max-mean problem), a fast exact solver for the max-sum diversity problem remains elusive. The max-sum problem remains the most widely studied problem variation, yet very few exact methods exist. One of the reasons for this might be that the problem is generally nonconcave, meaning the naive application of concave nonlinear programming techniques is not appropriate. However, when the distance measurements are taken as Euclidean, the problem exhibits certain special characteristics that allow for nonlinear programming techniques, particularly *cutting plane methods*, to be applied, even in the presence of nonconcavity.

The cutting plane method (or outer approximation) (Duran & Grossmann, 1986; Leyffer, 1993; Yuan et al., 1988) is one of several deterministic methods that provide general frameworks to tackle concave mixed integer problems. These methods require a concave objective function to guarantee convergence to optimality. For nonconcave quadratic problems, the cutting plane algorithm requires an extra concave reformulation step before applying the cutting planes procedure. In particular, using the property that $x_i = x_i^2$ for $x_i \in \{0,1\}$, the nonconcave objective $f(x) = \frac{1}{2}\langle Qx, x\rangle$ is replaced by a concave function $f'(x) := \frac{1}{2}\big(\langle (Q - \lambda I_n)x, x\rangle + \lambda \sum_{i=1}^{n} x_i\big)$, where λ is the largest eigenvalue of Q (Lima & Grossmann, 2017), and where I_n is the identity matrix of dimension n. This method has been implemented in commercial solvers like `CPLEX` and `Gurobi` (Bliek et al., 2014; Lima & Grossmann, 2017). However, this approach can be slow to converge, particularly when λ is large (Bliek et al., 2014; Bonami et al., 2022). We show in Section 2 that the cutting plane algorithm can solve the nonconcave (EMSDP) directly, without the need for a reformulation step, leading to much faster convergence.

The performance of the cutting plane algorithm is evaluated using two publicly available test sets from the `MDPLIB 2.0` test library (Martí et al., 2021), several randomly generated instances as well as a subset of problems from the `TSPLIB` test library. Numerical results show that the cutting plane algorithm is vastly superior to other exact solvers and is capable of solving large, two-coordinate problems of up to $n = 85{,}900$. The algorithm's performance deteriorates as the number of coordinates grows, however, even in these difficult instances it remains superior to other exact solvers, and is able to solve large 20-coordinate problems of up to $n = 2000$.

The paper is organized as follows. In Section 2, we present an exact cutting plane approach for solving (EMSDP). The convergence to optimality is established in Theorem 2. We then provide in Theorem 5 an estimation of how many non-optimal solutions each cutting plane eliminates at each iteration. Finally, in Section 3, we evaluate the effectiveness of the proposed cutting plane algorithm through extensive numerical experiments.

2. Cutting plane methodology

Denote the feasible set of (EMSDP) as

$$K := \left\{ x \in \{0,1\}^n : \sum_{i=1}^{n} x_i = p \right\}.$$

Let $h : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the tangent plane of f defined as follows,

$$h(x,y) := \langle \nabla f(y), x - y\rangle + f(y), \quad \forall x, y \in \mathbb{R}^n. \tag{7}$$

Given a set $A \subset K$, let $\Gamma_A \subset K \times \mathbb{R}$ be defined as

$$\Gamma_A := \left\{ (x,\theta) \in \mathbb{R}^{n+1} : \; x \in K, \; \theta \leq h(x,y), \; \forall y \in A \right\}.$$

We consider the following auxiliary linear maximization problem

$$\max_{(x,\theta)\in\Gamma_A} \theta. \tag{LP_A}$$

This linearization problem is known as the cutting-plane model of (EMSDP), and can be written explicitly as

$$\max \quad \theta$$

$$\text{s.t. } \theta \le h(x,y), \quad \forall y \in A,$$
$$x \in K. \tag{8}$$

We now briefly review some key properties of Euclidean distance matrices. An $n \times n$ matrix $D = [d_{ij}]$ ($n \ge 1$) is called a *squared* Euclidean distance matrix if there are n vectors $v_1, \ldots, v_n$ in a Euclidean space $\mathbb{R}^s$ ($s \ge 1$) such that $d_{ij} = \|v_i - v_j\|^2$ for all $i, j = 1, \ldots, n$, where $\|\cdot\|$ is the Euclidean norm (see Gower, 1982; Hayden et al., 1999; Schoenberg, 1937). Schoenberg (1935) proved that a symmetric nonnegative matrix D with zero diagonal is a squared Euclidean distance matrix if and only if D is *conditionally negative definite*, that is $\langle Dx, x\rangle \le 0$ for any $x \in \mathbb{R}^n$ with $\sum_{i=1}^n x_i = 0$. Furthermore, it is shown in Schoenberg (1937) that given any squared Euclidean distance matrix D, we can construct n points $u_1, \ldots, u_n$ in $\mathbb{R}^s$ such that

$$d_{ij} = \|v_i - v_j\|^2 = \|u_i - u_j\|, \quad \forall i, j = 1, \ldots, n.$$

As such, the Euclidean distance matrix Q in the (EMSDP) is also a squared Euclidean distance matrix, and therefore it is conditionally negative definite. We now show that when $A = K$, the linear problem (LP_K) is equivalent to the quadratic problem (EMSDP).

Proposition 1. *It holds that*

$$\max_{(x,\theta)\in\Gamma_K} \theta = \max_{x\in K} f(x).$$

Furthermore, if (x^*, θ^*) is a solution of (LP_K), then x^* is a solution of (EMSDP).

Proof. For any feasible solution (x, θ) of (LP_K), we have $x \in K$ and

$$\theta \le h(x,x) = f(x) \le \max_{z\in K} f(z).$$

Therefore, $\max_{(x,\theta)\in\Gamma_K} \theta \le \max_{x\in K} f(x)$. Now, we prove the reverse inequality. Taking into account that Q is conditionally negative definite, and for any feasible solutions $x, y \in K$ we have $\sum_{i=1}^n (x_i - y_i) = 0$, the following inequality holds

$$\langle Q(x-y), x-y\rangle \le 0.$$

The inequality above yields

$$\begin{aligned} h(x,y) - f(x) &= \langle Qy, x-y\rangle + \tfrac{1}{2}\langle Qy, y\rangle - \tfrac{1}{2}\langle Qx, x\rangle \\ &= \langle Qy, x-y\rangle + \tfrac{1}{2}\langle Q(y+x), y-x\rangle \\ &= -\tfrac{1}{2}\langle Q(x-y), x-y\rangle \ge 0. \end{aligned}$$

Thus,

$$h(x,y) \ge f(x), \quad \forall x, y \in K.$$

Taking minimum over all $y \in K$ in the inequality above yields $\min_{y\in K} h(x,y) \ge f(x)$ for each $x \in K$. Thus, $(x, \tilde{\theta}_x)$, with $\tilde{\theta}_x = \min_{y\in K} h(x,y)$, is a feasible solution of (LP_K) such that $\tilde{\theta}_x \ge f(x)$, and therefore

$$\max_{(x,\theta)\in\Gamma_K} \theta \ge \max_{x\in K} \tilde{\theta}_x \ge \max_{x\in K} f(x).$$

Hence, we have proved that $\max_{(x,\theta)\in\Gamma_K} \theta = \max_{x\in K} f(x)$. Finally, let (x^*, θ^*) be a solution of (LP_K). Then, x^* is feasible for (EMSDP), and

$$\max_{x\in K} f(x) = \theta^* \le h(x^*, x^*) = f(x^*),$$

proving the second assertion. □

Based on Proposition 1, we propose a cutting plane approach to solve the quadratic problem (EMSDP) by solving the linear problem (LP_K). Since it is not practical to generate a cutting plane $\theta \le h(x,y)$ for every $y \in K$, our cutting plane algorithm successively generates cuts of type (8) whenever a candidate solution is found. Let A_k denote the set A at iteration k, and let LB_k denote the lower bound at iteration k. We say $x \in K$ is a *candidate solution* if there is $(x, \theta) \in \Gamma_{A_k}$ such that $\theta > \text{LB}_k$. The Euclidean Diversity-Cut (EDC) Algorithm successively generates candidate solutions and adds the cutting planes to the linear model (LP_{A_k}) to eliminate non-optimal solutions until no candidate solution remains in the search space.

Algorithm 1: The Euclidean diversity-cut (EDC) algorithm.

```
Take x^0 ∈ K
Set A_0 ← {x^0}, LB_0 ← f(x_0), k ← 1
while ∃(x^k, θ^k) ∈ Γ_{A_{k-1}} s.t. θ^k > LB_{k-1} do
    LB_k ← max{LB_{k-1}, f(x^k)}
    A_k ← A_{k-1} ∪ {x^k}
    k ← k + 1
end
```

The EDC Algorithm does not require solving the linear problem (LP_{A_k}) to optimality whenever an additional cut is added. Rather, it looks for a feasible solution $(x^k, \theta^k) \in \Gamma_{A_{k-1}}$ that improves upon the current lower bound, i.e., $\theta^k > \text{LB}_{k-1}$. This algorithm outlines the framework of a general branch and cut procedure, where cuts are added during the solve process. A branch and cut procedure based on the EDC Algorithm can be implemented in standard MIP solvers using the *callback* functionality. Callbacks allow certain processes or algorithms to be implemented alongside general branch and bound or branch and cut procedures. In the case of the EDC Algorithm, we begin by solving (LP_{A_0}) with a single cut generated by some feasible solution. Whenever an incumbent solution is found during the solve process, a callback is used to add the associated cutting plane. This allows the MIP solver to preserve the information from previous steps and therefore generates only one search tree, improving the computational performance of the algorithm. A detailed demonstration of the EDC Algorithm implemented using branch and cut is shown in Appendix A.

We now prove that the EDC Algorithm converges to an optimal solution of the (EMSDP).

Theorem 2. *The sequence $\{x^k\} \subset K$ generated by the EDC Algorithm converges to an optimal solution of (EMSDP) after a finite number of steps.*

Proof. Consider the sequence $\{x^k\}$ generated by the EDC Algorithm. We first show that the EDC Algorithm converges after a finite number of steps. Suppose $x^{k_1} = x^{k_2}$ for some $0 \le k_1 < k_2$. Let $(x^{k_2}, \theta^{k_2}) \in \Gamma_{A_{k_2-1}}$. Then $\theta^{k_2} > \text{LB}_{k_2-1}$ and,

$$\theta^{k_2} \le h(x^{k_2}, x^{k_1}) = f(x^{k_1}) \le \text{LB}_{k_2-1}$$

which is a contradiction. This shows that the EDC Algorithm will not revisit a previous point. Since the set K is finite, we must have finite convergence.

We now prove that the algorithm terminates at an optimal solution. Suppose the algorithm terminates at step k, then for every $(\tilde{x}, \tilde{\theta}) \in \Gamma_{A_{k-1}}$, it holds that

$$\tilde{\theta} \le \text{LB}_{k-1} \le \max_{x\in K} f(x) = \max_{(x,\theta)\in\Gamma_K} \theta,$$

where the last equality follows from Proposition 1. Taking the maximum over all $(\tilde{x}, \tilde{\theta}) \in \Gamma_{A_{k-1}}$ in the first inequality, we obtain

$$\max_{(x,\theta)\in\Gamma_{A_{k-1}}} \theta \le \text{LB}_{k-1} \le \max_{(x,\theta)\in\Gamma_K} \theta. \tag{9}$$

Note that because $A_{k-1} \subset K$, we have $\Gamma_K \subset \Gamma_{A_{k-1}}$, and hence

$$\max_{(x,\theta)\in\Gamma_{A_{k-1}}} \theta \ge \max_{(x,\theta)\in\Gamma_K} \theta.$$

From (9), the inequality above and the definition of LB_{k-1}, the following equations hold

$$\max_{(x,\theta)\in\Gamma_{A_{k-1}}} \theta = \max_{(x,\theta)\in\Gamma_K} \theta = \max_{x\in K} f(x) = \mathrm{LB}_{k-1} = f(x^l),$$

for some $l \in \{0, 1, \ldots, k-1\}$, and hence x^l is optimal for (EMSDP). □

We now study the efficiency of the cutting planes by answering the question, after iteration k, how many non-optimal solutions are eliminated by the cut $\theta \le h(x, x^k)$? We first establish the following elementary results.

Proposition 3. *Let $x^k \in K$ ($k \ge 0$) be the iterate generated by the EDC Algorithm during iteration k. If $\nabla f(x^k) = 0$, then x^k is an optimal solution of (EMSDP) and the algorithm terminates.*

Proof. If $\nabla f(x^k) = 0$, then the cutting plane $h(x, x^k) \ge \theta$ becomes

$$f(x^k) \ge \theta. \tag{10}$$

Hence, $f(x^k) \ge \max_{(x,\theta)\in\Gamma_{A_{k+1}}} \theta \ge \max_{(x,\theta)\in\Gamma_K} \theta = \max_{x\in K} f(x)$, and the candidate x^k is an optimal solution of (EMSDP). The constraint (10) implies that there will be no candidate solution found in iteration $k+1$, and hence the EDC Algorithm must terminate. □

Lemma 4. *Let x^k be the iterate generated by the EDC Algorithm during iteration k. Then, for any subsequent iteration $l > k$, it holds that*

$$\frac{\mathrm{LB}_{l-1} - f(x^k)}{\left\|\nabla f(x^k)\right\|} < \left\|x^l - x^k\right\|. \tag{11}$$

Proof. Let $k < l$ be iterations of the EDC Algorithm, and let $(x^l, \theta^l) \in \Gamma_{A_{l-1}}$. Then

$$f(x^k) \le \mathrm{LB}_{l-1} < \theta^l \le f(x^k) + \left\langle \nabla f(x^k), x^l - x^k \right\rangle$$

and hence we have that

$$\mathrm{LB}_{l-1} - f(x^k) < \left\langle \nabla f(x^k), x^l - x^k \right\rangle \le \left\|x^l - x^k\right\| \cdot \left\|\nabla f(x^k)\right\|.$$

As $k < l$, the algorithm did not terminate at k and hence from Proposition 3, $\left\|\nabla f(x^k)\right\| \ne 0$. Therefore we have that

$$\frac{\mathrm{LB}_{l-1} - f(x^k)}{\left\|\nabla f(x^k)\right\|} < \left\|x^l - x^k\right\|$$

thus proving the assertion. □

Theorem 5. *Let k and l be iterations of the EDC Algorithm such that $k < l$. Suppose at iteration l there exists a non-negative integer N_l such that*

$$\sqrt{2N_l} \le \frac{\mathrm{LB}_{l-1} - f(x^k)}{\left\|\nabla f(x^k)\right\|}. \tag{12}$$

Then at step l onwards, the cutting plane $\theta \le h(x, x^k)$ removes at least

$$\sum_{q=0}^{N_l} \binom{p}{q}\binom{n-p}{q}$$

binary points from the feasible set K, where $\binom{a}{b} = \frac{b!(a-b)!}{a!}$ for all $a, b \in \mathbb{N}$, and $a \ge b$.

Proof. Let k and l be iterations of the EDC Algorithm such that $k < l$, and suppose (12) holds for some non-negative integer N_l. It follows from Lemma 4 that at iteration l, the cutting plane $\theta \le h(x, x^k)$ removes all points $x \in K$ such that

$$\left\|x - x^k\right\| \le \sqrt{2N_l}. \tag{13}$$

Consider two sets of indices

$$I_1 := \{i : x_i^k = 1\}, \quad I_2 := \{i : x_i^k = 0\}.$$

Since $x^k \in K$, we have $|I_1| = p$ and $|I_2| = n - p$. For any $x \in K$, we consider

$$I_1(x) := \{i : x_i = 1\}, \quad I_2(x) := \{i : x_i = 0\}.$$

Then for any point $x \in K$, we have that

$$\left\|x - x^k\right\|^2 = |I_1 \cap I_2(x)| + |I_2 \cap I_1(x)|$$

i.e., the squared distance between x and x^k is the sum of the indices that are in x^k but not in x, and the indices that are not in x^k but are in x. Furthermore, we can see that

$$\begin{aligned}
|I_1 \cap I_2(x)| &= \sum_{i=1}^n x_i^k(1 - x_i) \\
&= \sum_{i=1}^n \left(x_i^k - x_i^k x_i + x_i - x_i\right) \\
&= \sum_{i=1}^n \left(x_i^k + x_i(1 - x_i^k) - x_i\right) \\
&= \sum_{i=1}^n x_i(1 - x_i^k) + p - p \\
&= |I_2 \cap I_1(x)|
\end{aligned}$$

and hence

$$\left\|x - x^k\right\| = \sqrt{2|I_1 \cap I_2(x)|}.$$

Now, for any $q \in \{0, 1, \ldots, N_l\}$, consider all the solutions $x \in K$ such that

$$q = |I_1 \cap I_2(x)|. \tag{14}$$

Altogether there are $\binom{p}{q}\binom{n-p}{q}$ feasible solutions $x \in K$ that satisfy (14). Therefore in total, there are precisely $\sum_{q=0}^{N_l} \binom{p}{q}\binom{n-p}{q}$ feasible points in K that satisfy (13). This proves the assertion. □

Theorem 5 provides insight on the strength of individual cuts within the cutting plane algorithm as iterations progress. It shows that cuts are stronger when N_l is chosen larger. We now explore the general strength of cuts of type (8) by comparing them to the standard concave reformulation technique commonly used to solve binary quadratic problems such as (EMSDP). Let $f(x)$ be defined as in (EMSDP), then given a regulator $\rho \in \mathbb{R}$ let

$$f_\rho(x) = \tfrac{1}{2}\left(\langle (Q - \rho I_n)x, x\rangle + \rho \sum_{i=1}^n x_i\right)$$

define a ρ-perturbation of f, where I_n is the identity matrix of dimension n. Given x is binary we have that $x_i = x_i^2$ and hence $f(x) = f_\rho(x)$ for all $x \in \{0, 1\}^n$ and $\rho \in \mathbb{R}$. Therefore solving the perturbed problem given by

$$\max_{x\in K} f_\rho(x)$$

is equivalent to solving the original problem. Provided ρ is chosen such that the perturbed quadratic term is negative semi-definite, i.e., $Q - \rho I_n \preceq 0$, the objective function $f_\rho(x)$ becomes concave, thereby guaranteeing the global convergence of a cutting plane or outer approximation algorithm. This is a common technique found in many binary quadratic solvers (Billionnet & Elloumi, 2007; Lima & Grossmann, 2017). However, the perturbation term should be used with caution, as shown by the following result.

Proposition 6. *Similar to (7), let $h_\rho(x, y)$ denote the tangent plane of $f_\rho(x)$ at $y \in K$. If $\rho_1 \le \rho_2$, then*

$$h_{\rho_1}(x, y) \le h_{\rho_2}(x, y)$$

for all $x \in K$.

Proof. Let $e = (1, \dots, 1) \in \mathbb{R}^n$, then

$$\begin{aligned} h_{\rho_1}(x, y) &= f_{\rho_1}(y) + \left\langle Qy - \rho_1 y + \tfrac{1}{2}\rho_1 e, x - y \right\rangle, \\ &= f_{\rho_1}(y) + \langle Qy, x - y\rangle + \tfrac{1}{2}\rho_1 \langle e - 2y, x - y\rangle. \end{aligned}$$

Now, as $x_i^2 = x_i$ and $y_i^2 = y_i$ we have that

$$\begin{aligned} \langle e - 2y, x - y\rangle &= \sum_{i=1}^{n} \left(x_i - y_i - 2x_i y_i + 2y_i^2\right) = \sum_{i=1}^{n} \left(x_i^2 + y_i^2 - 2x_i y_i\right) \\ &= \sum_{i=1}^{n} (x_i - y_i)^2 \geq 0. \end{aligned}$$

Therefore,

$$\begin{aligned} h_{\rho_1}(x, y) &= f_{\rho_1}(y) + \langle Qy, x - y\rangle + \tfrac{1}{2}\rho_1 \langle e - 2y, x - y\rangle \\ &\leq f_{\rho_1}(y) + \langle Qy, x - y\rangle + \tfrac{1}{2}\rho_2 \langle e - 2y, x - y\rangle \\ &= f_{\rho_2}(y) + \langle Qy, x - y\rangle + \tfrac{1}{2}\rho_2 \langle e - 2y, x - y\rangle = h_{\rho_2}(x, y), \end{aligned}$$

as required. □

Let $\lambda_{\max}$ denote the largest eigenvalue of Q. It is proved in Hammer & Rubin (1970) that when $\rho \geq \lambda_{\max}$, $Q - \rho I_n$ is negative semidefinite and hence $f_\rho(x)$ is concave. As such, an outer approximation algorithm is guaranteed to converge thanks to concavity in $f_\rho(x)$. However, unlike outer approximation, our cutting plane algorithm does not require any perturbation or reformulation of the original problem. In other words, the EDC Algorithm converges to the global solution even for the case where $\rho = 0$. Moreover, from Proposition 6, we can see that cuts become weaker (remove fewer nonoptimal solutions) as ρ increases. Hence, the EDC Algorithm is expected to perform better than an outer approximation algorithm applied to the perturbed problem.

3. Numerical results

We now explore the performance of the EDC Algorithm on a range of test instances. The algorithm was implemented in CPLEX Version 22.1 using the callback functionality. As explained in the previous section, callbacks allow for cuts to be added to the model during the general solve procedure, thus generating only one branch and cut search tree. The program was compiled using g++ and run on a machine with a 2.3 GHz AMD EPYC processor with 32 GB of RAM, using a single thread.

The performance of the EDC Algorithm was evaluated on three publicly available and four randomly generated test libraries. The publicly available test library MDPLIB 2.0[1] (Martí et al., 2021) has commonly been used as a benchmark for the maximum diversity problem and contains many test sets. Within this test library, we use the test sets GKD-c and GKD-d. Test set GKD-c contains 20 Euclidean distance matrices of 500 locations. Each location is defined by 20 coordinates in the range of 0 to 100. Test set GDK-d contains 70 Euclidean distance matrices between randomly generated points with two coordinates in the range 0 to 100. There are ten matrices for each value $n = 25, 50, 100, 250, 500, 1000, 2000$. One of the major differences between these test sets is the number of coordinates of original locations. To explore further the effect the number of coordinates has on the algorithm, we randomly generate an additional four test sets similar to GKD-d, where each test set uses a different number of coordinates. Finally, in order to test the limits of the EDC Algorithm, we use a subset of the very large problems available within the TSPLIB test library.[2]

The algorithm was compared against three exact solution methods. The first method solves the Glover reformulation (6) using CPLEX. This linear reformulation was shown in Martí et al. (2022) to be competitive among other exact solvers. Additionally, as CPLEX can handle binary quadratic programs, we also solve the problem in its original form using quadratic CPLEX. The final exact approach is to apply outer approximation to the perturbed objective function, $f_\rho(x)$, where $\rho = \lambda_{\max}$. Such a perturbation makes the function concave and hence guarantees the global convergence of outer approximation. Note that when using outer approximation, calculating $\lambda_{\max}$ is done as a preprocessing step and is not considered to count towards the solver's runtime. Finally, we compared the performance of the algorithm against the heuristic algorithm OBMA (Zhou et al., 2017), which was shown in Martí et al. (2022) to be one of the most effective on Euclidean instances.

Table 1 summarises the performance of all solvers on test sets GKD-c and GKD-d. The table is broken down by time limit, test set, and $\frac{p}{n}$ ratio. Then, the average final optimality gap at the end of the time limit is reported for each exact solver. For all solvers, including the heuristic OBMA, the number of times the final solution matched the best-known solution is also reported.[3] Note that this is not necessarily the number of times the algorithm was able to confirm optimality but rather gives an indication of the solvers' ability to locate good solutions quickly.

On test set GKD-d, the EDC Algorithm was vastly superior to other exact solvers across all time limits. For the 10-seconds time limit, the algorithm was able to confirm optimality in almost all cases and locate more optimal solutions than the heuristic OBMA. Increasing the time limit to 100 seconds, the algorithm could easily solve all test instances of set GKD-d to optimality (including the large instances of $n = 2000$), representing a significant improvement when compared to the other exact algorithms. While still superior to the other exact solution methods, the algorithm's performance appears slightly worse on test set GKD-c. It is able to locate almost all optimal solutions within the 600-seconds time limit, however the algorithm struggles to close the optimality gap completely. That said, the performance is still a noticeable improvement compared to the other exact solvers.

Figure 1 shows the performance of the EDC Algorithm on test set GKD-d. The figure shows the average solve time and the average number of cuts required to solve to optimality for each value of n and p in the test set. Interestingly, the average solve time for large instances of $n = 2000$ remains less than 10 seconds. Furthermore, increasing the size of the problem from $n = 250$ to $n = 2000$ demands a similar number of cuts to prove optimality. This is testament to the strength of the cuts themselves and their ability to remove a vast number of nonoptimal solutions easily, as shown in Theorem 5. Finally, we note that substantially fewer cuts are required for $p = \lceil 0.5n \rceil$ compared with $p = \lceil 0.1n \rceil$ and $p = \lceil 0.2n \rceil$.

It is worth noting that the performance of the EDC Algorithm seems to contradict a previously held notion about the difficulty of diversity problems. Martí et al. (2022) state that as p approaches $n/2$, a problem instance becomes harder due to the larger number of feasible solutions. While this may be true for many existing exact and heuristic solvers, this result was not observed for the EDC Algorithm. The results in Table 1 and Fig. 1 show that when p is chosen as the larger value, fewer cuts are required on average, and therefore the problem is solved faster. This contradicts the statement in Martí et al. (2022) and shows that the run time does not increase for the EDC Algorithm as p approaches $n/2$.

Table 2 details the performance of the EDC Algorithm on test set GKD-c after an hour of solve time. For each value of $\frac{p}{n}$, there are 20 test instances, and we report the number of tests where the EDC Algorithm managed to prove optimality within the time limit, as well as the average solve time, optimality gap and number of cuts added. When $p = \lceil 0.5n \rceil$, the algorithm can still solve

[1] Available at https://www.uv.es/rmarti/paper/mdp.html.

[2] Available at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[3] The best-known values for large instances ($n \geq 500$) are listed in Appendix B.

Table 1
For every combination of time limit, test set and $\frac{p}{n}$ ratio, we report the average final gap as a percentage for all exact solvers. For all solvers (including the heuristic OBMA), we also report the number of times the final solution matched the best-known solution.

				Average gap (%)				Number best				
Time limit (seconds)	Set	Tests	$\frac{p}{n}$	EDC Algorithm	Glover CPLEX	Quadratic CPLEX	Outer approx	EDC algorithm	Glover CPLEX	Quadratic CPLEX	Outer approx	OBMA
10	GKD-d	70	0.1	0.0001	104.5611	3193.7143	1222.6446	69	20	28	10	60
10	GKD-d	70	0.2	0.0000	70.9375	851.8493	621.9356	70	13	25	1	68
10	GKD-d	70	0.5	0.0000	29.6054	128.7590	153.1647	70	20	29	1	70
10	GKD-c	20	0.1	0.1439	126.9552	921.7808	1610.4214	5	0	0	0	20
10	GKD-c	20	0.2	0.0178	123.7596	406.2149	731.2725	12	0	0	0	20
10	GKD-c	20	0.5	0.0002	67.1518	100.4986	178.9432	20	0	0	0	20
100	GKD-d	70	0.1	0.0000	97.6257	3156.0311	1139.4223	70	29	30	10	70
100	GKD-d	70	0.2	0.0000	73.0000	841.8472	580.3135	70	26	28	3	70
100	GKD-d	70	0.5	0.0000	29.9724	125.2907	145.2900	70	32	30	1	70
100	GKD-c	20	0.1	0.0995	116.9812	921.7808	1597.5409	9	0	0	0	20
100	GKD-c	20	0.2	0.0067	104.1303	406.2149	721.2425	18	0	0	0	20
100	GKD-c	20	0.5	0.0001	57.8678	100.4986	175.8901	20	0	0	0	20
600	GKD-d	70	0.1	0.0000	91.1853	3120.6212	1103.8050	70	40	30	10	70
600	GKD-d	70	0.2	0.0000	75.9072	833.0172	548.4910	70	37	29	10	70
600	GKD-d	70	0.5	0.0000	32.0015	121.4612	141.8368	70	40	30	1	70
600	GKD-c	20	0.1	0.0668	109.6791	904.1314	1588.1770	15	0	0	0	20
600	GKD-c	20	0.2	0.0022	93.2871	402.3066	715.2041	20	0	0	0	20
600	GKD-c	20	0.5	0.0000	51.5980	99.2862	175.7679	20	0	0	0	20

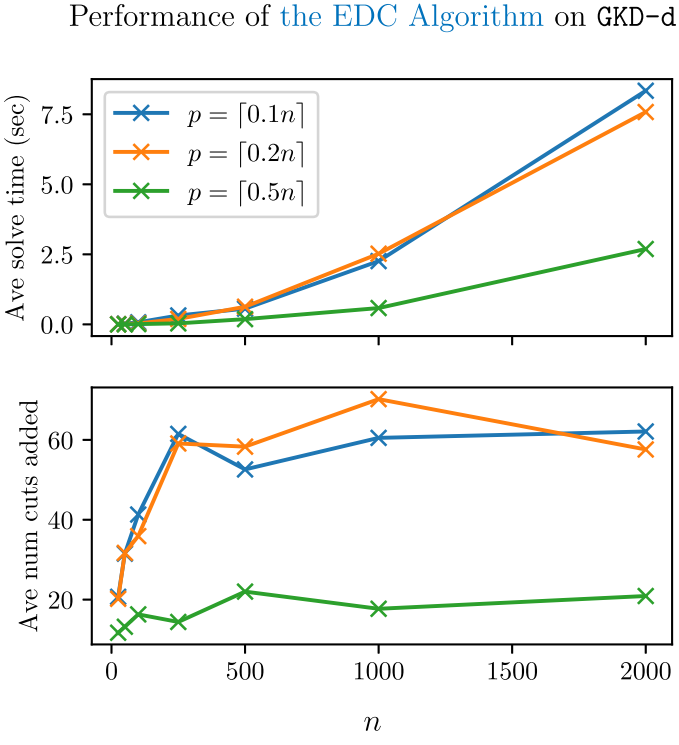

Fig. 1. Performance of the EDC Algorithm on test set GKD-d. For each value n and p, there are 10 problems to solve. We report the average solve time and the average number of cuts added for each pair n and p.

Table 2
Performance of the EDC Algorithm on test set GKD-c. For each value $\frac{p}{n}$, there are 20 problems to solve. The number of tests solved to optimality, average solve time (sec), gap (%) and cuts added after an hour time limit is reported.

$\frac{p}{n}$	Number optimal	Average solve time (seconds)	Average gap (%)	Average cuts added
0.1	8	2738.1935	0.0382	6704.65
0.2	19	643.5760	0.0001	2632.50
0.5	20	28.0190	0.0000	381.30

problem instances to optimality well within the time limit. However, when $p = \lceil 0.1n \rceil$, the algorithm could only solve eight tests within an hour. While the final optimality gap is small, the number of cuts required significantly increases compared to the results seen in Fig. 1. The number of cuts required to solve an instance in GKD-c can be more than 100 times that of a similar-sized problem in GKD-d, the key difference between these tests being the number of coordinates of each location. This suggests that the cut strength decreases as the number of original coordinates increases.

To explore this relationship further, four new tests sets are introduced. Each test set contains 5 Euclidean distance matrices for each value $n = 25, 50, 100, 250, 500, 1000, 2000$, totalling 35 distance matrices in each test set. The key difference between the test sets is the number of coordinates of original locations. The four test sets are denoted as GKD-d5 (with $s = 5$), GKD-d10 (with $s = 10$), GKD-d15 (with $s = 15$), GKD-d20 (with $s = 20$). Each coordinate is then uniformly randomly generated in the range 0 to 100. As before, every instance is then run with $p = \lceil 0.1n \rceil, \lceil 0.2n \rceil, \lceil 0.5n \rceil$, making three tests for each distance matrix.

Figure 2 outlines the performance profiles for the four exact solution methods on the four new test sets GKD-d5, GKD-d10, GKD-d15 and GKD-d20. The results show that as the number of coordinates increases from 5 to 20, the performance of the EDC Algorithm deteriorates substantially. This is in contrast to the other three exact solvers, whose performance remains fairly stable as the number of coordinates increases. Although the algorithm is not able to solve large coordinate instances as effectively, it still remains superior to the other exact solution methods. For GKD-d5, the EDC Algorithm can solve almost all tests to optimality within the 600-seconds time limit (including some of the large instances with $n = 2000$). However, once the number of coordinates increases to 20, the performance is almost halved. This appears to support the observation made on test sets GKD-c and GKD-d that the strength of the cuts decreases as the number of coordinates increases.

Table 3 outlines the performance of the EDC Algorithm on test sets GKD-d5, GKD-d10, GKD-d15 and GKD-d20 in more detail. For the large ratio problems where $p = \lceil 0.5n \rceil$, the algorithm can still easily solve all problem instances well within the time limit. However, across all ratios, the number of cuts required to prove optimality is significantly higher than the results seen on set

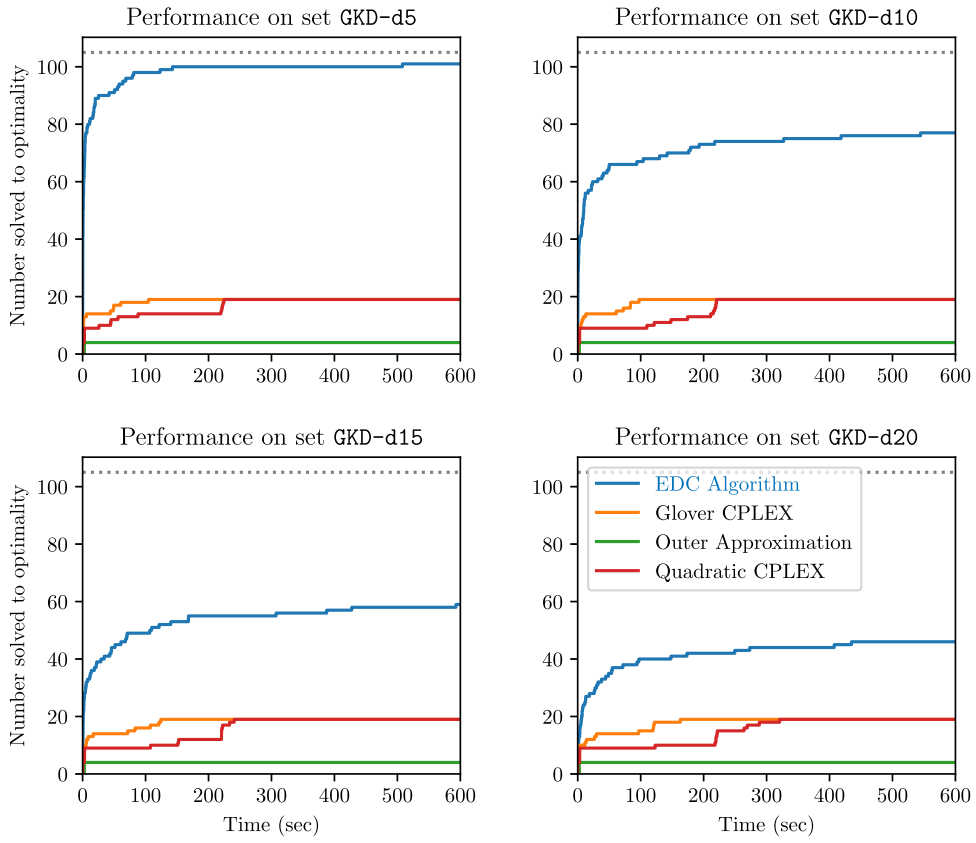

Fig. 2. Performance profile on test sets GKD-d5, GKD-d10, GKD-d15 and GKD-d20. For each test set, there are a total of 105 test instances. The number solved to optimality is shown for each of the exact solvers used.

Table 3
Performance of the EDC Algorithm on test sets GKD-d5, GKD-d10, GKD-d15 and GKD-d20 over an hour time limit. For each pair of test set and ratio $\frac{p}{n}$, 35 tests are solved. The number solved to optimality, average solve time, optimality gap and number of cuts are reported.

Set	$\frac{p}{n}$	Number optimal	Average solve time (seconds)	Average gap (%)	Average number cuts added
GKD-d5	0.1	32	322.0020	0.0006	990.0857
GKD-d5	0.2	35	29.7180	0.0000	418.4000
GKD-d5	0.5	35	1.1731	0.0000	46.1429
GKD-d10	0.1	23	1469.1109	0.0263	4005.7429
GKD-d10	0.2	32	554.1971	0.0005	2302.8286
GKD-d10	0.5	35	5.2383	0.0000	149.8286
GKD-d15	0.1	12	2395.1338	0.0909	6416.5938
GKD-d15	0.2	21	1824.7223	0.0276	5652.9143
GKD-d15	0.5	35	29.1460	0.0000	463.9143
GKD-d20	0.1	9	2604.2534	0.1462	7237.7586
GKD-d20	0.2	13	2325.6716	0.0391	6901.9355
GKD-d20	0.5	35	144.6117	0.0000	1310.8286

GKD-d. As such, the EDC Algorithm's performance on high coordinate instances with low $\frac{p}{n}$ ratio is considerably worse and is often unable to prove optimality within an hour time limit.

We now test the limits of the EDC Algorithm on a subset of test instances available within the TSPLIB test library. The library contains several location problems of very large dimensions (up to $n = 85,900$) and contains original coordinate locations. To solve such large instances, the computational implementation is modified slightly such that the pairwise distances between locations are only calculated when required. As such, the full pairwise distance matrix is no longer loaded into memory, only the original coordinates. In doing so, we avoid the memory burden that arises from saving large distance matrices. However, this strategy means that generating cuts requires calculating all pairwise distances associated with a given solution, creating extra steps to generate cuts. This is not expected to create significant issues, as we have already shown that for two-dimensional problems, the number of cuts required to prove optimality is very small.

Within the TSPLIB test library, 17 Euclidean two-coordinate test instances with $n \geq 2000$ are used. As before, every test instance is then run with $p = \lceil 0.1n \rceil, \lceil 0.2n \rceil, \lceil 0.5n \rceil$, comprising three tests for each set of locations. Each problem is then solved to proven optimality using the EDC Algorithm. In Table 4, we report the solve time in seconds and the number of cuts added across the three values of $\frac{p}{n}$. The results in Table 4 are consistent with previous tests. The number of cuts required to solve the problem to optimality remains small, indicating that the cuts are tight, and hence large problems are easily solved within a reasonable time

Table 4
Performance of the EDC Algorithm on a subset of tests within the TSPLIB test library. For each set of locations, the problem is run with $p = \lceil 0.1n \rceil$, $\lceil 0.2n \rceil$, and $\lceil 0.5n \rceil$, and we report the solve time in seconds and the number of cuts required to solve the problem to optimality.

		$p = \lceil 0.1n \rceil$		$p = \lceil 0.2n \rceil$		$p = \lceil 0.5n \rceil$	
Instance	n	Solve time (seconds)	Cuts	Solve time (seconds)	Cuts	Solve time (seconds)	Cuts
d2103.tsp	2103	8.43	98	7.49	62	6.54	22
u2152.tsp	2152	9.02	86	6.93	55	10.52	34
u2319.tsp	2319	6.58	74	6.30	43	12.19	34
pr2392.tsp	2392	7.43	76	8.20	53	10.59	28
pcb3038.tsp	3038	12.89	78	27.61	109	27.44	45
fl3795.tsp	3795	21.04	90	36.67	69	38.83	41
fnl4461.tsp	4461	48.09	143	35.09	64	66.42	48
rl5915.tsp	5915	65.14	117	116.67	121	116.75	49
rl5934.tsp	5934	54.27	93	51.94	54	81.12	34
pla7397.tsp	7397	121.92	118	166.82	106	198.15	54
rl11849.tsp	11,849	208.72	87	363.62	89	389.30	37
usa13509.tsp	13,509	675.39	197	324.01	64	339.21	26
brd14051.tsp	14,051	507.15	148	676.59	116	506.14	36
d15112.tsp	15,112	676.25	171	776.97	116	842.26	49
d18512.tsp	18,512	799.56	129	775.45	73	1379.92	56
pla33810.tsp	33,810	2053.07	113	3320.10	110	3519.85	47
pla85900.tsp	85,900	18291.56	151	16374.19	74	19986.27	38

frame. Even for the very large problems of $n = 85,900$, the number of cuts required to solve for $p = \lceil 0.1n \rceil$ is only 151. This is a very small number considering the size of the problem. The strength in cuts allows this very large problem to be solved to optimality in five hours.

4. Conclusions and future work

This paper presented a cutting plane algorithm for the max-sum diversity problem. While the problem is inherently nonconcave, the cuts are shown to be appropriate, and the algorithm converges to the optimal solution. As the cuts can be applied directly to the original problem, the algorithm can avoid the reformulation steps needed in some integer quadratic solvers such as CPLEX.

The EDC Algorithm's performance was evaluated on several test libraries, where it was found to be vastly superior to other exact solution methods. The algorithm is especially effective for low-coordinate problems where the cuts are tight, allowing the algorithm to solve large instances quickly. As the number of coordinates grows, the cuts become less effective, and the algorithm's performance deteriorates. The reason for this is unclear, however, we suspect it is closely related to the rank of the matrix. It is known that the rank of a Euclidean distance matrix of points in $\mathbb{R}^s$ is at most $s+2$ (see Dokmanic et al., 2015). Therefore, as the number of original coordinates increases, so does the rank of the matrix. We can think of this as increasing the complexity of the distance matrix and, therefore, the complexity of the maximum diversity problem. Future research should explore this relationship further and examine why the cuts are weaker for a larger number of coordinates and if there are ways to combat this challenge.

The cuts are appropriate because they do not remove feasible solutions and are upper planes of the objective in the domain of feasible solutions. However, the cuts are no longer valid if either the integrality condition or constraint (1) is relaxed. As such, this maximisation problem can be considered to be concave when the domain is restricted to purely discrete feasible solutions. This proof of concavity is different to much of the previous research into cutting plane algorithms for mixed-integer nonlinear programming, where concavity is usually assured by showing concavity when the integrality condition, constraint set, or both, are relaxed. Future research should (1) redefine concavity notions on the discrete domains and (2) uncover other well-known integer nonlinear maximisation problems where the problem is concave on the domain of feasible solutions, but nonconcave when the domain is relaxed.

Acknowledgements

The authors are supported by the Australian Research Council through the Centre for Transforming Maintenance through Data Science (grant number IC180100030). This work was supported by resources provided by the Pawsey Supercomputing Research Centre with funding from the Australian Government and the Government of Western Australia.

Appendix A. Worked example

We now demonstrate in detail the steps of the EDC Algorithm through a worked example. Suppose we are asked to solve

$$\max \quad f(x) = \frac{1}{2}\langle Qx, x\rangle, \tag{A.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_i = p, \tag{A.2}$$

$$x_i \in \{0,1\}, \qquad i = 1,\dots,n, \tag{A.3}$$

where Q is a symmetric hollow (zero diagonal) matrix. The first step is to confirm that Q is a Euclidean distance matrix. Given the original locations, this could easily be confirmed by finding the pairwise Euclidean distances between locations and confirming they agree with the corresponding component of Q. Without the original locations, one can use the *Schoenberg Criterion* to determine whether Q is a Euclidean distance matrix. We begin by constructing the $(n-1)\times(n-1)$ matrix $G = [g_{ij}]_{i,j=2,\dots,n}$ where

$$g_{ij} = \tfrac{1}{2}\big(q_{1i} + q_{1j} - q_{ij}\big).$$

It is then shown in Schoenberg (1935) that a symmetric hollow $n \times n$ matrix Q is a Euclidean distance matrix if and only if the matrix G is positive semidefinite. Then, provided Q is a Euclidean distance matrix, the EDC Algorithm will converge to a globally optimal solution.

The algorithm begins by finding a feasible solution to (A.1). This can be done using some heuristic procedure or by choosing an arbitrary set of p locations. Let the starting solution be denoted by

x^0, then $LB^0 = f(x^0)$. We are then required to find some feasible solution to the subproblem

$$\begin{aligned} \max \quad & \theta^0 \\ s.t. \quad & \theta^0 \leq f(x^0) + \left\langle \nabla f(x^0), x - x^0 \right\rangle \\ & \theta^0 > LB^0 \\ & (A.2), (A.3). \end{aligned} \tag{A.4}$$

Note that while the problem is to maximise θ^0, we may terminate whenever an integer feasible solution is found. While the optimal solution of the subproblem is not required, the maximisation highlights quality search directions. Suppose a feasible solution to (A.4) was given by x^1. Then $LB^1 = \max\{LB^0, f(x^1)\}$. Then the next subproblem is to solve

$$\max \quad \theta^1 \tag{A.5}$$

$$\begin{aligned} s.t. \quad & \theta^1 \leq f(x^0) + \left\langle \nabla f(x^0), x - x^0 \right\rangle \\ & \theta^1 \leq f(x^1) + \left\langle \nabla f(x^1), x - x^1 \right\rangle \\ & \theta^1 > LB^1 \\ & (A.2), (A.3). \end{aligned} \tag{A.6}$$

This process is repeated until no feasible solution exists. At which point, by Theorem 2, we will have converged to the optimal solution.

Appendix B. Summary of results

We now report the results of the EDC Algorithm on large test instances ($n \geq 500$) in GKD-c and GKD-d. For each instance, a time limit of one hour is set, and the final objective value, gap as a percentage and solve time in seconds is reported.

Set	Instance	n	p	Objective value	Gap (%)	Solve time (seconds)
GKD-c	GKD-c_1_n500_m50.txt	500	50	19483.736005	0.10799	3600.01
GKD-c	GKD-c_2_n500_m50.txt	500	50	19701.534791	0.0	1380.11
GKD-c	GKD-c_3_n500_m50.txt	500	50	19547.206837	0.07894	3600.03
GKD-c	GKD-c_4_n500_m50.txt	500	50	19596.468381	0.0	967.77
GKD-c	GKD-c_5_n500_m50.txt	500	50	19602.622005	0.0	2052.38
GKD-c	GKD-c_6_n500_m50.txt	500	50	19421.539095	0.04772	3600.01
GKD-c	GKD-c_7_n500_m50.txt	500	50	19534.303582	0.07854	3600.02
GKD-c	GKD-c_8_n500_m50.txt	500	50	19486.671883	0.09651	3600.01
GKD-c	GKD-c_9_n500_m50.txt	500	50	19221.628796	0.0	3152.37
GKD-c	GKD-c_10_n500_m50.txt	500	50	19703.339933	0.03193	3600.01
GKD-c	GKD-c_11_n500_m50.txt	500	50	19587.120163	0.06183	3600.04
GKD-c	GKD-c_12_n500_m50.txt	500	50	19360.223938	0.0	832.0
GKD-c	GKD-c_13_n500_m50.txt	500	50	19366.698508	0.01825	3600.01
GKD-c	GKD-c_14_n500_m50.txt	500	50	19458.564660	0.01806	3600.01

(continued on next column)

Set	Instance	n	p	Objective value	Gap (%)	Solve time (seconds)
GKD-c	GKD-c_15_n500_m50.txt	500	50	19422.146399	0.06996	3600.03
GKD-c	GKD-c_16_n500_m50.txt	500	50	19678.190189	0.08247	3600.01
GKD-c	GKD-c_17_n500_m50.txt	500	50	19331.388407	0.0	2037.43
GKD-c	GKD-c_18_n500_m50.txt	500	50	19461.394615	0.0	1134.67
GKD-c	GKD-c_19_n500_m50.txt	500	50	19474.800409	0.0711	3600.01
GKD-c	GKD-c_20_n500_m50.txt	500	50	19604.843569	0.0	6.94
GKD-c	GKD-c_1_n500_m50.txt	500	100	75809.909454	0.0	310.82
GKD-c	GKD-c_2_n500_m50.txt	500	100	76435.555515	0.0	171.99
GKD-c	GKD-c_3_n500_m50.txt	500	100	75838.544292	0.0	814.39
GKD-c	GKD-c_4_n500_m50.txt	500	100	75716.598228	0.0	2180.7
GKD-c	GKD-c_5_n500_m50.txt	500	100	75974.214099	0.0	126.84
GKD-c	GKD-c_6_n500_m50.txt	500	100	75701.792895	0.0	13.58
GKD-c	GKD-c_7_n500_m50.txt	500	100	75976.436247	0.00229	3600.01
GKD-c	GKD-c_8_n500_m50.txt	500	100	76019.127767	0.0	57.55
GKD-c	GKD-c_9_n500_m50.txt	500	100	75086.877566	0.0	100.94
GKD-c	GKD-c_10_n500_m50.txt	500	100	76248.246543	0.0	6.4
GKD-c	GKD-c_11_n500_m50.txt	500	100	76011.670781	0.0	11.07
GKD-c	GKD-c_12_n500_m50.txt	500	100	75016.432733	0.0	397.74
GKD-c	GKD-c_13_n500_m50.txt	500	100	75124.215072	0.0	2998.98
GKD-c	GKD-c_14_n500_m50.txt	500	100	75617.353164	0.0	10.87
GKD-c	GKD-c_15_n500_m50.txt	500	100	75294.900081	0.0	14.3
GKD-c	GKD-c_16_n500_m50.txt	500	100	76448.106158	0.0	48.55
GKD-c	GKD-c_17_n500_m50.txt	500	100	75250.955832	0.0	1057.48
GKD-c	GKD-c_18_n500_m50.txt	500	100	75553.610354	0.0	13.69
GKD-c	GKD-c_19_n500_m50.txt	500	100	75546.717875	0.0	75.56
GKD-c	GKD-c_20_n500_m50.txt	500	100	75735.371694	0.0	860.06
GKD-c	GKD-c_1_n500_m50.txt	500	250	443719.723950	0.0	0.83
GKD-c	GKD-c_2_n500_m50.txt	500	250	447225.685882	0.0	0.56
GKD-c	GKD-c_3_n500_m50.txt	500	250	443045.271559	0.0	1.54
GKD-c	GKD-c_4_n500_m50.txt	500	250	442681.290872	0.0	1.03
GKD-c	GKD-c_5_n500_m50.txt	500	250	445473.734823	0.0	8.61
GKD-c	GKD-c_6_n500_m50.txt	500	250	442339.168595	0.0	2.51
GKD-c	GKD-c_7_n500_m50.txt	500	250	445073.835956	0.0	8.2
GKD-c	GKD-c_8_n500_m50.txt	500	250	446734.873298	0.0	0.36
GKD-c	GKD-c_9_n500_m50.txt	500	250	439877.960575	0.0	0.78
GKD-c	GKD-c_10_n500_m50.txt	500	250	444247.120213	0.0	0.68
GKD-c	GKD-c_11_n500_m50.txt	500	250	443853.915174	0.0	516.78
GKD-c	GKD-c_12_n500_m50.txt	500	250	440215.996121	0.0	0.91
GKD-c	GKD-c_13_n500_m50.txt	500	250	440976.117254	0.0	2.95
GKD-c	GKD-c_14_n500_m50.txt	500	250	442437.866858	0.0	0.7

(continued on next page)

Set	Instance	n	p	Objective value	Gap (%)	Solve time (seconds)
GKD-c	GKD-c_15_n500_m50.txt	500	250	440651.808120	0.0	0.93
GKD-c	GKD-c_16_n500_m50.txt	500	250	448315.951367	0.0	0.2
GKD-c	GKD-c_17_n500_m50.txt	500	250	440843.055227	0.0	3.26
GKD-c	GKD-c_18_n500_m50.txt	500	250	441399.555997	0.0	7.14
GKD-c	GKD-c_19_n500_m50.txt	500	250	441966.308420	0.0	1.78
GKD-c	GKD-c_20_n500_m50.txt	500	250	440996.317489	0.0	0.63
GKD-d	GKD_d_1_n500_coor.txt	500	50	93273.991901	0.0	0.48
GKD-d	GKD_d_2_n500_coor.txt	500	50	95855.136509	0.0	0.69
GKD-d	GKD_d_3_n500_coor.txt	500	50	94219.878866	0.0	0.5
GKD-d	GKD_d_4_n500_coor.txt	500	50	95180.977517	0.0	0.7
GKD-d	GKD_d_5_n500_coor.txt	500	50	91962.114047	0.0	0.54
GKD-d	GKD_d_6_n500_coor.txt	500	50	95149.924437	0.0	0.53
GKD-d	GKD_d_7_n500_coor.txt	500	50	96618.050271	0.0	0.18
GKD-d	GKD_d_8_n500_coor.txt	500	50	94917.487984	0.0	0.64
GKD-d	GKD_d_9_n500_coor.txt	500	50	95234.618740	0.0	0.36
GKD-d	GKD_d_10_n500_coor.txt	500	50	95195.426995	0.0	0.99
GKD-d	GKD_d_1_n500_coor.txt	500	100	356991.795777	0.0	0.26
GKD-d	GKD_d_2_n500_coor.txt	500	100	365625.471836	0.0	1.51
GKD-d	GKD_d_3_n500_coor.txt	500	100	359815.333541	0.0	0.3
GKD-d	GKD_d_4_n500_coor.txt	500	100	361327.488337	0.0	0.42
GKD-d	GKD_d_5_n500_coor.txt	500	100	352597.651107	0.0	0.24
GKD-d	GKD_d_6_n500_coor.txt	500	100	362028.785981	0.0	0.61
GKD-d	GKD_d_7_n500_coor.txt	500	100	368290.092382	0.0	0.3
GKD-d	GKD_d_8_n500_coor.txt	500	100	359798.807175	0.0	0.99
GKD-d	GKD_d_9_n500_coor.txt	500	100	361444.899761	0.0	1.31
GKD-d	GKD_d_10_n500_coor.txt	500	100	363096.240153	0.0	0.35
GKD-d	GKD_d_1_n500_coor.txt	500	250	2003881.529806	0.0	0.18
GKD-d	GKD_d_2_n500_coor.txt	500	250	2046325.289184	0.0	0.09
GKD-d	GKD_d_3_n500_coor.txt	500	250	2013203.283427	0.0	0.15
GKD-d	GKD_d_4_n500_coor.txt	500	250	2013390.437355	0.0	0.2
GKD-d	GKD_d_5_n500_coor.txt	500	250	1995722.778598	0.0	0.08
GKD-d	GKD_d_6_n500_coor.txt	500	250	2016805.751319	0.0	0.12
GKD-d	GKD_d_7_n500_coor.txt	500	250	2060936.704385	0.0	0.08
GKD-d	GKD_d_8_n500_coor.txt	500	250	2004824.886174	0.0	0.3
GKD-d	GKD_d_9_n500_coor.txt	500	250	2031726.233231	0.0	0.32
GKD-d	GKD_d_10_n500_coor.txt	500	250	2036966.697518	0.0	0.35
GKD-d	GKD_d_1_n1000_coor.txt	1000	100	379396.664223	0.0	1.42
GKD-d	GKD_d_2_n1000_coor.txt	1000	100	372966.630874	0.0	1.56
GKD-d	GKD_d_3_n1000_coor.txt	1000	100	373355.875821	0.0	2.73
GKD-d	GKD_d_4_n1000_coor.txt	1000	100	378060.355307	0.0	2.06
GKD-d	GKD_d_5_n1000_coor.txt	1000	100	371493.807089	0.0	5.33
GKD-d	GKD_d_6_n1000_coor.txt	1000	100	379212.777302	0.0	1.23
GKD-d	GKD_d_7_n1000_coor.txt	1000	100	375718.555535	0.0	2.12
GKD-d	GKD_d_8_n1000_coor.txt	1000	100	381667.774945	0.0	2.33
GKD-d	GKD_d_9_n1000_coor.txt	1000	100	376493.160043	0.0	1.79
GKD-d	GKD_d_10_n1000_coor.txt	1000	100	375135.506418	0.0	2.02
GKD-d	GKD_d_1_n1000_coor.txt	1000	200	1438611.847242	0.0	1.98
GKD-d	GKD_d_2_n1000_coor.txt	1000	200	1420412.950815	0.0	1.07
GKD-d	GKD_d_3_n1000_coor.txt	1000	200	1421214.934140	0.0	5.44
GKD-d	GKD_d_4_n1000_coor.txt	1000	200	1437118.696240	0.0	1.89
GKD-d	GKD_d_5_n1000_coor.txt	1000	200	1415046.354927	0.0	4.44
GKD-d	GKD_d_6_n1000_coor.txt	1000	200	1437228.201619	0.0	1.54
GKD-d	GKD_d_7_n1000_coor.txt	1000	200	1430955.645799	0.0	1.85
GKD-d	GKD_d_8_n1000_coor.txt	1000	200	1451304.222995	0.0	1.92
GKD-d	GKD_d_9_n1000_coor.txt	1000	200	1435965.052080	0.0	1.62
GKD-d	GKD_d_10_n1000_coor.txt	1000	200	1423298.383413	0.0	3.47
GKD-d	GKD_d_1_n1000_coor.txt	1000	500	8042767.196980	0.0	0.48
GKD-d	GKD_d_2_n1000_coor.txt	1000	500	7939991.703937	0.0	0.69
GKD-d	GKD_d_3_n1000_coor.txt	1000	500	7983853.480681	0.0	0.53
GKD-d	GKD_d_4_n1000_coor.txt	1000	500	8036332.278672	0.0	0.64
GKD-d	GKD_d_5_n1000_coor.txt	1000	500	7936191.375905	0.0	0.49
GKD-d	GKD_d_6_n1000_coor.txt	1000	500	8002225.363103	0.0	0.4
GKD-d	GKD_d_7_n1000_coor.txt	1000	500	7993013.768650	0.0	0.74
GKD-d	GKD_d_8_n1000_coor.txt	1000	500	8101496.623474	0.0	0.58
GKD-d	GKD_d_9_n1000_coor.txt	1000	500	8007673.872360	0.0	0.82
GKD-d	GKD_d_10_n1000_coor.txt	1000	500	7976595.297690	0.0	0.46
GKD-d	GKD_d_1_n2000_coor.txt	2000	200	1500742.236348	0.0	10.44
GKD-d	GKD_d_2_n2000_coor.txt	2000	200	1512175.815531	0.0	5.94
GKD-d	GKD_d_3_n2000_coor.txt	2000	200	1498803.361784	0.0	6.8
GKD-d	GKD_d_4_n2000_coor.txt	2000	200	1509820.670307	0.0	4.53
GKD-d	GKD_d_5_n2000_coor.txt	2000	200	1503100.556353	0.0	12.7
GKD-d	GKD_d_6_n2000_coor.txt	2000	200	1508381.015547	0.0	6.98
GKD-d	GKD_d_7_n2000_coor.txt	2000	200	1506677.211363	0.0	7.22
GKD-d	GKD_d_8_n2000_coor.txt	2000	200	1521157.939744	0.0	7.47
GKD-d	GKD_d_9_n2000_coor.txt	2000	200	1497554.074402	0.0	8.31
GKD-d	GKD_d_10_n2000_coor.txt	2000	200	1492059.155164	0.0	13.01
GKD-d	GKD_d_1_n2000_coor.txt	2000	400	5705960.536860	0.0	5.45
GKD-d	GKD_d_2_n2000_coor.txt	2000	400	5743090.860038	0.0	8.74

(*continued on next column*)

(*continued on next page*)

Set	Instance	n	p	Objective value	Gap (%)	Solve time (seconds)
GKD-d	GKD_d_3_n2000_coor.txt	2000	400	5692144.007735	0.0	8.89
GKD-d	GKD_d_4_n2000_coor.txt	2000	400	5742603.152477	0.0	8.41
GKD-d	GKD_d_5_n2000_coor.txt	2000	400	5730547.159057	0.0	5.28
GKD-d	GKD_d_6_n2000_coor.txt	2000	400	5738995.070550	0.0	7.86
GKD-d	GKD_d_7_n2000_coor.txt	2000	400	5742235.066713	0.0	6.44
GKD-d	GKD_d_8_n2000_coor.txt	2000	400	5777956.898553	0.0	6.96
GKD-d	GKD_d_9_n2000_coor.txt	2000	400	5698890.335320	0.0	6.95
GKD-d	GKD_d_10_n2000_coor.txt	2000	400	5685547.519134	0.0	10.84
GKD-d	GKD_d_1_n2000_coor.txt	2000	1000	31937349.129220	0.0	2.48
GKD-d	GKD_d_2_n2000_coor.txt	2000	1000	31940376.184016	0.0	2.44
GKD-d	GKD_d_3_n2000_coor.txt	2000	1000	31862045.862833	0.0	1.93
GKD-d	GKD_d_4_n2000_coor.txt	2000	1000	32192249.230364	0.0	4.37
GKD-d	GKD_d_5_n2000_coor.txt	2000	1000	32134403.082757	0.0	1.68
GKD-d	GKD_d_6_n2000_coor.txt	2000	1000	32276048.370771	0.0	3.07
GKD-d	GKD_d_7_n2000_coor.txt	2000	1000	32070339.848466	0.0	2.17
GKD-d	GKD_d_8_n2000_coor.txt	2000	1000	32443307.901856	0.0	3.86
GKD-d	GKD_d_9_n2000_coor.txt	2000	1000	31862204.959855	0.0	3.1
GKD-d	GKD_d_10_n2000_coor.txt	2000	1000	31922071.525118	0.0	1.81

References

Billionnet, A., & Elloumi, S. (2007). Using a mixed integer quadratic programming solver for the unconstrained quadratic 0–1 problem. *Mathematical Programming, 109*, 55–68. https://doi.org/10.1007/s10107-005-0637-9.

Bliek, C., Bonami, P., & Lodi, A. (2014). Solving mixed-integer quadratic programming problems with IBM-CPLEX: A progress report. In *Proceedings of the twenty-sixth ramp symposium* (pp. 16–17).

Bonami, P., Lodi, A., & Zarpellon, G. (2022). A classifier to decide on the linearization of mixed-integer quadratic problems in CPLEX. *Operations Research*. https://doi.org/10.1287/opre.2022.2267.

Church, R. L., & Garfinkel, R. S. (1978). Locating an obnoxious facility on a network. *Transportation Science, 12*(2), 107–118. https://doi.org/10.1287/trsc.12.2.107.

Dokmanic, I., Parhizkar, R., Ranieri, J., & Vetterli, M. (2015). Euclidean distance matrices: Essential theory, algorithms, and applications. *IEEE Signal Processing Magazine, 32*, 12–30. https://doi.org/10.1109/MSP.2015.2398954. Conference Name: IEEE Signal Processing Magazine

Duran, M. A., & Grossmann, I. E. (1986). An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming, 36*(3), 307–339. https://doi.org/10.1007/BF02592064.

Eremeev, A. V., Kelmanov, A. V., Kovalyov, M. Y., & Pyatkin, A. V. (2019). Maximum diversity problem with squared Euclidean distance. In M. Khachay, Y. Kochetov, & P. Pardalos (Eds.), *Mathematical optimization theory and operations research*. In *Lecture notes in computer science* (pp. 541–551). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-22629-9_38.

Erkut, E. (1990). The discrete p-dispersion problem. *European Journal of Operational Research, 46*(1), 48–60. https://doi.org/10.1016/0377-2217(90)90297-O.

Ferrero-Guillén, R., Díez-González, J., Verde, P., Martínez-Gutiérrez, A., Alija-Pérez, J.-M., & Álvarez, R. (2022). Optimal chair location through a maximum diversity problem genetic algorithm optimization. In I. Rojas, O. Valenzuela, F. Rojas, L. J. Herrera, & F. Ortuño (Eds.), *Bioinformatics and biomedical engineering*. In *Lecture notes in computer science* (pp. 417–428). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-07704-3_34.

Garraffa, M., Della Croce, F., & Salassa, F. (2017). An exact semidefinite programming approach for the max-mean dispersion problem. *Journal of Combinatorial Optimization, 34*, 71–93. https://doi.org/10.1007/s10878-016-0065-1.

Glover, F. (1975). Improved linear integer programming formulations of nonlinear integer problems. *Management Science, 22*(4), 455–460. https://doi.org/10.1287/mnsc.22.4.455.

Glover, F., & Woolsey, E. (1974). Technical note-converting the 0–1 polynomial programming problem to a 0–1 linear program. *Operations Research, 22*(1), 180–182. https://doi.org/10.1287/opre.22.1.180.

Gower, J. C. (1982). Euclidean distance geometry. *Mathematical Sciences, 7*(1), 1–14.

Hammer, P. L., & Rubin, A. A. (1970). Some remarks on quadratic programming with 0–1 variables. *Revue Française d'Informatique et de Recherche Opérationnelle. Série Verte, 4*(V3), 67–79.

Hayden, T. L., Reams, R., & Wells, J. (1999). Methods for constructing distance matrices and the inverse eigenvalue problem. *Linear Algebra and Its Applications, 295*(1–3), 97–112.

Kuby, M. J. (1987). Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems. *Geographical Analysis, 19*, 315–329. https://doi.org/10.1111/J.1538-4632.1987.TB00133.X.

Kuo, C.-C. C., Glover, F., & Dhir, K. S. (1993). Analyzing and modeling the maximum diversity problem by zero-one programming*. *Decision Sciences, 24*, 1171–1185. https://doi.org/10.1111/J.1540-5915.1993.TB00509.X.

Leyffer, S. (1993). *Deterministic methods for mixed integer nonlinear programming*. Citeseer Ph.D. thesis..

Lima, R. M., & Grossmann, I. E. (2017). On the solution of nonconvex cardinality Boolean quadratic programming problems: A computational study. *Computational Optimization and Applications, 66*(1), 1–37. https://doi.org/10.1007/s10589-016-9856-7.

Martí, R., Duarte, A., Martínez-Gavara, A., & Sánchez-Oro, J. (2021). The MDPLIB 2.0 library of benchmark instances for diversity problems. https://www.uv.es/rmarti/paper/mdp.html.

Martí, R., Gallego, M., & Duarte, A. (2010). A branch and bound algorithm for the maximum diversity problem. *European Journal of Operational Research, 200*, 36–44. https://doi.org/10.1016/J.EJOR.2008.12.023.

Martí, R., Martínez-Gavara, A., Pérez-Peló, S., & Sánchez-Oro, J. (2022). A review on discrete diversity and dispersion maximization from an OR perspective. *European Journal of Operational Research, 299*, 795–813. https://doi.org/10.1016/j.ejor.2021.07.044.

Parreño, F., Álvarez Valdés, R., & Martí, R. (2021). Measuring diversity. A review and an empirical analysis. *European Journal of Operational Research, 289*, 515–532. https://doi.org/10.1016/J.EJOR.2020.07.053.

Pisinger, D. (2006). Upper bounds and exact algorithms for p-dispersion problems. *Computers and Operations Research, 33*, 1380–1398. https://doi.org/10.1016/J.COR.2004.09.033.

Porter, W., Rawal, K., Rachie, K., Wien, H., & Williams, R. (1975). Cowpea germplasm catalog no 1. In *International institute of tropical agriculture, Ibadan, Nigeria.*

Ravi, S. S., Rosenkrantz, D. J., & Tayi, G. K. (1994). Heuristic and special case algorithms for dispersion problems. *Operations Research, 42*(2), 299–310. https://doi.org/10.1287/opre.42.2.299.

Roberge, M.-È., & van Dick, R. (2010). Recognizing the benefits of diversity: When and how does diversity increase group performance? *Human Resource Management Review, 20*(4), 295–308. https://doi.org/10.1016/j.hrmr.2009.09.002.

Sayyady, F., & Fathi, Y. (2016). An integer programming approach for solving the p-dispersion problem. *European Journal of Operational Research, 253*, 216–225. https://doi.org/10.1016/j.ejor.2016.02.026.

Schoenberg, I. J. (1935). Remarks to maurice fréchet's article "sur la définition axiomatique d'une classe d'espaces distanciés vectoriellement applicable sur l'espace de Hilbert". *Annals of Mathematics, 36*(3). https://doi.org/10.2307/1968654.

Schoenberg, I. J. (1937). On certain metric spaces arising from Euclidean spaces by a change of metric and their imbedding in Hilbert space. *Annals of Mathematics*, 787–793. https://doi.org/10.2307/1968835.

Yuan, X., Zhang, S., Pibouleau, L., & Domenech, S. (1988). Une méthode d'optimisation non linéaire en variables mixtes pour la conception de procédés. *RAIRO-Operations Research, 22*(4), 331–346.

Zhou, Y., Hao, J.-K., & Duval, B. (2017). Opposition-based memetic search for the maximum diversity problem. *IEEE Transactions on Evolutionary Computation, 21*, 731–745. https://doi.org/10.1109/TEVC.2017.2674800. Conference Name: IEEE Transactions on Evolutionary Computation