School of Electrical Engineering,
Computing and Mathematical Sciences

Exact Cutting Plane Methods for Quadratic Programming Problems with Applications

Sandy Peter Spiers
0000-0002-0961-0425

This thesis is presented for the Degree of

Doctor of Philosophy
of
Curtin University

June 2024

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature:	____________________
	Sandy Spiers
Date:	June 21, 2024

Abstract

Optimisation methods for nonconcave maximisation problems are fundamentally different from those for their concave counterparts. Concave problems benefit from the helpful property that every local solution is also a global one, enabling the development of computationally efficient algorithms. In contrast, nonconcave problems lack such guarantees. Consequently, the application of concave programming techniques to nonconcave problems cannot guarantee global optimality, nor can they provide valid objective bounds.

This thesis seeks to bridge this methodological divide, by adapting cutting plane methods, traditionally reserved for concave problems, to nonconcave mixed-integer quadratic programming problems. Cutting planes are particularly appealing due to their ease of implementation and versatility across both continuous and integer variable types. Moreover, as these methods result in linear approximations, they can be easily integrated alongside other mathematical programming techniques.

We achieve this algorithmic advancement by developing the novel concept of *directional concavity*, which asserts the concavity of a quadratic function along a given direction. With this knowledge we can determine when and where we can approximate a quadratic function by its tangents, and use this information to search for a globally optimal solution.

Establishing directional concavity for general quadratic functions involves three critical steps. First, we define tractable conditions that can identify concave directions of a quadratic function. To illustrate this, we present some example conditions that apply to a special class of quadratic function characterised by a Euclidean distance matrix. Next, we use strategic functional decomposition to break down a quadratic function into simpler, more manageable components. Identifying concave directions within each component is noticeably easier, and often times more effective. Lastly, we combine these ideas with innovative algorithmic techniques that allow us to effectively explore the entire search space while always staying on concave directions. This integration results in a globally convergent cutting plane algorithm applicable to general quadratic programming problems.

We systematically develop the concept of directional concavity by focusing each chapter of this thesis on one of the steps mentioned above. Each development step is tested against various classes of quadratic programming problems, demonstrating the algorithmic advantage of this novel approach, in some cases reducing solve times by a factor of more than 1000 against the current state-of-the-art methods. This work not only bridges a methodological gap between concave and nonconcave optimisation techniques, but also opens new avenues for advancements in cutting plane methods. In the final chapter of this thesis, we explore an important application of cutting plane-type methods in maintenance scheduling for refinery operations.

Research Outputs

The following papers were completed during the PhD candidature and have either been published or are under review in international peer-reviewed journals:

- Spiers, S., Bui, H. T., & Loxton, R. (2023b). An exact cutting plane method for the Euclidean max-sum diversity problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2023.05.014
- Spiers, S., Bui, H. T., & Loxton, R. (2023a). Coordinate partitioning for difficult Euclidean max-sum diversity problems. *Under review*
- Bui, H. T., Spiers, S., & Loxton, R. (2024). Solving Euclidean Max-Sum problems exactly with cutting planes. *Computers & Operations Research*, *168*, 106682. https://doi.org/10.1016/j.cor.2024.106682
- Spiers, S., Bui, H. T., Loxton, R., Mansour, M. R., Hollins, K., Francis, R., Martindale, C., & Pimpale, Y. (2023). Bayer digestion maintenance optimisation with lazy constraints and Benders decomposition. *Annals of Operations Research*. https://doi.org/10.1007/s10479-023-05561-6

Acknowledgements

First and foremost, I would like to express my deepest thanks to my incredible supervisors, Dr. Hoa Bui and Prof. Ryan Loxton. Thank you for your amazing support, mentorship and friendship. You have made the PhD experience unforgettable, and I know I will remember these years as some of my fondest. Most importantly, thank you for introducing me to the world of optimisation research, way back in my honours year, and for encouraging me to pursue a PhD. If not for your support and guidance, this thesis would not exist.

My PhD was supported by the Center for Transforming Maintenance through Data Science (CTMTDS), for which I am extremely grateful. To my officemates and CTMTDS colleagues, thank you for creating such an amazing and productive research environment. In particular I want to thank Natasha Bartlett, who was always there to listen and encourage. Our early morning coffee chats kept me motivated, excited and only occasionally distracted. Thank you for your incredible mentorship, and for always going the extra mile. A special thanks to Prof. Aloke Phatak for always providing a fresh perspective on the PhD hurdles, and helping me through many of the challenges. To my fellow PhD comrades, Gabriel, Ryan, Ponpot, Yingying & Srimali, thanks for being wonderful lunchtime, pubtime and travel companions. Your support and friendship have made this journal all the more enjoyable.

A big thank you to my industry supervisors, co-authors and mentors for having faith in me and letting me into your world. I am particularly grateful for the support of Dr. Moussa Mansour, who opened my eyes to the world of mathematics beyond the walls of a university. Your insights and encouragement have been instrumental in shaping my research and broadening my horizons.

A special and heartfelt thanks to my family - Mum, Dad and Hugo - for your unwavering love, support and encouragement. I couldn't ask for a better team to have by my side. You are a constant source of inspiration and motivation, and I owe so much of this thesis to you all.

Finally, to Sam, my best friend and better half, your endless and unconditional support has meant everything to me. Words cannot express how grateful I am to have you by my side. Thanks for helping me celebrate all the little wins, and encouraging me to keep pushing. Thanks for always showing your undivided interest in my obscure work. I could not have got here without you.

Thank you all for your invaluable contributions to my academic and personal growth.

Acknowledgement of Country

In the spirit of reconciliation, I acknowledge and pay my deepest respect to the Traditional Owners of the land on which this research was conducted, the Whadjuk people of the Noongar Nation, whose ancestral lands Curtin University is built upon. I also wish to acknowledge the Traditional Owners of the lands I have been fortunate enough to travel on during my PhD, and their enduring connections to land, sea, and community. I pay my respects to their Elders, past and present. This always was, and always will be, Aboriginal land.

Copyright Statement

I have obtained permission from the copyright owners to use any of my own published work (e.g. journal articles) in which the copyright is held by another party (i.e. publisher, co-author).

Contents

1 Introduction

1.1 Motivation and Background

Optimisation is fundamentally a decision science, deeply woven into the fabric of our daily lives. Every day, we are faced with decisions, big and small, that shape our reality and our future. While the weight of our choices may vary, they typically seek a similar goal, to maximise the benefits and desired outcomes of our actions. Although certain aspects of our lives remain beyond our control, the deliberate application of mathematical principles to enhance our decision-making processes is what we refer to as *optimisation*. This field encompasses a broad spectrum of topics, disciplines and philosophies that all share a common objective: to extract the maximum possible value from the resources and options available to us.

In the study of optimisation, we translate real-world decision making into a structured mathematical framework. This transformation involves distilling and encoding the decisions, mechanisms and outcomes of an optimisation problem into a precise mathematical representation. Atop this foundation, we construct the constraints and objectives which, when combined, give rise to our mathematical optimisation model. This model typically manifests in a form such as

$$\begin{aligned} \max \quad & f(x) && (1.1) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \dots, k, && (1.2) \\ & x \in K \subset \mathbb{R}^n. \end{aligned}$$

Here, x are called our decision variables, representative of the real-world options available to a decision-maker. The options we may choose from are represented by the set $K \subset \mathbb{R}^n$, which is often referred to as the domain of x. The constraints that must be adhered to are translated into the general functions $g_i(x)$, which must satisfy (1.2), and the objective is given by $f(x)$, which in this case is to be maximised.

While an optimisation model follows a logical and deterministic structure, in its initial form it stands as nothing more than an abstract algebraic construct. It is only through the application of sophisticated solution methods and algorithms that we may provide concrete values to our decision variables, effectively translating our once abstract model into actionable solutions.

We interact with optimisation solutions every day. Every time you enter a destination into your favourite maps application or receive a parcel from across the country, there is an optimisation model working behind the scenes to find the best solution (Braekers et al., 2016). Even the mammoth task of keeping the world's passenger aeroplanes on time and on route involves the work of massive optimisation models (Bazargan, 2016).

1.2 Solving Optimisation Problems

Extracting solutions from an algebraic optimisation model is usually very challenging. It is our role as optimisation practitioners to propose and develop efficient solution algorithms, appropriate for the problem at hand. While solving small test models may not be particularly taxing, practical problems are often of a much larger scale. In such cases it becomes vital to have solution methods that are accurate, robust and pragmatic. Developing such an algorithm requires a sophisticated understanding of the model's structure and requirements of a potential solution. By unlocking these keys, we can use optimisation techniques to solve complex real-world problems.

However, not all optimisation problems are alike. While problem size is a tangible indicator of potential difficulty, structure arguably plays a more important role. For instance a linear programming model (whereby $f(x)$ and each $g_i(x)$ are affine functions and $K = \mathbb{R}^n$) can be easily solved to global optimality using the well known simplex method (Nelder & Mead, 1965). This means that the solution returned by the algorithm is known to have the best objective function value of any other feasible solution. In practice, the simplex method works very well, and there are many available linear programming solvers that can tackle large problems with relative ease.

While easy to solve, pure linear programming is rare in practical applications. This is because many of the decisions we seek to optimise are discrete. Questions like 'what day should this be scheduled?', 'which train is going first?' or 'how many welders do we need?' all have answers that come from a discrete set, such as the set of days, a set of trains, or the set of integers. The challenge with optimising over a discrete set is the combinatorial effect this has on our solution space. Adding just a single extra binary decision to a model doubles the number of possible combinations. This exponential increase becomes a major burden when solving large scale integer programs.

Optimising over discrete variables is known as integer programming, or mixed-integer programming in the case where there are both discrete and continuous variables. Usually, these problems are solved using a branch and bound algorithm, which works by breaking down the problem into easier to manage pieces (Land & Doig, 1960). Typically, this involves solving a *relaxation* of the original problem. A relaxation is simply a reformulation of the model that involves an expansion of the feasible region that allows the problem to be more easily solved. By virtue of this expansion, the optimal value of the relaxation provides an upper bound of the original problem. Furthermore, the solution found is likely to be infeasible for the original problem. For instance, if $f(x)$ and $g_i(x)$ are all linear, we would typically use the continuous relaxation, whereby an integer variable $x_i \in \mathbb{Z}$ is relaxed to $x_i \in \mathbb{R}$. The continuous relaxation is then easily solved using the simplex method. However, while being easier to solve, the solution of the relaxation is likely to contain some fractional variables which do not satisfy the original constraints.

To refine these solutions towards integer feasibility we can use branching rules by creating subtrees with tighter local bounds on a fractional variable. For instance, if x_i is a variable that should be binary, but has value 0.5 after solving the continuous relaxation, we form two branches, one with $x_i = 0$ and the other with $x_i = 1$. The relaxation of these two child nodes is then solved, and if the upper bound (given by the solution of the relaxation) drops below the objective value of the best known solution, then the optimal solution cannot exist in this branch and hence the node is pruned. The branch and bound algorithm converges globally to the solution of a mixed-integer program by continuing this procedure until an entire search tree is developed.

Formulating a tight relaxation of the problem is crucial for the success of a branch and bound algorithm. A tight relaxation is one whose upper bound provides a good approximation of the actual optimal value. This allows the algorithm to more effectively eliminate branches, and hence converge to the optimal solution faster. If, on the other hand, the relaxation provides a poor bound, the algorithm may descend down unhelpful and misleading branches. The highly combinatorial nature of integer programming problems then means that a huge branching tree is formed, making it difficult for the algorithm to converge. Poor relaxation is a common difficulty found in discrete optimisation, and there are many examples of large integer problems that are easier to solve than small problems with poor relaxations.

Adding to the complexity, real-world optimisation problems often include nonlinear elements which are generally harder to solve than linear equivalents. Unlike in linear programs, asserting global optimality in a nonlinear program can be very difficult, dependent largely on the structure of the overall problem. For continuous problems, the Karush-Kuhn-Tucker (KKT) conditions are a key result used to prove local optimality, also known as first order stationarity, of a solution (Kuhn & Tucker, 1951; Slater, 1950). In the concave problems, i.e., where $f(x)$ is a concave function and each $g_i(x)$ is convex, we have the useful property that any local solution is also a global one. As such, the KKT conditions can be used to prove global optimality.

Note that much of the existing literature refers to the maximisation of a concave function over a convex set as a *convex* problem, since it can be transformed into the minimisation of the convex function $-f(x)$. However, this thesis is primarily interested in maximisation problems, and hence is it is more natural and convenient to refer to them as *concave* problems.

Conversely, *nonconcave* problems may contain many local solutions of varying quality.

Finding global optimality for such problems therefore becomes incredibly difficult, so much so that in many cases finding local stationarity is the only realistic goal. While much research has been conducted on this topic, many of the algorithms proposed are efficient only on a particular class of problem. As stated by the *No Free Lunch* theorems, universal algorithm dominance is impossible to achieve (Wolpert & Macready, 1997). This is especially true for nonconcave programming over discrete variables, which represents a pinnacle of complexity in the study of optimisation.

1.3 Exact Quadratic Programming

Mixed-integer quadratic programming is the problem of maximising a quadratic function over both integer and continuous decision variables. Mathematically, these problems take on the form

$$\begin{aligned} \max \quad & f(x) = \langle Qx + p, x \rangle && (1.3) \\ \text{s.t.} \quad & x \in K \subset \mathbb{R}^n, && (1.4) \end{aligned}$$

where $Q \in \mathbb{R}^{n \times n}$ is a symmetric matrix, $p \in \mathbb{R}^n$ is a real vector and K contains both integer and continuous dimensions.

Quadratic terms are a particularly common form of nonlinearity seen in practice and hence (1.3) has many important applications. In finance and economics, quadratic programs have commonly been used to model the complex interdependencies involved in portfolio selection optimisation (Mencarelli & D'Ambrosio, 2019). Similarly, Aboudolas et al. (2010) demonstrated the application of quadratic programming techniques in optimising urban traffic control systems in a Greek city. Beyond these applications, the well-established least squares problem in statistical regression represents a fundamental example of quadratic programming. More recently, these techniques have gained interest in the field of machine learning, particularly for feature selection of large models (Rodriguez-Lujan et al., 2010).

These problems are known to be NP-hard, and are therefore particularly challenging to solve (Pia et al., 2017). As such, many authors have proposed efficient heuristic, metaheuristic and approximation algorithms for a wide range of both classical and practical problems. Conversely, the literature on exact methods has followed a natural divergence, based on the concavity of the problem. This distinction has a profound effect on the way in which we might solve the problem.

1.3.1 Concave Problems

For the case where Q is negative semi-definite, the continuous relaxation of (1.3) satisfies the Slater conditions, and hence the problem is said to be concave. As such, many of the techniques used in concave nonlinear programming can be extended to mixed-integer quadratic programming. In particular, given we can solve exactly the continuous relaxation, we can use this to formulate a branch and bound search tree, similar to that described earlier. An example of this idea can be seen in Gupta and Ravindran (1985) and Leyffer (2001). In fact, many of today's notable mixed-integer and nonlinear quadratic programming solvers implement this key idea (Kronqvist et al., 2019).

When employing relaxation techniques within a branch and bound framework, there is an important trade-off that needs to be balanced between the tightness of the relaxation and the computational effort required to solve it. If solving the relaxation cannot be done efficiently, then nodes cannot be propagated quickly, leading to slow convergence.

An alternative technique is to form a relaxation that approximates the objective function by cutting planes, thereby keeping the problem linear. For concave objective functions, this is easily achieved through the use of tangent planes, thanks to the fact that

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle \tag{1.5}$$

holds for all $x, y \in \mathbb{R}^n$. Thus, the tangent plane of any y provides a valid upper bound for the function value of x. This important property means we can use cutting planes to form a linear upper approximation of the objective function, and use this as our relaxation. Given a set of cut generating points $A \subset \mathbb{R}^n$, the cutting plane model of (1.3) is given as

$$\begin{aligned} \max \quad & \theta \\ \text{s.t.} \quad & \theta \leq f(y) + \langle \nabla f(y), x - y \rangle, \quad \forall y \in A, \\ & x \in K \subset \mathbb{R}^n. \end{aligned} \tag{1.6}$$

This formulation gives rise to the following important properties.

Proposition 1. *Suppose (1.3) is feasible and bounded. Then, for any nonempty subset $A \subset \mathbb{R}^n$, either (1.6) is unbounded, or its optimal value provides an upper bound for (1.3). Furthermore, if $A = K$, then a solution to (1.6) is also a solution to (1.3).*

Proof. Since (1.3) is feasible, there exists an $x \in K$ and sufficiently small $\theta \in \mathbb{R}$ such that (θ, x) is feasible for (1.6). Suppose (1.6) is bounded and let (θ, x) be an optimal solution. Finally, let x^* be an optimal solution of (1.3). Then from (1.5) and the fact that (θ, x) is optimal, we have that

$$\begin{aligned} \theta &= \min_{y \in A} \{ f(y) + \langle \nabla f(y), x - y \rangle \} \\ &\geq \min_{y \in A} \{ f(y) + \langle \nabla f(y), x^* - y \rangle \} \geq f(x^*). \end{aligned}$$

Therefore (1.6) provides an upper bound for (1.3). Now, suppose $A = K$. Given $x \in K$ and the result above we then have

$$f(x^*) \leq \theta = \min_{y \in K} \{ f(y) + \langle \nabla f(y), x - y \rangle \} = f(x).$$

However, as x^* is optimal we also have $f(x^*) \geq f(x)$ and hence $f(x) = f(x^*)$. Therefore x must also be a solution of (1.3). $\square$

In practice, we rarely require $A = K$ to recover an optimal solution of (1.3). Instead, it is common to implement an algorithm that can find a sufficiently large A so as to also solve (1.3).

The *Extended Cutting Plane Algorithm* from Westerlund and Pettersson (1995) achieves this by iteratively solving (1.6) to optimality and adding the new solution to the set of cuts. The cutting plane model is then re-solved with this additional cut, and the process is repeated until the upper bound provided by (1.6) equals the known lower bound, at which point we have converged to an optimal solution. The benefit of the extended cutting plane algorithm is that each iteration's subproblem is a mixed-integer linear program. As such, we have broken down a quadratic programming problem into a series of linear problems. If (1.6) is relatively easy to solve, cutting plane methods can be more efficient than branch and bound techniques.

We can improve upon this idea by employing an *Outer Approximation* methodology (Duran & Grossmann, 1986). As the problem contains some integer variables, a branch and bound tree must be formed in order to solve (1.6) to optimality. However, we can see from Proposition 1 that A can be *any* nonempty set. Duran and Grossmann (1986) showed that whenever an integer feasible solution is found *during* the branch and bound process, its tangent can be added immediately. This allows cuts to be added 'on-the-fly', meaning our linear approximation is continuously tightened during the search for an optimal solution. As such, only one mixed-integer search tree is required, thereby reducing the computational load.

Outer approximation has become a very popular way in which to solve concave versions of (1.3). Furthermore, the method is easily implementable in modern integer programming solvers using the *lazy constraint callback* functionality. This allows for the injection of user generated code into the general branch and bound solver routine. By using this capability, we can create our own tangent generating functions and therefore easily implement an outer approximation algorithm within our desired mixed-integer programming solver.

1.3.2 Nonconcave Problems

In a nonconcave setting, the problem becomes much more difficult. This is because many of the properties utilised in a concave setting, such as valid tangent planes or that all local solutions are also global solutions, no longer apply. Consequently, even the continuous relaxation of (1.3) is a global optimisation problem, and is thus very challenging.

A common solution methodology used in general mixed-integer nonconcave programming that is easily extensible to (1.3) is the use of spatial branching with convex envelopes. A convex envelope is a set of linear inequalities that define upper and lower approximations of a nonlinear, potentially nonconcave, function. In the case of (1.3), this is usually achieved through the introduction of an auxiliary variable $y_{ij} = x_i x_j$. This formulation can then be linearised by the well known McCormick (1976) inequalities, given by

$$\begin{aligned}
y_{ij} &\geq \overline{x_i} x_j + x_i \overline{x_j} - \overline{x_i}\,\overline{x_j} \\
y_{ij} &\geq \underline{x_i} x_j + x_i \underline{x_j} - \underline{x_i}\,\underline{x_j} \\
y_{ij} &\leq \overline{x_i} x_j + x_i \underline{x_j} - \overline{x_i}\,\underline{x_j} \\
y_{ij} &\leq \underline{x_i} x_j + x_i \overline{x_j} - \underline{x_i}\,\overline{x_j}
\end{aligned}$$

where $\overline{x_i}$ and $\underline{x_i}$ are upper and lower bounds on x_i. However, this linearisation is usually only tight at the bounds of x_i and x_j. To overcome this, we can iteratively tighten the envelope around y_{ij} by branching on the bounds of x_i or x_j. This creates two child nodes with updated local bounds on x_i, which can be used to tighten the $y_{ij} = x_i x_j$ approximation. By integrating this tightening with the general branch and bound procedure for integer feasibility, we eventually converge to global optimality.

For pure binary problems, this linearisation can be very effective, since it is known to be tight when x_i and x_j are at their upper or lower bounds. If $x \in \{0, 1\}^n$, then the McCormick inequalities reduce to

$$y_{ij} \geq x_j + x_i - 1 \tag{1.7}$$
$$y_{ij} \geq 0 \tag{1.8}$$
$$y_{ij} \leq x_j \tag{1.9}$$
$$y_{ij} \leq x_i. \tag{1.10}$$

This provides a valid linearisation such that $y_{ij} = x_i x_j$ where $x_i, x_j \in \{0, 1\}$. As such, (1.3) can be reformulated as the equivalent binary linear program

$$\begin{aligned}
\max \quad & \sum_{i,j=1}^{n} q_{ij} y_{ij} + \sum_{i=1}^{n} p_i x_i \\
\text{s.t.} \quad & (1.7)\text{-}(1.10) \\
& y \in \mathbb{R}^{n \times n}, \\
& x \in K \subset \{0, 1\}^n.
\end{aligned} \tag{1.11}$$

While effective for small scale problems, this technique introduces n^2 additional auxiliary variables and thus does not scale efficiently. When Q contains only nonnegative entries, we may use the more compact linearisation technique outlined in Glover (1975) and given as

$$\begin{aligned} \max \quad & \sum_{i=1}^{n-1} w_i + \sum_{i=1}^{n} p_i x_i \\ \text{s.t.} \quad & w_i \leq x_i \sum_{j=i}^{n} q_{ij}, \quad i = 1, \dots, n-1, \\ & w_i \leq \sum_{j=i}^{n} x_j q_{ij}, \quad i = 1, \dots, n-1, \\ & w \in \mathbb{R}^{n-1}, \\ & x \in K \subseteq \{0,1\}^n. \end{aligned} \tag{1.12}$$

This formulation only introduces $n-1$ additional variables, however has a slightly weaker continuous relaxation compared to (1.11).

Another method that has recently received significant attention for solving (1.3) in both purely binary and mixed-integer domains is the application of semidefinite relaxations. Similar to the McCormick inequalities, we introduce auxiliary variables $Y = xx^T$, however these are now relaxed such that $Y - xx^T \succeq 0$, where $M \succeq 0$ if and only if M is positive semi-definite. The resultant problem can then be solved using any available semidefinite programming solver. In many cases, this formulation is known to provide some of the tightest available relaxations of (1.3) (Burer & Vandenbussche, 2009; Chen & Burer, 2012). Furthermore, semidefinite programs have been shown to be not much more difficult than linear programming (Vandenberghe & Boyd, 1996). However, the number of new decision variables in this approach is n^2, and hence it scales poorly. As such, the use of semidefinite relaxations can struggle to solve large problem sizes.

Cutting plane algorithms can be used for (1.3) when x is binary, however they generally require an extra concave reformulation step. In particular, using the property that $x_i = x_i^2$ for $x_i \in \{0,1\}$, the nonconcave objective $f(x) = \langle Qx, x \rangle + \langle p, x \rangle$ can be replaced by a concave function

$$f'(x) = \langle (Q - \lambda I_n)\, x, x \rangle + \lambda \sum_{i=1}^{n} x_i + \langle p, x \rangle,$$

where λ is the largest eigenvalue of Q, and where I_n is the identity matrix of dimension n (Lima & Grossmann, 2017). The resultant matrix $Q - \lambda I_n$ then has only negative eigenvalues, and hence the quadratic term is concave. We can therefore use cutting plane methods to solve as we might normally with a concave problem. This approach has been implemented in commercial solvers such as `CPLEX` and `Gurobi` (Bliek et al., 2014; Lima & Grossmann, 2017). However, it can be slow to converge, particularly when λ is large (Bliek et al., 2014; Bonami et al., 2022).

Importantly, the method described above demonstrates how cutting plane methods can be extended to nonconcave settings. This has the potential to be quite effective, since concave programming techniques need only search for one local solution. Thus, an important question arises: when and how can we extend cutting plane techniques to nonconcave settings? Specifically, under what circumstances can we use (1.6) to solve (1.3), even in the case where $f(x)$ is nonconcave? For instance, if we knew that $f(x)$ was concave for all feasible solutions $x \in K$, then this may be sufficient. If not, could we augment our search algorithm to ensure we always stay on concave directions? Or divide the feasible region based on the concavity of $f(x)$? An answer to any of these questions would allow us to extend the efficiency of cutting plane methods to difficult nonconcave settings.

1.4 Thesis Overview

This thesis focuses on developing and advancing cutting plane methods, usually reserved for concave problems, to nonconcave quadratic programming. We investigate the conditions under which these methods are not only applicable, but also yield substantial computational improvements, outperforming state-of-the-art commercial solvers and heuristic methods. By extending concave programming techniques to these nonconcave settings, this research aims to address the key questions of *when* and *how* such extensions can be effectively applied.

However, doing so is far from trivial. To achieve extension, we introduce the novel concept of *directional concavity*. Directional concavity allows us to determine whether a function is concave on a given $x - y$ direction, thereby allowing (1.5) to hold. By understanding the concave directions of a quadratic function, we can better utilise cutting plane techniques to solve nonconcave problems. To establish the requirements of directional concavity, each chapter of this thesis will focus on a specific challenge, and examine ways to overcome each.

In Chapter 2, we look at a classical binary quadratic optimisation problem, known as the *diversity problem*. This well known problem seeks to maximise a quadratic objective defined by a Euclidean distance matrix, over a simple cardinality constraint. We prove the interesting property that, although the Euclidean distance matrix is nonconcave, (1.5) holds for any x, y that are *feasible* for the problem. In other words, we always have valid tangents between feasible solutions. This allows us to solve the problem using cutting plane methods, and the resultant algorithm proves to be incredibly computationally efficient on a majority of test instances.

Using the results from Chapter 2 as a motivator, Chapter 3 looks more closely at *when* a cutting plane methodology performs poorly, and why this might be the case. This leads us to an exploration of *functional decomposition*, and how breaking down the objective into its key components can lead to much tighter approximations via tangent planes. Using the diversity problem as a case study, we introduce some heuristic decomposition strategies and show the performance improvement that can arise from these reformulations.

Chapter 4 extends the results from Chapter 2 and shows how cutting plane methods can be used on *general* quadratic programming problems where the objective is defined by a Euclidean distance matrix. Unlike the diversity problem, we no longer have that (1.5) holds for *all* feasible solutions. However, that is not to say we cannot still use cutting plane techniques to solve the problem. This chapter uses the concept of directional concavity to determine the directions of valid tangents. With knowledge of the concave directions of the objective, we can augment our search algorithm to always stay on these directions. This allows us to solve nonconcave problems using concave techniques by simply changing our search strategy.

In Chapter 5, we expand upon the notion of directional concavity to encompass general matrices, moving beyond the confines of only Euclidean distance matrices. By integrating the results, techniques, and perspectives of the previous chapters, we introduce an exact algorithm that revolutionises the application of cutting plane methodologies in scenarios previously deemed unsuitable. This novel approach offers a cohesive strategy for applying cutting plane methods across a broader class of problems. The performance and outcomes of this algorithm are very encouraging, and highlight the benefits of this new strategy.

Finally, in Chapter 6 we explore some practical applications of optimisation and develop a maintenance scheduling optimisation model for digester banks. Digester banks are network-connected assets that lie on the critical path of the Bayer process, a chemical refinement process that converts bauxite ore into alumina. Given the complexity of scheduling maintenance for large fleets of digester banks, a continuous-time, mixed-integer linear program is formulated to find the cost-minimising maintenance schedule that satisfies all required constraints. A solution approach that employs lazy constraints and Benders decomposition is proposed to solve the model. We solve the scheduling model for realistic scenarios involving two Bayer refineries based in Western Australia. The purpose of this chapter is to highlight the importance of pragmatic solution algorithms, tailored to the problem at hand.

2 The Max-Sum Diversity Problem[1]

This chapter 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 maximisation 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 (EMSDP). 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.

2.1 Introduction

The problem of maximising diversity and dispersion arises in many practical settings. It involves selecting a subset of elements from a larger set to maximise 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 maximise the sum of distances between selected elements, and the max-min diversity problem (Erkut, 1990), which is to maximise the minimum distance among selected points. In this chapter, we focus our attention on the Euclidean max-sum diversity problem (EMSDP).

Given a set of n predefined locations $u_1, \dots, u_n$ 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 maximised. Here, we consider d_{ij} to be the distance between locations i and j defined by $d_{ij} = \|u_i - u_j\|$ where $\|\cdot\|$ is the standard Euclidean distance in $\mathbb{R}^s$. Let $D = [d_{ij}]$ denote the full distance matrix where $i = 1, \dots, n$ and $j = 1, \dots, n$. The (EMSDP) is then

[1]This chapter is based on Spiers, Bui, and Loxton (2023b).

given as

$$\begin{aligned} \max \quad & f(x) = \frac{1}{2}\langle Dx, x\rangle, \\ \text{s.t.} \quad & x \in K \end{aligned} \tag{EMSDP}$$

where

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

The (EMSDP) is known to be strongly NP-hard (Eremeev et al., 2019; Kuo et al., 1993; Ravi et al., 1994). Practical applications of the maximum diversity problem are vast. One of the first examples presented in the literature is in locating unwanted facilities on a network (Church & Garfinkel, 1978; Kuby, 1987). This application was recently used to find the optimal location of temporary waste collection points in an Indonesian city (Julianto et al., 2023). Furthermore, diversity maximisation has also been used to find the optimal locations of chairs for COVID-19 social distancing (Ferrero-Guillén et al., 2022).

Research into solution methods for the max-sum diversity problem has mainly focused on heuristics and meta-heuristics. Recently, Zhou et al. (2017) introduced an opposition-based memetic algorithm for the (EMSDP). They define *opposite solutions* as being $\bar{x} = x - 1$, and use these solutions to diversify the search of a machine learning algorithm. The algorithm achieved impressive results, and was later shown in Martí et al. (2022) to be one of the best performers for Euclidean instances of the (EMSDP). For further discussion on heuristics for the (EMSDP), see the excellent review paper by Martí et al. (2022).

While heuristic methods for the (EMSDP) have made significant progress, the development 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 linearisation technique presented in Glover and 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

$$\max \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} y_{ij}, \tag{2.1}$$

$$\text{s.t.} \quad y_{ij} \geq x_i + x_j - 1, \qquad 1 \leq i < j \leq n,$$

$$y_{ij} \leq x_i, \qquad 1 \leq i < j \leq n, \tag{2.2}$$

$$y_{ij} \leq x_j, \qquad 1 \leq i < j \leq n, \tag{2.3}$$

$$y_{ij} \geq 0, \qquad 1 \leq i < j \leq n,$$

$$x \in K,$$

where constraints (2.2)-(2.3) enforce $y_{ij} = x_i x_j$. Note that this formulation is equivalent to the McCormick envelopes introduced in Chapter 1.3.2 where x_i and x_j are binary. A second reformulation that uses inequalities and real variables to handle quadratic terms, a technique first outlined in Glover (1975), is given as

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{n-1} w_i, && \\
\text{s.t.} \quad & w_i \leq x_i \sum_{j=i+1}^{n} d_{ij}, && 1 \leq i \leq n-1, \\
& w_i \leq \sum_{j=i+1}^{n} d_{ij} x_j, && 1 \leq i \leq n-1, \\
& w_i \geq 0, && 1 \leq i \leq n-1, \\
& x \in K.
\end{aligned}
\tag{2.4}
$$

This formulation was shown in Martí et al. (2010) to be far more efficient than (2.1). 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 Lagrangian 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 (2.4), the numerical results show that it struggles to solve instances of $n = 150$ in under an hour of computation time.

This chapter 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 use of cutting plane methods for (EMSDP) have seen limited application in existing literature. However, this technique has been applied to closely related problems, such as the unconstrained binary quadratic problem, standard quadratic programming problem and quadratic knapsack problem. In the majority of cases, this has been through the application of the concave reformulation step outlined in Chapter 1.3.2. As such, these methods have rarely seen particularly impressive performances. In this chapter, we show that the cutting plane algorithm can solve the nonconcave (EMSDP) directly, without the need for a reformulation step.

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 $n = 85900$. 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 chapter is organised as follows. In Section 2.2 we derive some well known and important results pertaining to Euclidean distance matrices. Section 2.3 uses these results to formulate an exact cutting plane approach for solving (EMSDP). The convergence to optimality is established in Theorem 10. We then provide in Theorem 13 an estimation of how many non-optimal solutions each cutting plane eliminates at each iteration. Finally, in Section 2.4, we evaluate the effectiveness of the proposed cutting plane algorithm through extensive numerical experiments.

2.2 Euclidean Distance Geometry

We begin by deriving some important results regarding Euclidean distance matrices. An $n \times n$ matrix $D = [d_{ij}^2]_{1 \le i,j \le n}$ is called a *squared* Euclidean distance matrix if there are n vectors $v_1, \ldots, v_n$ in a Euclidean space $\mathbb{R}^s$ such that $d_{ij}^2 = \|v_i - v_j\|^2$ for all $i, j = 1, \ldots, n$, where $\|\cdot\|$ is the Euclidean norm (see Schoenberg, 1937, Gower, 1982, Bapat & Raghavan, 1997, Hayden et al., 1999 and citations therein). The existing literature on Euclidean geometry has mainly focused on squared distance variety, as opposed to the real distance matrix defined by $[d_{ij}]_{1 \le i,j \le n}$ and used in the (EMSDP). We now demonstrate how these differing definitions are intrinsically linked.

Let $V = [v_1, \ldots, v_n] \in \mathbb{R}^{s \times n}$ be a real matrix whose columns are given by the locations $v_1, \ldots, v_n \in \mathbb{R}^s$. The *Gram* matrix of V is given by the square, symmetric, $n \times n$ matrix,

$$G = V^T V = \left[\langle v_i, v_j \rangle\right]_{1 \le i,j \le n}.$$

Recall from the law of cosines,

$$\left\|v_i - v_j\right\|^2 = \left\|v_i\right\|^2 + \left\|v_j\right\|^2 - 2\left\langle v_i, v_j\right\rangle.$$

This identity provides an alternative definition of G, given by

$$G = \frac{1}{2}\left[\left\|v_i\right\|^2 + \left\|v_j\right\|^2 - \left\|v_i - v_j\right\|^2\right]_{1\le i,j\le n}.$$

This definition becomes especially useful when we only have knowledge of D, i.e., we know the pairwise distances but not the locations. Given pairwise distances are translation invariant, let us suppose v_1 defines the origin. Then the $(n-1)\times(n-1)$ Gram matrix of the distance matrix D is given by

$$G = \frac{1}{2}\left[d_{1i}^2 + d_{1j}^2 - d_{ij}^2\right]_{2\le i,j\le n}. \tag{2.5}$$

The following results are credited to various works by Schoenberg and prove some important properties relating a distance matrix to its Gram matrix.

Theorem 2 (Schoenberg, 1935). *A symmetric, hollow (zero diagonal), nonnegative matrix $D \in \mathbb{R}^{n\times n}$ is a squared Euclidean distance matrix of n points in $\mathbb{R}^s$ (but not $\mathbb{R}^{s-1}$) if and only if its Gram matrix is positive semidefinite with rank s.*

Proof. We begin by proving the forward statement. Suppose D is a squared Euclidean distance matrix of points $v_1, \dots, v_n \in \mathbb{R}^s$. Then by the law of cosines,

$$d_{1i}^2 + d_{1j}^2 - d_{ij}^2 = \left\|v_i - v_1\right\|^2 + \left\|v_j - v_1\right\|^2 - \left\|v_i - v_j\right\|^2 = 2\left\langle v_i - v_1, v_j - v_1\right\rangle.$$

Therefore its Gram matrix G, given by (2.5), is equivalent to

$$G = \left[\left\langle v_i - v_1, v_j - v_1\right\rangle\right]_{2\le i,j\le n} = W^T W$$

where $W = [v_2 - v_1, \dots, v_n - v_1] \in \mathbb{R}^{s\times(n-1)}$. Hence, G is clearly positive semidefinite since its principal root W is real. Furthermore, given these points are not embeddable in $\mathbb{R}^{s-1}$ they must be linearly independent in $\mathbb{R}^s$. Therefore, W is of full row rank and hence the rank of G is s.

For the reverse statement, we have that G is the Gram matrix of D, defined by (2.5). Given G is positive semidefinite, its principal root exists and is real. Suppose this is given as $W = [v_2, \dots, v_n] \in \mathbb{R}^{s\times(n-1)}$, and let $v_1 = \mathbf{0}$ define the origin. Then from (2.5), for $i = 2, \dots, n$, we have

$$\left\|v_i - v_1\right\|^2 = \left\|v_i - \mathbf{0}\right\|^2 = \left\|v_i\right\|^2 = \langle v_i, v_i\rangle = g_{ii} = d_{1i}^2.$$

Similarly, for $i, j = 2, \dots, n$,

$$\left\|v_i - v_j\right\|^2 = \left\|v_i\right\|^2 + \left\|v_j\right\|^2 - 2\left\langle v_i, v_j\right\rangle = g_{ii} + g_{jj} - 2g_{ij} = d_{1i}^2 + d_{1j}^2 - d_{1i}^2 - d_{1j}^2 + d_{ij}^2 = d_{ij}^2.$$

Therefore, $D = \left[\left\|v_i - v_j\right\|^2\right]_{1\le i,j\le n}$ as required. This concludes the proof. □

Theorem 3. *A symmetric, hollow, nonnegative matrix $D \in \mathbb{R}^{n\times n}$ is a squared Euclidean distance matrix of n points in $\mathbb{R}^s$ if and only if it is conditionally negative definite, that is that $\langle Dx, x\rangle \le 0$ for all $x \in \mathbb{R}^n$ such that $\sum_{i=1}^n x_i = 0$.*

Proof. It suffices to prove that D is conditionally negative definite if and only if its Gram matrix G is positive semidefinite. Let $x \in \mathbb{R}^n$ be such that $\sum_{i=1}^n x_i = 0$. Then we have from (2.5) that

$$\begin{aligned}
\sum_{i,j=1}^n x_i x_j \left(d_{1i}^2 + d_{1j}^2 - 2d_{ij}^2\right) &= \sum_{i,j=1}^n x_i x_j d_{1i}^2 + \sum_{i,j=1}^n x_i x_j d_{1j}^2 - 2\sum_{i,j=1}^n x_i x_j d_{ij}^2 \\
&= 2\sum_{i=1}^n x_i \left(\sum_{j=1}^n x_j d_{1j}^2\right) - 2\sum_{i,j=1}^n x_i x_j d_{ij}^2 \\
&= 2\left(\sum_{i=1}^n x_i\right)\left(\sum_{j=1}^n x_j d_{1j}^2\right) - 2\sum_{i,j=1}^n x_i x_j d_{ij}^2 \\
&= -2\sum_{i,j=1}^n x_i x_j d_{ij}^2 \ge 0.
\end{aligned}$$

This concludes both forward and reverse proofs. □

Theorem 4. *If D is a squared Euclidean distance matrix, then it has exactly one positive eigenvalue.*

Proof. Firstly, D must be hollow, since $d_{ii} = \|v_i - v_i\| = 0$. Let $\lambda_1 \ge \cdots \ge \lambda_n$ and $v_1, \dots, v_n \in \mathbb{R}^n$ denote the eigenvalues and eigenvectors of D. Then given $\sum_{i=1}^n \lambda_i = \operatorname{tr}(D) = 0$, we must have that D has at least one positive eigenvalue. Assume, for a contradiction, that D has 2 or more positive eigenvalues, i.e., $\lambda_2 > 0$. Let

$$\alpha = \begin{cases} 0, & \text{if } \sum_{i=1}^n v_{1i} = 0, \\ 1, & \text{if } \sum_{i=1}^n v_{1i} \ne 0 \text{ and } \sum_{i=1}^n v_{2i} = 0, \\ \sum_{i=1}^n v_{1i} / \sum_{i=1}^n v_{2i}, & \text{otherwise}, \end{cases}$$

and let $x = v_1 - \alpha v_2$. Then $\langle Dx, x\rangle = \lambda_1 + \alpha^2\lambda_2 > 0$. However, note that $\sum_{i=1}^n x_i = 0$. Consequently, $\langle Dx, x\rangle \le 0$ since D is also conditionally negative definite, creating a contradiction. Therefore D has exactly one positive eigenvalue. □

Lemma 5 (Schoenberg, 1938b, see also Schoenberg, 1938a)**.** *If $v_1, \dots, v_n \in \mathbb{R}^s$ then*

$$\sum_{i,j=1}^n e^{-\lambda^2\|v_i - v_j\|^2} x_i x_j \ge 0$$

for all $x \in \mathbb{R}^n$ and $\lambda > 0$.

Proof. Let $w \in \mathbb{R}$ be a real number, and recall the well known Fourier transform of a Gaussian function (see Abramowitz & Stegun, 1968, pp. 302; Bracewell & Kahn, 1966, pp. 105-108),

$$e^{-w^2} = (4\pi)^{-1/2} \int_{-\infty}^{\infty} e^{iwu} e^{-u^2/4} du.$$

Then, observe that for any $v \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$ we have

$$\begin{aligned}
e^{-\lambda^2\|v\|^2} &= e^{-(\lambda v_1)^2} \cdots e^{-(\lambda v_n)^2} \\
&= (4\pi)^{-1/2} \int_{-\infty}^{\infty} e^{i\lambda v_1 u_1} e^{-u_1^2/4} du_1 \cdots (4\pi)^{-1/2} \int_{-\infty}^{\infty} e^{iv_n u_n} e^{-u_n^2/4} du_n \\
&= (4\pi)^{-n/2} \int_{\mathbb{R}^n} e^{i\lambda\langle v,u\rangle} e^{-\|u\|^2/4} d^n u \\
&= (4\pi)^{-n/2} \left(\int_{\mathbb{R}^n} \cos\left(\lambda\left\langle v,u\right\rangle\right) e^{-\|u\|^2/4} d^n u + i \int_{\mathbb{R}^n} \sin\left(\lambda\left\langle v,u\right\rangle\right) e^{-\|u\|^2/4} d^n u \right) \\
&= (4\pi)^{-n/2} \int_{\mathbb{R}^n} \cos\left(\lambda\left\langle v,u\right\rangle\right) e^{-\|u\|^2/4} d^n u, \qquad (2.6)
\end{aligned}$$

where (2.6) comes from the fact that $\sin(\cdot)$ is an odd function. Therefore, for all $x \in \mathbb{R}^n$,

$$\sum_{j,k=1}^{n} e^{-\lambda^2\|v_j - v_k\|^2} x_j x_k = (4\pi)^{-n/2} \int_{\mathbb{R}^n} \left(\sum_{j,k=1}^{n} \cos\left(\lambda\left\langle v_j - v_k, u\right\rangle\right) x_j x_k \right) e^{-\|u\|^2/4} d^n u. \qquad (2.7)$$

Furthermore, observe that

$$\begin{aligned}
\sum_{j,k=1}^{n} \cos\left(\lambda\left\langle v_j - v_k, u\right\rangle\right) x_j x_k &= \sum_{j,k=1}^{n} \cos\left(\lambda\left\langle v_j, u\right\rangle\right) \cos\left(\lambda\left\langle v_j, u\right\rangle\right) x_j x_k \\
&\quad + \sum_{j,k=1}^{n} \sin\left(\lambda\left\langle v_j, u\right\rangle\right) \sin\left(\lambda\left\langle v_j, u\right\rangle\right) x_j x_k \\
&= \left(\sum_{j=1}^{n} \cos\left(\lambda\left\langle v_j, u\right\rangle\right) x_j \right)^2 + \left(\sum_{j=1}^{n} \sin\left(\lambda\left\langle v_j, u\right\rangle\right) x_j \right)^2 \geq 0
\end{aligned}$$

and $e^{-\|u\|^2} \geq 0$ and therefore we must have (2.7) is nonnegative as required. □

Theorem 6 (Schoenberg, 1935). *$D = [d_{ij}^2]_{1\leq i,j\leq n}$ is a squared Euclidean distance matrix if and only if $D^{(\alpha)} = [d_{ij}^{2\alpha}]_{1\leq i,j\leq n}$ is also a squared Euclidean distance matrix where* $0 < \alpha < 1$.

Proof. The following proof relies on the existence of the helper integral

$$I(\alpha) = \int_0^{\infty} \left(1 - e^{-\lambda^2}\right) \lambda^{-1-2\alpha} d\lambda. \qquad (2.8)$$

We begin by proving the existence of this integral for $0 < \alpha < 1$. Using the substitution $\lambda = \sqrt{\mu}$ we get

$$I(\alpha) = \int_0^\infty (1 - e^{-\mu})\, \mu^{(-1-2\alpha)/2} \mu^{-1/2} \tfrac{1}{2} d\mu = \tfrac{1}{2} \int_0^\infty (1 - e^{-\mu})\, \mu^{-1-\alpha} d\mu.$$

Then using integration by parts,

$$I(\alpha) = \frac{-1}{2\alpha} \left[\mu^{-\alpha} (1 - e^{-\mu})\right]_0^\infty + \frac{1}{2\alpha} \int_0^\infty \mu^{-\alpha} e^{-\mu} d\mu.$$

Then, for $0 < \alpha < 1$ we have the following,

$$\lim_{\mu \to 0} \{\mu^{-\alpha}(1 - e^{\mu})\} = \lim_{\mu \to 0} \left\{\frac{-e^{\mu}}{\alpha \mu^{\alpha - 1}}\right\} = \frac{1}{\alpha} \lim_{\mu \to 0} \left\{-e^{\mu} \mu^{1-\alpha}\right\} = \frac{1}{\alpha} \left(-e^0\right) \left(0^{1-\alpha}\right) = 0, \tag{2.9}$$

$$\lim_{\mu \to \infty} \{\mu^{-\alpha}(1 - e^{-\mu})\} = 0,$$

$$\int_0^\infty \mu^{(1-\alpha)-1} e^{-\mu} d\mu = \Gamma(1 - \alpha), \tag{2.10}$$

where (2.9) comes from L'Hospital's rule (Taylor, 1952) and (2.10) comes from the definition of a Gamma function (see Davis, 1959). Therefore for $0 < \alpha < 1$,

$$I(\alpha) = \frac{1}{2\alpha} \Gamma(1 - \alpha) \geq 0$$

and hence the integral exists.

Substituting $\lambda = t\mu$ where $t \geq 0$ into (2.8) we get

$$I(\alpha) = \int_0^\infty \left(1 - e^{-t^2 \mu^2}\right) (t\mu)^{-1-2\alpha} t d\mu = t^{-2\alpha} \int_0^\infty \left(1 - e^{-t^2 \mu^2}\right) \mu^{-1-2\alpha} d\mu$$

and therefore

$$t^{2\alpha} = \frac{1}{I(\alpha)} \int_0^\infty \left(1 - e^{-t^2 \mu^2}\right) \mu^{-1-2\alpha} d\mu.$$

Forward Proof. Let $t = d_{ij}$, then

$$d_{ij}^{2\alpha} = \frac{1}{I(\alpha)} \int_0^\infty \left(1 - e^{-d_{ij}^2 \mu^2}\right) \mu^{-1-2\alpha} d\mu.$$

Since D is a squared Euclidean distance matrix there exists $v_1, \ldots, v_n \in \mathbb{R}^n$ such that $d_{ij} = \|v_i - v_j\|^2$ for $i, j = 1, \ldots, n$ and hence by Lemma 5 we have $\sum_{i,j=1}^n e^{-d_{ij}^2 \mu^2} x_i x_j \geq 0$ for all $x \in \mathbb{R}^n$. Then, for all x with $\sum_{i=1}^n x = 0$, we get

$$\sum_{i,j=1}^n d_{ij}^{2\alpha} x_i x_j = \frac{-1}{I(\alpha)} \int_0^\infty \left(\sum_{i,j=1}^n e^{-d_{ij}^2 \mu^2} x_i x_j\right) \mu^{-1-2\alpha} d\mu \leq 0 \tag{2.11}$$

and hence $D^{(\alpha)}$ is conditionally negative definite meaning it is a squared Euclidean distance matrix.

Reverse Proof. For the reverse proof, we have that $D^{(\alpha)}$ is conditionally negative definite and hence (2.11) holds. Furthermore, observe that for $0 < \alpha < 1$ we have $I(\alpha) = \frac{1}{2\alpha}\Gamma(1-\alpha) \geq 0$ and $\mu^{-1-2\alpha} \geq 0$ for $\mu \geq 0$. As such,

$$\begin{aligned}\sum_{i,j=1}^{n} d_{ij}^2 x_i x_j &= \lim_{\alpha\to 1^-} \sum_{i,j=1}^{n} d_{ij}^{2\alpha} x_i x_j \\ &= \lim_{\alpha\to 1^-} \frac{-1}{I(\alpha)} \int_0^\infty \left(\sum_{i,j=1}^{n} e^{-d_{ij}^2\mu^2} x_i x_j \right) \mu^{-1-2\alpha} d\mu \leq 0\end{aligned}$$

as required. □

Corollary 7. *Given a set of points $u_1, \dots, u_n \in \mathbb{R}^s$, there exists a set of points $v_1, \dots, v_n \in \mathbb{R}^t$ such that $\|u_i - u_j\| = \|v_i - v_j\|^2$ for all i, j.*

Proof. By the previous theorem, $D^{(1/2)} = [\|u_i - u_j\|]_{1\leq i,j\leq n}$ is a squared Euclidean distance matrix. Therefore there must exist a set of locations $v_1, \dots, v_n \in \mathbb{R}^t$ such that $\|u_i - u_j\| = \|v_i - v_j\|^2$ for all i, j. □

Corollary 8. *The value of t from the previous corollary is either $n-1$ or $n-2$.*

Proof. From Theorem 4 of Schoenberg (1937), the matrix $D^{(2\alpha)}$ from Theorem 6 is of full rank. Therefore, $D^{(1/2)} = [\|u_i - u_j\|]_{1\leq i,j\leq n} = \left[\|v_i - v_j\|^2\right]_{1\leq i,j\leq n}$ must also be of full rank. By Theorem 6 of Gower (1985), the rank of a squared Euclidean distance matrix is either $t+1$ or $t+2$, where t is the number of coordinates of the locations that construct it. Therefore, since the rank of $D^{(1/2)}$ is n, either $t = n-1$ or $t = n-2$. □

Note that for the remainder of this chapter, we do not need to know concrete values for $v_1, \dots, v_n$. We simply require an assertion that they exist. However, it is easy to generate these locations by conducting eigenvalue decomposition on the Gram matrix defined by (2.5), where $d_{ij}^2 = \|u_i - u_j\|$. We discuss this methodology in further detail in Chapter 3.3.1.

2.3 The Euclidean Diversity Cut Algorithm

The results of the previous section outline some crucial results regarding Euclidean distance matrices. We can use these results to prove the following interesting property, that ensures the objective function of (EMSDP) is concave on its feasible domain. Let $h : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the tangent plane of f defined as follows,

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

The following result shows the validity of tangent planes on feasible solutions.

Proposition 9. *For all $x, y \in K$, we have $f(x) \le h(x, y)$.*

Proof. Note that

$$\begin{aligned} f(x) - h(x,y) &= \frac{1}{2}\langle Dx, x\rangle - \frac{1}{2}\langle Dy, y\rangle - \langle Dy, x-y\rangle \\ &= \frac{1}{2}\langle D(x+y), x-y\rangle - \langle Dy, x-y\rangle \\ &= \frac{1}{2}\langle D(x-y), x-y\rangle, \end{aligned}$$

therefore it is sufficient to prove $\langle D(x-y), x-y\rangle \le 0$. By Corollary 7, D is also a squared Euclidean distance matrix and therefore by Theorem 3 it is also conditionally negative definite. Now, for all $x, y \in K$ have $\sum_{i=1}^{n}(x_i - y_i) = 0$ and therefore $\langle D(x-y), x-y\rangle \le 0$ as required. □

Concavity in $f(x)$, albeit only on feasible solutions, ensures that tangents planes of feasible solutions provide valid upper approximations of an optimal solution. Let $A \subset K$ be any subset of feasible solutions of the (EMSDP) and let $\Gamma_A \subset K \times \mathbb{R}$ be defined as

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

The cutting plane model of (EMSDP), denoted by (Θ_A), is then given as the following linear maximisation problem

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

which can be written out explicitly as

$$\begin{aligned} \max \quad & \theta \\ \text{s.t.} \quad & \theta \le h(x,y), \quad \forall y \in A, \\ & x \in K. \end{aligned} \tag{2.12}$$

Then, whenever $A \subset K$, Proposition 9 holds and hence the results of Proposition 1 hold analogously. In other words, (Θ_A) provides a valid upper bound of (EMSDP) and when $A = K$, the problems are equivalent.

Based on this result, we propose a cutting plane approach to solve the nonconcave quadratic problem (EMSDP) by solving the linear problem (Θ_K). However, 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 (2.12) 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 $\theta > \mathrm{LB}_k$. The Euclidean Diversity-Cut (EDC) Algorithm, outlined in Algorithm 1, successively generates candidate solutions and adds the cutting planes to the linear model (Θ_{A_k}) to eliminate non-optimal solutions until no candidate solution remains in the search space.

Algorithm 1: The Euclidean Diversity Cut (EDC) Algorithm.

```
function EuclideanDiversityCut $(f,p)$:
    Take $x^0 \in K$
    Set $A_0 \leftarrow \{x^0\}$, $LB_0 \leftarrow f(x_0)$, $k \leftarrow 1$
    while $\exists(x^k, \theta^k) \in \Gamma_{A_{k-1}}$ s.t. $\theta^k > LB_{k-1}$ do
        $LB_k \leftarrow \max\{LB_{k-1}, f(x^k)\}$
        $A_k \leftarrow A_{k-1} \cup \{x^k\}$
        $k \leftarrow k+1$
    return $LB_k$
```

The EDC Algorithm does not require solving the linear problem (Θ_{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., $\theta^k > \mathrm{LB}_{k-1}$. 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 (Θ_{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.

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

Theorem 10. *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 $0 \leq k_1 < k_2$. Let $(x^{k_2}, \theta^{k_2}) \in \Gamma_{A_{k_2-1}}$. Then $\theta^{k_2} > \mathrm{LB}_{k_2-1}$ and,

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

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

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

where the last equality follows from Propositions 1 and 9. Taking the maximum over all $(\tilde{x}, \tilde{\theta}) \in \Gamma_{A_{k-1}}$ in the first inequality, we obtain

$$\max_{(x,\theta)\in\Gamma_{A_{k-1}}} \theta \leq \mathrm{LB}_{k-1} \leq \max_{(x,\theta)\in\Gamma_K} \theta. \tag{2.13}$$

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

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

From (2.13), 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 $l \in \{0, 1, \dots, k-1\}$, and hence x^l is optimal for (EMSDP). □

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

Proposition 11. *Let $x^k \in K$ ($k \geq 0$) 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 $h(x, x^k) \geq \theta$ becomes

$$f(x^k) \geq \theta. \tag{2.14}$$

Hence, $f(x^k) \geq \max_{(x,\theta)\in\Gamma_{A_{k+1}}} \theta \geq \max_{(x,\theta)\in\Gamma_K} \theta = \max_{x\in K} f(x)$, and the candidate x^k is an optimal solution of (EMSDP). The constraint (2.14) implies that there will be no candidate solution found in iteration $k+1$, and hence the EDC Algorithm must terminate. □

Lemma 12. *Let x^k be the iterate generated by the EDC Algorithm during iteration k. Then, for any subsequent iteration $l > k$, it holds that*

$$\frac{LB_{l-1} - f(x^k)}{\left\|\nabla f(x^k)\right\|} < \left\|x^l - x^k\right\|.$$

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

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

and hence we have that

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

As $k < l$, the algorithm did not terminate at k and hence from Proposition 11, $\left\|\nabla f(x^k)\right\| \neq 0$. 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 13. *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*

$$\sqrt{2N_l} \leq \frac{LB_{l-1} - f(x^k)}{\|\nabla f(x^k)\|}. \tag{2.15}$$

Then at step l onwards, the cutting plane $\theta \leq h(x, x^k)$ 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 $a \geq b$.

Proof. Let k and l be iterations of the EDC Algorithm such that $k < l$, and suppose (2.15) holds for some non-negative integer N_l. It follows from Lemma 12 that at iteration l, the cutting plane $\theta \leq h(x, x^k)$ removes all points $x \in K$ such that

$$\left\|x - x^k\right\| \leq \sqrt{2N_l}. \tag{2.16}$$

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 $q \in \{0, 1, \dots, N_l\}$, consider all the solutions $x \in K$ such that
$$q = |I_1 \cap I_2(x)|\,. \tag{2.17}$$
Altogether there are $\binom{p}{q}\binom{n-p}{q}$ feasible solutions $x \in K$ that satisfy (2.17). Therefore in total, there are precisely $\sum_{q=0}^{N_l} \binom{p}{q}\binom{n-p}{q}$ feasible points in K that satisfy (2.16). This proves the assertion. □

Theorem 13 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 can be chosen larger. We now explore the general strength of cuts of type (2.12) 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
$$f_\rho(x) = \frac{1}{2}\left(\langle (D - \rho I_n)x, x\rangle + \rho \sum_{i=1}^{n} x_i \right)$$
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., $D - \rho I_n \preceq 0$, 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 14. *Let $h_\rho(x, y)$ denote the tangent plane of $f_\rho(x)$. 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
$$\begin{aligned} h_{\rho_1}(x, y) &= f_{\rho_1}(y) + \left\langle Dy - \rho_1 y + \tfrac{1}{2}\rho_1 e, x - y \right\rangle, \\ &= f_{\rho_1}(y) + \langle Dy, x - y\rangle + \tfrac{1}{2}\rho_1 \langle e - 2y, x - y\rangle\,. \end{aligned}$$

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,

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

as required. □

Let $\lambda_{\max}$ denote the largest eigenvalue of D. It is proved in Hammer and Rubin (1970) that when $\rho \geq \lambda_{\max}$, $D - \rho I_n$ 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 14, 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.

2.4 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 2.3GHz 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`[2] (Martí et al., 2021) has commonly been used as a benchmark for the maximum

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

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[3].

The algorithm was compared against three exact solution methods. The first method solves the Glover reformulation (2.4) 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 2.1 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. 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-second 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-second time

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

				Average Gap (%)				Number Best				
Time Limit (sec)	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

Table 2.1: 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.

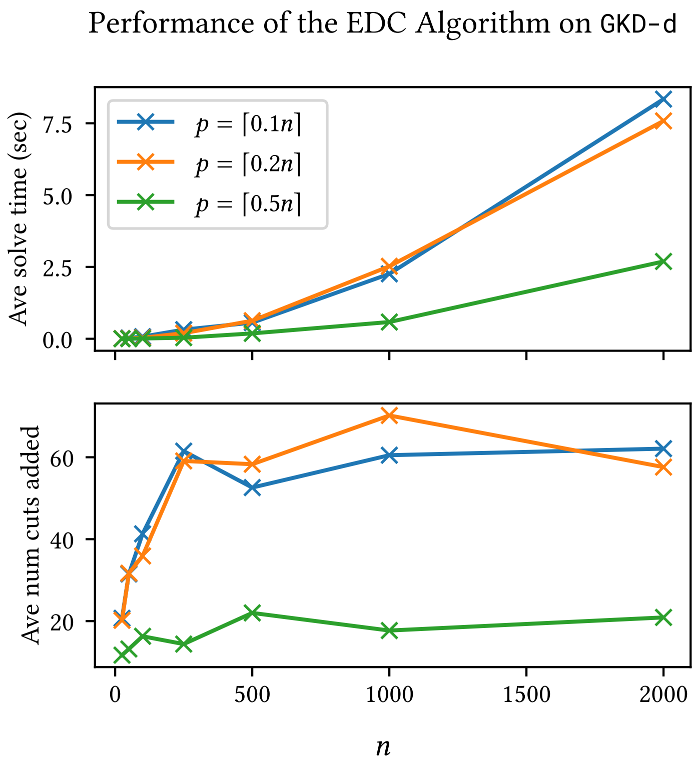

Figure 2.1: Performance of the EDC Algorithm on test set `GKD-d`. For each value n and p, there are 10 problems to solve. We report the average solve time and the average number of cuts added for each pair n and p.

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 2.1 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 13. 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 2.1 and

$\frac{p}{n}$	Number Optimal	Average Solve Time (sec)	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

Table 2.2: 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.

Figure 2.1 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 2.2 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 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 Figure 2.1. 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 2.2 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

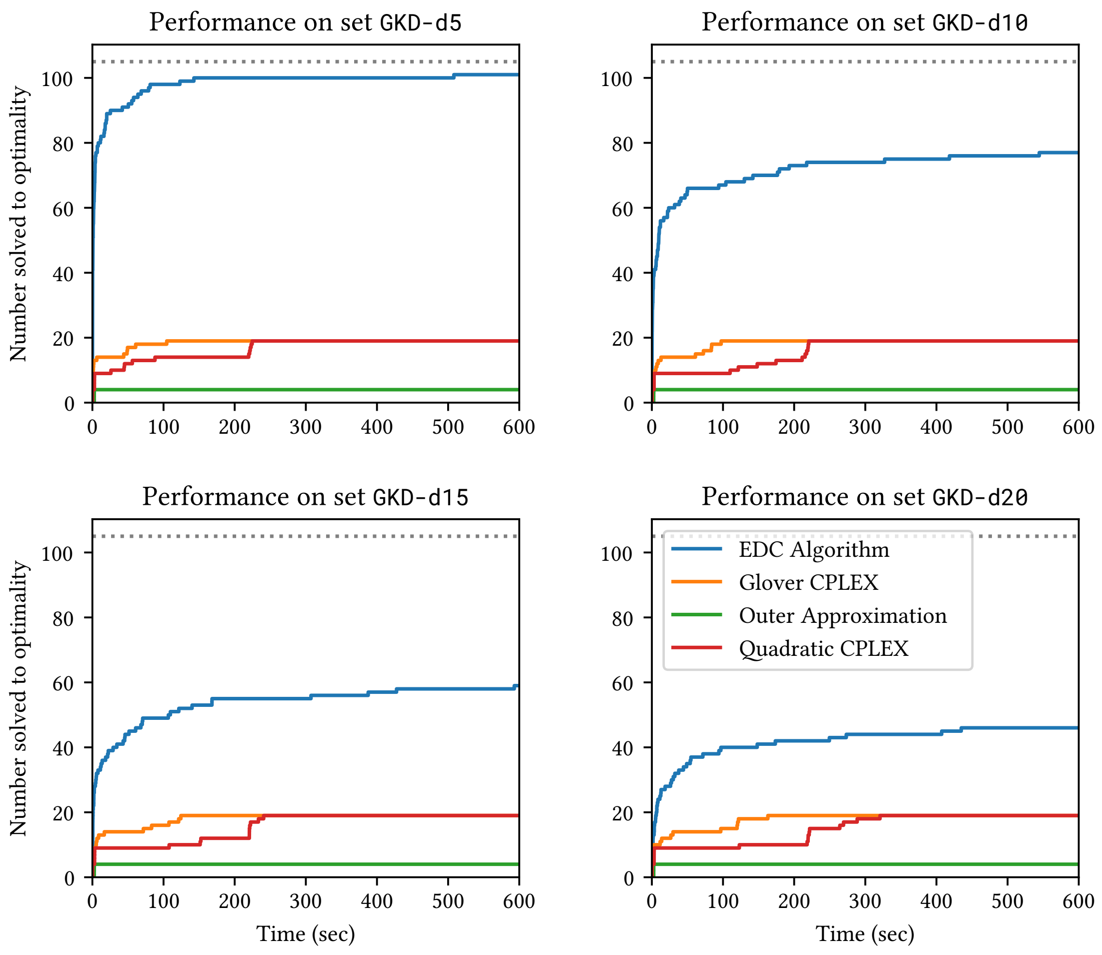

Figure 2.2: Performance profile on test sets GKD-d5, GKD-d10, GKD-d15 and GKD-d20. For each test set, there are a total of 105 test instances. The number solved to optimality is shown for each of the exact solvers used.

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-second 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 2.3 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,

Set	$\frac{p}{n}$	Number Optimal	Average Solve Time (sec)	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

Table 2.3: 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.

across all ratios, the number of cuts required to prove optimality is significantly higher than the results seen on set `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 $n = 85900$) 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 $p = \lceil 0.1n \rceil, \lceil 0.2n \rceil, \lceil 0.5n \rceil$,

Instance	n	$p = \lceil 0.1n \rceil$ Solve Time (sec)	Cuts	$p = \lceil 0.2n \rceil$ Solve Time (sec)	Cuts	$p = \lceil 0.5n \rceil$ Solve Time (sec)	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	11849	208.72	87	363.62	89	389.30	37
usa13509.tsp	13509	675.39	197	324.01	64	339.21	26
brd14051.tsp	14051	507.15	148	676.59	116	506.14	36
d15112.tsp	15112	676.25	171	776.97	116	842.26	49
d18512.tsp	18512	799.56	129	775.45	73	1379.92	56
pla33810.tsp	33810	2053.07	113	3320.10	110	3519.85	47
pla85900.tsp	85900	18291.56	151	16374.19	74	19986.27	38

Table 2.4: 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.

comprising three tests for each set of locations. Each problem is then solved to proven optimality using the EDC Algorithm. In Table 2.4, 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 2.4 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 frame. Even for the very large problems of $n = 85900$, 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.

2.5 Conclusion

This chapter 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. This trend was not observed on other exact methods, however. In the following chapter, we examine in detail why high coordinate instances are difficult for the cutting plane method, and propose an effective technique to improve performance on these difficult instances.

3 Coordinate Partitioning for Difficult Diversity Problems[1]

The Euclidean max-sum diversity problem becomes substantially more difficult as the number of coordinates increases, despite the number of decision variables not changing. In this chapter, we overcome this complexity by constructing a new set of locations whose squared Euclidean distances equal that of the original. Using squared distances allows the objective function to be decomposed into the sum of pairwise distances within each coordinate. We use a partition set of the coordinates to enable a functional decomposition of the objective. Each functional component is expected to be simpler than the original, and therefore easier to approximate via cutting plane methods. We prove the global convergence of our new approach and introduce several partitioning strategies. Furthermore, we show how a principal component analysis of coordinate influence can be conducted with minimal extra computation, the results of which can be used to guide the partitioning process. Extensive numerical results prove the efficiency of the partitioned cutting plane method, with the algorithm able to solve large, 20-coordinate problems of up to 1000 locations. Finally, we introduce a new class of challenging diversity problems, characterised by locations situated on the edge of a ball.

3.1 Introduction

In Chapter 2, we discussed applications of (EMSDP) concerning mainly location-based applications. However, since the problems conception, many researchers have relaxed the notion of distance to more general settings. One example is in genetics, where breeders attempt to maximise the diversity of traits among a breeding stock (Porter et al., 1975). In these slightly more abstract applications, Euclidean distance has often been used as a measure of dissimilarity between data points (Shirkhorshidi et al., 2015). For example, by using Euclidean dissimilarity, the (EMSDP) can help in forming skilfully and socially diverse teams (Hochbaum et al., 2023), which have become highly desirable in many workplace settings (Roberge & van Dick, 2010). Furthermore, Euclidean dissimilarity is especially common in clustering applications (Charikar et al., 2019; Lloyd, 1982), such as in the Maximally Diverse Grouping Problem (Feo et al., 1992; O'Brien & Mingers, 1997).

[1]This chapter is based on Spiers, Bui, and Loxton (2023a)

This problem is an extension of the (EMSDP) which considers the clustering of nodes into groups of given sizes, such that the sum of intra-group Euclidean dissimilarity is maximised.

Practical applications of the (EMSDP) for facility location problems typically involve a small number of coordinates, given that real-world scenarios often focus on locations in either $\mathbb{R}^2$ or $\mathbb{R}^3$. However, when we instead use Euclidean distance as a measure of dissimilarity, as in the Maximally Diverse Grouping Problem, the number of coordinates is equal to the number of features in the data. Although the size of the (EMSDP) remains unaffected by an increasing number of coordinates, our work in the previous chapter demonstrates that, surprisingly, this has a significant impact on problem complexity. Consequently, problems with large feature sets have the potential to generate challenging instances of the (EMSDP).

Gaining insight into the main factors that contribute to instance difficulty can provide invaluable insights into problem structure. This understanding can subsequently guide the improvement and development of new solution methods. As highlighted by the 'no-free-lunch' theorem (Wolpert & Macready, 1997), universal algorithm superiority across all problem classes or instances is unattainable. The inability of an algorithm to adequately solve a particular class of problem has led to the development of many novel solution approaches. It is common to find several potential solution algorithms for the same problem, each targeting a specific complexity (K. Smith-Miles & Lopes, 2012). Moreover, instance analysis can help in identifying which algorithm is best suited to different problem types (K. A. Smith-Miles, 2009).

In this chapter, we conduct a comprehensive analysis into the complexity of high coordinate instances of the (EMSDP), the results of which motivate our new approach. Cutting plane and outer approximation algorithms are most effective when their tangent planes provide a tight approximation of the objective function. However, as seen in the previous chapter, when the number of coordinates increases the tangent planes become weak. As cuts weaken, they can no longer effectively direct the search for the optimal solution, leading to an overall slowdown in convergence. By better understanding the problem structure, we can attempt to overcome this difficulty and improve cut tightness.

In certain scenarios, improving the approximation of the objective can be achieved through functional decomposition. This process involves breaking down the objective into its essential functional components and applying outer approximation to each component separately. Consider a concave objective function defined by $f(x) = \sum_{i=1}^{m} f_i(x)$. It is well known that tighter approximations arise when outer approximation is applied to each $f_i(x)$, rather than solely $f(x)$ (Kronqvist et al., 2018). Nevertheless, achieving a functional decomposition such as this is rarely straightforward. In Tawarmalani and Sahinidis (2005), the authors show how composition functions such as $h = g \circ f(x)$ can be decomposed into $g(u)$ and $f(x)$. By applying outer approximation to $g(u)$ and $f(x)$ separately, the authors achieved a tighter approximation of $h(x)$ compared to applying outer approximation directly to $h(x)$. Despite possibly introducing extra decision variables and competing objectives, functional decomposition can be very effective when the improvement in objective approximation outweighs these added complexities.

To overcome the difficulty of high-coordinate instances, we present a method for partitioning a Euclidean distance matrix into its distinct coordinate spaces, thereby enabling a functional decomposition of the objective function. This process involves generating a new set of points whose squared pairwise distances are equal to the corresponding Euclidean distances. The coordinate space of these new points is then partitioned, allowing the objective to be separated such that $f(x) = \sum_{i=1}^{m} f_i(x)$. Each Euclidean objective function now contains fewer coordinates, and as such is expected to be easier to approximate.

The remainder of this chapter is structured as follows. In Section 3.2, we explore in detail the difficulty of high coordinate instances by quantifying the relative strength of tangent planes. This insight leads to a new class of challenging problems, where locations are situated on the edge of a ball. Then, in Section 3.3, we formulate a functional decomposition of the (EMSDP) by partitioning the distance matrix into its distinct coordinate subspaces. We prove the global convergence of this new algorithm, and outline several heuristic partitioning strategies. Finally, Section 3.4 presents the results from comprehensive numerical experiments used to evaluate the performance of the newly proposed algorithm and partition strategies.

3.2 Geometric Insights into Cutting Planes

To help explain why the cuts weaken as the number of coordinates increases, we now present a geometric interpretation of (2.12) that identifies regions of excluded solutions, referred to as *exclusion zones*. We further show how the density of these regions, and therefore the number of solutions removed, decreases exponentially as the number of coordinates increases. This new insight offers a deeper understanding of the relative difficulty of (EMSDP) instances, allowing us to introduce a new class of challenging problems.

Let us begin by quantifying the relative strength of different cuts. Given a feasible solution $y \in K$ and an appropriate lower bound $LB \geq f(y)$, let

$$R(y, LB) = \{x \in K \ : \ h(x, y) \leq LB\}$$

denote the set of solutions removed by the tangent plane of y. These solutions cannot improve upon the current lower bound and therefore never satisfy line 4 of Algorithm 1. While determining the size of $R(y, LB)$ can provide a valuable measure of cut strength, this is generally considered an intractable and highly combinatorial problem for even moderately sized instances. However, significant intuition can be gained by considering solutions with a strictly decreasing gradient, i.e., those where $\langle \nabla f(y), x - y \rangle \leq 0$, as these are known to be included in $R(y, LB)$. Let $I(x) = \{i : x_i = 1\}$ denote the set of locations included in the solution of x, and recall that

$$\frac{\partial f(y)}{\partial y_i} = \sum_{j \in I(y)} D_{ij} = \sum_{j \in I(y)} \|u_i - u_j\| .$$

Hence, $\langle \nabla f(y), x - y \rangle \leq 0$ is equivalent to

$$\begin{aligned} & \sum_{i=1}^{n} x_i \frac{\partial f(y)}{\partial y_i} \leq \sum_{i=1}^{n} y_i \frac{\partial f(y)}{\partial y_i} \\ \iff \quad & \sum_{i \in I(x)} \sum_{j \in I(y)} \|u_i - u_j\| \leq \sum_{i \in I(y)} \sum_{j \in I(y)} \|u_i - u_j\| . \end{aligned} \tag{3.1}$$

In other words, the sum of the distances from the locations in y to those in x must be less than or equal to the total sum of the distances between the locations in y.

With this intuition in mind, how can we easily construct a solution that satisfies (3.1)? One approach is to consider solutions generated by single location swaps, i.e., $x = y + e_i - e_j$, where $i \notin I(y)$ and $j \in I(y)$ and where e_i is a vector with 1 in the ith component and 0 elsewhere. This simplifies (3.1) to:

$$\sum_{k \in I(y)} \|u_i - u_k\| \leq \sum_{k \in I(y)} \|u_j - u_k\| . \tag{3.2}$$

In other words, if the sum of the distances to the incoming location is less than or equal to the sum of the distances to the outgoing location, then $x \in R(y, LB)$. These sums can be visualized by considering locations in $\mathbb{R}^2$. Figure 3.1 shows in blue the five locations that make up the solution of y. In each subplot, we consider swapping the location depicted by the blue square with that depicted by the yellow star. The red lines represent the sum of distances to the outgoing location (the right-hand side of (3.2)), while the green lines represent the sum of distances to the incoming location (the left-hand side of (3.2)).

Two observations become immediately apparent from this visualisation. Firstly, since the left hand side of (3.2) contains the additional nonzero distance from the outgoing to incoming locations, swaps are more likely to satisfy (3.2) when these locations are close together. This can be seen in Figure 3.1, where there are five incoming distances to consider, yet only four outgoing. Secondly, swapping to a location closer to the geometric centre of all locations is more likely to result in a smaller sum distance, thereby increasing the likelihood of satisfying (3.2). Considering the two subplots, the swap on the left is more likely to satisfy (3.2) compared with that on the right, as the incoming location is closer to the geometric centre of the locations.

Guided by the interpretation provided in (3.2), let

$$O(w) = \left\{ o \in \mathbb{R}^s : \sum_{k \in I(y)} \|o - u_k\| \leq \sum_{k \in I(y)} \|w - u_k\| \right\} \tag{3.3}$$

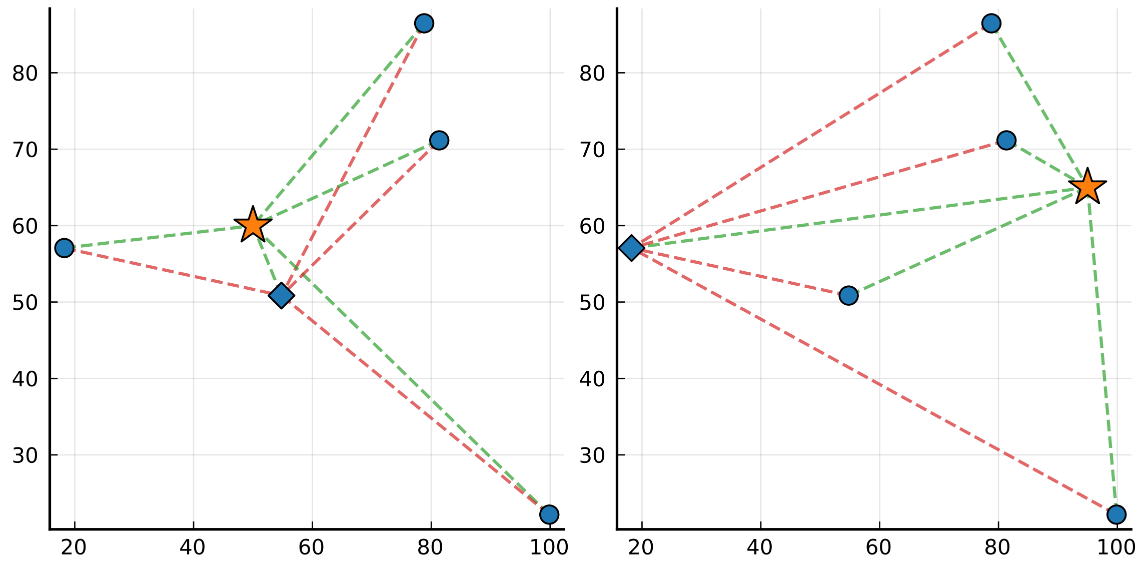

Figure 3.1: Geometric representation of (3.2). The five locations that form the solution y are shown in blue. The red lines indicate the sum of distances to the outgoing location, depicted by the blue square, while the green lines represent the sum of distances to the incoming location, depicted by the yellow star. These correspond to the right-hand and left-hand sides of (3.2), respectively.

define a region in $\mathbb{R}^n$. If $o \in O(w)$, then the sum of the distances from the points in $I(y)$ to o is less than or equal to the sum of the distances to w. Therefore, if we then swap a location u_j with $u_i \in O(u_j)$, we get that (3.2) holds and hence $\langle \nabla f(y), x - y\rangle = \left\langle \nabla f(y), e_i - e_j \right\rangle \leq 0$. Note that the boundary of $O(w)$ forms an n-ellipse, with the foci being the locations in $I(y)$, and w lying on its perimeter. In the following proposition, we generalise this result to consider up to $m \leq p$ potential swaps.

Proposition 15. *Let $x, y \in K$ be such that $I(x) = \{i_1, \dots, i_p\}$ and similarly $I(y) = \{j_1, \dots, j_p\}$. If $u_{i_k} \in O(u_{j_k})$ for $k = 1, \dots, p$, then $x \in R(y, LB)$ where $LB \geq f(y)$.*

Proof. If $u_{i_k} \in O(u_{j_k})$ holds for $k = 1, \dots, p$, then (3.2) also holds for each pair of u_i and u_j. Aggregating over $k = 1, \dots, p$ we see that (3.1) also holds and hence $\langle \nabla f(y), x - y\rangle \leq 0$. Therefore, $x \in R(y, LB)$. □

We can think of $O(u_{j_1}), \dots, O(u_{j_p})$ as representing the exclusion zones of y, as they identify regions of nonoptimal solutions that have been removed from the search space. These regions can be visualised by considering locations in $\mathbb{R}^2$. Each row of subplots in Figure 3.2 shows a different solution for the same 50 locations in $\mathbb{R}^2$, where $p = 5$. For each solution shown on the left, we plot the boundary of the regions defined by $O(u_{j_1}), \dots, O(u_{j_5})$ on the right. By Proposition 15, any solution x created by choosing a unique location in each $O(u_{j_1}), \dots, O(u_{j_5})$ is in $R(y, LB)$, and therefore never satisfies line 4 of Algorithm 1. These are the solutions that are known to be removed by the cut of y. Note that the last row Figure 3.2 shows the exclusion zones of the optimal solution for the (EMSDP) for these locations.

From Figure 3.2 it is clear to see that the cuts are more effective at removing non-optimal locations in the relative interior. This backs up the intuition from Figure 3.1 and demonstrates the strong functional approximation that can be achieved for points closer to the geometric centre of selected locations. Consequently, a higher density of interior points facilitates the removal of nonoptimal locations more easily, as we get an improved outer approximation for a larger proportion of search space.

However, as the number of coordinates grows, the volume of the entire space expands exponentially, leading to an exponential decrease in density. As the density of interior locations decreases, our functional approximation weakens. Consequently, the algorithm struggles to prove whether there exists an additional x satisfying line 4 of Algorithm 1, leading to an overall slowdown in algorithm convergence.

Furthermore, we can see in Figure 3.3 the affect a large p/n ratio has on exclusions zones. Observe how the optimal solution for this set of locations contains almost all remaining points in the smallest exclusion zone, leading to a much stronger outer approximation. This backs up the observations in from the previous chapter, where the cutting plane algorithm was shown to perform strongly on (EMSDP) instances with large p/n ratios.

To confirm our understanding regarding exclusion zones and interior density, we now conduct a small simulation study. Using the following proposition, we can count the number of solutions that satisfy Proposition 15, thereby providing a lower bound for the size of $R(y, LB)$.

Proposition 16. *Let y be such that $\frac{\partial f(y)}{\partial y_{j_1}} \leq \dots \leq \frac{\partial f(y)}{\partial y_{j_p}}$ and let $H(j) = \{i : u_i \in O(u_j)\}$ give the set of locations in the exclusion zone of u_j. Finally, let*

$$h(k) = \begin{cases} H(j_1), & \text{if } k = 1, \\ H(j_k) \setminus H(j_{k-1}), & \text{otherwise}. \end{cases}$$

The number of solutions that satisfy Proposition 15 for a given y is given by the recursive sum

$$L := \sum_{w_1=1}^{\min\{|h(1)|,p\}} \sum_{w_2=\max\{2-w_1,0\}}^{\min\{|h(2)|,p-w_1\}} \cdots \sum_{w_p=\max\{p-\sum_{l=1}^{p-1} w_l,0\}}^{\min\{|h(p)|,p-\sum_{k=1}^{p-1} w_l\}} \prod_{k=1}^{p} \binom{|h(k)|}{w_k} \tag{3.4}$$

where $LB \geq f(y)$. Furthermore, $|R(y, LB)| \geq L$.

Proof. The proof is given in Appendix A.1. □

Applying this counting technique to the solutions depicted in Figure 3.2, we find that y_1 excludes at least 0.1% of the total search space, y_2 excludes at least 7.1% and the optimal

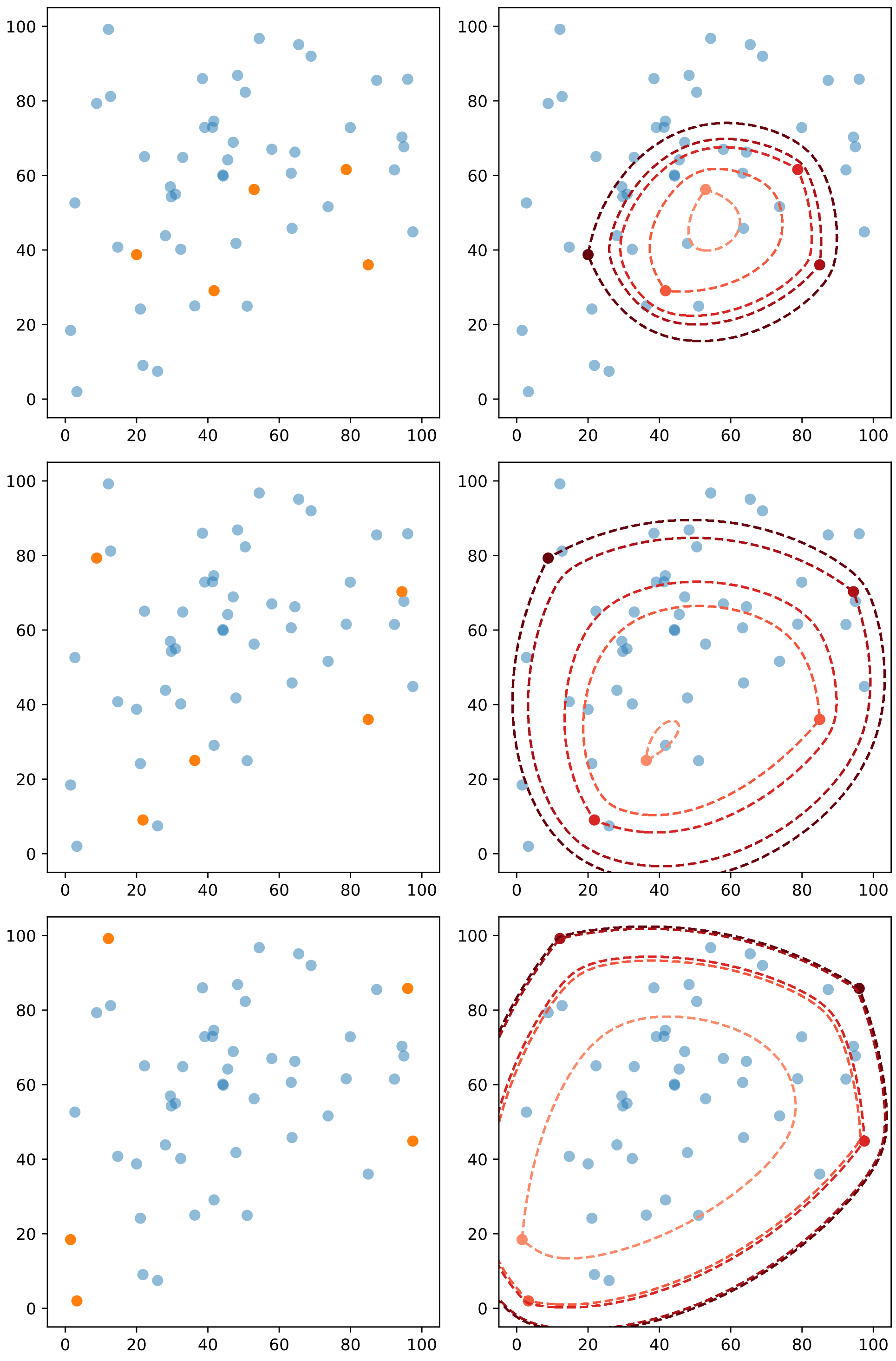

Figure 3.2: Example exclusions zones of solutions y_1, y_2, and y_3 on a set of 50 random locations in $\mathbb{R}^2$ where $p = 5$. Note that $f(y_1) < f(y_2) < f(y_3)$ and y_3 is the optimal solution for the (EMSDP) for these locations.

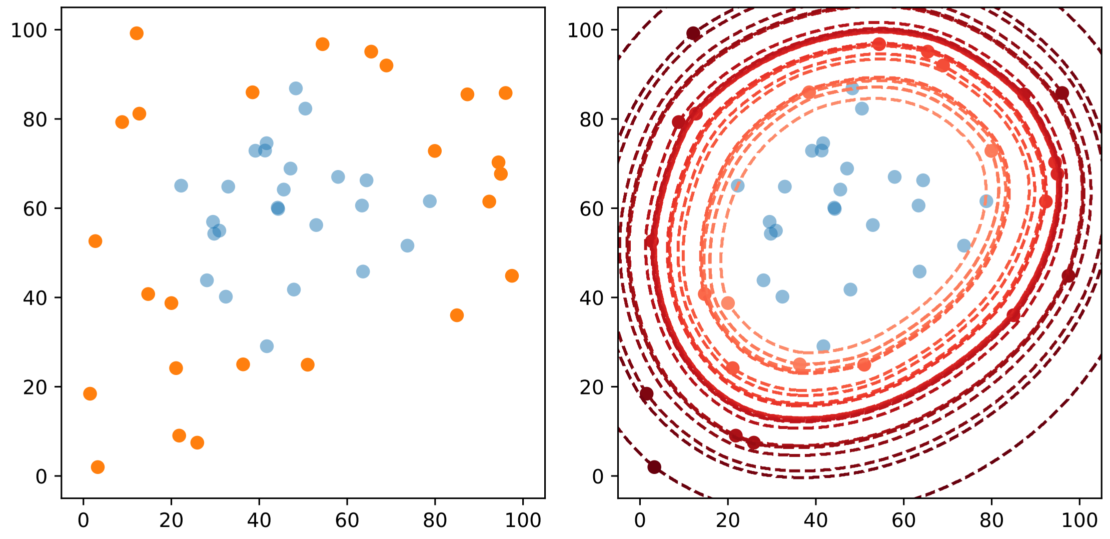

Figure 3.3: Example exclusions zones for the optimal solution of a set of 50 locations in $\mathbb{R}^2$, where $p = 25$.

solution y_3 removes more than 96.6% of the search space. For the large p/n solution shown in Figure 3.3, this solution alone is able to remove more than 99.9% of the search space.

We can relate the number of solutions excluded by a cut to the number of coordinates by calculating (3.4) for the optimal solutions of randomly generated instances of the (EMSDP). Following the approach of the previous chapter, we generated random instances with $n = 30$, $p = 3, 6, 15$, and $s = 2, 5, 10, 15, 20$. For every combination of n, p, and s, we generated and solved a total of 50 problems to optimality. We then determined the minimum proportion of search space removed by the optimal solution, based on the lower bound provided by (3.4). The results of this simulation are presented in Figure 3.4, and reflect the findings from the previous chapter. As the number of coordinates increases, the quality of outer approximation decreases exponentially, making it difficult to remove a substantial proportion of the search space. Furthermore, increasing p/n leads to a much stronger outer approximation. Most notably, for cases where $p/n = 0.5$ and $s = 2$, the optimal solution alone is able to remove almost all the search space.

This insight sheds new light on the factors that make certain instances of (EMSDP) challenging. For instance, consider a set of locations that are all positioned on the edge of a ball. As previously observed, exclusion zones are generally more effective at capturing points in their relative interior. Consequently, if points sit on a circle, their exclusion zones are unlikely to be effective. An example of this is shown in Figure 3.5. Clearly, the exclusion zones encompass significantly fewer locations due to the minimal interior density. Furthermore, when considering Figure 3.5.a, the strongest functional

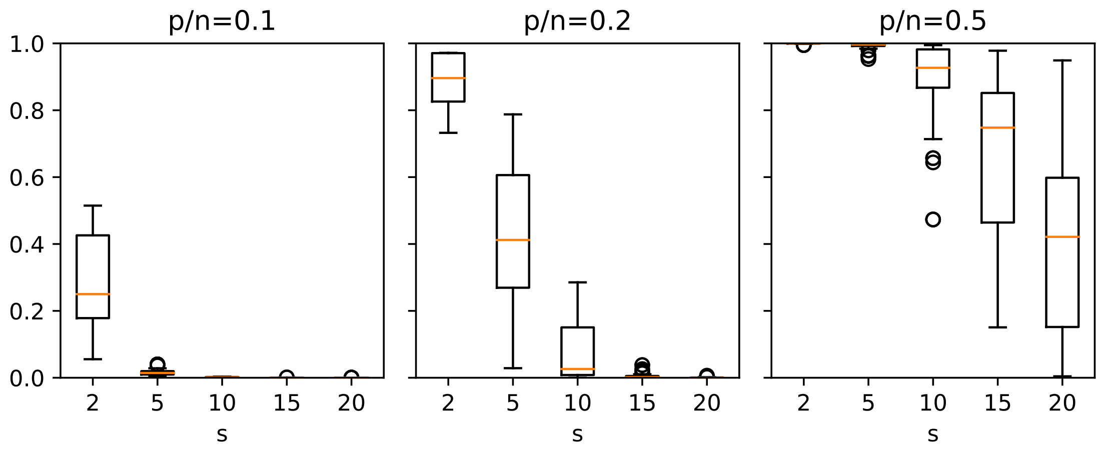

Figure 3.4: Minimum proportion of solution space removed by the optimal solution of the (EMSDP) for different values of p and s.

approximation is clustered around the lower-left corner, near to the geometric median of the selected locations. However, this covers only a small proportion of the total perimeter, meaning there remains many x that satisfy line 3 of Algorithm 1. This trend appears to get worse with an improved solution quality. The exclusion zones of the solution shown in Figure 3.5.b, which has a far better objective value than that in 3.5.a, contain none of the other locations. If we consider this cut on its own, then any $x \neq y$ satisfies line 4 of Algorithm 1 and hence $R(y, f(y)) = \{y\}$. This suggests that the tangent planes of near optimal solutions may struggle to remove nonoptimal solutions due to the low interior density, and existence of many nondominanted, near optimal solutions. We further explore the complexities of the ball instances in Chapter 3.4, demonstrating that they indeed constitute a particularly challenging class of problem.

3.3 Methodology

We now present our new methodology for solving difficult instances of the (EMSDP). The principle behind this method is to project high-coordinate problems down into smaller coordinate subspaces. In doing so, we expect the cuts to be stronger and more effective within their corresponding lower coordinate space. Figure 3.6 illustrates an example of this idea, where an instance in $\mathbb{R}^2$ is projected down into two separate instances in $\mathbb{R}$. The exclusion zones of the same solution in the lower coordinate space appear to be stronger, and solving these individual problems is expected to be easier.

We begin by outlining how to construct points $v_1, \dots, v_n$ such that $\|u_i - u_j\| = \|v_i - v_j\|^2$. As these new locations use squared Euclidean distances, the objective function can be

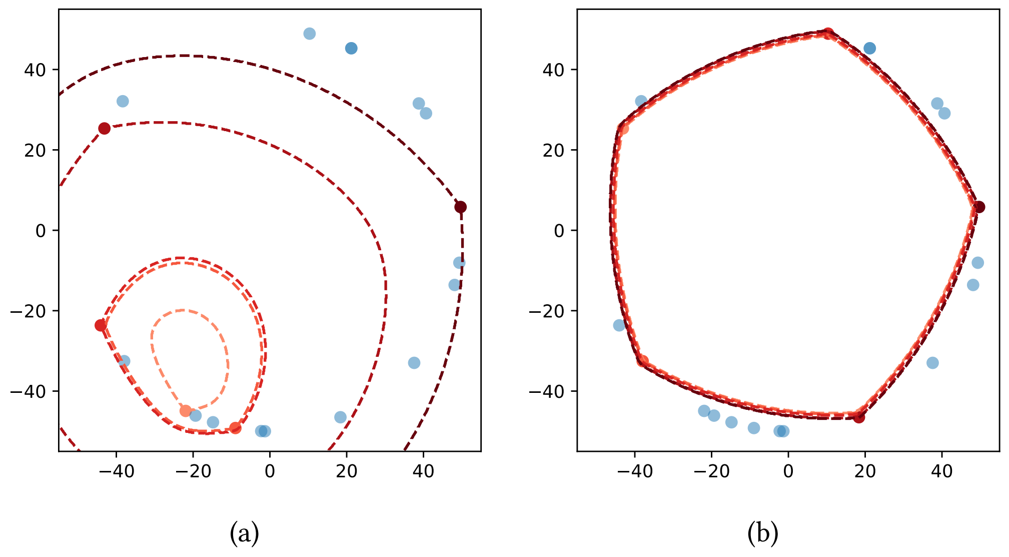

Figure 3.5: Exclusion zones of a solution where all locations lie on the edge of a ball.

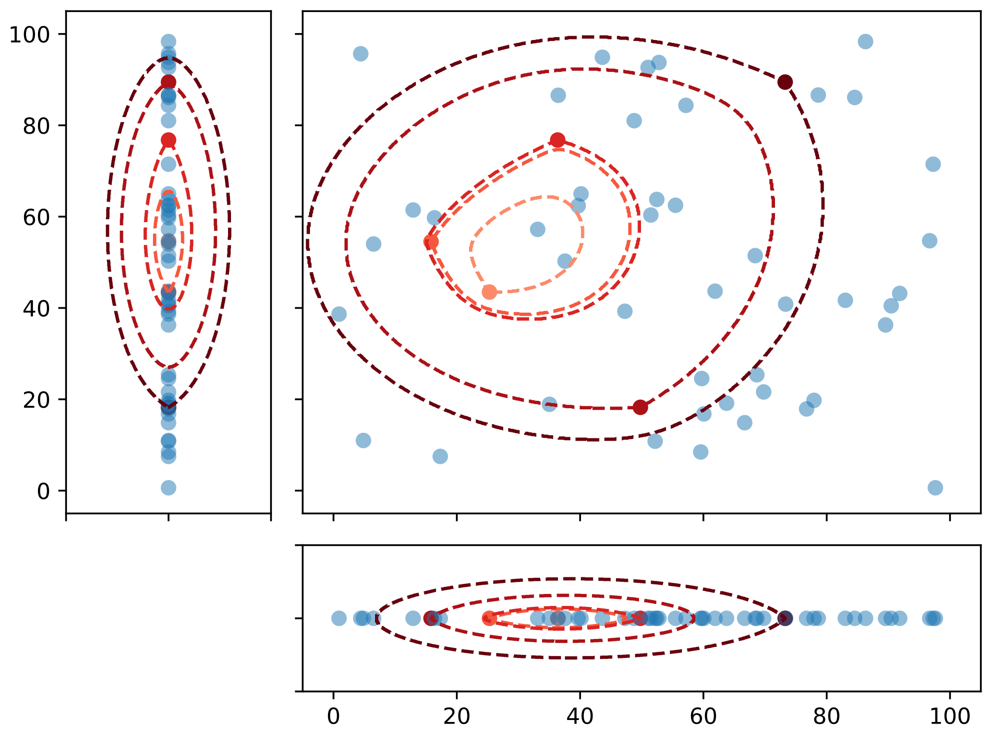

Figure 3.6: Exclusion zones of a solution in $\mathbb{R}^2$, as well as the zones generated by projecting the solution down into two subspaces in $\mathbb{R}$.

broken up into partitions of coordinates. These partitions contain fewer coordinates and are therefore expected to be easier to solve. We then show how principal component analysis can assist in generating effective coordinate partitions, and outline three partitioning strategies.

3.3.1 Location Recovery

Theorem 6 establishes the existence of a set of points $v_1, \dots, v_n \in \mathbb{R}^t$ such that $\|u_i - u_j\| = \|v_i - v_j\|^2$ for all i, j. While these locations are not originally known, they can be reconstructed by using the pairwise distances between u points. Firstly, recall from the law of cosines that

$$\|u_i - u_j\| = \|v_i - v_j\|^2 = \|v_i\|^2 + \|v_j\|^2 - 2\left\langle v_i, v_j \right\rangle. \tag{3.5}$$

Let us assume that v_1 lies at the origin. This assumption is valid since pairwise distances are rotation and translation invariant. The first column of the distance matrix Q then gives

$$d_{i1}^2 = \|v_i - v_1\|^2 = \|v_i - \mathbf{0}\|^2 = \|v_i\|^2.$$

Hence we can rewrite (3.5) as,

$$\left\langle v_i, v_j \right\rangle = \frac{1}{2}\left(\|v_i\|^2 + \|v_j\|^2 - \|u_i - u_j\|\right) = \frac{1}{2}\left(d_{i1}^2 + d_{j1}^2 - d_{ij}^2\right). \tag{3.6}$$

This result allows us to reconstruct the Gram matrix of vectors $v_1, \dots, v_n$. Let $G = [g_{ij}]_{1 \le i,j \le n}$ be an $n \times n$ matrix where

$$g_{ij} = \frac{1}{2}\left(d_{i1}^2 + d_{j1}^2 - d_{ij}^2\right). \tag{3.7}$$

Then from (3.6),

$$G = \left[\left\langle v_i, v_j \right\rangle\right]_{1 \le i,j \le n} = V^T V$$

where V is an $t \times n$ matrix with locations $v_1, \dots, v_n$ as its columns, i.e., $V = [v_1, \dots, v_n]$. The existence of the real matrix V has already been established in Theorem 2, and therefore, we have that G is always a positive semidefinite (PSD) matrix.

Using eigenvalue decomposition, we can factorise G as

$$G = W \Lambda W^T$$

where W is an orthonormal matrix whose columns are the eigenvectors of G, and Λ is a diagonal matrix whose entries are the eigenvalues of G. Being PSD, the eigenvalues of G are nonnegative, and hence we can recover points $v_1, \dots, v_n$ by finding

$$V = W\sqrt{\Lambda}.$$

Note that the number of coordinates of v points, denoted by t, is equal to the number of nonzero eigenvalues of G.

3.3.2 Coordinate Partitioning

After recovering points $v_1, \dots, v_n$, we can express the original objective function as,

$$f(x) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \sum_{k=1}^{t} \left(v_{ik} - v_{jk}\right)^2 \tag{3.8}$$

where v_{ik} gives the kth coordinate of location v_i. Let $T = \{T_1, \dots, T_m\}$ denote a partition set containing m subsets of the coordinates $\{1, \dots, t\}$. As T is a partition set, we have that

1. $T_r \subset \{1, \dots, t\}$ for $r = 1, \dots, m$,
2. $\bigcup_{r=1}^{m} T_r = \{1, \dots, t\}$, and
3. $T_\alpha \cap T_\beta = \emptyset$ for all $\alpha \neq \beta$.

Using this partition set, (3.8) can be written as

$$\begin{aligned} f(x) &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \sum_{r=1}^{m} \sum_{k \in T_r} \left(v_{ik} - v_{jk}\right)^2 \\ &= \sum_{r=1}^{m} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \sum_{k \in T_r} \left(v_{ik} - v_{jk}\right)^2 \\ &= \sum_{r=1}^{m} f_r(x), \end{aligned}$$

where $f_r(x)$ is the sum of the pairwise squared Euclidean distances of the T_r subset of coordinates of points $v_1, \dots, v_n$.

Crucially, each $f_r(x)$ represents the sum of pairwise squared Euclidean distances between exactly p points. Therefore, by the same logic as Proposition 9, every $f_r(x)$ is a concave function on $x \in K$, and hence

$$f_r(x) \leq f_r(y) + \langle \nabla f_r(y), x - y \rangle$$

holds for all $x, y \in K$ and $r = 1, \dots, m$. Then, given a subset of feasible solutions $A \subset K$ and valid partition set T, the coordinate partitioned cutting plane model of the (EMSDP), denoted by (Θ_{TA}), is given as

$$\begin{aligned} \max \quad & \sum_{r=1}^{m} \theta_r, && \\ \text{s.t.} \quad & \theta_r \leq f_r(y) + \langle \nabla f_r(y), x - y \rangle, && \forall y \in A, r = 1, \dots, m, \\ & \theta_r \geq 0, && r = 1, \dots, m, \\ & x \in K. && \end{aligned} \tag{Θ_{TA}}$$

Algorithm 2: Partitioned Diversity-Cut method for solving (EMSDP).

```
function PartitionedDiversityCut $(T, f, p)$:
    Choose a feasible $x \in K$
    Set $A \leftarrow \{x\}$, $LB \leftarrow f(x)$
    while $\exists(x', \theta'_1, \dots, \theta'_m) \in \Gamma_{TA}$ s.t. $\sum_{r=1}^{m} \theta'_r > LB$ do
        $LB \leftarrow \max\{LB, \sum_{r=1}^{m} f_r(x')\}$
        $A \leftarrow A \cup \{x'\}$
    return $LB$
```

As with the standard cutting plane algorithm, the partitioned cutting plane formulation (Θ_{TA}) can be solved using a similar branch and cut framework. An outline of this approach is shown in Algorithm 2.

This method allows us to reduce the complexity of the problem by breaking it down into coordinate partitions. As each partition contains a smaller number of coordinates, their associated cuts are expected to be tighter. In fact, it is shown in the following proposition that for any possible partition set, the upper bound provided by (Θ_{TA}) is always at least as good as that provided by (Θ_A).

Proposition 17. *For any subset of feasible solutions $A \subset K$ and valid partition set T, we have that $\Theta_{TA} \leq \Theta_A$.*

Proof. Let $(x^*, \theta^*_1, \dots, \theta^*_m)$ be an optimal solution of (Θ_{TA}). Then it follows that

$$\theta^*_r = \min_{y \in A} \{f_r(y) + \langle \nabla f_r(y), x^* - y \rangle\}, \quad r = 1, \dots, m.$$

By aggregating this expression over $r = 1, \dots, m$, we have that

$$\begin{aligned} \Theta_{TA} = \sum_{r=1}^{m} \theta^*_r &= \sum_{r=1}^{m} \min_{y \in A} \{f_r(y) + \langle \nabla f_r(y), x^* - y \rangle\}, \\ &\leq \min_{y \in A} \left\{ \sum_{r=1}^{m} (f_r(y) + \langle \nabla f_r(y), x^* - y \rangle) \right\}, \\ &= \min_{y \in A} \{f(y) + \langle \nabla f(y), x^* - y \rangle\}. \end{aligned}$$

Now, as x^* is a feasible solution of (Θ_A), we also have that

$$\min_{y \in A} \{f(y) + \langle \nabla f(y), x^* - y \rangle\} \leq \Theta_A$$

and therefore $\Theta_{TA} \leq \Theta_A$ as required. $\square$

In many cases, the upper bound provided by the partitioned problem can be far tighter than that of the original. This is most effective when partitions contain few coordinates.

However, we saw in Corollary 8 that either $t = n - 2$ or $t = n - 1$. Therefore, to achieve low coordinate partitions, a large m is potentially required. This in turn increases the size of the partitioned problem (Θ_{TA}). It is therefore crucial to partition the set of coordinates strategically.

3.3.3 Principal Component Analysis

Understanding the relative importance of each coordinate can then help in formulating strategic partition sets. This understanding can be garnered by conducting a principal component analysis (PCA) on the set of locations $v_1, \dots, v_n$ to identify the contribution each coordinate makes to overall variance.

To carry out PCA on the set of locations $v_1, \dots, v_n$, we begin by centering the locations such that the means of each coordinate is 0. Let

$$\bar{v}_{ik} = v_{ik} - \frac{\sum_{l=1}^{n} v_{lk}}{n}$$

be the centred kth coordinate of v_i, and let $\overline{V} = [\bar{v}_1, \dots, \bar{v}_n]$ be the matrix of centred locations. Note that centering each coordinate individually does not change pairwise distances,

$$\begin{aligned}
\left\|\bar{v}_i - \bar{v}_j\right\|^2 &= \sum_{k=1}^{t} \left(\bar{v}_{ik} - \bar{v}_{jk}\right)^2 \\
&= \sum_{k=1}^{t} \left(v_{ik} - \frac{\sum_{l=1}^{n} v_{lk}}{n} - v_{jk} + \frac{\sum_{l=1}^{n} v_{lk}}{n}\right)^2 \\
&= \sum_{k=1}^{t} \left(v_{ik} - v_{jk}\right)^2 = \left\|v_i - v_j\right\|^2 .
\end{aligned}$$

Let $\overline{G} = \overline{V}^T \overline{V}$ be the Gram matrix of the centred locations and let $\overline{G} = \overline{W}\,\overline{\Lambda}\,\overline{W}^T$ be its spectral decomposition, where $\lambda_1 \geq \cdots \geq \lambda_t$ are the nonzero eigenvalues associated with coordinates $1, \dots, t$. It follows from the theory of PCA that the proportion of overall variance explained by coordinate k is given as the ratio

$$\frac{\lambda_k}{\sum_{l=1}^{t} \lambda_l}.$$

We can use this ratio to strategically partition the set of coordinates. However, in order to conduct PCA, we must first recover locations $v_1, \dots, v_n$, and then carry out eigenvalue decomposition on the centred locations, leading to eigenvalue decomposition being carried out twice.

Interestingly, we can calculate $\overline{G}$ without prior knowledge of the non-centred locations $v_1, \dots, v_n$, and therefore only need to carry out eigenvalue decomposition once. Let

$\overline{G} = [\overline{g}_{ij}]_{1 \le i,j \le n}$, then it follows that

$$\begin{aligned}
\overline{g}_{ij} &= \langle \overline{v}_i, \overline{v}_j \rangle \\
&= \sum_{k=1}^{t} \overline{v}_{ik} \overline{v}_{jk} \\
&= \sum_{k=1}^{t} \left(v_{ik} - \frac{\sum_{l=1}^{n} v_{lk}}{n} \right) \left(v_{jk} - \frac{\sum_{l=1}^{n} v_{lk}}{n} \right) \\
&= \sum_{k=1}^{t} \left(v_{ik} v_{jk} - v_{ik} \frac{\sum_{l=1}^{n} v_{lk}}{n} - v_{jk} \frac{\sum_{l=1}^{n} v_{lk}}{n} + \left(\frac{\sum_{l=1}^{n} v_{lk}}{n} \right)^2 \right) \\
&= \sum_{k=1}^{t} \left(v_{ik} v_{jk} - \frac{1}{n} \sum_{l=1}^{n} v_{ik} v_{lk} - \frac{1}{n} \sum_{l=1}^{n} v_{jk} v_{lk} + \frac{1}{n^2} \sum_{l=1}^{n} \sum_{q=1}^{n} v_{lk} v_{qk} \right) \\
&= \sum_{k=1}^{t} v_{ik} v_{jk} - \frac{1}{n} \sum_{l=1}^{n} \sum_{k=1}^{t} v_{ik} v_{lk} - \frac{1}{n} \sum_{l=1}^{n} \sum_{k=1}^{t} v_{jk} v_{lk} + \frac{1}{n^2} \sum_{l=1}^{n} \sum_{q=1}^{n} \sum_{k=1}^{t} v_{lk} v_{qk} \\
&= g_{ij} - \frac{1}{n} \sum_{l=1}^{n} g_{il} - \frac{1}{n} \sum_{l=1}^{n} g_{jl} + \frac{1}{n^2} \sum_{l=1}^{n} \sum_{q=1}^{n} g_{lq}, \qquad (3.9)
\end{aligned}$$

where (3.9) comes from the fact $\sum_{k=1}^{t} v_{ik} v_{jk} = \langle v_i, v_j \rangle = g_{ij}$. Therefore, we can reconstruct G as before using (3.7), and then generate $\overline{G}$ using (3.9). This allows us to determine the *centred* Gram matrix of v points, $\overline{G}$, using only the locations $u_1, \dots, u_n$. By then conducting eigenvalue decomposition on $\overline{G}$, we not only recover the locations $\overline{v}_1, \dots, \overline{v}_n$, but also their eigenvalues, which are now representative of the importance of each coordinate.

3.3.4 Partition Strategies

Determining strategic partitions is crucial. While fully partitioning the problem may yield the strongest individual cuts within each partition, this introduces extra decision variables and potentially an exponential number of additional constraints. An effective partition strategy should balance the increased computational load of these additional variables and constraints with the benefits of achieving stronger cuts in each partition.

We now present three partitioning strategies, that use the results from PCA to generate effective partitions. Each strategy aims to partition coordinates $1, \dots, t$ into m subsets, such that each partition is easily approximated through cutting planes, and competing objectives are easily balanced, thus improving the effectiveness of Algorithm 2. For each approach, let m give the desired number of subsets, and let the coordinates be ordered based on their importance such that $\lambda_1 \ge \dots \ge \lambda_t$.

The first approach involves a greedy selection of coordinates based on their importance, to form partitions of similar size. An outline of this strategy is shown in Algorithm 3, which results in $t\%m$ partitions of size $\lfloor t/m \rfloor + 1$, and $m - t\%m$ partitions of size $\lfloor t/m \rfloor$,

Algorithm 3: Greedy partitions of t coordinates into m partitions.

```
function GreedyPartitions ($t, m$):
    $T_1, \dots, T_m \leftarrow \emptyset$
    $c \leftarrow 1$
    for $r = 1, \dots, t\%m$ do
        $T_r \leftarrow \{c, \dots, c + \lfloor t/m \rfloor + 1\}$
        $c \leftarrow c + \lfloor t/m \rfloor + 2$
    for $r = t\%m + 1, \dots, m$ do
        $T_r \leftarrow \{c, \dots, c + \lfloor t/m \rfloor\}$
        $c \leftarrow c + \lfloor t/m \rfloor + 1$
    $T \leftarrow \{T_1, \dots, T_m\}$
    return $T$
```

where % represents the modulo function. Coordinates are added to these partitions in order of importance. As such, early partitions contain more important coordinates and can thus have a larger potential influence on the objective function. Furthermore, within each partition coordinate importance is expected to be similar.

Algorithm 4: Stepped partitions of t coordinates into m partitions.

```
function SteppedPartitions ($t, m, \lambda_1, \dots, \lambda_t$):
    $T_1, \dots, T_m \leftarrow \emptyset$
    $k \leftarrow 1$ , $r \leftarrow 1$
    $\Lambda \leftarrow 0$
    while $k \leq t$ do
        $T_r \leftarrow T_r \cup \{k\}$
        $\Lambda \leftarrow \Lambda + \lambda_k$
        $k \leftarrow k + 1$
        if $\Lambda / \sum_{i=1}^{t} \lambda_i \geq r/m$ then $r \leftarrow r + 1$
    $T \leftarrow \{T_1, \dots, T_m\}$
    return $T$
```

Algorithm 3 can be adjusted to ensure that all partitions account for a similar amount of total variance while maintaining a greedy selection of coordinates. Consequently, earlier partitions may contain only a handful of coordinates, whereas later partitions should contain more. This approach is detailed in Algorithm 4, and referred to as stepped partitions. The result is a set of partitions where each subset accounts for roughly $1/m$ of the total variance, however, the number of coordinates within each partition may vary substantially.

Algorithm 5: Stratified partitions of t coordinates into m partitions.

```
function StratifiedPartitions $(t, m)$:
    $T_1, \dots, T_m \leftarrow \emptyset$
    $r \leftarrow 1$
    for $k = 1, \dots, t$ do
        $T_r \leftarrow T_r \cup \{k\}$
        $r \leftarrow r + 1$
        if $r = m + 1$ then $r \leftarrow 1$
    $T \leftarrow \{T_1, \dots, T_m\}$
    return $T$
```

The final strategy aims to establish partitions with large internal variances while trying to form partitions of similar sizes and total variance explained. This method emphasises selecting diverse coordinates within each partition. To achieve this, coordinates are allocated to partitions from 1 to m in descending order of significance, before looping back to the first partition and repeating. This stratified approach is outlined in Algorithm 5. The result is a series of similar-sized partitions, with diverse interval variances.

3.3.5 Illustrative Example

We conclude this section with a brief illustrative example to demonstrate the effectiveness of the partitioned cutting plane approach. In this example, we randomly generated 51 locations in $\mathbb{R}^{10}$, and selected a heuristic solution where $p = 5$. Figure 3.7 displays the first two coordinates of these locations as well as the chosen solution. Using the lower bound from (3.4), we see that this solutions removes at least 1.45% of the search space.

To implement the partitioned cutting plane algorithm, we begin by computing the centred Gram matrix from (3.9). We then employ eigenvalue decomposition to construct locations $v_1, \dots, v_n \in \mathbb{R}^t$, resulting in 50 non-zero eigenvalues, and thus $t = 50$. PCA reveals that the variance explained by each of the 50 coordinates ranges from 9.62% to 0.57%. Therefore, despite the large number, not all coordinates are expected to provide a significant contribution to the objective. Finally, we use Algorithm 5 to create 25 stratified partitions, each with 2 coordinates.

Figure 3.8 presents the distribution of locations in every partition. It also displays the exclusion zones of the solution from Figure 3.7, calculated using (3.3) with squared Euclidean distances. Clearly, partitioning the coordinates leads to a substantial improvement in the approximation of the objective function. In a majority of partitions, the exclusion zones cover a notable portion of other locations, and in some cases they cover almost the entire space.

Furthermore, we calculate (3.4) for the provided solution within each partition. While

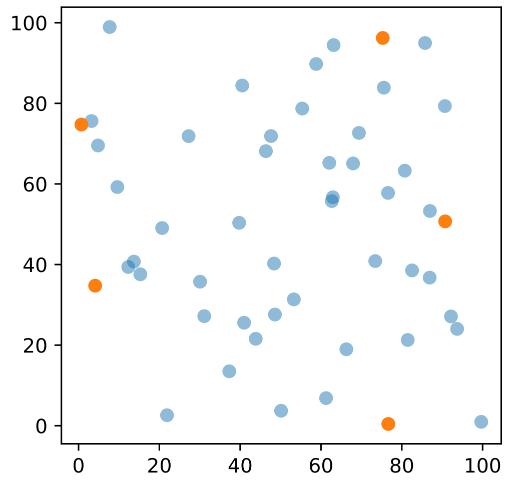

Figure 3.7: First two coordinates of 51 locations in $\mathbb{R}^{10}$, as well as a heuristic solution with $p = 5$. This solution removes at least 1.45% of the search space.

this value no longer has the same interpretation as before due to the objective changing to $\sum_{r=1}^{m} \theta_r$, it still provides a measure of relative strength of each cut. For instance, in partition 1 we see that more than 77.7% of the search space leads to a smaller value of θ_1. Therefore, for an incumbent solution to satisfy line 3 of Algorithm 2, it must counteract the decrease in θ_1 by a relative increase in objective contribution from another partition. However, this is difficult to achieve, as almost all other partitions exclude a greater proportion of the search space compared to the original solution in the non-partitioned space. Therefore, coordinate partitioning should lead to significant improvements in the approximation of the objective function as well as the removal of nonoptimal solutions.

3.4 Numerical Results

We now present numerical results for the partitioned cutting plane algorithm and accompanying partition strategies. The algorithm is implemented in C++ using `CPLEX` version 22.1.1 as its mixed-integer linear solver. The *lazy constraint callback* functionality is used to allow tangent planes to be added during the branch and bound procedure. Eigenvalue decomposition is conducted using the `Eigen` 3.4.0 library (available at https://eigen.tuxfamily.org). The source code of our implementation can be accessed at https://github.com/sandyspiers/coordinate_partitioning. All tests were conducted on a machine with a 2.3 GHz AMD EPYC processor with 64GB RAM, using a single thread.

The solution algorithms are evaluated using two classes of randomly generated in-

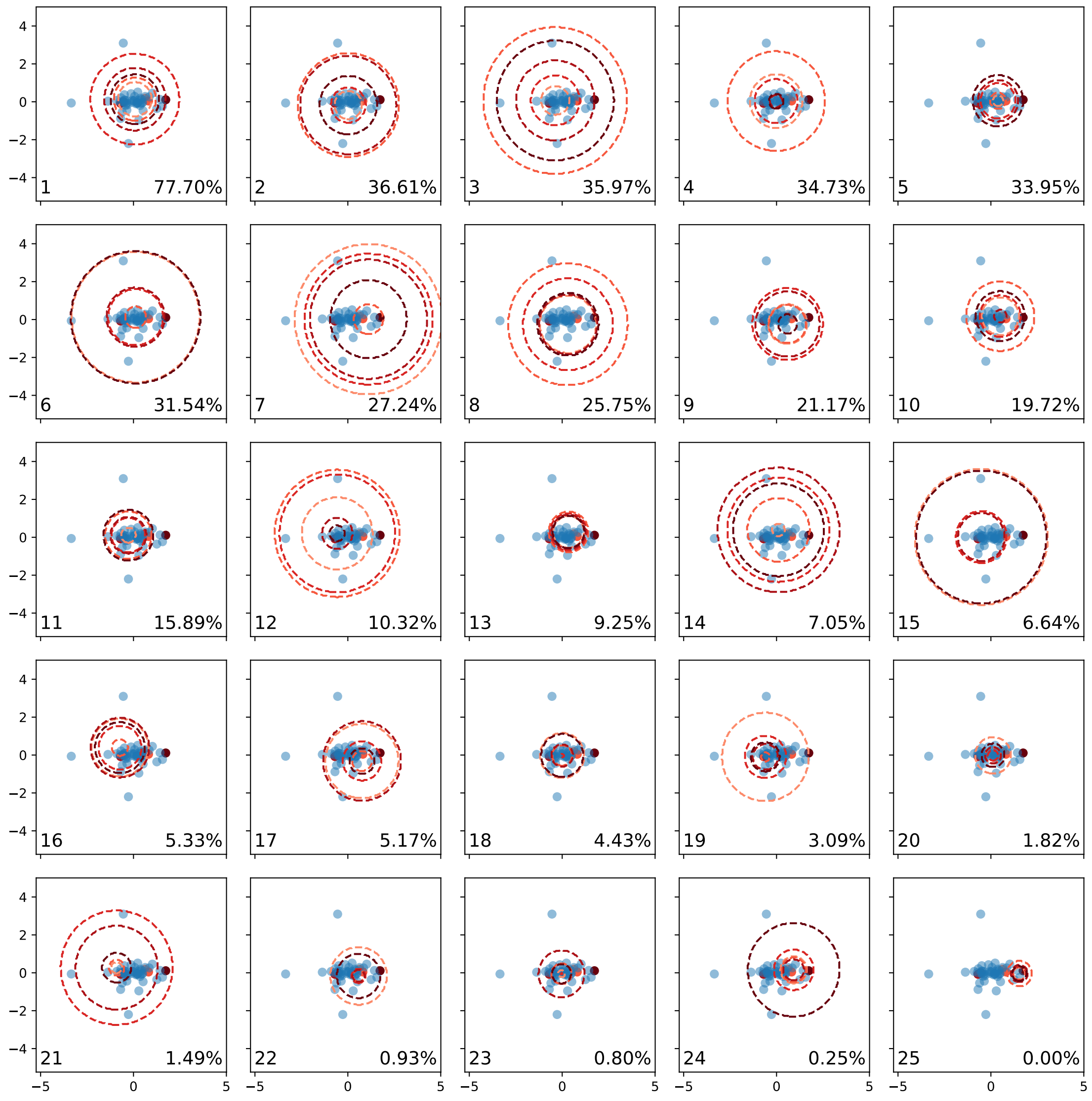

Figure 3.8: The 25 two-coordinate partitions of the locations from Figure 3.7. For each partition we show the exclusion zones calculated from (3.3) using squared Euclidean distances, as well as the lower bound (3.4).

stances. The first class, referred to as cube instances, has locations uniformly distributed within a hypercube of dimension s. These instances align with the framework suggested in Chapter 2, which was originally an extension of that proposed in Glover et al. (1995), with additional emphasis put on the number of coordinates of original locations. Additionally, we test the solvers on ball instances, where locations are situated on the boundary of a hyperball of dimension s. This class of problem, first introduced in Section 3.2, remains unexplored in existing diversity problem literature.

Algorithm 6: Random partitions of t coordinates into m partitions.

```
1 function RandomPartitions ($t, m$):
2     $C \leftarrow \{1, \dots, t\}$ , $T_1, \dots, T_m \leftarrow \emptyset$ , $r \leftarrow 1$
3     while $C \neq \emptyset$ do
4         Choose a random $c \in C$
5         $C \leftarrow C \setminus \{c\}$ , $T_r \leftarrow T_r \cup \{c\}$ , $r \leftarrow r + 1$
6         if $r = m + 1$ then $r \leftarrow 1$
7     $T \leftarrow \{T_1, \dots, T_m\}$
8     return $T$
```

Random partitions of similar size are used as a benchmark against our three proposed partitioning techniques. These benchmark partitions are generated using Algorithm 6, which results in $t\%m$ partitions of size $\lfloor t/m \rfloor + 1$ and $m - t\%m$ partitions of size $\lfloor t/m \rfloor$. This creates a similar partition set to Algorithm 3, however now coordinates are chosen at random. This strategy will be used as a control, to test whether the proposed partitioning methods produce significant improvements.

3.4.1 Cube Instances

We begin by assessing the performance of the partitioned cutting plane algorithm on a large set of randomly generated cube instances. Location sets are generated in sizes $n = 100, 250, 500, 1000$, each with coordinate counts $s = 2, 5, 10, 15, 20$. Every coordinate of a location is uniformly randomly generated in the range $[0, 100]$. Consequently, locations are uniformly distributed inside a hypercube of dimension s, such as those shown in Figure 3.2 of Section 3.2. For every combination of number of locations and number of coordinates, we generate five unique sets of locations. Then, for each set of locations, we solve the (EMSDP) where $p = 0.1n$ and $p = 0.2n$, resulting in a total of 200 test instances.

Each problem instance is solved by the partitioned cutting plane algorithm using one of the three proposed partitioning strategies, as well as a random partition set generated by Algorithm 6. The number of desired partitions is set to $m = 0.1n, 0.25n, 0.5n$, or $0.75n$. We then further solve the problem using a full partitioning ($m = n$), as well as the

n	Cutting Plane	Partition Ratio				
		0.10	0.25	0.50	0.75	1.0
100	0.00	0.00	0.00	0.00	0.01	0.01
250	0.00	0.03	0.04	0.07	0.10	0.12
500	0.00	0.23	0.34	0.53	0.74	0.90
1000	0.01	1.59	2.43	3.90	5.78	8.36

Table 3.1: Average setup time in seconds for cube test instances at various partition ratios. This includes the time to conduct eigenvalue decomposition and to construct m distance matrices.

cutting plane method from Chapter 2 as a benchmark. This makes a total of 17 solver configurations.

Before we delve into the performance of the various solution methods, it's worth noting that in order to use the partitioned cutting plane algorithm, we must first perform eigenvalue decomposition to determine locations $v_1, \dots, v_n$, and then construct m distance matrices, one for each partition. While this can be a computationally demanding task, we show in the remainder of this section how solve time is generally far greater than setup time. Nonetheless, the average time required to setup the partitioned cutting plane solver for each partition ratio is detailed in Table 3.1. Notably, the setup time only becomes significant when $n = 1000$. Moreover, we know that the computational load of eigenvalue decomposition should be consistent for a given n. Therefore, the longer setup times seen at high partition ratios are due to the extra time required to construct and save to memory a large number of distance matrices. This time could possibly be improved by using a more sophisticated, on-the-fly approach to pairwise distance calculations, however, we leave this task as an avenue for future improvements.

A summary of performance for each solver configuration on the random cube instances is shown in Table 3.2. This table displays the average total time in seconds, including both setup and solve times, where the latter is capped at 1000 seconds. The results of the standard cutting plane algorithm are consistent with the EDC Algorithm from Chapter 2, with the method being very efficient given a small number of coordinates, but struggles as s grows. In fact, for $s = 2$, the cutting plane method remains the most efficient and in many cases vastly outperforms Algorithm 2 at every partition configuration. With fewer coordinates, the objective function is easy to approximate, and hence excess partitions can slow this process down. However, as the number of coordinates grows, partitions help to better approximate the objective, leading to an overall improvement in runtime. Comparing the different strategies, greedy partitions appear to be the worst strategy by a considerable margin. In many cases, greedy partitions can lead to more than double

s	n	Cutting Plane	Random				Greedy				Stepped				Stratified				Fully Partitioned
			0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75	0.10	0.25	0.50	0.75	
2	100	0.03	**0.03**	0.06	0.11	0.19	**0.06**	0.09	0.11	0.16	**0.02**	0.03	0.05	0.07	**0.03**	0.05	0.08	0.13	0.17
	250	0.09	**0.23**	0.50	1.07	1.45	**0.51**	1.11	1.38	2.00	**0.22**	0.39	0.71	1.15	**0.21**	0.52	0.91	1.46	1.96
	500	0.40	**1.77**	4.00	7.58	11.62	**5.28**	8.35	11.31	16.83	**1.64**	3.81	6.44	9.55	**1.66**	3.68	7.39	11.44	14.81
	1000	0.53	**10.67**	20.66	42.65	67.27	**26.34**	54.84	61.74	101.94	**9.13**	18.37	35.31	50.51	**8.80**	21.99	37.57	62.26	84.43
5	100	2.08	0.33	**0.26**	0.30	0.54	0.93	0.56	**0.41**	0.66	0.41	0.25	**0.23**	0.28	0.27	**0.22**	0.28	0.40	0.46
	250	2.92	**1.11**	1.31	2.12	3.28	4.29	4.03	**2.87**	5.48	**0.90**	1.14	1.46	1.96	**0.89**	1.22	2.02	3.01	4.04
	500	29.87	**4.12**	6.53	10.40	17.50	29.60	16.32	**15.63**	29.78	**3.46**	5.68	7.91	12.45	**4.03**	5.51	9.27	16.12	19.01
	1000	86.93	**19.13**	33.98	69.71	86.53	107.39	84.21	**83.38**	131.90	**15.70**	29.85	47.83	78.71	**19.16**	35.76	54.69	92.36	118.91
10	100	282.98	4.02	1.71	**1.61**	2.19	13.99	2.85	**1.71**	3.06	6.65	3.13	1.63	**1.52**	3.41	1.48	**1.17**	2.22	1.89
	250	432.62	86.52	35.41	**31.58**	39.52	324.19	87.45	**36.05**	71.17	254.54	82.01	38.49	**28.98**	70.09	27.94	**23.59**	32.42	36.95
	500	509.59	53.99	**43.86**	48.16	77.57	319.85	122.03	**70.08**	131.95	93.36	54.58	**49.84**	50.37	53.36	**39.47**	49.28	71.97	79.60
	1000	696.31	**142.37**	155.78	225.63	302.03	728.21	393.82	**310.55**	574.97	**168.30**	185.14	200.49	221.56	**132.31**	136.73	195.58	319.59	347.27
15	100	903.94	321.39	51.01	**34.01**	44.16	483.02	87.16	**38.29**	80.75	387.59	219.36	55.09	**38.51**	307.41	49.57	**26.97**	35.67	39.13
	250	889.99	250.57	87.07	**69.72**	94.78	470.36	234.84	**96.59**	167.47	307.22	243.68	94.42	**75.28**	243.81	91.92	**62.80**	74.14	79.21
	500	845.83	350.48	200.02	**197.36**	246.15	604.86	417.11	**249.67**	429.08	456.71	353.21	226.35	**171.68**	312.71	188.61	**164.44**	232.46	261.62
	1000	946.62	592.59	**561.84**	600.46	690.40	937.49	734.99	**682.33**	845.65	607.43	**600.24**	608.67	636.41	567.46	**551.39**	576.88	694.44	735.38
20	100	780.19	161.40	26.46	**15.96**	18.56	327.71	45.21	**22.51**	37.99	190.93	80.20	27.40	**17.12**	155.33	24.08	**15.09**	17.33	18.17
	250	1000.02	571.26	457.35	**422.75**	463.26	894.25	523.21	**458.98**	504.26	686.43	532.27	463.18	**446.71**	541.09	455.54	**414.57**	445.68	429.50
	500	973.22	687.14	561.68	**470.84**	575.18	893.22	699.99	**565.95**	687.74	797.52	650.64	523.47	**469.82**	674.88	498.60	**450.38**	487.02	496.08
	1000	879.52	577.01	**537.34**	567.22	645.88	889.86	736.29	**654.25**	785.02	680.76	592.67	586.58	**563.46**	559.20	552.66	**552.01**	630.60	708.71

Table 3.2: Comparison of average run time in seconds (across 10 problem instances) for the standard and partitioned cutting plane algorithms using four partition strategies at various ratios and the full partition. Time includes setup and solve time, where the solve time is capped at 1000 seconds. The best partition ratio for each strategy is shown in bold.

the average solve time compared to the random partitions benchmark. In contrast, the stepped and stratified strategies are usually able to improve on the random benchmark, however, this improvement is often marginal. As expected, the full partition approach only becomes competitive for large s, however the improvement is often minimal.

To more clearly assess the impact of partition ratio on runtime, Figure 3.9 illustrates how the optimal partition ratio changes as the number of coordinates increases. The figure plots the average runtime for each coordinate count, as well as the number of cuts added per partition. For fewer coordinates, specifically $s = 2, 5$, there is a clear correlation between average solve time and partition count, with additional partitions slowing down runtime. However, by $s = 10$, this trend starts to shift. We can consider this as the cross-over point, after which we notice the opposite trend, where partitioning up to the $m = 0.5n$ ratio leads to faster run times.

Examining the number of cuts needed per partition, it's clear that increasing the number of partitions can considerably reduce the number of cuts required. This is especially true when compared with the cutting plane algorithm at high coordinates. This outcome is to be expected, as partitions with fewer coordinates should be simpler to approximate and thus require fewer cuts. However, while a full partitioning results in the fewest cuts per partition, it results in the largest overall problem size. As a result, its average run time is not the most efficient. Considering the range of coordinates tested, a partition ratio of 0.5 appears to be the most well-balanced strategy, as it is competitive at both high and low number of coordinates. Although this ratio is slightly slower than the cutting

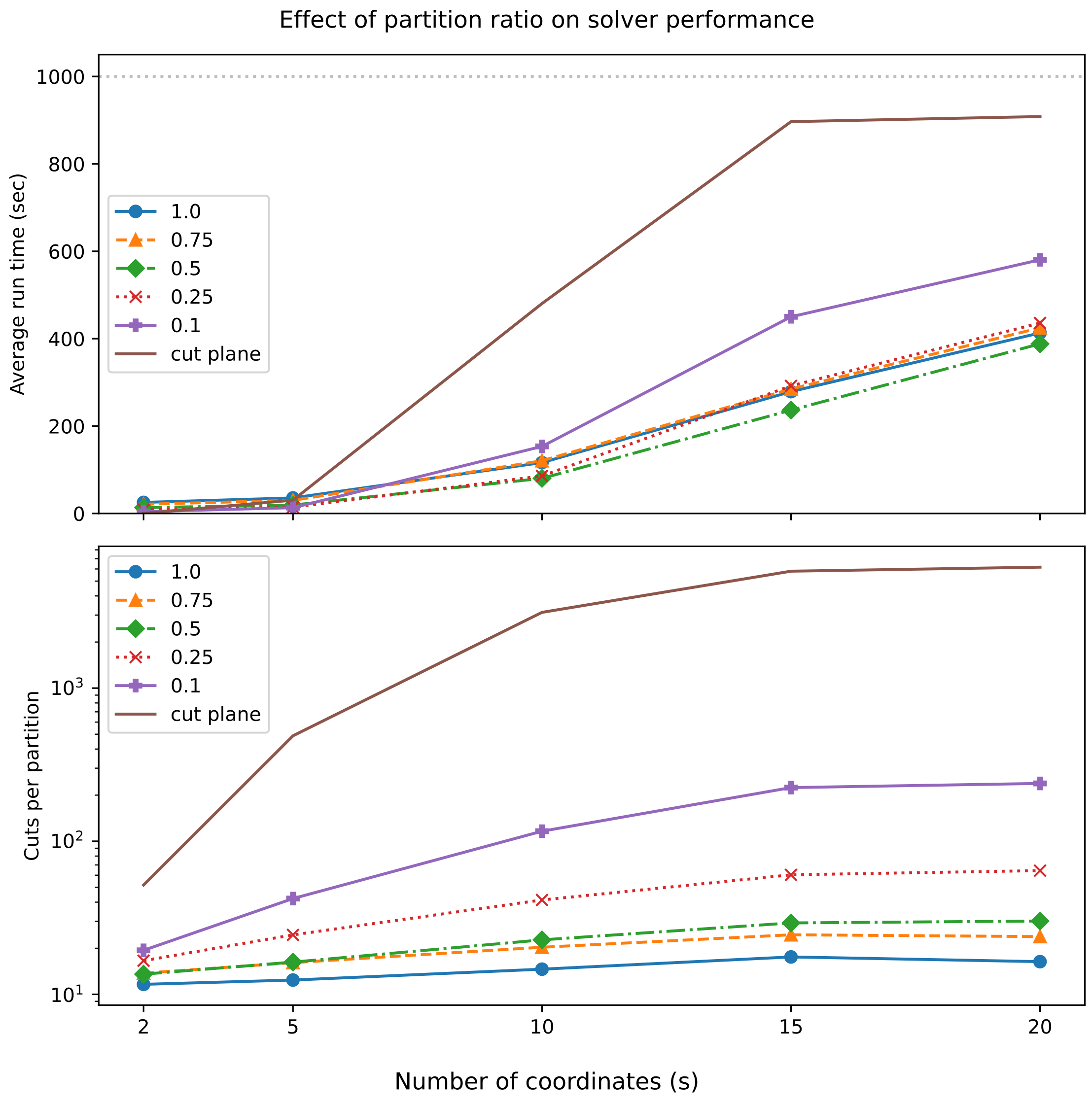

Figure 3.9: The average run time in seconds, and average number of cuts added per partition for different number of coordinates, broken down by partition ratio. Time includes setup and solve time, where the solve time is capped at 1000 seconds. Note that a 1.0 partition ratio refers to a full partitioning.

	Fully Partitioned	Random	Greedy	Stepped	Stratified
Random	1.2405e-28		6.8159e-29	8.2721e-01	1.0
Greedy	2.3191e-06	1.0		1.0	1.0
Stepped	5.8087e-14	1.7279e-01	3.5090e-16		9.8398e-01
Stratified	3.9121e-33	3.3279e-10	1.2583e-32	1.6023e-02	

Table 3.3: Results of the paired Wilcoxon signed-rank test comparing partition strategies at a 0.5 partition ratio, as well as the fully partitioned strategy. Each entry shows the p-value for the hypothesis that the strategy in the row leads to faster solve times compared to the strategy in the column, across all cube test instances.

plane algorithm at $s = 2$, it's a minor difference compared to the significant runtime improvements observed at higher coordinates. Therefore, a 0.5 ratio appears to be the best balance, offering a significant reduction in cuts per partition without excessively increasing the number of objective terms.

Figure 3.10 compares the performance of the four partition strategies at a ratio of 0.5, across the five different values of s. The standard cutting plane algorithm is used as a benchmark. The results clearly show how partitioning the coordinates makes the solver far more resilient to increasing the number of coordinates. At $s = 2$, while the partitioned strategies do perform worse than the original cutting plane algorithm, they are still able to efficiently solve all test instances well within the 1000-second time limit. Increasing to $s = 5$, all approaches are comparable, with only slight variations in performance. However, this would appear to be the turning point. At a larger number of coordinates, the partitioned cutting plane algorithm begins to vastly outperform the standard approach. Impressively, at the $s = 20$ level, the partitioned approach can solve more than half the test set, whereas the standard cutting plane algorithm is only able to solve a small number.

Comparing the various partitioning strategies is somewhat challenging using the performance profile alone. However, the stratified approach appears to be the best overall performer at the 0.5 ratio, frequently outperforming the random benchmark. Notably, the stepped approach also competes closely with this benchmark, despite resulting in partitions of significantly varied sizes.

To verify the statistical significance of these observations, Table 3.3 presents the results of a paired Wilcoxon signed-rank test, comparing the solve time of different partition strategies at a 0.5 partition ratio across all cube test instances. Each entry in the table tests the hypothesis that the strategy in the corresponding row results in a faster solve time compared to the strategy in the corresponding column.

The results support the observation that the stratified partition strategy is the fastest at a 0.5 partition ratio. At a 5% level of significance, we can conclude that this partitioning

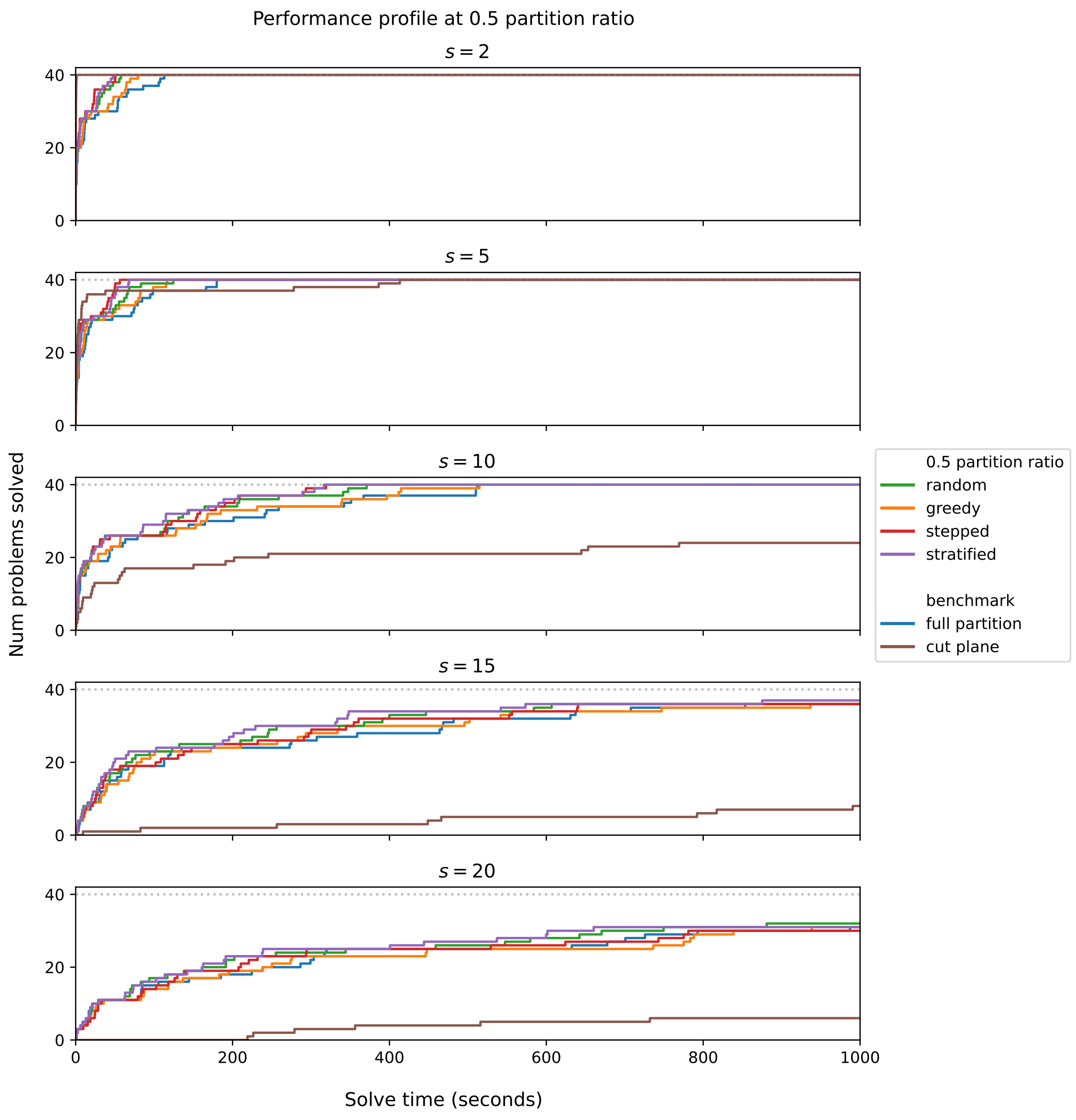

Figure 3.10: Solver performance on the cube instances at various number of coordinates, with partition ratio as $m = 0.5n$. We additionally show the performance of the fully partitioned approach and the standard cutting plane algorithm.

strategy is the most efficient across the cube test instances. Additionally, the random benchmark outperforms the greedy partition strategy (p-value 6.8159e-29), indicating that this was potentially a poor strategy to use for this test set.

These findings suggest that using eigenvalues and PCA to guide partitioning is a promising technique that can lead to effective and deterministic partitions. Having a deterministic strategy is crucial to ensuring the robustness of Algorithm 2. For example, if one was to rely random allocations alone, then this might result in partitions similar to those produced by the greedy approach. However, we can see from the previous results that greedy partitions lead to the worst average solve time. As such, using PCA to deterministically partition coordinates appears to be very effective, and could offer significant benefits for other problem classes as well.

Finally, while the partitioned cutting plane algorithm is more robust than the original for increasing values of s, it still suffers a minor deterioration in performance. This indicates that some of the structure of points $u_1, \dots, u_n$ that make tangent planes weak (such as low interior density), remain even after mapping to points $v_1, \dots, v_n$. While partitioning functional components remedies most of these issues, some difficulties still remain. As such, we still see the partitioned approach worse with increasing s. With that said, it is far more robust than the original.

3.4.2 Ball Instances

We complete this section by exploring the performance of the partitioned cutting plane algorithm on the ball instances first mentioned in Section 3.2. These instances are such that every location is situated on the edge of a hyperball of dimension s, such as those shown in Figure 3.5. We generate instances of sizes of $n = 25, 50, 100$, with the number of coordinates as $s = 2, 5$. Each coordinate of a location is randomly generated from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$. Subsequently, every location $x \in \mathbb{R}^s$ is translated to $x \leftarrow 50x/\|x\|$, thereby positioning the location on the edge of a hyperball with radius 50. For every combination of number of locations and number of coordinates, we generate five unique sets of locations. Then, for each set of locations, we solve the (EMSDP) with $p = 5$ for $n = 25$ and $p = 0.1n, 0.2n$ for $n = 50, 100$, resulting in a total of 50 test instances.

Although the cutting plane algorithm has been shown to be the leading solution method for cube instances, this cannot be said for the ball instances, as they have not yet been explored in the literature. Therefore, in addition to the standard and partitioned cutting plane algorithm, we also benchmark the partitioned cutting plane algorithm against the Glover reformulation detailed in (2.4). This mixed-integer linear reformulation is subsequently solved using `CPLEX`.

Table 3.4 displays the average run time and average final optimality gap for the ball instances, where solve time is limited to 1000 seconds. Firstly, it is clear that the ball

s	n	Cutting Plane	Glover	Random			Stratified			Fully Partitioned
				0.25	0.5	0.75	0.25	0.5	0.75	
2	25	1.01 (0.00)	2.26 (0.00)	0.46 (0.00)	0.46 (0.00)	0.72 (0.00)	0.47 (0.00)	0.46 (0.00)	0.63 (0.00)	0.68 (0.00)
	50	572.83 (0.53)	602.72 (26.32)	267.73 (0.00)	153.69 (0.00)	265.43 (0.00)	234.38 (0.00)	172.94 (0.00)	207.21 (0.00)	200.17 (0.00)
	100	1000.02 (1.28)	1000.01 (119.19)	1000.01 (0.42)	1000.03 (0.38)	1000.02 (0.38)	1000.01 (0.40)	1000.02 (0.37)	1000.02 (0.36)	1000.01 (0.37)
5	25	36.89 (0.00)	2.60 (0.00)	3.56 (0.00)	2.60 (0.00)	3.73 (0.00)	3.55 (0.00)	2.25 (0.00)	3.17 (0.00)	3.69 (0.00)
	50	1009.15 (6.76)	603.60 (19.95)	736.52 (0.65)	634.82 (0.48)	675.29 (0.54)	696.62 (0.63)	618.39 (0.48)	673.30 (0.48)	641.76 (0.49)
	100	1000.02 (4.58)	1000.00 (97.41)	1000.03 (2.02)	1000.04 (1.81)	1000.02 (1.84)	1000.05 (2.02)	1000.02 (1.76)	1000.05 (1.79)	1000.12 (1.71)

Table 3.4: Average runtime in seconds across the various solver configurations for the ball test instances, where solve time is capped at 1000 seconds. We show in parenthesis the average final optimality gap as a percentage.

instances represent a particularly challenging class of the (EMSDP). When considering the $n = 100$ problem size, none of the solution methods were able to solve a single problem to optimality within the time limit. While there is an improvement in performance for the smaller $n = 50$ instances, all the solvers lagged significantly when compared with the outcomes from the cube instances. At $s = 2$, the partitioned cutting plane algorithm is clearly the most effective, consistently yielding the smallest average runtime and optimality gap. Furthermore, the 0.5 partition ratio once again provides the best balance between problem size and cut strength. However, for $s = 5$, the performance of the partitioned approach deteriorate noticeably. In fact, Glover linearisation becomes fairly competitive for these test instances and achieved similar average run times. That said, Glover linearisation yields very poor bounds if the problem is not solved within the time limit, resulting in a very large average final gaps. In contract, all partition strategies yield fairly small optimality gaps, with the largest being 2.02%.

The challenge of the ball instances becomes clear when examining the performance breakdown in Figure 3.11. This figure shows the average primal-dual gap and number of cuts added at 4 time points across the 1000-second limit. Within the first 60 seconds, the algorithm is capable of achieving a respectable bound of less than 4%. However, this gap is not closed significantly when extending out to 1000 seconds. This is in contrast to the number of cuts added, which climbs substantially over the solve time. In some cases, the number of cuts almost doubles over this time frame. Therefore, even though the problem size is small, the tangent planes are not able to efficiently direct the search. As a result, the solver struggles to close the gap, leading to poor performance when compared with the cube instances.

3.5 Conclusion

This chapter introduces a novel approach to tackle challenging Euclidean max-sum diversity problems. By constructing a new set of locations, whose squared Euclidean

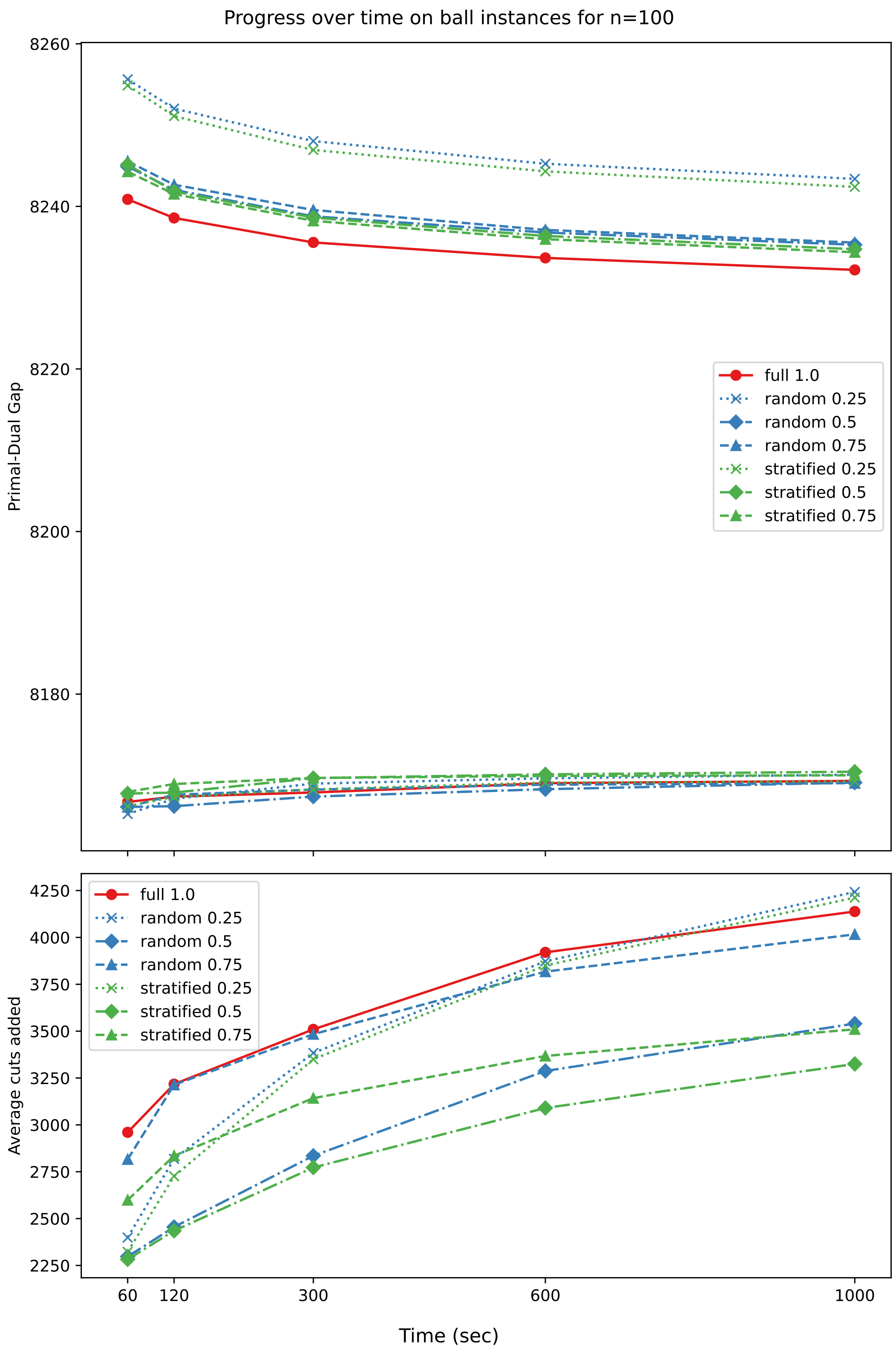

Figure 3.11: The average primal-dual gap and average number of cuts added during the solve procedure for the ball test instances with $n = 100$ for various solver configurations.

distances match the Euclidean distances of the original, we can employ a functional decomposition of the objective. This allows us to break up the objective based on subsets of coordinates. We show how, after partitioning the set of coordinates, the new objective terms are expected to be easier to approximate by tangent planes, making the problem easier to solve. Furthermore, we show how principal component analysis can be conducted alongside location recovery, thus requiring minimal excess computation. The results from PCA highlight the relative importance of coordinates, and can therefore be used to guide the partitioning process.

Extensive numerical results prove the effectiveness of this new approach. By breaking down the objective function, the cutting plane algorithm becomes far more resilient to an increasing number of coordinates. As such, it can solve large, 20-coordinate instances of up to 1000 locations. Moreover, we introduce a new class of problems characterized by locations positioned on the edge of a ball. These instances have not been researched before in the context of diversity problems, and are shown to be very challenging. Nevertheless, the partitioned cutting plane algorithm remains the standout performer in this problem class, achieving far tighter best bounds.

4 Euclidean Max-Sum Problems[1]

This chapter studies binary quadratic programs in which the objective is defined by the maximisation of a Euclidean distance matrix, subject to a general polyhedral constraint set. This class of nonconcave maximisation problems, which we refer to as the Euclidean Max-Sum problem, includes the capacitated, generalised and max-sum diversity problems as special cases. Due to the nonconcave objective, traditional cutting plane algorithms are not guaranteed to converge globally. In this chapter, we introduce two exact cutting plane algorithms to address this limitation. The new algorithms remove the need for a concave reformulation, which is known to significantly slow down convergence. We establish exactness of the new algorithms by examining the concavity of the quadratic objective in a given direction, a concept we refer to as *directional concavity.* Numerical results show that the algorithms outperform other exact methods for benchmark diversity problems (capacitated, generalised and max-sum) and can easily solve problems of up to three thousand variables.

4.1 Introduction

In this chapter, we show how cutting planes can be used to generate exact solutions for the problem of maximising the sum of pairwise Euclidean distances between selected points, subject to general polyhedral constraints, hereafter referred to as the *Euclidean Max-Sum problem* (EMSP). The (EMSP) is a generalisation of the Euclidean Max-Sum diversity problem from previous chapters, in which the cardinality constraint is replaced by a general polyhedral set. More precisely, given a set of locations $u_1, \dots, u_n \in \mathbb{R}^s$ ($s \geq 1$), the (EMSP) is defined as the following nonconcave binary maximisation problem,

$$\begin{aligned} \max \quad & f(x) = \frac{1}{2}\langle Dx, x\rangle, \\ \text{s.t.} \quad & x \in P \cap \{0,1\}^n, \end{aligned} \tag{EMSP}$$

where $D = [d_{ij}]_{i,j=1,\dots,n}$ is an $n \times n$ Euclidean distance matrix defined by $d_{ij} = \|u_i - u_j\|$, and where $P \subset \mathbb{R}^n$ is a polyhedral set defined by

$$P = \{x \in \mathbb{R}^n \,:\, Ax \leq a\},$$

[1]This chapter is based on Bui et al. (2024)

where $A \in \mathbb{R}^{m \times n}$ and $a \in \mathbb{R}^m$. Here, the definition of x can be easily generalised to include both integer and continuous variables. We showed in Chapter 2.2, Corollary 7 that D is also a *squared* Euclidean distance matrix. As such, it is conditionally negative definite, i.e., $\langle Dx, x \rangle \leq 0$ if $\sum_{i=1}^n x_i = 0$, and has exactly one positive eigenvalue. In this chapter, we exploit this property to show how the cutting plane methodology, which is normally restricted to concave maximisation problems, can be applied to find an optimal solution of (EMSP).

The Euclidean max-sum problem has various important practical applications. In machine learning and statistical analysis, Euclidean distance is often used as a measure of dissimilarity between data points in clustering algorithms (Madhulatha, 2012; Shirkhorshidi et al., 2015). By maximising the Euclidean distance between points, clusters can be formed based on their dissimilarity, allowing for effective grouping and classification of data. An example of this is the Maximally Diverse Grouping Problem (Feo et al., 1992; O'Brien & Mingers, 1997). Furthermore, in various practical applications such as urban planning or network design, there is a need to strategically locate unwanted facilities such as waste disposal sites or polluting industries (Erkut & Neuman, 1989; Kuby, 1987). Maximising the distance between these unwanted (but necessary) facilities and sensitive areas such as residential zones or environmental conservation areas helps minimise the negative impact on surrounding communities or ecosystems. Lastly, maximising Euclidean distances allows for the selection of points that capture diverse characteristics or represent different regions of interest, thereby enhancing the coverage and diversity of the chosen set.

In the previous chapters, we formulated a cutting plane algorithm for the Euclidean max-sum diversity problem by establishing the concavity of the objective function on the hyperplane $\sum_{i=1}^n x_i = p$, which ensures that tangent planes of feasible solutions serve as valid upper planes. As such, our cutting plane algorithm is globally convergent for the Euclidean max-sum diversity problem. The resultant exact algorithm is competitive with heuristic and meta-heuristic methods and can solve two-coordinate instances of up to eighty thousand variables. However, without the cardinality constraint, the objective function is not concave over the feasible set, and hence tangent planes do not always form valid cuts. The purpose of this chapter is to develop a new cutting plane methodology that still converges for this more general problem, where concavity is not guaranteed.

To the best of our knowledge, outside of the Euclidean max-sum diversity problem, quadratic maximisation problems defined by Euclidean distance matrices have never been explored in isolation. One reason for this is that these maximisation problems are, in general, nonlinear and nonconcave. Mixed-integer nonlinear programming is one of the most challenging classes of optimisation problems. Although there are several exact methods that provide general frameworks to tackle *concave* maximisation problems, including outer approximation (Duran & Grossmann, 1986; Kronqvist et al., 2020; Leyffer, 1993; Lubin et al., 2018), branch and bound (Bonami et al., 2013; Gupta & Ravindran, 1983; Vielma et al., 2008), and cutting plane methods (Kronqvist et al., 2016; Lundell et al., 2022; Westerlund & Pettersson, 1995), advancements in exact algorithms for *nonconcave* problems are still modest. The most common way to handle binary nonconcave maximisation is to reformulate the problem into an equivalent *concave problem* by using a penalty approach, before applying exact methods to the new concave problem such as that outlined in Chapter 1.3.2.

This chapter extends the cutting plane algorithm to general Euclidean distance maximisation by relaxing the requirement for a cardinality constraint. This is achieved by exploiting the property that Euclidean distance matrices have exactly one positive eigenvalue. To provide intuition on the key idea, consider a full eigenvalue decomposition of the objective function,

$$f(x) = \frac{1}{2}\langle Dx, x\rangle = \frac{1}{2}x^T\left(\sum_{i=1}^{n} \lambda_i v_i v_i^T\right)x = \frac{1}{2}\sum_{i=1}^{n} \lambda_i x^T\left(v_i v_i^T\right)x,$$

where $\lambda_1 \geq \cdots \geq \lambda_n$ and $v_1, \dots, v_n$ denote the eigenvalues and eigenvectors of D. This expresses the quadratic objective as a sum of functional components which are either convex or concave, depending on the sign of their respective eigenvalues. However, as D is a Euclidean distance matrix, we have from Theorem 4 that it contains exactly one positive eigenvalue, and therefore $f(x)$ has only one convex component. By restricting our search domain to exclude directions that traverse this convex component, our objective function can effectively be treated as a concave function (see Lemma 18).

As an example, consider the hyperbolic paraboloid defined by

$$g(x, y) = xy = \frac{1}{4}(x + y)^2 - \frac{1}{4}(x - y)^2.$$

Clearly, whenever $ax + by = 0$ ($ab > 0$), the function reduces to $g(x, y) = x(-\frac{ax}{b}) = -\frac{a}{b}x^2$, which is concave. Hence, while $g(x, y)$ is nonconcave for $x, y \in \mathbb{R}$, it is concave on the $ax + by = 0$ plane. The resultant concave parabola is shown in red in Figure 4.1. This is essentially the technique used in Chapter 2, where the Euclidean distance matrix is known to contain exactly one positive eigenvalue, and hence the objective has one convex functional component. The cardinality constraint then ensures that the feasible set excludes this convex component, and the quadratic function can be treated as concave. For the general problem (EMSP), which may not include a cardinality constraint, the main idea of our approach is to only generate the tangent planes on concave directions. By doing so the cutting planes are valid and the algorithm always converges to an optimal solution.

The remainder of this chapter is organised as follows. In Section 4.2, we formalise the concept of directional concavity and, based on this, formulate two key sufficient conditions for valid tangent planes, as detailed in Theorem 20. These conditions then form the basis of two exact cutting plane algorithms which vary in their approach to

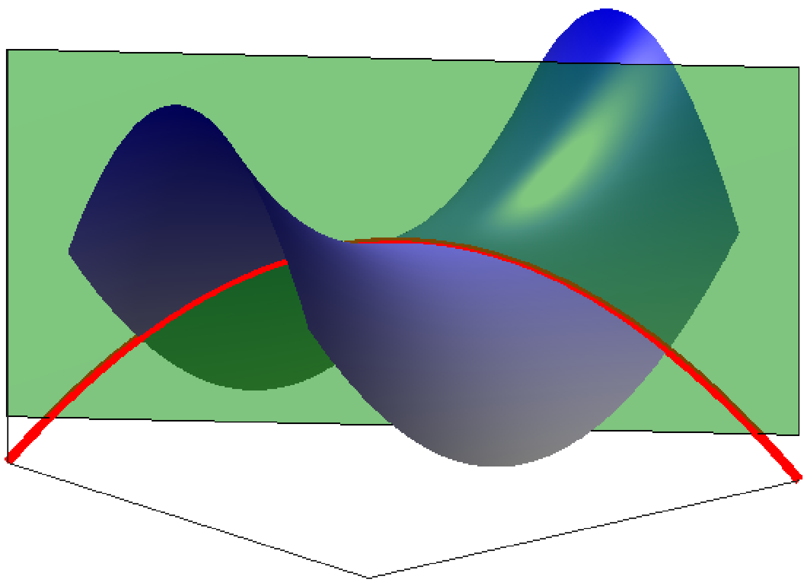

Figure 4.1: The intersection of a paraboloid and a hyperplane is either convex or concave.

generating new cuts. Finally, in Section 4.3 we conduct extensive numerical experiments to evaluate the effectiveness of the proposed solution approaches.

4.2 Methodology

We denote the feasible set of (EMSP) as $K = \{x \in \{0,1\}^n \ : \ x \in P\} \setminus \{0\}$, where $x = 0$ is excluded since $f(x) \geq 0 = f(0)$ for every $x \in K$. As before, let $h(x, y)$ be the tangent plane of y evaluated at x. We say that the tangent plane at a feasible solution $y \in K$ forms a *valid cut* if it provides an upper approximation for the optimal value of (EMSP), i.e, $f(x^*) \leq h(x^*, y)$, where x^* is an optimal solution of (EMSP). This differs from the majority of the existing literature, where valid cuts provide an upper approximation of the objective function at *all* feasible solutions (not just at an optimal solution).

Since the function f in (EMSP) is not concave, not every feasible solution $y \in K$ generates a valid cut. In this section, we establish sufficient conditions for when the tangent plane $h(x, y)$ is valid. The key to our approach is to study the concavity of the function f when restricted to a given direction, exploiting the observation that the restriction of a quadratic function to a line is either concave or convex.

4.2.1 Directional Concavity

Given a vector $u \in \mathbb{R}^n \setminus \{0\}$, we say that u is a concave direction of D if $\langle Du, u \rangle \leq 0$; this means that $f(x)$ is concave along the line with direction u emanating from the origin. Conversely, a vector $v \in \mathbb{R}^n \setminus \{0\}$ is a convex direction of D if $\langle Dv, v \rangle \geq 0$. Note that $x - y$ is a concave direction of D if and only if

$$f(x) - h(x, y) = \frac{1}{2} \langle D(x - y), x - y \rangle \leq 0.$$

Thus, the tangent plane at y is an upper approximation of $f(x)$ when the line from y to x is a concave direction of D. We now show that $u = x - y$ is a concave direction of the matrix D if u is orthogonal to Dz, where z is a convex direction of D.

Lemma 18. *Suppose $x, y \in \mathbb{R}^n$, and there is vector $z \in \mathbb{R}^n \setminus \{0\}$ such that*

a. $\langle Dz, z \rangle \geq 0$, *and*

b. $\langle Dz, x - y \rangle = 0$.

Then, $h(x, y) \geq f(x)$.

Proof. Recall from the proof of Proposition 9 that the inequality $h(x, y) \geq f(x)$ is equivalent to $\langle D(x - y), x - y \rangle \leq 0$. We suppose to the contrary that $\langle D(x - y), x - y \rangle > 0$. Because D is a Euclidean distance matrix, by Theorem 4 it has exactly one positive eigenvalue. Furthermore, because D is a real symmetric matrix, it is orthogonally diagonalisable. Let $\lambda_1 > 0 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of D, and let $v_1, \ldots, v_n$ be the corresponding eigenvectors, which are normalised and orthogonal. Then, we can express $x - y$ and z on the basis $\{v_1, \ldots, v_n\}$ as follows,

$$x - y = \sum_{i=1}^{n} \alpha_i v_i, \quad z = \sum_{i=1}^{n} \beta_i v_i,$$

for some $\alpha_i, \beta_i \in \mathbb{R}$ $(i = 1, \ldots, n)$. Then,

$$\langle Dz, z \rangle = \sum_{i=1}^{n} \lambda_i \beta_i^2 \geq 0, \tag{4.1}$$

$$\langle Dz, x - y \rangle = \sum_{i=1}^{n} \lambda_i \beta_i \alpha_i = 0, \tag{4.2}$$

$$\langle D(x - y), x - y \rangle = \sum_{i=1}^{n} \lambda_i \alpha_i^2 > 0. \tag{4.3}$$

Because $\lambda_i \leq 0$ $(i = 2, \ldots, n)$, inequality (4.1) and $z \neq 0$ imply that $\beta_1 \neq 0$, and (4.3) implies that $\alpha_1 \neq 0$. Therefore, we can multiply both sides of (4.1) by $\alpha_1^2 > 0$, (4.2) by $-2\alpha_1\beta_1 \neq 0$, and (4.3) by $\beta_1^2 > 0$, and sum up to obtain

$$0 < \left(\lambda_1\beta_1^2\alpha_1^2 + \alpha_1^2\sum_{i=2}^n \lambda_i\beta_i^2\right) - 2\left(\lambda_1\beta_1^2\alpha_1^2 + \alpha_1\beta_1\sum_{i=2}^n \lambda_i\beta_i\alpha_i\right) + \left(\lambda_1\beta_1^2\alpha_1^2 + \beta_1^2\sum_{i=2}^n \lambda_i\alpha_i^2\right)$$
$$= \sum_{i=2}^n \lambda_i(\alpha_1^2\beta_i^2 - 2\alpha_1\beta_1\alpha_i\beta_i + \beta_1^2\alpha_i^2) = \sum_{i=2}^n \lambda_i(\alpha_1\beta_i - \alpha_i\beta_1)^2.$$

The inequality above only holds when there is at least one positive eigenvalue among $\lambda_2, \dots, \lambda_n$, which is a contradiction. Hence, it must hold that $\langle D(x-y), x-y\rangle \le 0$. □

Recall that the Euclidean distance matrix D is *conditionally negative definite*. The next result exploits this fact to replace condition (b) in Lemma 18 with two new conditions.

Lemma 19. *Suppose $x, y \in \mathbb{R}^n$, and there is $z \in \mathbb{R}^n \setminus \{0\}$ such that*

a. $\langle Dz, z\rangle \ge 0$,

b. $\langle Dz, x-y\rangle \le 0$, *and*

c. either $\frac{\sum_{i=1}^n (x_i - y_i)}{\sum_{i=1}^n z_i} \ge 0$, *or* $\sum_{i=1}^n (x_i - y_i) = \sum_{i=1}^n z_i = 0$.

Then, $h(x,y) \ge f(x)$.

Proof. Similar to Lemma 18, $f(x) \le h(x,y)$ is equivalent to $\langle D(x-y), x-y\rangle \le 0$. Let $u = x - y$, and choose $w \in \mathbb{R}^n$ such that

$$w = \alpha z, \qquad \text{where } \alpha = \begin{cases} 1 & \text{if } \sum_{i=1}^n u_i = \sum_{i=1}^n z_i = 0, \\ \frac{\sum_{i=1}^n u_i}{\sum_{i=1}^n z_i} & \text{otherwise.} \end{cases}$$

From (c), $\alpha \ge 0$ and $\sum_{i=1}^n u_i = \sum_{i=1}^n w_i$, or equivalently $\sum_{i=1}^n (u_i - w_i) = 0$. Note that from (a) and (b), we have

$$\langle Dw, w\rangle = \alpha^2 \langle Dz, z\rangle \ge 0, \quad \langle Dw, u\rangle \le 0.$$

Because D is conditionally negative definite, we have $\langle D(w-u), w-u\rangle \le 0$. Combining this with $\langle Dw, w\rangle \ge 0$ and $\langle Dw, u\rangle \le 0$, we get

$$\langle Du, u\rangle = \langle D(w-u), w-u\rangle - \langle Dw, w\rangle + 2\langle Dw, u\rangle \le 0,$$

thus giving $f(x) \le h(x,y)$. □

Using Lemmas 18 and 19, we now establish conditions for when a tangent plane $h(x,y)$ provides an upper approximation for higher value solutions in K, i.e., $h(x,y) \ge f(x)$ for all x such that $f(x) \ge f(y)$.

Theorem 20. *Suppose $x, y \in \mathbb{R}^n_+ \setminus \{0\}$, such that $f(x) \geq f(y)$. Then, $h(x, y) \geq f(x)$ if either*

a. $\sum_{i=1}^n x_i \leq \sum_{i=1}^n y_i$, *or*

b. there is $w \in \mathbb{R}^n_+ \setminus \{0\}$ such that $\langle Dw, x - y\rangle \leq 0$.

Proof. Because $f(x) \geq f(y)$, we have

$$\langle D(x + y), x - y\rangle \geq 0. \tag{4.4}$$

a. Suppose $\sum_{i=1}^n x_i \leq \sum_{i=1}^n y_i$, and choose $z := -(x + y)$. Since D, x and y all have only nonnegative entries, it follows that

$$\langle Dz, z\rangle = \langle -D(x + y), -(x + y)\rangle = \langle D(x + y), x + y\rangle \geq 0.$$

Then from (4.4) we have $\langle Dz, x - y\rangle \leq 0$. Taking into account that $\sum_{i=1}^n x_i \leq \sum_{i=1}^n y_i$ and $x + y \in \mathbb{R}^n_+$, condition (c) in Lemma 19 is fulfilled. Hence, by Lemma 19, the inequality $h(x, y) \geq f(x)$ holds.

b. Suppose there is $w \in \mathbb{R}^n_+ \setminus \{0\}$ such that $\langle Dw, x - y\rangle \leq 0$. Then, given (4.4), there exists a $z \in \mathbb{R}^n_+ \setminus \{0\}$ on the line between w and $x + y$ such that $\langle Dz, x - y\rangle = 0$. Note that D has zero diagonal and positive off-diagonal entries, hence $\langle Dz, z\rangle > 0$. Therefore by Lemma 18, we have that the inequality $h(x, y) \geq f(x)$ holds.

□

4.2.2 Cutting Plane Algorithms

We now introduce two cutting plane algorithms designed to solve (EMSP). Let $A \subset \mathbb{R}^n_+$ denote an arbitrary finite set of points that generate valid tangent planes, such that for all $y \in A$ we have $f(x^*) \leq h(x^*, y)$ where x^* is an optimal solution. Then, we define

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

The cutting plane model of the (EMSP) is then given as the following mixed-integer linear program,

$$\max_{(x,\theta)\in\Gamma_A} \; \theta. \tag{ILP$_A$}$$

Since the points in A generate valid tangents we have $h(x^*, y) \geq f(x^*)$ for all $y \in A$ and hence $(x^*, f(x^*))$ is feasible for (ILP$_A$), meaning the optimal value of (ILP$_A$) is a valid upper bound for (EMSP). We now present two algorithms for solving the (EMSP) that iteratively generate new, valid tangent planes, thereby tightening the approximation of (ILP$_A$). Provided the first cut added is valid, both methods are guaranteed to converge to an optimal solution of the (EMSP). Note that from Theorem 20.a, we can always choose the first cut to be the solution to the maximum cardinality problem. Let y be the solution to $\max_{x \in K} \sum_{i=1}^{n} x_i$, then $y \in K$, $f(x^*) \geq f(y)$ and $\sum_{i=1}^{n} x_i^* \leq \sum_{i=1}^{n} y_i$. Therefore, by Theorem 20.a, y generates a valid tangent.

The first algorithm makes use of the following proposition, which asserts that the tangent plane at the optimal solution of (ILP_A) is always valid.

Proposition 21. *Given $A \subset \mathbb{R}_+^n$ is a set of points that generate valid tangents, let (x, θ) be an optimal solution of the cutting plane problem (ILP_A). Then $f(x^*) \leq h(x^*, x)$, where x^* is an optimal solution of (EMSP).*

Proof. We begin by proving that there is a $y \in A$ such that $\langle Dy, x^* - x \rangle \leq 0$. Suppose, for a contradiction, that for all $y \in A$ we have $\langle Dy, x^* - x \rangle > 0$, or equivalently, $\langle Dy, x^* \rangle > \langle Dy, x \rangle$. Then,

$$\theta \leq h(x, y) < h(x^*, y)$$

holds for all $y \in A$. Let $\hat{\theta}$ be such that,

$$\hat{\theta} = \min_{y \in A} h(x^*, y) > \theta.$$

However, $(x^*, \hat{\theta}) \in \Gamma_A$, and $\hat{\theta} > \theta$, which contradicts (x, θ) being an optimal solution. Hence, the first assertion is settled. The second assertion is a direct consequence of Theorem 20.b, where $w = y \neq 0$ (since otherwise y would not generate a valid cut), and noting that $f(x) \leq f(x^*)$. Hence, $f(x^*) \leq h(x^*, x)$. □

Using this result, we can now solve the (EMSP) by repeatedly solving (ILP_A) to optimality, and using the solutions as new valid tangent planes. An implementation of this approach is shown in Algorithm 7, and its convergence is established in Proposition 22.

Algorithm 7: Repeated (ILP_A) method for solving (EMSP).

```
function RepeatedILP ($f, K, \epsilon$):
    $k \leftarrow 0, UB_k \leftarrow +\infty$
    Take $x^0 \in \arg\max_{x \in K} \sum_{i=1}^{n} x_i$
    Set $A_1 \leftarrow \{x^0\}$, $LB_k \leftarrow f(x^k)$
    while $\frac{UB_k - LB_k}{LB_k} > \epsilon$ do
        $k \leftarrow k + 1$
        Solve ($\text{ILP}_{A_k}$) to obtain $(x^k, \theta^k)$
        $UB_k \leftarrow \theta_k$, $LB_k \leftarrow \max\{LB_{k-1}, f(x^k)\}$
        $A_{k+1} \leftarrow A_k \cup \{x^k\}$
    return $LB_k$
```

Proposition 22. *Algorithm 7 converges to an optimal solution of the (EMSP) in a finite number of steps.*

Proof. As every (ILP_{A_k}) is solved to optimality, we have from Proposition 21 that the tangent plane of every x^k is valid. This implies that $(x^*, f(x^*))$ is always feasible at every step k, i.e., $(x^*, f(x^*)) \in \Gamma_{A_k}$ for all $k \geq 0$. Thus,

$$\text{UB}_k = \max_{(x,\theta)\in\Gamma_{A_k}} \theta \geq f(x^*) = \max_{x\in K} f(x) \geq \text{LB}_k.$$

Because the feasible region K is finite (variables x are discrete and the polyhedral set P is bounded), there is a step k such that the optimal solution (x^k, θ^k) of (ILP_{A_k}) satisfies $x^k \in A_k$. In this case, we have $\text{UB}_k = \theta^k \leq h(x^k, x^k) = f(x^k) \leq \text{LB}_k$, and hence, $\text{UB}_k = \text{LB}_k$. When $\text{UB}_k = \text{LB}_k$, we have $\theta^k = \max_{x\in K} f(x)$, and therefore Algorithm 7 has converged to an optimal solution. □

Note that the repeated (ILP_A) algorithm is similar to the extended cutting plane method presented in Westerlund and Pettersson (1995), with a modification on the first cut added $h(x, x^0)$. The main iterations of each algorithm are the same, as they solve the linearised problem to optimality, and add the solution to the set of cuts. However, while these algorithms are similar, the extended cutting plane algorithm is designed for concave problems. As such, it is generally not applicable to the (EMSP) without modifying the first cut added.

Although Algorithm 7 is globally convergent, it requires solving (ILP_A) to optimality at every iteration, since only the optimal solution is guaranteed to generate a valid cut. Depending on K, this potentially represents a difficult mixed-integer programming problem. We now describe an alternative algorithm in which cuts are added at intermediate feasible points to accelerate convergence.

Recall from Theorem 20.a that $f(x) \leq h(x, y)$ whenever $f(x) \geq f(y)$ and $\sum_{i=1}^n x_i \leq \sum_{i=1}^n y_i$. As such, consider the restriction of the (EMSP) to points with cardinality $c \in \mathbb{N}$. The feasible region of this problem is then given by $K_c = \{x \in K : \sum_{i=1}^n x_i = c\}$. We can solve $\max_{x\in K_c} f(x)$ exactly by instead solving the following linear cutting plane problem,

$$\begin{aligned} \max \quad & \theta && (4.5)\\ \text{s.t.} \quad & \theta \leq h(x, y), \quad \forall y \in K_c, && (4.6)\\ & x \in K_c. \end{aligned}$$

Although there are an exponential number of constraints in (4.6), we can accelerate the solution process by using a branch and cut methodology, whereby cuts are added on the fly during the search procedure.

Given we can solve $\max_{x\in K_c} f(x)$ by the cutting plane problem (4.5), we can therefore decompose (EMSP) such that

$$\max_{x\in K} f(x) = \max_{c=1,\dots,C} \max_{x\in K_c} f(x) = \max_{c=1,\dots,C} \max_{x,\theta} \{\theta : x \in K_c, \theta \leq h(x, y), \forall y \in K_c\}$$

where C is the maximum cardinality achievable in K. Algorithm 8 below executes this decomposition by solving the inner maximisation problem repeatedly for decreasing cardinality c. At each step k (corresponding to cardinality c_k) the algorithm imposes not only the cuts corresponding to points in K_{c_k}, but also additional cuts generated by feasible points with higher cardinality from previous iterations. By Theorem 20.a, for such feasible points y, the cut $\theta \leq h(x, y)$ does not exclude improved solutions with cardinality less than or equal to y, and hence the decomposition remains valid. Line 10 of the algorithm finds the upper bound of the remaining iterations with a lesser cardinality, meaning we do not need to iterate over every possible cardinality. The process repeats until the optimal solution is found.

Algorithm 8: Forced cardinality method for solving (EMSP).

```
1  function ForcedCardinality (f,K,ϵ):
2      $k \leftarrow 0, UB_0 \leftarrow +\infty$
3      Take $x^0 \in \arg\max_{x \in K} \sum_{i=1}^n x_i$
4      $LB_k \leftarrow f(x^k)$
5      $c_1 \leftarrow \sum_{i=1}^n x_i^0,\ A_1 \leftarrow \{x^0\}$
6      while $\frac{UB_k - LB_k}{LB_k} > \epsilon$ and $c_k > 0$ do
7          $k \leftarrow k + 1$
8          Solve $\max_{x,\theta}\{\theta : x \in K_{c_k}, \theta \leq h(x, y), \forall y \in K_{c_k} \cup A_k\}$ for $(x^k, \theta^k)$ using
             branch and cut, saving all cuts found and adding their corresponding
             points to $A_{k+1}$
9          $LB_k \leftarrow \max\{LB_{k-1}, f(x^k)\}$
10         Solve $\max_{(x,\theta) \in \Gamma_{A_{k+1}}}\{\theta : \sum_{i=1}^n x_i \leq c_k - 1\}$ for $UB_k$
11         $c_{k+1} \leftarrow c_k - 1$
12     return $LB_k$
```

Proposition 23. *Algorithm 8 converges to an optimal solution of the (EMSP) in a finite number of steps.*

Proof. Let x^* be an optimal solution of the (EMSP) and suppose $\sum_{i=1}^n x_i^* = c_k$. We will prove two things: at step k, x^k is a solution of (EMSP), and prior to step k, UB_k is an upper bound for the optimal value of (EMSP), ensuring the algorithm does not terminate early. For the first assertion, observe that $\sum_{i=1}^n x_i^* \leq \sum_{i=1}^n y_i$ and $f(x^*) \geq f(y)$ for all $y \in K_{c_k} \cup A_k$. Therefore, by Theorem 20.a, $f(x^*) \leq h(x^*, y)$ for all $y \in K_{c_k} \cup A_k$. Hence, $(x^*, f(x^*))$ is a feasible solution to the subproblem on line 8 of Algorithm 8. Suppose that (x^k, θ^k) is optimal for this subproblem. Then we have that

$$f(x^*) \leq \theta^k \leq h(x^k, x^k) = f(x^k).$$

Therefore x^k obtained at step k must be optimal for (EMSP), and $LB_k = f(x^*)$.

We now prove that the upper bound determined on line 10 of Algorithm 8 is always valid if the optimal solution has not yet been reached, i.e., at iterations $l < k$ where $\sum_{i=1}^n x_i^* = c_k$. At step $l < k$, the set A_{l+1} contains only solutions with cardinality at least c_l. Therefore, $\sum_{i=1}^n x_i^* = c_k \leq c_l - 1 < \sum_{i=1}^n y_i$ and $f(x^*) \geq f(y)$ for all $y \in A_{k+1}$. By Theorem 20.a, this ensures that $f(x^*) \leq h(x^*, y)$, meaning $(x^*, f(x^*))$ is a feasible solution for the subproblem on line 10 of Algorithm 8. Hence, the optimal value of this subproblem UB_l is an upper bound of the globally optimal solution, i.e.,

$$UB_l = \max_{(x,\theta)\in\Gamma_{A_{l+1}}} \left\{ \theta \, : \, \sum_{i=1}^n x_i \leq c_l - 1 \right\} \geq f(x^*).$$

As such, the algorithm does not terminate until a globally optimal solution has been found. □

Note that, the lower bound at each iteration k of Algorithm 8 satisfies

$$LB_k = \max_{c=c_k,\dots,C} \max_{x\in K_c} f(x) = \max_{x\in K} \left\{ f(x) \, : \, \sum_{i=1}^n x_i \geq c_k \right\}. \tag{4.7}$$

In other words, this is the best function value achievable for the current and higher cardinalities. We can show this by induction. For $k = 1$, this statement is clearly true since $A_1 = \{x^0\} \subset K_{c_1}$ meaning that the subproblem on line 8 of Algorithm 8 reduces to problem (4.5), and hence by the arguments given above, x^1 is optimal for $\max_{x\in K_{c_1}} f(x)$ and consequently $LB_1 = \max_{x\in K_{c_1}} f(x)$. Suppose the assertion holds at step $k-1$. Then, for step k, there are two cases to consider;

(i) $\max_{x\in K_{c_k}} f(x) < LB_{k-1}$, or

(ii) $\max_{x\in K_{c_k}} f(x) \geq LB_{k-1}$.

For case (i), given (4.7) holds for $k-1$ we have that

$$\begin{aligned} LB_k &= \max\{LB_{k-1}, f(x^k)\} \\ &= LB_{k-1} \\ &= \max_{c=c_{k-1},\dots,C} \max_{x\in K_c} f(x) \\ &= \max_{c=c_k,\dots,C} \max_{x\in K_c} f(x) \end{aligned}$$

as required. For case (ii), suppose x' is optimal for $\max_{x\in K_{c_k}} f(x)$. Then we must have $f(x') \geq f(y)$ and $\sum_{i=1}^n x_i' \leq \sum_{i=1}^n y_i$ for all $y \in A_k$, and hence from Theorem 20.a, $f(x') \leq h(x', y)$. Therefore, $(x', f(x'))$ is feasible for the linear subproblem on line 8 of Algorithm 8. This implies

$$f(x') \leq \theta^k \leq h(x^k, x^k) = f(x^k).$$

Thus, given (4.7) holds for $k-1$, we have

$$LB_k = \max\left\{LB_{k-1}, f(x^k)\right\} = f(x^k) = \max_{c=c_k,\dots,C} \max_{x\in K_c} f(x)$$

as required.

In difficult instances of the (EMSP), a large number of tangent planes are potentially required to sufficiently approximate the objective function (such as with high-coordinate instances of the diversity problem). To accelerate cut generation, recall that the Euclidean distance matrix is conditionally negative definite, and hence $\langle D(x-y), x-y\rangle \leq 0$ holds for all $x, y \in \mathbb{R}^n$ with $\sum_{i=1}^n (x_i - y_i) = 0$. This implies that we can generate valid cuts even for non-integer y; specifically, $f(x) \leq h(x,y)$ holds for any y (integer or continuous) that has the same cardinality as x. Therefore, Proposition 23 still holds if we modify the subproblem on line 8 of Algorithm 8 to include additional cuts generated by non-integer points with cardinality c_k. These tangents can be generated by solving the linear relaxation, and are therefore computationally cheap to generate and may improve the approximation.

However, using the linear relaxation potentially introduces a large integrality gap, possibly reducing the effectiveness of these cuts. A good strategy to reduce this gap is to ensure additional cuts are generated close to the best-known solution. This is achieved by employing a trust-region methodology, where the continuous solutions are constrained to the region defined by $\left\|x - y^k\right\|_1 \leq \gamma$, where y^k is the current best-known integer solution up to step k and $\gamma \geq 0$ is a given parameter. When $x \in [0,1]^n$, this is easily enforced by the following constraint

$$\sum_{\substack{i=1\\ y_i^k=0}}^{n} x_i + \sum_{\substack{i=1\\ y_i^k=1}}^{n} (1-x_i) \leq \gamma. \tag{4.8}$$

Algorithm 9 outlines the process for generating LP tangents, where ϵ is the maximum allowable relative tolerance between bounds and M is the maximum number of cuts to be added. This algorithm is called during each iteration k, before solving the subproblem on line 8, and its main inputs are A_k and the current best-known solution y^k. Note that if $\gamma = 0$ then the trust region contains only y^k, and hence this is the only solution returned by Algorithm 9. The cuts from Algorithm 9 can then be introduced by replacing the subproblem in line 8 with

$$\max_{x,\theta}\left\{\theta \,:\, x \in K_{c_k}, \theta \leq h(x,y), \forall y \in K_{c_k} \cup A_k \cup L_k\right\},$$

where L_k comes from Algorithm 9. Note that as the points $y \in L_k$ are not necessarily integer feasible, it is not always true that $f(x^*) \geq f(y)$ and hence they do not satisfy the requirements for Theorem 20. As such, the tangents of these solutions may only be used for the current iteration.

Algorithm 9: LP-relaxation cuts for (EMSP).

```
1 function GetLPTangents ($f, K, A_k, c_k, \gamma, y^k, \epsilon, M$):
2     $p \leftarrow 0, UB_0 \leftarrow +\infty, LB_0 \leftarrow f(y^k)$
3     $L \leftarrow \emptyset$.
4     while $\frac{UB_p - LB_p}{LB_p} > \epsilon$ and $p \leq M$ do
5         $p \leftarrow p + 1$
6         Solve the continuous relaxation of
            $\max_{x,\theta}\{\theta : x \in K_{c_k}, \theta \leq h(x, y), \forall y \in A_k \cup L\}$ with trust-region
            constraint (4.8) to obtain $(x^p, \theta^p)$
7         $UB_p \leftarrow \theta_p$, $LB_p \leftarrow \max\{LB_{p-1}, f(x^p)\}$, $L \leftarrow L \cup \{x^p\}$
8     return $L$
```

4.3 Numerical Results

We now present numerical results for Algorithms 7 and 8. These algorithms were implemented in Julia 1.10 using the JuMP mathematical programming package (Lubin et al., 2023) and Gurobi version 11.0 as the mixed-integer linear solver. The branch and cut method in Algorithm 8 utilised the *lazy constraint callback* function, enabling the addition of tangent planes as constraints during the branch and bound procedure. For Algorithm 8, we add LP-tangents by using Algorithm 9 with $\gamma = 0, 0.5n$, or n and with a maximum iteration limit of $M = 100$. Note that when $\gamma = 0$ the trust region contains only the best known solution (labelled y^k in Algorithm 9). This results in four distinct solver configurations.

Our implementation's source code, including the raw results data, can be accessed at https://github.com/sandyspiers/EuclideanMaximisation/tree/v1.0-julia[2]. The results data also includes tests using CPLEX 22.1.1 as the MIP solver, which produce similar results to those reported here. A relative termination tolerance of $\epsilon = 10^{-6}$ was used for the main iterations of Algorithms 7-9, as well as for any mixed-integer subproblems solved by Gurobi. All other mixed-integer programming parameters were set to their defaults. All tests were conducted on a machine with a 2.3 GHz AMD EPYC processor with 64GB RAM, using a single thread.

The performance of the algorithms was evaluated against the well-known Glover linearisation of the objective function. This reformulation was first introduced in Glover

[2]Commit reference `b921170`.

(1975) and is given as

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n-1} w_i, && (4.9)\\
\text{s.t.} \quad & x \in P \cap \{0,1\}^n, \\
& w_i \leq x_i \sum_{j=i+1}^{n} d_{ij}, \quad 1 \leq i \leq n-1, \\
& w_i \leq \sum_{j=i+1}^{n} d_{ij} x_j, \quad 1 \leq i \leq n-1, \\
& w_i \geq 0, \quad 1 \leq i \leq n-1.
\end{aligned}$$

In addition to (4.9), we solve (EMSP) using the mixed-integer quadratic programming solver available within Gurobi.

4.3.1 Capacitated Diversity Problem

We begin by evaluating the performance of the different solution methods for solving the capacitated diversity problem. In this problem, the constraint set P contains only the following knapsack constraint,

$$\sum_{i=1}^{n} c_i x_i \leq b,$$

where $c_i \in \mathbb{R}_+$ $(i = 1, \ldots, n)$, and $\min_{i=1,\ldots,n} c_i \leq b < \sum_{i=1}^{n} c_i$. As such, (EMSP) then becomes the problem of selecting a subset of predefined locations, each with a weight, to maximise the sum of the pairwise distances, while keeping the total weight less than or equal to a given limit. The capacitated diversity problem belongs to the family of diversity problems, which has a wide variety of practical applications, including facility location, social network analysis, and ecological conservation (Lai et al., 2018; Z. Lu et al., 2023; Peiró et al., 2021).

The test instances used are derived from the publicly available `MDPLIB 2.0`[3] test library (Martí et al., 2021). Within this test library, we use the Euclidean instances of the capacitated diversity problem. This includes 10 instances each of sizes 50, 150, and 500. These instances were generated such that the weight of each node was randomly generated in the range $[1, 1000]$, with the capacity set to $b = 0.2 \sum_{i=1}^{n} c_i$ and $b = 0.3 \sum_{i=1}^{n} c_i$, making 60 instances in total.

In addition to the previous publicly available test sets, we randomly generated some larger instances of the capacitated diversity problem. These instances are made up of either 1000, 1500, 2000, 2500, or 3000 nodes, where each node contains either 2, 10, or 20 coordinates. Each coordinate of a location was uniformly randomly generated in

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

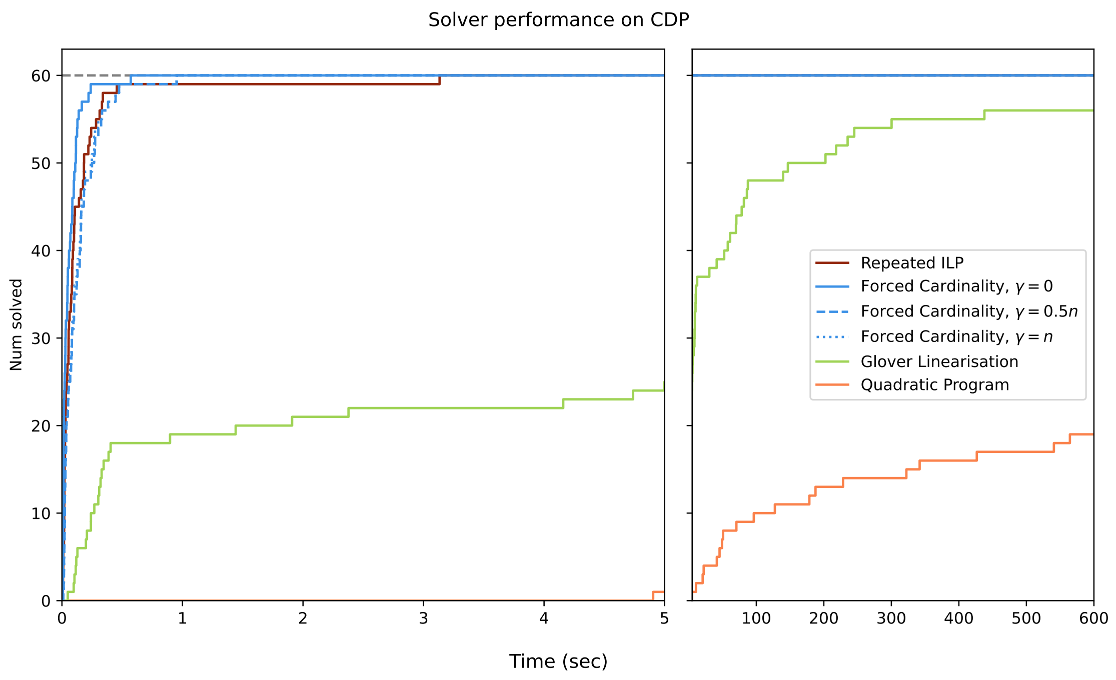

Figure 4.2: Solver performance on the 60 capacitated diversity problem instances available within the `MDPLIB 2.0` test library, using a single thread for computation. The time axis is split at 5 seconds due to marked differences in solver performance.

the range $[0, 100]$. The weight of each node was uniformly randomly generated in the range $[1, 1000]$, and the capacity was set to $b = 0.2\sum_{i=1}^{n} c_i$ or $b = 0.3\sum_{i=1}^{n} c_i$. For every combination of the number of nodes and the number of coordinates, we generated 5 instances, comprising a total of 150 test instances in total.

The performance of various solvers for the benchmark problem instances (labelled CDP) and randomised problem instances (labelled RCDP) over a 600-second time limit is displayed in Figures 4.2 and 4.3, respectively. For CDP test instances, Algorithms 7 and 8 exhibit similar performance, both efficiently solving the entire test set within a maximum of 3.14 seconds. This represents a substantial improvement compared to the other exact methods. While Glover linearisation can solve some of the smaller instances within 5 seconds of run time, it fails to solve the entire problem set within the 600-second time limit. Solving the problem in its original quadratic form proved to be the least effective method by a significant margin, solving fewer than 20 instances. The use of LP-tangents did not appear to improve Algorithm 8 and, in fact, worsened the runtime slightly.

On the larger RCDP test instances, we continue to see impressive performances from both Algorithms 7 and 8. Remarkably, even with the immense size of these instances, the repeated (ILP_A) method was still able to solve all instances in under 5 seconds. We also begin to see a difference in performance between the two algorithms, with the forced cardinality method becoming less effective for these very large problem sizes. That said,

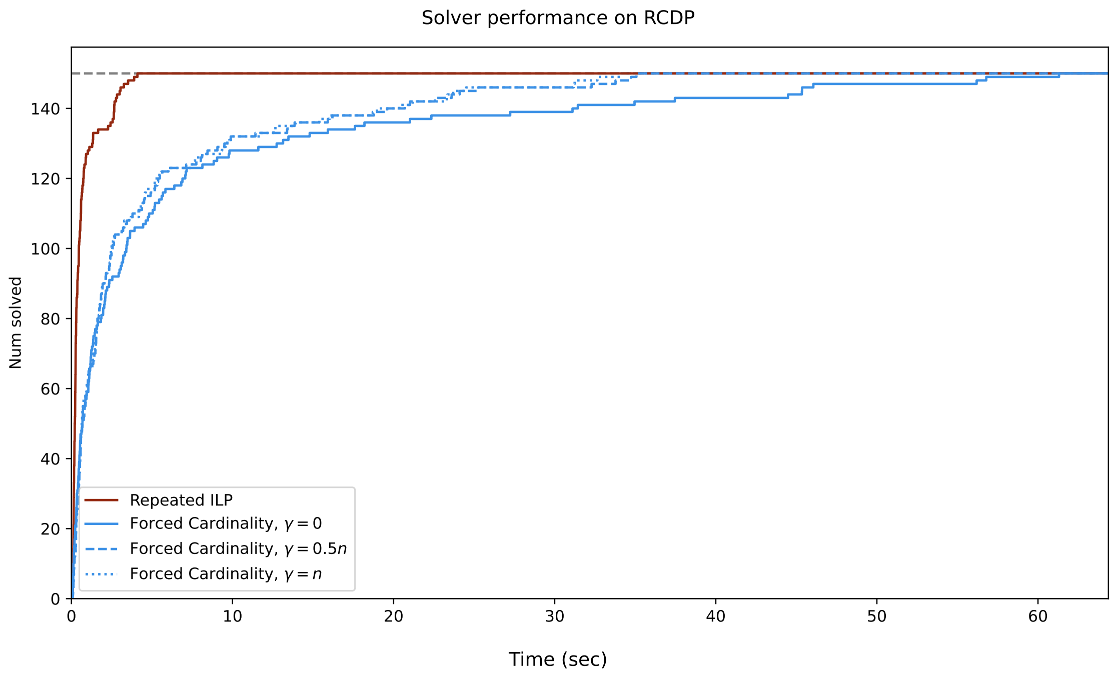

Figure 4.3: Solver performance on the 150 randomly generated capacitated diversity problem instances.

it is still able to solve all instances within 60 seconds of run time.

The results from these tests are further summarised in Table 4.1, which shows the average solve time for each test set, broken down by problem size. It clearly shows how, on the capacitated diversity problem, the performance of Algorithm 7 remains stable for increasing problem size.

4.3.2 Generalised Diversity Problem

The generalised diversity problem (GDP) represents a fundamental optimisation problem in the fields of facility location, supply chain management, and network design (Martinez-Gavara et al., 2021). At its core, the GDP seeks to strategically position a set of facilities on a network to efficiently serve a given demand distribution. This entails optimising not only the allocation of facilities to locations but also considering the spread of these

Type	n	Repeated (ILP$_A$)	Forced Cardinality			Glover Linearisation	Quadratic Programming
			$\gamma = 0$	$\gamma = 0.5n$	$\gamma = n$		
CDP	50	0.21	0.05	0.16	0.15	0.40	196.52
	150	0.08	0.04	0.10	0.10	12.59	600.00
	500	0.15	0.09	0.14	0.14	245.49	600.01
RCDP	1000	0.58	0.64	0.63	0.58	-	-
	1500	0.68	1.93	1.75	1.71	-	-
	2000	0.80	3.88	3.36	3.37	-	-
	2500	0.54	12.48	7.09	6.94	-	-
	3000	0.59	12.55	9.35	9.16	-	-
GDP	50	0.25	0.19	0.14	0.05	0.21	0.13
	150	0.39	0.08	0.11	0.11	4.01	3.67
	500	0.36	0.32	0.39	0.39	36.43	600.01
RGDP	1000	0.80	1.83	1.73	1.69	-	-
	1500	0.93	4.58	4.32	4.29	-	-
	2000	1.22	10.15	7.72	7.62	-	-

Table 4.1: Average solve time in seconds of the various solver setups, broken down by test set and test size. Each problem is solved with a time limit of 600 seconds, using a single thread.

facilities. The max-sum GDP is given as

$$\begin{aligned} \max \quad & f(x) && \text{(GDP-f)} \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_i x_i \geq B, \\ & \sum_{i=1}^{n} a_i x_i \leq K, \\ & x_i \in \{0,1\}, \quad i = 1, \dots, n, \end{aligned}$$

where c_i and a_i represent the capacity and cost of site i. Sites must be chosen such that the minimum demand B is met, and setup cost is kept below the maximum K. The formulation in (GDP-f) considers the capacity to be fixed once a facility is open. A more realistic model considers variable setup costs, where extra capacity can be achieved at a given cost, once the facility is open. The variable cost version of the GDP is given as

$$\begin{aligned} \max \quad & f(x) && \text{(GDP-v)} \\ \text{s.t.} \quad & \sum_{i=1}^{n} t_i \geq B, \\ & \sum_{i=1}^{n} (a_i x_i + b_i t_i) \leq K, \\ & t_i \leq c_i x_i, \quad i = 1, \dots, n, \\ & t_i \in \mathbb{Z}, x_i \in \{0,1\}, \quad i = 1, \dots, n. \end{aligned}$$

We note that (GDP-f) and (GDP-v) were first introduced in Martinez-Gavara et al. (2021) where the objective was to maximise the minimum distance, however, for our purposes we have changed this objective to maximise the sum of pairwise distances.

For the GDP, we again use the Euclidean test instances available within the `MDPLIB 2.0` test library on the (GDP-v) model. All parameters were uniformly randomly generated as follows. The capacity c_i was generated in the range $[1, 1000]$, the fixed cost a_i in the range $[c_i/2, 2c_i]$ and finally the variable cost b_i in $[\min\{1, a_i\}/100, \max\{1, a_i\}/100]$. The minimum capacity is set at either $B = 0.2 \sum_{i=1}^{n} c_i$ or $B = 0.3 \sum_{i=1}^{n} c_i$. Finally, the maximum budget is set as $K = \phi \sum_{i=1}^{n} (a_i + b_i c_i)$, where $\phi = 0.5$ or $\phi = 0.6$. As before, there are 10 instances each of size 50, 150 and 500, making a total of 120 test instances.

To test the solution algorithms on a larger scale, we generated several large instances of GDP-v. These instances were generated similarly to the method described above; however we increased the number of locations to 1000, 1500, and 2000 and generated locations with 2, 10, and 20 sets of coordinates. Furthermore, to reduce the cardinality of the optimal solution and avoid the possibility of full solutions, we decreased the minimum capacity and maximum budget parameters such that $B = 0.05 \sum_{i=1}^{n} c_i$ or $B = 0.1 \sum_{i=1}^{n} c_i$

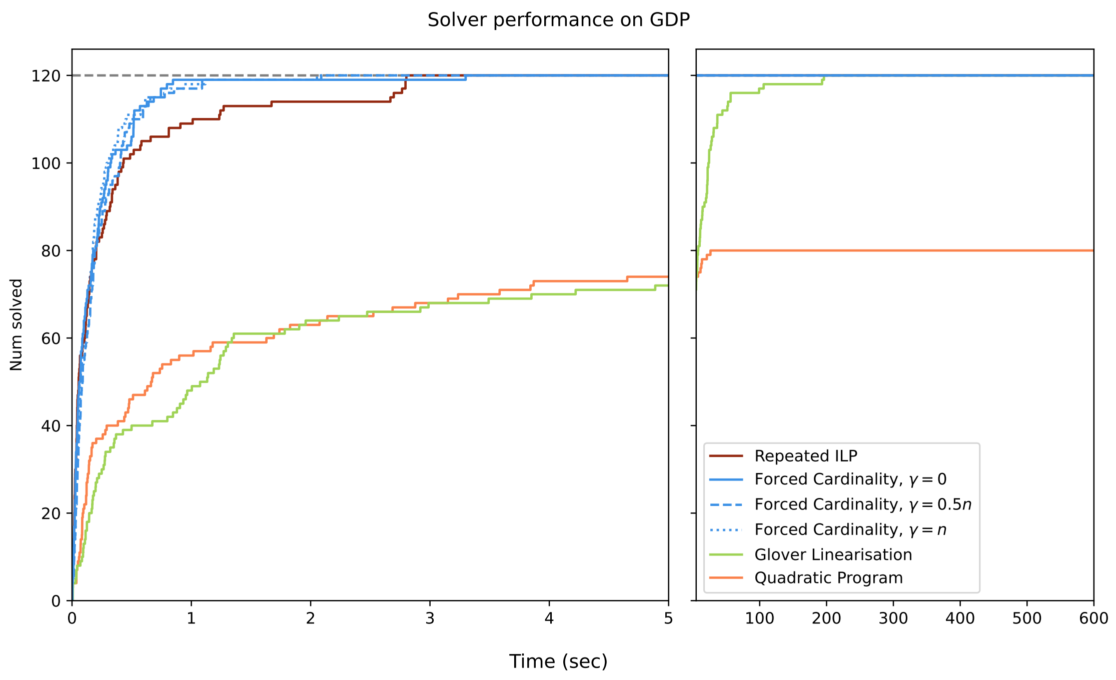

Figure 4.4: Solver performance on the 120 variable cost generalised diversity problem instances within the `MDPLIB 2.0` test library. The time axis is split at 5 seconds due to marked differences in solver performance.

and $K = \phi \sum_{i=1}^{n} (a_i + b_i c_i)$, where $\phi = 0.1$ or $\phi = 0.2$. For every combination of the number of nodes, the number of coordinates, minimum capacity, and maximum budget, we generated 5 instances, comprising a total of 180 test instances in total.

The performance of different solver setups for the benchmark instances (labelled GDP) and random instances (labelled RGDP) over a 600-second time limit is displayed in Figures 4.4 and 4.5, respectively. For GDP instances, Algorithms 7 and 8 exhibit similar performance, both efficiently solving nearly the entire set within 4 seconds. The incorporation of LP-tangent planes for Algorithm 8 has a negligible effect on its solve time. However, both Glover linearisation and the quadratic programming approach find this test set comparatively easier than the capacitated diversity problem, as the Glover linearisation model can solve over half the instances within five seconds and the full set in under 200 seconds. Turning to the results of the RGDP instances shown in Figure 4.5, the repeated (ILP_A) method continues to outperform other solver setups. Moreover, the results suggest that introducing LP-tangent planes can marginally improve Algorithm 8 at large problem sizes. A summary of solve times for these larger instances is provided in Table 4.1.

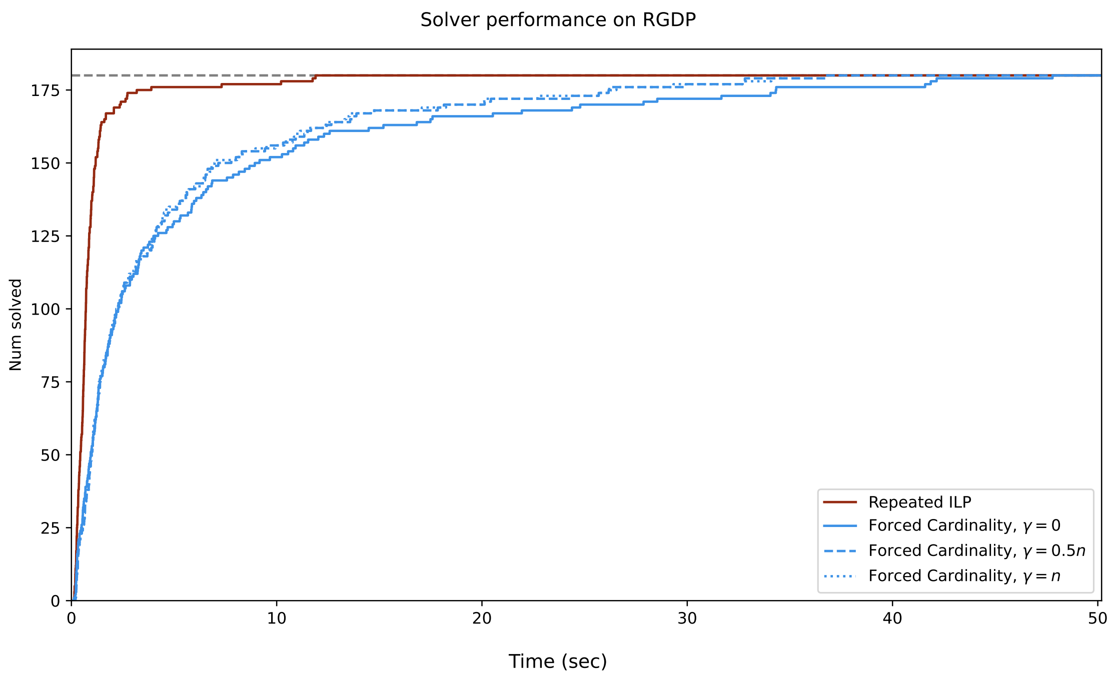

Figure 4.5: Solver performance on the 180 randomised variable cost generalised diversity problem instances.

4.3.3 Cardinality and Cut Strength

To gain a deeper insight into the strength of the cutting planes, we present a breakdown of the number of each type of cut added in Figure 4.6. The figure shows the number of integer cuts (defined by points in A_k) and LP-tangents (defined by points in L_k) across the four solver setups for the CDP, RCDP, GDP, and RGDP test instances. Interestingly, the repeated (ILP_A) method consistently outperforms the forced cardinality method in almost all test sets, despite the latter introducing significantly more cutting planes since all intermediate feasible solutions are used to generate cuts. This suggests that by solving (ILP_A) to optimality, the cut generated provides a very tight approximation of the objective function at the optimal solution. Therefore, in many cases, it is worth taking the extra time to solve the (ILP_A) subproblem to optimality, as the cut generated is expected to be tight. This also explains why the addition of LP-tangent planes does not seem to provide much computational benefit to either approach. As these cuts are generated on the continuous relaxation, they are expected to be even further away from the optimal solution than any integer solution, and hence provide a worse approximation. Tightening the trust region by decreasing γ also seems to have little effect. While LP-tangents are easy to generate and can therefore introduce a large number of cuts, they do not provide a good approximation of the objective function, and hence they do not substantially reduce the number of integer tangents required.

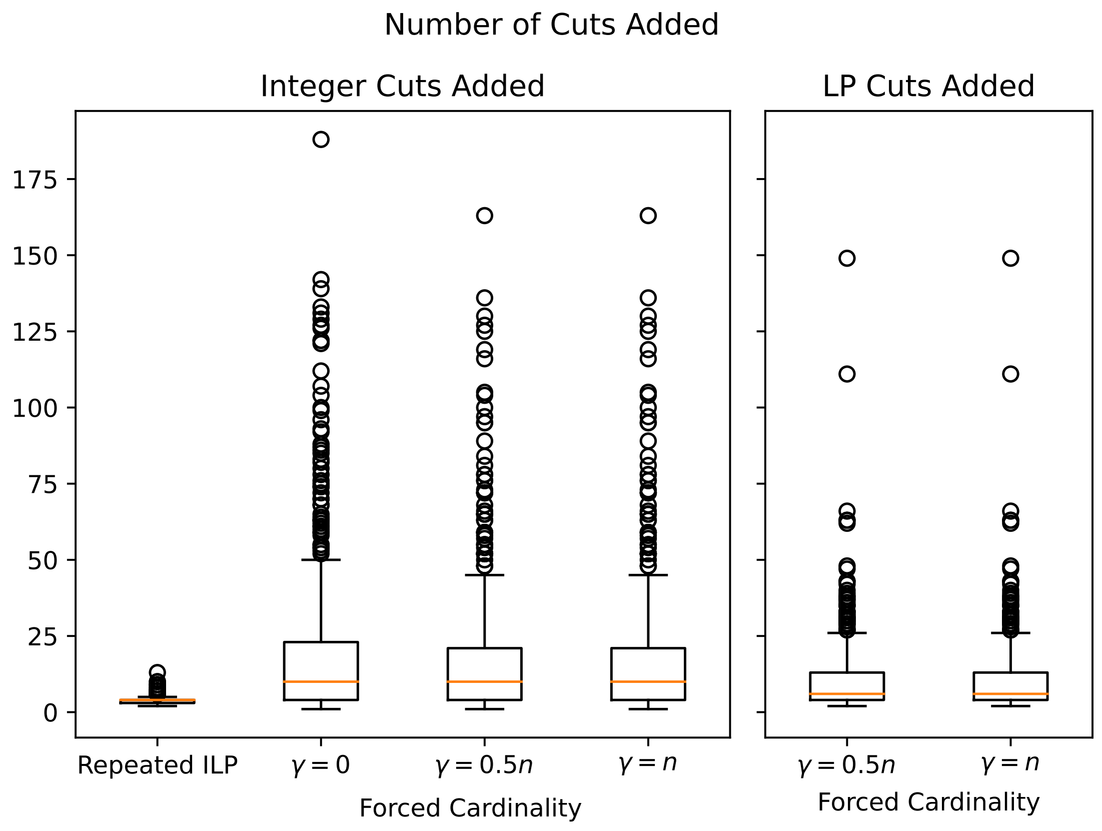

Figure 4.6: Breakdown of the number of integer- and LP-tangents added across the CDP, RCDP, GDP and RGDP test instances.

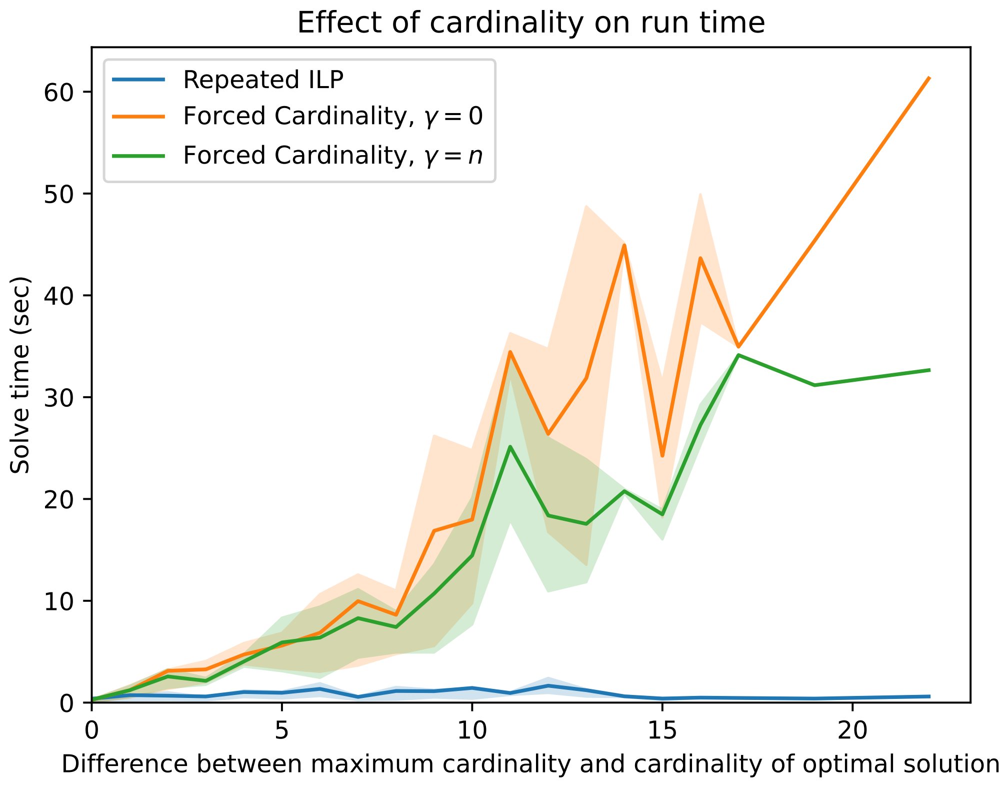

Figure 4.7: Average solve time based on the difference between the maximum cardinality and cardinality of the optimal solution, across the CDP, RCDP, GDP and RGDP test instances. The shaded regions denote the range bounded by the minimum and maximum solve times.

One example of where LP-tangents become highly beneficial is in problems that have a large difference between the maximum cardinality and the cardinality of an optimal solution. In such cases, the forced cardinality approach must solve many iterations before reaching an iteration that contains an optimal solution. Figure 4.7 shows the minimum, average, and maximum solve time at each difference between the maximum cardinality and the cardinality of the found solution across the CDP, RCDP, GDP, and RGDP test instances. The repeated (ILP_A) method is virtually unaffected by this metric, and its average runtime remains steady. However, the forced cardinality method performs substantially worse as this number increases. That said, LP-tangents appear to improve performance by quickly solving earlier iterations, thereby reducing overall solve time. As such, they become fairly beneficial in these cases.

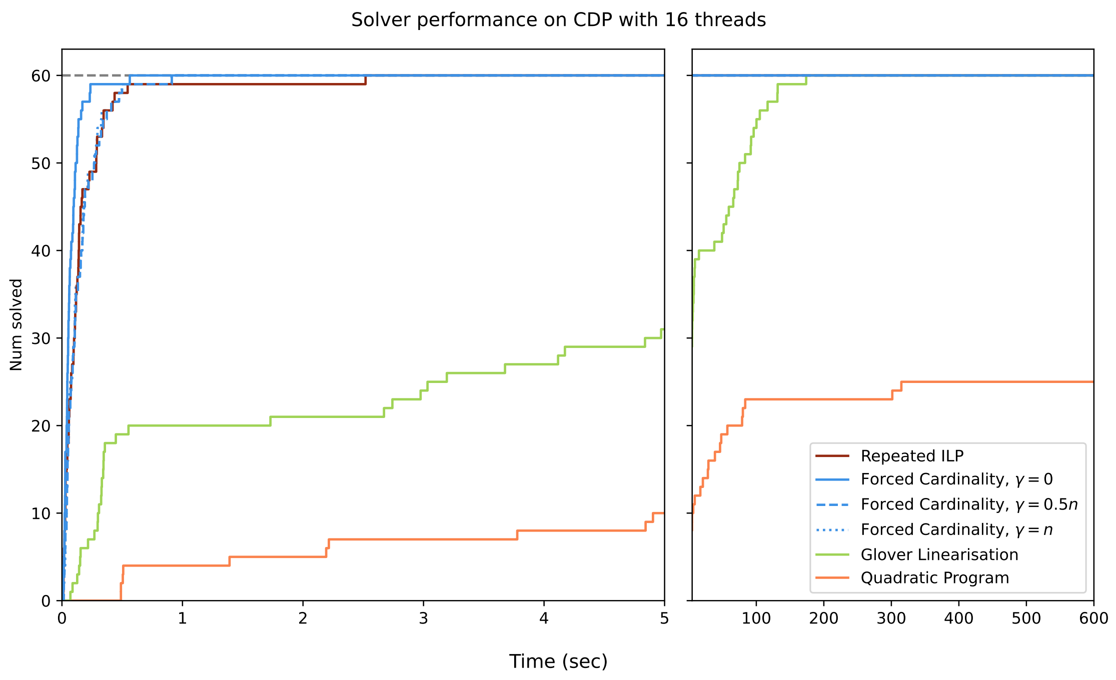

Figure 4.8: Solver performance on the 60 capacitated diversity problem instances available within the `MDPLIB 2.0` test library, using all 16 threads for computation.

4.3.4 Multi-threaded Tests

While all tests mentioned thus far use a single thread for computation to provide a fair test setup, this is rarely required in practice. As such, we now revisit test sets CDP and GDP, allowing the solver to use all 16 available threads. Note that for Algorithms 7-9, the main loop iterations are still single-threaded, but the mixed-integer solver may now use all threads to solve the required subproblems.

The results on sets CDP and GDP are shown in Figures 4.8 and 4.9 respectively. Given their already short runtimes, the performances of Algorithms 7 and 8 do not improve substantially. This is partly explained by the fact that each subproblem is easy to solve, and hence does not benefit greatly from parallelism. For Glover linearisation, the performance difference within the first 5 seconds is marginal, solving only a few extra instances in each case. However, after this time frame, the solver benefits greatly and sees marked improvements, especially on the CDP instances. Finally, the quadratic programming approach benefits the most from parallelism, allowing it to perform comparably with Glover linearisation on the GDP instances. That said, the cutting plane algorithms remain the best performers on each test set.

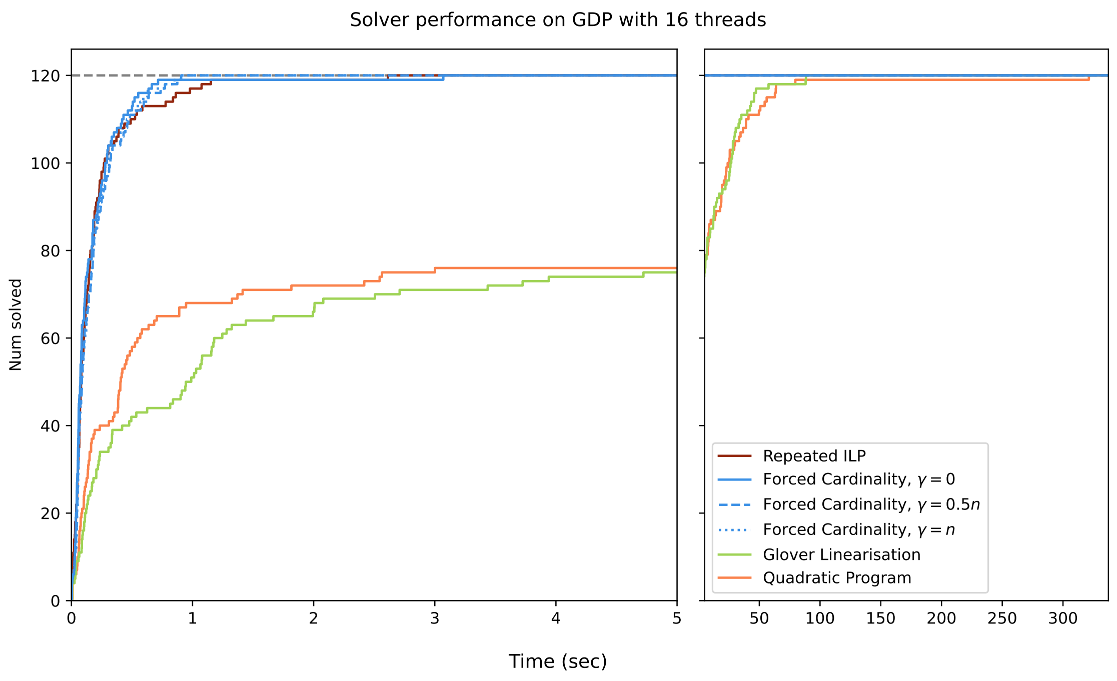

Figure 4.9: Solver performance on the 120 generalised diversity problem instances available within the `MDPLIB 2.0` test library, using all 16 threads for computation.

4.3.5 Max-sum Diversity Problem

We finish this section by looking at difficult instances of the max-sum diversity problem. This is similar to the capacitated diversity problem visited earlier, except the knapsack constraint is replaced by the following cardinality constraint,

$$\sum_{i=1}^{n} x_i = p.$$

The problem has many real-world applications and fits the structure of (EMSP). While this problem can be efficiently solved using the cutting plane approaches of the previous chapters, we can use this problem to test the robustness of Algorithms 7 and 8. Notably, we showed in Chapter 2 that instances with a large number of coordinates are particularly difficult to approximate by cutting planes, thereby resulting in poor performance. As such, we use the test instances in set `GKD-c` of `MDPLIB2.0`. These 20 instances each contain 500 locations with 20 coordinates, and where $p = 50$.

Table 4.2 shows the average gap as a percentage, average objective value, average number of integer cuts added, and the number of problems solved to optimality on the `GKD-c` test set over a 600-second time limit. While Algorithm 7 is only able to solve 3 out of 20 instances, the average final gap is very small at just 0.07%. Furthermore, it is able to achieve this gap with an average of only 141 cuts. In contrast, the forced cardinality approach is only able to solve a single instance to optimality. For the remaining instances,

	Repeated (ILP$_A$)	Forced Cardinality			Glover Linearisation	Quadratic Programming
		$\gamma = 0$	$\gamma = 0.5n$	$\gamma = n$		
Ave Gap (%)	0.07	∞	∞	∞	115.59	702.83
Ave Objective Value	19500.83	19490.72	19482.78	19482.78	18985.52	19501.70
Ave Number Integer Cuts	141.45	3820.10	4015.80	4001.40	-	-
Number Solved	3	1	0	0	0	0

Table 4.2: Solver performance on test set `GKD-c` using a single thread over a 600-second time limit.

the algorithm is never able to reach its upper bounding subproblem (line 10, Algorithm 8) as it times out beforehand. Consequently, the algorithm never determines a valid upper bound. This represents a major shortcoming of this approach. That said, it can still achieve a decent lower bound, close to that of Algorithm 7. Interestingly, the standard quadratic programming approach achieved the best average objective value and yet a very poor objective bound. This is possibly indicative of the fact that the problem is inherently nonconcave, and hence Gurobi is most likely relying on over-estimators to provide objective bounds. However, these bounds appear to perform very poorly and result in a very large optimality gap.

4.4 Conclusion

In this chapter, we presented two exact cutting plane algorithms for the general Euclidean distance maximisation problem. We establish the validity of tangents by introducing the concept of *directional concavity*. This notion led to the formulation of two important sufficient conditions for valid cuts, shown in Theorem 20. Two cutting plane solution algorithms were then introduced. The algorithms exploit Theorem 20 to ensure the search for the optimal solution always stays on a concave direction of the objective function, therefore ensuring all cuts are valid. This was achieved by either repeatedly solving the cutting plane subproblem to optimality, or by iteratively forcing and decreasing the cardinality of the problem.

Extensive numerical experiments were conducted to test the suggested solution algorithms. The results are very promising, with all proposed methods easily able to solve capacitated diversity problem instances with 3000 locations in under 60 seconds. This represents a significant improvement compared to other exact methods for the (EMSP). The repeated (ILP$_A$) method appeared to be the best overall performer as it generates tight cuts and remains fairly stable for increasing dimensions. Additionally, the approach is still able to provide a good upper bound even for very difficult instances, unlike the forced cardinality approach. Therefore, the choice of which approach to use should depend on the specific problem structure, especially the expected difference between the maximum cardinality and the cardinality of an optimal solution.

5 Directionally Concave Branch and Cut

We now combine the directionally concave cutting plane techniques of the previous chapters into a unified algorithm to solve general mixed-integer quadratic programming problems. The approach uses functional decomposition to create objective components, each with at most one positive eigenvalue. Using an extension of the results from the previous chapter, we formulate directionally concave upper planes of each functional component using a combination of tangent planes and convex over-envelopes. The resultant branch and cut algorithm is globally convergent, and extensive numerical experiments demonstrate the promising performance of this approach.

5.1 Introduction

In this chapter, we develop a novel, exact branch and cut algorithm to solve general quadratic programming problems of the form,

$$\begin{aligned} z \;=\; \max \quad & f(x) = \langle Qx, x\rangle + \langle p, x\rangle \\ \text{s.t.} \quad & x \in K \subset \mathbb{R}^n. \end{aligned} \tag{5.1}$$

Here, $Q \in \mathbb{R}^{n\times n}$ represents a square, symmetric matrix with exactly m strictly positive eigenvalues, $p \in \mathbb{R}^n$ is a vector, and K is a bounded subset of $\mathbb{R}^n$ that may contain both integer and continuous dimensions.

As discussed in Chapter 1, problems of this form have many important practical applications in a wide range of fields. However, they are generally very challenging to solve, particularly when the objective function is nonconcave. Throughout the previous chapters, we exploited specific characteristics of the matrix Q to solve the problem via cutting plane methods. For instance, Chapter 2 demonstrated how restricting the feasible domain, such as with a cardinality constraint, can ensure valid tangents. Additionally, Chapter 3 discussed how functional decomposition can help form tighter cuts, and therefore improve the linear approximation. Finally, the previous chapter introduced the novel concept of directional concavity, and began exploring ways to assert whether a function is concave on a given $x - y$ direction.

Based on these insights, this chapter aims to combine and extend the results from earlier chapters to develop a sophisticated branch and cut algorithm for (5.1). To achieve this, we introduce a new directionally concave tangent plane, based off the results from the previous chapter. Originally, directional concavity was proved on only Euclidean distance matrices by exploiting the property of having a single positive eigenvalue. Here, we expand these results to encompass general matrices Q. Furthermore, we provide sufficient conditions that can identify regions where a tangent plane is known to be valid. These conditions are practical and useful in an algorithmic setting. By combining directionally concave tangent planes with over-envelopes, we can formulate a tight upper approximation of the objective function. This forms the basis of our exact branch and cut approach.

Formulating directionally concave tangent planes requires using eigenvalue decomposition to identify regions of concavity and convexity within Q. While eigenvalue decomposition is a well-established tool in quadratic programming, it is typically used to partition the search space into fully concave and convex components. For example, Bomze (2002) use eigenvalue decomposition to formulate the objective function of the standard quadratic programming problem as a difference-of-convex function, i.e., $Q = P - S$, where both P and S are positive semidefinite. The authors then used a branch and bound algorithm to converge on a local solution. Similarly, in C. Lu et al. (2017), the authors use eigenvalue decomposition to derive a tight semidefinite relaxation for the nonconcave part of a quadratic constraint. This relaxation is then integrated into a branch and bound algorithm, where branching directions are guided by the eigenvalues of the functional components.

The novelty of our approach lies in decomposing the problem into nonconcave functional components. Being nonconcave, these components have both positive and negative eigenvalues. We then apply our directionally concave tangents, enabling us to leverage the efficiency of cutting plane methods even for nonconcave functions.

5.2 A Directionally Concave Branch and Cut Algorithm

In this section, we introduce our novel directionally concave branch and cut algorithm, designed to solve quadratic programming problems of type (5.1). Key to our approach is the use of directionally concave upper planes, used to approximate the nonconcave objective function. These planes are generated through the use of auxiliary vectors that assert whether the tangent plane of y is valid at an x. Ensuring the validity of these tangents is challenging however, especially when Q contains many positive eigenvalues. Recognising these difficulties, we look at the special case of a single positive eigenvalue, and show it is possible to formulate a tight upper approximation of $f(x)$ using a combination of directionally concave tangent planes and convex over-envelopes. By decomposing Q into a set of matrices each with a single positive eigenvalue, we can use these upper approximations to formulate a globally convergent branch and cut algorithm.

5.2.1 Directional Notation

Before commencing the main analysis, let us begin by introducing some notation that will be used throughout the chapter. Let

$$\lambda_1 \geq \cdots \geq \lambda_m > 0 \geq \lambda_{m+1} \geq \cdots \geq \lambda_n$$

and $v_1, \dots, v_n$ denote the eigenvalues and eigenvectors of the matrix. Note that we assume Q has exactly m strictly positive eigenvalues. We can write Q on the basis of its eigenvectors such that $Q = \sum_{i=1}^n \lambda_i v_i^T v_i$. Consider the matrices Q^+ and Q^-, defined by

$$Q^+ = \sum_{i=1}^{m} \lambda_i v_i^T v_i, \qquad Q^- = \sum_{j=m+1}^{n} \lambda_j v_j^T v_j.$$

Then $Q = Q^+ + Q^-$. Furthermore, Q^+ is positive definite as it has only positive eigenvalues, and similarly Q^- is negative semi-definite. As such, we can think of Q^+ and Q^- as representing the *convex* and *concave* functional components, respectively.

Given eigenvectors are orthonormal, we can express any vector $x \in \mathbb{R}^n$ using the basis of eigenvectors such that $x = \sum_{i=1}^n \alpha_i v_i$, where $\alpha_1, \dots, \alpha_n \in \mathbb{R}$. Let us now extend our new directional notation by defining

$$x^+ = \sum_{i=1}^{m} \alpha_i v_i, \qquad x^- = \sum_{j=m+1}^{n} \alpha_j v_j.$$

Then, analogous to before, $x = x^+ + x^-$. This notation allows us to consider the convex and concave components of a vector. The following remark summarises some well-known matrix and vector properties using this new notation.

Remark 24. Let Q be defined as above, and let $x, y \in \mathbb{R}^n$. Then the following holds,

1. $\langle x, y \rangle = \langle x^+, y^+ \rangle + \langle x^-, y^- \rangle$,
2. $(Qx)^+ = Q^+ x^+$, and similarly $(Qx)^- = Q^- x^-$,
3. $\langle Qx, y \rangle = \langle Q^+ x^+, y^+ \rangle + \langle Q^- x^-, y^- \rangle$, and
4. $\langle Q^+ x^+, x^+ \rangle = 0$ if and only if $x^+ = \mathbf{0}$.

Proof. Let $x = \sum_{i=1}^n \alpha_i v_i = x^+ + x^-$ and $y = \sum_{i=1}^n \beta_i v_i = y^+ + y^-$. The proofs of each property are shown below,

1. Observe $\langle x, y\rangle = \left\langle x^+ + x^-, y^+ + y^-\right\rangle = \left\langle x^+, y^+\right\rangle + \left\langle x^+, y^-\right\rangle + \left\langle x^-, y^+\right\rangle + \left\langle x^-, y^-\right\rangle$. Furthermore, $\left\langle x^+, y^-\right\rangle = \left\langle \sum_{i=1}^m \alpha_i v_i, \sum_{j=m+1}^n \beta_j v_j \right\rangle = \sum_{i=1}^m \sum_{j=m+1}^n \alpha_i \beta_j \left\langle v_i, v_j\right\rangle = 0$, and analogously, $\left\langle x^-, y^+\right\rangle = 0$. Hence, $\langle x, y\rangle = \left\langle x^+, y^+\right\rangle + \left\langle x^-, y^-\right\rangle$.

2. Firstly, consider that

$$Qx = \left(\sum_{i=1}^n \lambda_i v_i^T v_i\right) x = \sum_{i=1}^n \lambda_i \begin{pmatrix} v_{i1}v_{i1} & \cdots & v_{i1}v_{in} \\ \vdots & \ddots & \vdots \\ v_{in}v_{i1} & \cdots & v_{in}v_{in} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$
$$= \sum_{i=1}^n \lambda_i \begin{pmatrix} v_{i1} \sum_{j=1}^n v_{ij}x_j \\ \vdots \\ v_{in} \sum_{j=1}^n v_{ij}x_j \end{pmatrix} = \sum_{i=1}^n \lambda_i \left\langle v_i, x\right\rangle v_i,$$

and hence,

$$(Qx)^+ = \left(\sum_{i=1}^n \lambda_i \left\langle v_i, x\right\rangle v_i\right)^+ = \sum_{i=1}^m \lambda_i \left\langle v_i, x\right\rangle v_i = \left(\sum_{i=1}^m \lambda_i v_i^T v_i\right) x = Q^+ x.$$

Furthermore, for $i = 1, \dots, m$, we have $\left\langle v_i, x\right\rangle = \left\langle v_i, x^+ + x^-\right\rangle = \left\langle v_i, x^+\right\rangle$ and consequently

$$(Qx)^+ = \sum_{i=1}^m \lambda_i \left\langle v_i, x\right\rangle v_i = \sum_{i=1}^m \lambda_i \left\langle v_i, x^+\right\rangle v_i = \left(\sum_{i=1}^m \lambda_i v_i^T v_i\right) x^+ = Q^+ x^+.$$

The proof of $(Qx)^- = Q^- x^-$ is analogous.

3. From 1. and 2., $\langle Qx, y\rangle = \left\langle (Qx)^+, y^+\right\rangle + \left\langle (Qx)^-, y^-\right\rangle = \left\langle Q^+x^+, y^+\right\rangle + \left\langle Q^-x^-, y^-\right\rangle$.

4. We first prove the forward statement. Observe that $\left\langle Q^+x^+, x^+\right\rangle = \sum_{i=1}^m \lambda_i \alpha_i^2 = 0$. Given $\lambda_i > 0$ by definition, this holds only if $\alpha_i = 0$ for $i = 1, \dots, m$, and therefore $x^+ = \mathbf{0}$. The reverse holds trivially.

□

This new notation, and results from the previous remark, allow us to view Q as a difference-of-convex function. Furthermore, it shows how the function value $f(x)$ depends on x's projection into the convex and concave subspaces, and its contribution in each. These results help in building an understanding of how Q operates on x.

5.2.2 Directional Concavity

At the heart of our branch and cut algorithm is the concept of *directional concavity*. Directional concavity asserts whether a particular direction of a nonconcave function is known to be concave. In other words, the $u = x - y$ direction is concave if

$$f(x) - h(x, y) = \langle Q(x - y), x - y\rangle = \langle Qu, u\rangle \leq 0$$

holds, where $h(x, y)$ denotes the tangent plane of f at y. The following results establish this concept, and formulate sufficient conditions based on the existence of some auxiliary vectors. The existence of these vectors helps to determine whether $x - y$ is a concave direction of Q.

Lemma 25. *Let Q, x, y be defined as above. If there exists a $w \in \mathbb{R}^n$ such that*

1. $\langle Qw, w\rangle \geq 0$,
2. $\langle Qw, x - y\rangle = 0$, *and*
3. $w^+ = (x - y)^+$,

then $f(x) \leq h(x, y)$.

Proof. Firstly, observe that $f(x) \leq h(x, y)$ is equivalent to $\langle Q(x - y), x - y\rangle \leq 0$ and hence it suffices to prove $\langle Qz, z\rangle \leq 0$ where $z = x - y$. Aggregating condition 1 with -2 times condition 2 yields

$$0 \leq \langle Qw, w\rangle - 2\langle Qw, z\rangle = \langle Qw, w\rangle - 2\langle Qw, z\rangle + \langle Qz, z\rangle - \langle Qz, z\rangle\,,$$

which can be factorised and rearranged to achieve

$$\langle Qz, z\rangle \leq \langle Q(w - z), w - z\rangle\,.$$

Then from Remark 24.6 we have that

$$\langle Qz, z\rangle \leq \left\langle Q^+(w - z)^+, (w - z)^+\right\rangle + \langle Q^-(w - z)^-, (w - z)^-\rangle\,.$$

However from condition 3, $\mathbf{0} = w^+ - z^+ = (w - z)^+$ and hence

$$\left\langle Q^+(w - z)^+, (w - z)^+\right\rangle = \left\langle Q^+\mathbf{0}, \mathbf{0}\right\rangle = 0.$$

Finally, given Q^- is negative semi-definite it follows that

$$\langle Q^-(w - z)^-, (w - z)^-\rangle \leq 0.$$

Therefore, $\langle Qz, z\rangle \leq 0$ as required. □

Theorem 26. *Let $u_1, \ldots, u_k \in \mathbb{R}^n$ be a set of k vectors that all satisfy conditions 1 and 2 of Lemma 25. If*

1. $\left\langle Qu_i, u_j\right\rangle = 0$ *for all $i \neq j$, and*
2. $(x - y)^+ \in span\{u_1^+, \ldots, u_k^+\}$

then $f(x) \leq h(x, y)$.

Proof. If $(x-y)^+ \in \operatorname{span}\{u_1^+, \dots, u_k^+\}$ then there exists $\alpha_1, \dots, \alpha_k \in \mathbb{R}$ such that $(x-y)^+ = \sum_{i=1}^k \alpha_i u_i^+$. Suppose $w = \sum_{i=1}^k \alpha_i u_i$, then we have that

1. $\langle Qw, w\rangle = \sum_{i=1}^k \sum_{j=1}^k \alpha_i \alpha_j \langle Qu_i, u_j\rangle = \sum_{i=1}^k \alpha_i^2 \langle Qu_i, u_i\rangle \geq 0$,
2. $\langle Qw, x-y\rangle = \sum_{i=1}^k \alpha_i \langle Qu_i, x-y\rangle = 0$, and
3. $w^+ = \sum_{i=1}^k \alpha_i u_i^+ = (x-y)^+$.

Therefore w satisfies conditions 1-3 of Lemma 25 and hence $f(x) \leq h(x,y)$. □

Corollary 27. *Let $u_1, \dots, u_m \in \mathbb{R}^n$ be a set of m vectors that all satisfy conditions 1 and 2 of Lemma 25 and condition 1 of Theorem 26. If $u_1^+, \dots, u_m^+$ are linearly independent and nonzero then $f(x) \leq h(x,y)$.*

Proof. If $u_1^+, \dots, u_m^+$ are linearly independent and nonzero then

$$\operatorname{span}\{u_1^+, \dots, u_m^+\} = \operatorname{span}\{v_1^+, \dots, v_m^+\}$$

and hence we always have $(x-y)^+ \in \operatorname{span}\{u_1^+, \dots, u_m^+\}$. Therefore, by Theorem 26, $f(x) \leq h(x,y)$. □

Theorem 26 extends the principles of directional concavity, initially proposed in Chapter 4, to matrices with any number of positive eigenvalues. This is achieved by identifying sufficiently many auxiliary vectors that ensure the existence of a vector w that fulfils the conditions of Lemma 25. Interestingly, Theorem 26 reveals that finding m helper vectors is not always necessary. The key is that one only needs to find sufficiently many such that Lemma 25 holds. If K exhibited some special structure that meant $(x-y)^+$ was restricted in some predetermined way, then this result may be particularly helpful in identifying directionally valid tangents. This information is not often known in advance, and hence in many cases Corollary 27 becomes more useful. Even so, finding m vectors that satisfy condition 1 of Theorem 26 is challenging in its own right, particularly when m is large. We can avoid this challenge altogether by considering the special case where $m = 1$ (thereby making this condition obsolete), and using the following sufficient conditions.

Proposition 28. *If there exists a $u \in \mathbb{R}^n$ such that*

1. $\langle Qu, u\rangle > 0$,
2. $\langle Qu, x-y\rangle \leq 0$, *and*
3. $\langle v_i, u\rangle \langle v_i, x-y\rangle \geq 0$

for some $i \in \{1, \dots, m\}$, then there exists a $w \in \mathbb{R}^n$ such that

4. *$\langle Qw, w\rangle > 0$, and*

5. *$\langle Qw, x-y\rangle = 0$*

Proof. The first case to consider is when $\langle v_i, x-y\rangle = 0$. In this case, simply let $w = v_i$, then $\langle Qw, w\rangle = \lambda_i > 0$ and $\langle Qw, x-y\rangle = \lambda_i \langle v_i, x-y\rangle = 0$ as required. Alternatively, assume that $\langle v_i, x-y\rangle \neq 0$ and let $w = u + \alpha v_i$. From condition 5,

$$\langle Qw, x-y\rangle = \langle Qu, x-y\rangle + \langle Q\alpha v_i, x-y\rangle = \langle Qu, x-y\rangle + \alpha\lambda_i \langle v_i, x-y\rangle = 0$$

and hence let

$$\alpha = \frac{-\langle Qu, x-y\rangle}{\lambda_i \langle v_i, x-y\rangle}.$$

We now show that if α satisfies the above, then condition 4 also holds. In other words, it suffices to show

$$\langle Qw, w\rangle = \langle Qu, u\rangle + 2\alpha\langle Qv_i, u\rangle + \alpha^2 \langle Qv_i, v_i\rangle = \langle Qu, u\rangle + 2\lambda_i\alpha\langle v_i, u\rangle + \alpha^2\lambda_i \tag{5.2}$$

is positive. Note that from condition 3 and the fact that $\langle v_i, x-y\rangle \neq 0$ we have $\frac{\langle v_i, u\rangle}{\langle v_i, x-y\rangle} \geq 0$. Furthermore, given $\lambda_i > 0$ and $-\langle Qu, x-y\rangle \geq 0$ it follows that

$$\alpha\langle v_i, u\rangle = \frac{-\langle Qu, x-y\rangle}{\lambda_i \langle v_i, x-y\rangle}\langle v_i, u\rangle \geq 0.$$

Therefore from (5.2) we have $\langle Qw, w\rangle > 0$ as required and hence w satisfies conditions 4 and 5. □

Proposition 29. *Suppose Q has exactly one positive eigenvalue, i.e., $m = 1$ and let $x, y \in \mathbb{R}^n$. If there exists a $u \in \mathbb{R}^n$ that satisfies conditions 1-3 of Proposition 28, then $f(x) \leq h(x, y)$.*

Proof. Given x, y, u satisfy Proposition 28, there exists a z such that $\langle Qz, x-y\rangle = 0$ and $\langle Qz, z\rangle > 0$. Note that $\langle Qz, z\rangle > 0$ implies $z^+ \neq \mathbf{0}$. Therefore, by Corollary 27 and the fact that $m = 1$, we have $f(x) \leq h(x, y)$ as required. □

Note that condition 3 of Proposition 28 simply ensures that $\langle v_i, u\rangle$ and $\langle v_i, x-y\rangle$ have the same sign, with the inclusion of the case where either (or both) is zero. In other words, it ensures that u and $x-y$ are in the same relative direction with respect to v_i.

While Proposition 28 does not specifically require $m = 1$, it becomes particularly useful in this case. Furthermore, finding realistic candidates for this proposition is far easier than for Theorem 26, as we require only a single vector, and may settle for an inequality (rather than the equality of condition 2 of Lemma 25). This makes the result very attractive for algorithmic use, offering a simpler and more direct approach to directionally concavity.

5.2.3 One-Positive Matrices

We now show how, when Q has one positive eigenvalue, we can combine directionally concave tangent planes with convex over-envelopes to formulate a tight upper approximation of $f(x)$. For the remainder of this subsection, assume Q has exactly one positive eigenvalue, and let $f^+(x) = \langle Q^+x, x\rangle = \lambda_1 \langle v_1, x\rangle^2$ and $f^-(x) = \langle Q^-x, x\rangle$. Furthermore, let $h^+(x, y)$ and $h^-(x, y)$ be defined similarly. Finally, for the sake of brevity, let $\lambda = \lambda_1$ and $v = v_1$ denote the single positive eigenvalue and associated eigenvector.

Given a suitable $u \in \mathbb{R}^n$ with $\langle Qu, u\rangle > 0$, Proposition 29 can be used to generate a directionally concave tangent, as well as define a region where this tangent is known to provide a valid upper approximation. Outside of this region, however, we cannot be certain whether or not the tangent remains valid. Instead, we can use an over-envelop of the convex component $f^+(x) = \lambda \langle v, x\rangle^2$, and use this to approximate the region not covered by the tangent.

Lemma 30. *Let $l, t, u \in \mathbb{R}$ be such that $l \leq t \leq u$. Then it follows that $t^2 \leq O(t, l, u)$ where*

$$O(t, l, u) = t(l + u) - lu.$$

Proof. Let $t = \alpha l + (1 - \alpha)u$ where $\alpha \in [0, 1]$. Then

$$t^2 \leq \alpha l^2 + (1 - \alpha)u^2 = \alpha\left(l^2 - u^2\right) + u^2 = \alpha(l + u)(l - u) + u^2.$$

Substituting $\alpha = \frac{t-u}{l-u}$ yields

$$t^2 \leq \frac{t - u}{l - u}(l + u)(l - u) + u^2 = (t - u)(l + u) + u^2 = t(l + u) - lu,$$

as required. □

Observe that the u defined in Proposition 29 is not strictly dependent on x or y. As such, we can provide multiple candidates for u in order to define a larger region. The following result uses two candidates to define a region of directionally concave tangent planes, and uses over-envelopes to approximate the remainder.

Proposition 31. *Let $x, y \in \mathbb{R}^n$ be such that $v^l \leq \langle v, x\rangle \leq v^u$ for some $v^l, v^u \in \mathbb{R}$. Furthermore, suppose $u, w \in \mathbb{R}^n$ are such that $\langle Qu, u\rangle > 0$, $\langle Qw, w\rangle > 0$, and $\langle v, u\rangle \langle v, w\rangle \geq 0$. Then we have that*

$$f(x) \leq \begin{cases} h(x, y), & \text{if } \langle Qu, x - y\rangle \langle Qw, x - y\rangle \leq 0, \\ h^-(x, y) + \lambda O\big(\langle v, x\rangle, \langle v, y\rangle, v^u\big), & \text{else if } \langle v, x - y\rangle \geq 0, \\ h^-(x, y) + \lambda O\big(\langle v, x\rangle, v^l, \langle v, y\rangle\big), & \text{else if } \langle v, x - y\rangle < 0. \end{cases} \tag{5.3}$$

Proof. We will prove this by considering the cases defined in (5.3), beginning with case one. If $\langle Qu, x - y\rangle \langle Qw, x - y\rangle \leq 0$ then either

(i) $\langle Qu, x-y\rangle \langle v, u\rangle \langle v, x-y\rangle \leq 0$, or

(ii) $\langle Qw, x-y\rangle \langle v, w\rangle \langle v, x-y\rangle \leq 0$

since $\langle v, u\rangle \langle v, w\rangle \geq 0$. Without loss of generality, suppose (i) holds. Observe that this condition is equivalent to one of

(iii) $\langle Qu, x-y\rangle \leq 0, \langle v, u\rangle \langle v, x-y\rangle \geq 0$, or

(iv) $\langle Qu, x-y\rangle \geq 0, \langle v, u\rangle \langle v, x-y\rangle \leq 0$.

For (iii), x, y and u all satisfy the conditions of Proposition 29 and hence $f(x) \leq h(x, y)$. Alternatively, for (iv), notice that $\langle Qu, y-x\rangle \leq 0$ and $\langle v, u\rangle \langle v, y-x\rangle \geq 0$ and therefore analogously we have $f(y) \leq h(y, x)$. This is equivalent to $\langle Q(y-x), y-x\rangle \leq 0$ and therefore it must also be true that $\langle Q(x-y), x-y\rangle \leq 0$ and hence $f(x) \leq h(x, y)$. The proof of the case where (ii) holds is analogous,

For condition 2 of (5.3), notice that $\langle v, y\rangle < \langle v, x\rangle \leq v^u$. Therefore, by Lemma 30, $\langle v, x\rangle^2 \leq O(\langle v, x\rangle, \langle v, y\rangle, v^u)$ and hence $f^+(x) \leq \lambda O(x, l, u)$. This implies

$$f(x) = f^+(x) + f^-(x) \leq f^+(x) + h^-(x, y) \leq \lambda O(x, z^l, y) + h^-(x, y)$$

as required. The proof for condition 3 is analogous.

Finally, as the cases do not exclude any $x \in \mathbb{R}^n$, we have that (5.3) holds everywhere. □

We can think of (5.3) as effectively dividing $\mathbb{R}^n$ into three spaces, and providing a linear upper approximation in each, either by directionally concave tangent planes, or by tangent planes and over-envelopes. By focusing our search on one region at a time, we can use the resultant upper plane to formulate a linear cutting plane model of (5.1).

5.2.4 Space Partitioning

As mentioned previously, our selection of u and w is not strictly dependent on x or y, but rather influences the size of the area defined by condition one of (5.3). Given our only requirement is to have $\langle Qu, u\rangle > 0$, $\langle Qw, w\rangle > 0$, and $\langle v, u\rangle \langle v, w\rangle \geq 0$, we could simply choose $u = w = v$ (as $\langle Qv, v\rangle = \lambda > 0$). However, this means that condition one becomes $\lambda^2 \langle v, x-y\rangle \langle v, x-y\rangle \leq 0$ and therefore we can only use the directionally concave tangent plane when $\langle v, x-y\rangle = 0$.

Where applicable, tangent planes generally offer a tighter approximation than those provided by over-envelopes. As a result, it is advantageous to utilise case one of (5.3) wherever possible, thereby minimising the reliance on over-envelopes. This can be achieved by maximising the angle between the vectors Qu and Qw. Let $\phi \in \mathbb{R}$ denote this angle, then by the law of cosines,

$$\cos(\phi) = \frac{\langle Qw, Qu\rangle}{\|Qw\| \, \|Qu\|}.$$

Therefore, to maximise ϕ we need to find the u and w that minimises this expression. However, this represents a fairly challenging nonlinear programming problem.

An easier approach is to instead maximise the angle between Qu and v, and then determine w by reflecting Qu about v. Without loss of generality, suppose u and w are such that $\langle v, u\rangle \geq 0$ and $\langle v, w\rangle \geq 0$ (since if case one of (5.3) holds for u, w then it also holds for $-u, -w$). Then the angle ϑ between Qu and v is given by

$$\cos(\vartheta) = \frac{\langle v, Qu\rangle}{\|v\|\,\|Qu\|} = \frac{\lambda\langle v, u\rangle}{\|Qu\|} \propto \frac{\langle v, u\rangle}{\|Qu\|}.$$

Therefore, the best selection of u can be found by solving the auxiliary nonlinear programming problem

$$\begin{aligned} \min \quad & \frac{\langle v, u\rangle}{\|Qu\|} \\ \text{s.t.} \quad & \langle Qu, u\rangle > 0 \\ & \langle v, u\rangle \geq 0 \\ & u \in \mathbb{R}^n. \end{aligned}$$

Note that with respect to the Proposition 29, u is scale invariant. Therefore, it may be easier to instead solve the following quadratic programming problem

$$\begin{aligned} \max \quad & \|Qu\| \\ \text{s.t.} \quad & \langle Qu, u\rangle \geq \epsilon \\ & \langle v, u\rangle = \frac{1}{\lambda} \\ & u \in \mathbb{R}^n, \end{aligned} \tag{5.4}$$

where $\epsilon > 0$. This problem can be solved to local optimality using an appropriate nonlinear programming solver. Once we have found a solution for u, we can determine a valid w by reflecting Qu about v. This reflection is given as

$$\begin{aligned} Qw = 2v - Qu = \frac{2}{\lambda}Qv - Qu = Q\left(\frac{2}{\lambda}v - u\right), \\ \iff \quad w = \frac{2}{\lambda}v - u. \end{aligned} \tag{5.5}$$

Then, since $\langle v, u\rangle = \frac{1}{\lambda}$, it follows that

1. $\langle Qw, w\rangle = \frac{4}{\lambda^2}\lambda - \frac{4}{\lambda}\lambda\langle v, u\rangle + \langle Qu, u\rangle = \frac{4}{\lambda} - \frac{4}{\lambda} + \langle Qu, u\rangle > 0$, and
2. $\langle v, w\rangle = \frac{2}{\lambda} - \frac{1}{\lambda} \geq 0$,

as required.

5.2.5 Upper Plane Visualisation

We can visualise the upper plane defined in (5.3) by considering one-positive matrices in $\mathbb{R}^{2\times 2}$. Let $w_1, w_2 \in [0,1]^2$ be two vectors whose components are uniformly randomly generated in the range $[0,1]$. Then we can construct a one-positive matrix by finding

$$Q = w_1 w_1^T - w_2 w_2^T.$$

This comes from the fact that the rank one matrices of $w_1 w_1^T$ and$w_2 w_2^T$ are both positive definite. Once we have generated Q, we can calculate u and w using the procedure outlined above and plot the nonlinear upper plane defined in (5.3).

Each row of Figure 5.1 shows a randomly generated Q and three unique cut generating points $y \in [-1,1]^2$. In green we show the directionally concave tangent planes (case one of (5.3)) and in orange we show the tangent and over-envelope component (cases two and three). The figure clearly outlines the effectiveness of (5.3). For example, in row one, we achieve an angle between Qu and Qw of 168°, allowing us to substantially improve the approximation compared with using purely over-envelopes. Similarly, in row two we achieve an angle of 101°. However, in cases of dominating convex components, such as in row three, we achieve a much smaller angle of just 6°, leading to only a minor improvement in approximation.

In higher dimension, we are not able to easily visualise these upper planes, however we can still measure the angles between Qu and Qw, and analyse the effect that the number of negative eigenvalues has on this metric. Let $l \geq 1$ be the number of desired negative eigenvalues. Similar to before, we can construct a randomly generated matrix in $\mathbb{R}^{n\times n}$ with one positive and $l \leq n-1$ negative eigenvalues by finding

$$Q = w_1 w_1^T - \alpha \sum_{j=2}^{l+1} w_j w_j^T$$

where each $w_j \in [0,1]^n$ for $j = 1, \ldots, l+1$ and where $\alpha \geq 0$. Figure 5.2 shows the average angle, in degrees, between Qu and Qw across 25 randomly generated matrices where $n = 10$ for ranging values of α and l. It clearly shows how increasing either α or l leads to more dominating concave components and hence larger resultant angles. By the results seen in Figure 5.1, these should lead to tighter upper planes from (5.3).

5.2.6 A Partial Cutting Plane Model

To formulate our cutting plane model, let us decompose Q into a set of matrices with at most one positive eigenvalue. This can be achieved by writing $f(x)$ as

$$f(x) = g(x) + \sum_{i=1}^{m} f_i(x)$$

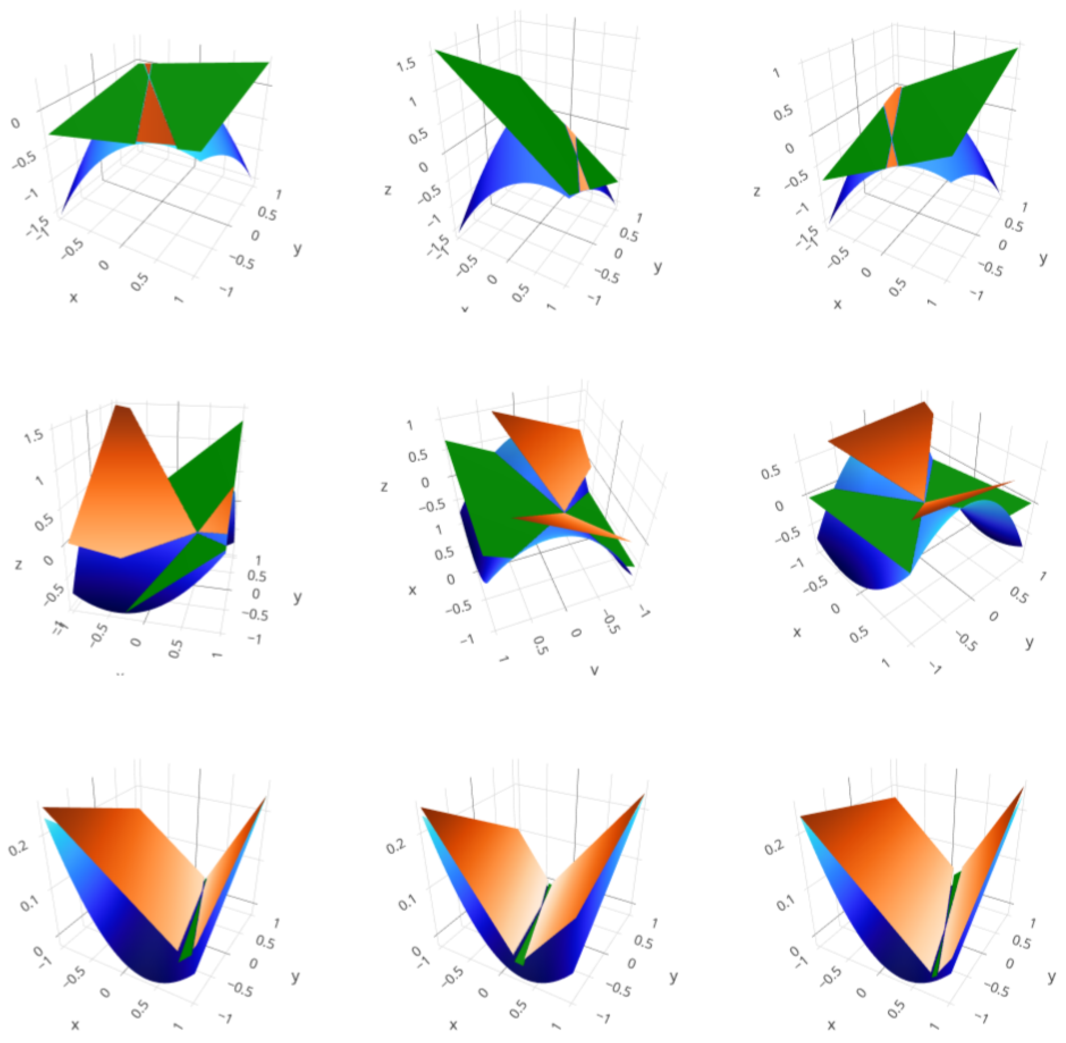

Figure 5.1: Each row contains a randomly generated one-positive matrix (shown in blue) and the resultant upper plane from (5.3) for three random generating points y. In green we show the directionally concave tangent component, and in orange the over-envelope component. The angles between Qu and Qw for each row are 168°, 101° and 6° respectively.

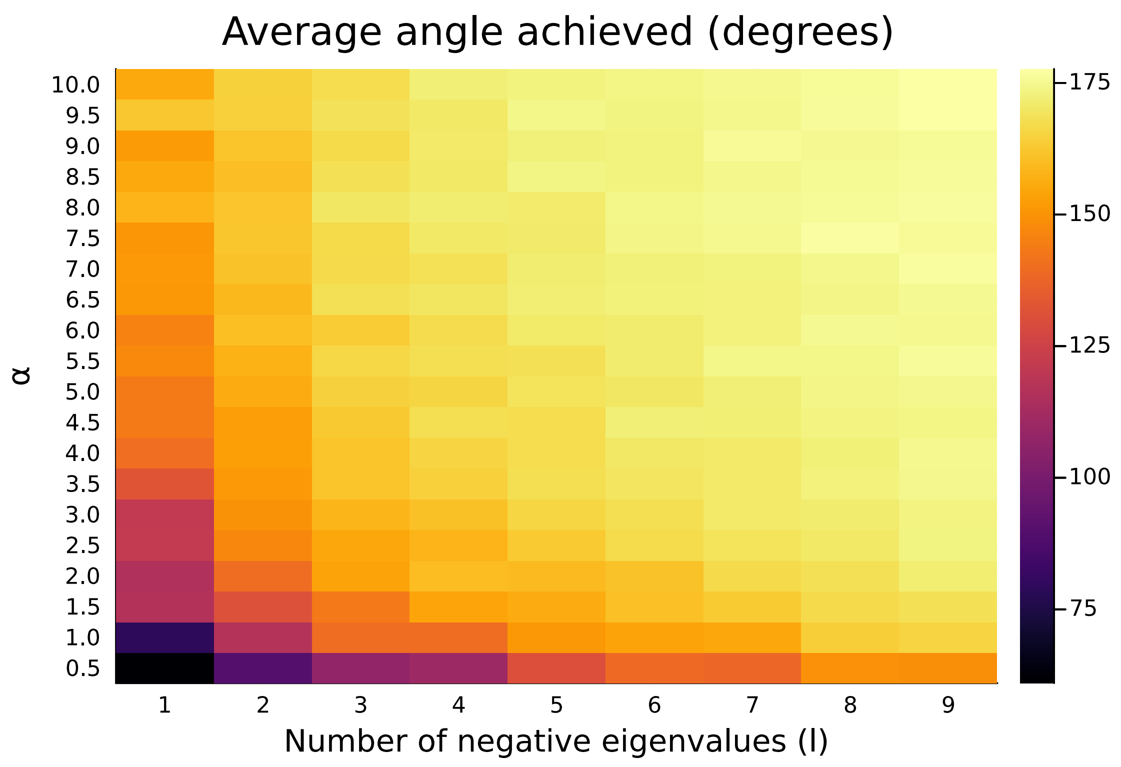

Figure 5.2: Average angle in degrees between Qu and Qw across 25 randomly generates one-positive matrices $Q \in \mathbb{R}^{10\times 10}$ for ranging values of α and l.

where $g(x) = \langle Wx, x\rangle$ and W is negative semi-definite, and $f_i(x) = \langle Q_i x, x\rangle$ where Q_i has one positive eigenvalue. We can then use Proposition 31 to build an upper approximation of each $f_i(x)$ functional component. Note that we can always choose $g(x) = 0$ and $f_i(x) = \lambda_i \langle v_i, x\rangle^2 + \frac{1}{m}\langle Q^- x, x\rangle$ as a valid decomposition. We discuss decomposition strategies further in a later section.

Let u_i be a solution of (5.4) for Q_i and let w_i be given by (5.5). Then let us define the following regions in $\mathbb{R}^n$,

$$\begin{aligned}
H_i^1(y) &= \{x \in \mathbb{R}^n \,:\, \langle Q_i u_i, x - y\rangle \le 0, \langle Q_i w_i, x - y\rangle \ge 0\},\\
H_i^2(y) &= \{x \in \mathbb{R}^n \,:\, \langle Q_i u_i, x - y\rangle \ge 0, \langle Q_i w_i, x - y\rangle \le 0\},\\
H_i^3(y) &= \{x \in \mathbb{R}^n \,:\, \langle Q_i u_i, x - y\rangle \ge 0, \langle Q_i w_i, x - y\rangle \ge 0\},\\
H_i^4(y) &= \{x \in \mathbb{R}^n \,:\, \langle Q_i u_i, x - y\rangle \le 0, \langle Q_i w_i, x - y\rangle \le 0\}.
\end{aligned}$$

Note that $H_i^1(y)$ and $H_i^2(y)$ satisfy condition one of (5.3). Furthermore, we have from (5.5) that

$$\begin{aligned}
&\langle Q_i w_i, x - y\rangle = 2\langle v_i, x - y\rangle - \langle Q_i u_i, x - y\rangle\\
\iff\ &\langle v_i, x - y\rangle = \frac{1}{2}\left(\langle Q_i w_i, x - y\rangle + \langle Q_i u_i, x - y\rangle\right).
\end{aligned}$$

Therefore, if $x \in H_i^3(y)$ then $\langle v_i, x - y\rangle \ge 0$ (since $\langle Q_i u_i, x - y\rangle \ge 0$ and $\langle Q_i w_i, x - y\rangle \ge 0$). Similarly if $x \in H_i^4(y)$ then $\langle v_i, x - y\rangle \le 0$. As such, $H_i^3(y)$ and $H_i^4(y)$, satisfy conditions two and three of (5.3), respectively. This allows us to linearise the conditions of (5.3) into the four spaces defined by $H_i^1(y), \dots, H_i^2(y)$.

Let $\mathbf{v}^l = \left(v_1^l, \dots, v_m^l\right)$ and $\mathbf{v}^u = \left(v_1^u, \dots, v_m^u\right)$ provide estimates of the lower and upper bounds of $\langle v_i, x\rangle$ for $i = 1, \dots, m$. Then we can introduce the following epigraph sets that relate the sets $H_i^1(y), \dots, H_i^4(y)$ with their associated upper place from (5.3),

$$E_i^1(y) = \left\{(\theta, x) \in \mathbb{R} \times H_i^1(y) \,:\, \theta \le h_i(x, y)\right\}, \tag{5.6}$$
$$E_i^2(y) = \left\{(\theta, x) \in \mathbb{R} \times H_i^2(y) \,:\, \theta \le h_i(x, y)\right\}, \tag{5.7}$$
$$E_i^3(y) = \left\{(\theta, x) \in \mathbb{R} \times H_i^3(y) \,:\, \theta \le h_i^-(x, y) + \lambda O\big(\langle v_i, x\rangle, \langle v_i, y\rangle, v_i^u\big)\right\}, \tag{5.8}$$
$$E_i^4(y) = \left\{(\theta, x) \in \mathbb{R} \times H_i^4(y) \,:\, \theta \le h_i^-(x, y) + \lambda O\big(\langle v_i, x\rangle, v_i^l, \langle v_i, y\rangle\big)\right\}. \tag{5.9}$$

Provided the bounds given by $\mathbf{v}^l$ and $\mathbf{v}^l$ are valid, we have that for any $x, y \in \mathbb{R}^n$ and $i \in \{1, \dots, m\}$, there exists a $k \in \{1, \dots, 4\}$ such that $(f_i(x), x) \in E_i^k(y)$. Furthermore, give K is bounded, it is always possible to determine $\mathbf{v}^l$ and $\mathbf{v}^l$ that are valid for all $x \in K$.

Our cutting plane model takes a set of points in $\mathbb{R}^n$, each assigned to a functional component and epigraph set, and adds the associated upper plane and region constraints. Let $A_i^k \subset \mathbb{R}^n$ give the set of points used to approximate $f_i(x)$ restricted to $x \in H_i^k(y)$, and let $\mathbf{A} = \left(A_1^1, \dots, A_m^4\right)$. Note that in practice, since $H_i^{k_1} \cap H_i^{k_2} = \emptyset$, we try to assure that $A_i^{k_1} \cap A_i^{k_2} = \emptyset$ for all $k_1 \neq k_2$. Then our partial cutting plane problem, denoted by $\Gamma(\mathbf{A}, \mathbf{v^l}, \mathbf{v^u})$, is given by the following quadratic concave maximisation problem,

$$\begin{aligned} \Gamma(\mathbf{A}, \mathbf{v^l}, \mathbf{v^u}) \;=\; \max \quad & g(x) + \sum_{i=1}^{m} \theta_i + \langle p, x\rangle \\ \text{s.t} \quad & (\theta_i, x) \in E_i^k(y), && \forall y \in A_i^k, i = 1, \dots, m, k = 1, \dots, 4, \\ & \theta_i \in \mathbb{R}, && i = 1, \dots, m, \\ & x \in K. \end{aligned} \tag{5.10}$$

Note that we call this a *partial* cutting plane model as the inclusion of $x \in H_i^k(y)$ means we only approximate a subset of K. However, this restriction grants us the following important result, which asserts that $\Gamma(\mathbf{A}, \mathbf{v^l}, \mathbf{v^u})$ provides an upper bound of $f(x)$ for all x that are feasible for this problem.

Theorem 32. *Let $\mathbf{A} = \left(A_1^1, \dots, A_m^k\right)$ be defined as above, and consider the following quadratic problem,*

$$\begin{aligned} \Phi(\mathbf{A}) \;=\; \max \quad & f(x) \\ s.t \quad & x \in H_i^k(y), \quad \forall y \in A_i^k, i = 1, \dots, m, k = 1, \dots, 4, \\ & x \in K. \end{aligned} \tag{5.11}$$

If $\mathbf{v}^l = \left(v_1^l, \dots, v_m^l\right)$ and $\mathbf{v}^u = \left(v_1^u, \dots, v_m^u\right)$ are such that $v_i^l \leq \langle v_i, x\rangle \leq v_i^u$ holds for all $i = 1, \dots, m$ and x that are feasible for (5.11), then $\Phi(\mathbf{A}) \leq \Gamma(\mathbf{A}, \mathbf{v^l}, \mathbf{v^u})$.

Proof. Let x be optimal for $\Phi(\mathbf{A})$, then for all $k = 1, \dots, 4$, $i = 1, \dots, m$ and $y \in A_i^k$ we must have $x \in H_i^k(y)$. Furthermore, given that $v_i^l \leq \langle v_i, x\rangle \leq v_i^u$, we have by Proposition 31 that $(f_i(x), x) \in E_i^k(y)$. As this holds for all $i = 1, \dots, m$ we have that $(f_1(x), \dots, f_m(x), x)$ is a feasible solution to (5.10), and therefore $\Phi(\mathbf{A}) \leq \Gamma(\mathbf{A}, \mathbf{v^l}, \mathbf{v^u})$ as required. □

Therefore, our partial cutting plane problem provides a valid upper approximation of the objective function, but only on its feasible domain, which is a subset of K. To then use this result to solve (5.1), we must employ a branching strategy to ensure we search the entire space.

5.2.7 Branch and Cut Algorithm

In order to use our partial cutting plane model to provide a linear approximation of the objective function everywhere, we must employ a branch and cut algorithm. Whenever a cut is added to the model to approximate an $f_i(x)$ functional component, the branch and cut algorithm generates four child subproblems, one for each of the epigraph sets (5.6)-(5.9). In this way, the feasible space is divided into four and an appropriate upper plane is added to each. Given a cut generating point y and functional component i, the process of generating these four child nodes is shown in Algorithm 10. This child generating algorithm is then used inside the branch and cut procedure shown in Algorithm 11.

Algorithm 10: Create the 4 child nodes associated with the tangent of y on functional component i.

```
1 function children ($y, i, \mathbf{A}, \mathbf{v}^l, \mathbf{v}^u$):
2     $Children \leftarrow \emptyset$
3     for $k = 1, \dots, 4$ do
4         $\mathbf{Child} \leftarrow \mathbf{A}$
5         $Child_i^k \leftarrow Child_i^k \cup \{y\}$
6         if $k = 3$ then $v_i^l \leftarrow \langle v_i, y \rangle$
7         if $k = 4$ then $v_i^u \leftarrow \langle v_i, y \rangle$
8         $Children \leftarrow Children \cup \{\Gamma(\mathbf{Child}, \mathbf{v}^l, \mathbf{v}^u)\}$
9     return $Children$
```

Theorem 33. *Algorithm 11 converges to an optimal solution of* (5.1).

Proof. We will prove this statement by showing the following two properties hold.
1. Every path in the search tree converges.
Let $(x^1, \theta^1), (x^2, \theta^2), \dots,$ be the sequence of solutions along any descending path in the search tree. Since the cut of a solution is tight at its generating point, the question is whether the gap between each epigraph variable and its true function value closes, such that $\theta_i = f_i(x)$. Suppose, to the contrary, that the sequence does not converge. Then there exists a functional component $i \in \{1, \dots, m\}$ and $\varepsilon > 0$ such that for all $N \geq 1$ there exists an $r \geq N + 1$ with

$$0 < \varepsilon \leq \left\| \theta_i^r - f_i(x^r) \right\| = \theta_i^r - f_i(x^r). \tag{5.12}$$

Observe that for any previous solution x^q ($q \leq r - 1$) and $k \in \{1, \dots, 4\}$ we have

$$\theta^r \leq \max \left\{ \tilde{\theta} \ : \ \left(\tilde{\theta}, x^r \right) \in E_i^k \left(x^q \right) \right\}. \tag{5.13}$$

For $k = 1, 2$, the solution to (5.13) is given by a tangent and hence (5.12) becomes

$$\begin{aligned} \theta_i^r - \langle Q_i x^r, x^r \rangle &\leq h(x^r, x^q) - \langle Q_i x^r, x^r \rangle \\ &= \langle Q_i x^q, x^q \rangle + 2 \langle Q_i x^q, x^r - x^q \rangle - \langle Q_i x^r, x^r \rangle \\ &= \langle Q_i (x^r - x^q), x^r - x^q \rangle \\ &\leq \| Q_i \| \, \| x^r - x^q \| , \end{aligned} \tag{5.14}$$

Algorithm 11: An exact branch and cut method for solving (5.1).

```
function BranchAndCut (Q, p, K):
    Decompose $Q$ to get $W, Q_1, \dots, Q_m$ and $v_1, \dots, v_m$
    Solve (5.4) with $Q_1, \dots, Q_m$ and $v_1, \dots, v_m$ for $u_1, \dots, u_m$
    Use (5.5) to determine $w_1, \dots, w_m$
    Find $\mathbf{v}^l = \left(v_1^l, \dots, v_m^l\right)$ such that $v_i^l \le \min_{x \in K} \langle v_i, x \rangle$
    Find $\mathbf{v}^u = \left(v_1^u, \dots, v_m^u\right)$ such that $v_i^u \ge \min_{x \in K} \langle v_i, x \rangle$
    Find heuristic $x \in K$ and choose an $i \in \{1, \dots, m\}$
    $LB \leftarrow \langle Qx + p, x \rangle$
    $\mathbf{A} \leftarrow (\emptyset, \dots, \emptyset)$
    $Nodes \leftarrow$ children$(x, i, \mathbf{A}, \mathbf{v}^l, \mathbf{v}^u)$
    while $Nodes \neq \emptyset$ do
        Choose a $\Gamma(\mathbf{A}, \mathbf{v}^l, \mathbf{v}^u) \in Nodes$
        Solve $\Gamma(\mathbf{A}, \mathbf{v}^l, \mathbf{v}^u)$ for $(\theta, x)$
        $LB \leftarrow \max\{LB, \langle Qx + p, x \rangle\}$
        if $\sum_{i=1}^m \theta_i \le LB$ then continue
        Choose an $i \in \{1, \dots, m\}$ with $\langle Q_i x, x \rangle < \theta_i$
        $Nodes \leftarrow Nodes \cup$ children$(x, i, \mathbf{A}, \mathbf{v}^l, \mathbf{v}^u)$
    return $LB$
```

where $\|Q_i\|$ is the Frobenius norm of Q_i. If $k = 3$, then (5.13) is given (5.8) and hence (5.12) becomes

$$\begin{aligned} \theta_i^r - \langle Q_i x^r, x^r\rangle &\le h_i^-(x^r, x^q) + \lambda O\left(\langle v_i, x^r\rangle, \langle v_i, x^q\rangle, v_i^u\right) - \langle Q_i x^r, x^r\rangle \qquad (5.15)\\ &\le h_i^-(x^r, x^q) + \lambda O\left(\langle v_i, x^r\rangle, \langle v_i, x^q\rangle, v_i^u\right) - \left\langle Q_i^- x^r, x^r\right\rangle - \left\langle Q_i^+ x^r, x^r\right\rangle\\ &= h_i^-(x^r, x^q) - \left\langle Q_i^- x^r, x^r\right\rangle + \lambda\left(O\left(\langle v_i, x^r\rangle, \langle v_i, x^q\rangle, v_i^u\right) - \langle v_i, x^r\rangle^2\right). \end{aligned}$$

Then analogous to (5.14),

$$h_i^-(x^r, x^q) - \left\langle Q_i^- x^r, x^r\right\rangle \le \left\|Q_i^-\right\| \left\|x^r - x^q\right\|.$$

Furthermore,

$$\begin{aligned} O\left(\langle v_i, x^r\rangle, \langle v_i, x^q\rangle, v_i^u\right) - \langle v_i, x^r\rangle^2 &= \langle v_i, x^r\rangle \left(\langle v_i, v^q\rangle + v_i^u\right) - \langle v_i, v^q\rangle v_i^u - \langle v_i, x^r\rangle^2\\ &= \langle v_i, x^r - v^q\rangle \left(v_i^u - \langle v_i, x^r\rangle\right)\\ &\le \langle v_i, x^r - v^q\rangle \qquad (5.16)\\ &\le \|v_i\| \left\|x^r - v^q\right\| \end{aligned}$$

where (5.16) comes from the fact $v_i^u \ge \langle v_i, x^r\rangle$. Therefore from (5.15) it follows that

$$\theta_i^r - \langle Q_i x^r, x^r\rangle \le \left(\left\|Q_i^-\right\| + |\lambda| \|v_i\|\right) \left\|x^r - x^q\right\|.$$

The above holds analogously in the case where $k = 4$. Therefore, we have that

$$\left\|x^r - x^q\right\| \ge \frac{\varepsilon}{\min\left\{\|Q_i\|, \left\|Q_i^-\right\| + |\lambda| \|v_i\|\right\}} > 0.$$

However this implies that the sequence of solutions contains no Cauchy subsequence, which is impossible since K is compact set. Therefore, the sequence must converge.
2. The algorithm locates an optimal solution.
Let x^* be an optimal solution of (5.1), and suppose this solution is feasible for an arbitrary node $\Gamma(\mathbf{A}, \mathbf{v}^l, \mathbf{v}^u) \in Nodes$. Let (θ, x) be the solution to this node. Then either $f(x) = LB = f(x^*)$, in which case an optimal solution has been located, or $f(x) \le LB < f(x^*)$. Since the cuts contained in $\Gamma(\mathbf{A}, \mathbf{v}^l, \mathbf{v}^u)$ are all valid and x^* is a feasible solution, we must have $f(x^*) \le \sum_{i=1}^m \theta_i$ and hence $LB < \sum_{i=1}^m \theta_i$. Therefore, this node is not pruned and since $f(x) = \sum_{i=1}^m \langle Q_i x, x\rangle < \sum_{i=1}^m \theta_i$, there must exist an i with $\langle Q_i x, x\rangle < \theta_i$. But if $\langle Q_i x, x\rangle < \theta_i$ then $x \notin A_i^k$ for $k = 1, \ldots, m$, since from (5.3) the cut of any $y \in A_i^k$ is tight at y. Therefore, every child node in `children`$(x, i, \mathbf{A}, \mathbf{v}^l, \mathbf{v}^u)$ gets a new unique cut not present in $\Gamma(\mathbf{A}, \mathbf{v}^l, \mathbf{v}^u)$. As $\bigcup_{k=1,\ldots,4} H_i^k(x) = \mathbb{R}^n$ there must exist a child node that still contains x^*. Finally, as x^* is feasible for at least one of the root nodes generated by line 9 of Algorithm 11, we have that the algorithm locates an optimal solution. □

It is worth pointing out that in the case where a functional component has no concave component, i.e., a rank 1 matrix with a positive eigenvalue, such as $Q = vv^T$, the only option for u and w is $u = w = v$. However, this results in only two child nodes, since there is no concave component and hence no concave region. This reduces the methodology to a more standard branch and bound approach for solving nonconcave quadratic programs.

Finally, a note regarding our chosen method to solve a node $\Gamma(\mathbf{A}, \mathbf{v}^l, \mathbf{v}^u) \in \mathit{Nodes}$. While this problem is concave and can therefore be solved to global optimality using any capable mixed-integer nonlinear programming solver, we suggest the use of outer approximation techniques. This is because the tangent planes used to approximate $g(x)$ are always valid, since the function is concave, and hence may persist amongst all nodes. Additionally, these techniques are globally convergent regardless of integrality of K. If K is defined by only a polyhedral set, and hence x is continuous, then the node subproblem can be solved using the extended cutting plane algorithm (Westerlund & Pettersson, 1995). Alternatively, if K is integer, we can use a branch and cut outer approximation framework such as that defined in Duran and Grossmann (1986). In either case, the problem is solved to global optimality by generating sufficiently many tangent planes. For the purposes of of Algorithm 11, these tangents are always valid, and hence may be reused amongst all nodes, rather than having to generate a new set every time. This can help in avoiding unnecessary computation, and potentially speed up solve times of later nodes.

5.2.8 Functional Decomposition and Branching Strategies

The functional decomposition of $f(x)$ should attempt to generate one-positive matrices with large angles between $Q_i u_i$ and $Q_i w_i$, as this allows us to approximate more of the function via tangent planes, rather than over-envelopes. Recall from Figure 5.2 that we can achieve large angles by either including more negative eigenvalues, or by placing more weight on the concave component. In line with this motivation, we suggest three potential strategies to decompose $f(x)$. For each strategy, let

$$\lambda_1 \geq \cdots \geq \lambda_m > 0 > \lambda_{m+1} \geq \cdots \geq \lambda_{m+l}$$

be the nonzero eigenvalues and let $v_1, \ldots, v_{m+l}$ be the associated eigenvectors. Then the strategies are as follows, and their validity is proved in Appendix A.2.

1. *Basic:* The most basic decomposition is to simply set $g(x) = 0$ and

$$f_i(x) = \lambda_i \langle v_i, x \rangle^2 + \frac{1}{m} \langle Q^- x, x \rangle, \quad i = 1, \ldots, m.$$

2. *Greedy:* In a greedy approach, we attempt to create functional components by pairing the largest eigenvalue with the smallest and so on, until we run out of either positive or negative eigenvalues. If there are remaining negative eigenvalues, these

all get combined and incorporated into $g(x)$. Alternatively, if there are remaining positive eigenvalues, these becomes their own functional components that have no negative eigenvalues. Hence, the decomposition results in

$$f_i(x) = \begin{cases} \lambda_i \langle v_i, x\rangle^2 + \lambda_{m+l+1-i} \langle v_{m+l+1-i}, x\rangle^2, & \text{if } i \leq l \\ \lambda_i \langle v_i, x\rangle^2, & \text{if } i > l \end{cases}, \quad i = 1, \dots, m,$$

and

$$g(x) = \begin{cases} \sum_{i=m+1}^{l} \lambda_i \langle v_i, x\rangle^2, & \text{if } m < l, \\ 0, & \text{otherwise.} \end{cases}$$

3. *Stratified:* Our last strategy considers a more stratified approach to eigenvalue pairings. Rather than pairing the largest eigenvalue with the smallest, this approach attempts to pair the largest eigenvalue with the lth eigenvalue if it is negative, or otherwise the $m+1$ eigenvalue. The decomposition results in

$$f_i(x) = \begin{cases} \lambda_i \langle v_i, x\rangle^2 + \lambda_{\max\{m,l\}+i} \langle v_{\max\{m,l\}+i}, x\rangle^2, & \text{if } i \leq l \\ \lambda_i \langle v_i, x\rangle^2, & \text{if } i > l \end{cases}, \quad i = 1, \dots, m,$$

and

$$g(x) = \begin{cases} \sum_{i=m+1}^{l} \lambda_i \langle v_i, x\rangle^2, & \text{if } m < l, \\ 0, & \text{otherwise.} \end{cases}$$

Another vitally important point to consider is the branching strategy used. This determines which i, or functional component, to branch off in line 16 of Algorithm 11. We suggest one of three branching strategies. For each, the idea is to favour producing approximations on a certain functional component, based on some metric.

1. *Favour gap.* Choose the i that has the largest value of $\theta_i - \langle Q_i x, x\rangle$.

2. *Favour contribution.* Choose the i that has the largest value of $\langle Q_i x, x\rangle$, such that $\theta_i > \langle Q_i x, x\rangle$.

3. *Favour angle.* Choose the i that has the largest angle between $Q_i u_i$ and $Q_i w_i$, such that $\theta_i > \langle Q_i x, x\rangle$.

The chosen strategy can then be used to determine i on lines 7 and 16 of Algorithm 11.

5.3 Numerical Results

We now evaluate the performance of Algorithm 11 across a range of quadratic programming test problems, including both continuous and discrete instances. The algorithm was implemented in Julia version 1.10.0 using Gurobi version 11.0.1 as its lower-level solver. The angle maximisation auxiliary problem (5.4) was solved to local stationarity using the Ipopt solver using all default settings (Wachter & Biegler, 2006).

For continuous problems, the node subproblems were solved using Gurobi's concave quadratic programming solver using all default settings. For integer problems, each concave quadratic subproblem was solved using an outer approximation methodology, for the reasons motivated earlier. The tangents of $g(x)$ used to solve the node subproblems are propagated to the root node, and hence persist throughout the branch and cut search tree. These concave tangents are added using the *lazy constraint callback* functionality, allowing them to be introduced *during* the branch and bound process. As such, only one MIP search tree is formulated at each node.

We benchmark our methodology against Gurobi's nonconcave quadratic programming solver, using all default settings. All tests were conducted on a machine with a 2.3 GHz AMD EPYC processor with 64GB RAM, using a single thread and a ten minute time limit.

Throughout our experimentation, we attempt to control the number of positive eigenvalues of Q. In addition, we ensure the matrix contains majority nonnegative elements, thereby avoiding the situation where the zero vector is in fact the optimal solution. To randomly generate a matrix with m positive and l negative eigenvalues, we find $V \in \mathbb{R}^{n\times m}$ and $W \in \mathbb{R}^{n\times l}$ by uniformly randomly generating each element in the range $[0, 1]$. We then set

$$Q = lVV^T - mWW^T.$$

From computational experience, when $m + l \leq n$, this matrix always contains m positive eigenvalues, and has mostly positive elements. We use this generation procedure throughout this section.

5.3.1 Functional Decompositions

We begin by exploring of the effectiveness of each of the proposed decomposition strategies by measuring the angle between each $Q_i u_i$ and $Q_i w_i$, similar to the analysis outlined in Section 5.2.8. To do so, we randomly generated 16 matrices using the generation procedure outlined above, where $m = 25$, $l = 25$ and $n = 50$. Then, using each of the functional decomposition strategies from Section 5.2.8, we calculate the angle between $Q_i u_i$ and $Q_i w_i$ for each $i = 1, \ldots, 25$. Figure 5.3 shows a frequency plot of the angles achieved across each of the 16 randomly generated matrices. For the basic strategy, we see some fairly poor performance, with a majority of angles found being less than 10°. Interestingly, the greedy approach has a notable cluster around 90°, but achieves very few large angles, with only 2 above 150°. Finally, while the stratified approach as a fairly even spread, it is able to achieve the most large angles, including several close to 180°.

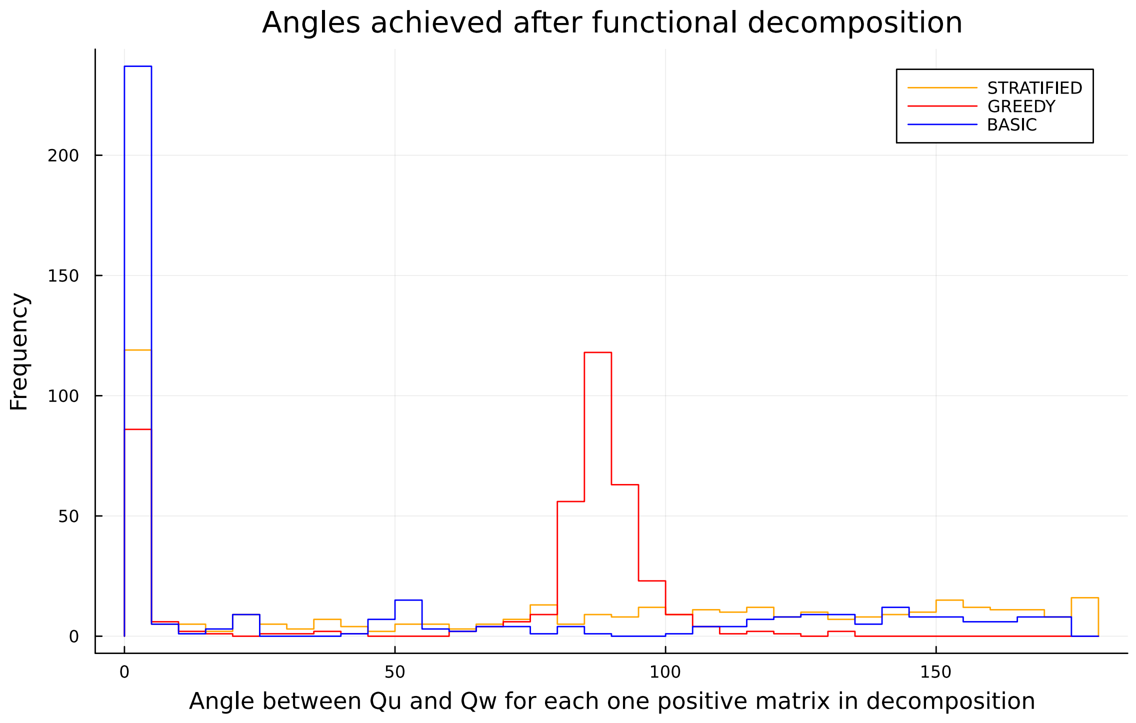

Figure 5.3: Angle between Qu and Qw for each one matrix after functional decomposition, aggregated across 16 randomly generated matrices with $m = 25$, $l = 25$ and $n = 50$.

5.3.2 Continuous Problems

For continuous instances, we examine two specific problem formulations. The first is the straightforward box constrained quadratic programming problem, given by

$$\begin{aligned} \max \quad & \langle Qx, x \rangle \\ \text{s.t.} \quad & x \in [0,1]^n. \end{aligned}$$

The matrix defining each instance is randomly generated using the procedure outlined previously. We generate five instances of $n = 50$ and five of $n = 100$, with $m = 1$, 2, or 3 and $l = 10$, making for a total of 30 test problems. In addition, we look at the continuous quadratic knapsack problem, following an identical formulation to the box constrained quadratic programming problem, except for the addition of a single knapsack constraint. This constraint is constructed in line with Gallo et al. (1980), whereby we randomly generate each integer weight $w_i \in [1, 100]$ and set the capacity to a random integer in the range $[50, \sum_{i=1}^{n} w_i]$. As before, we generate five instances of $n = 50$ and five of $n = 100$, with $m = 1$, 2, or 3 and $l = 10$, making for a total of 30 test problems.

The results from continuous test instances, using a 600 second time limit, are shown in Table 5.1. Note that for $m = 1$, the branching rule is redundant, as there is only one functional component to choose. For $m = 1$, we get very good performance, with all setups able to solve all problems with ease. However, increasing to $m = 2$, we begin to see the importance of good branching rules. Only the gap branching rule was able to solve all ten instances, performing well. Interestingly, it appeared that the basic decomposition approach performed the best across the two problem types. Finally, at $m = 3$, we see a continued trend of decreasing performance with increasing number of positive eigenvalues. That said, using a gap branching rule meant all decomposition strategies were able to solve almost all instances.

Figure 5.4 shows a performance profile of the three functional decomposition strategies using a largest gap branching rule. This figure also includes the performance of the Gurobi benchmark. While no setup is able to improve on Gurobi across the entire test set, they all perform comparably. Additionally, as seen in the previous table, poor performance is often due to going down misleading branches, as branching rules play an important role in overall performance. Given it's mature status, Gurobi has very sophisticated and optimised branching rules, and hence even a comparable performance indicates that the key idea of directionally concave tangent planes is effective on these problems.

5.3.3 Binary Problems

We now explore the case where $K \subset \{0, 1\}^n$, i.e., problems where all decision variables are binary, beginning with the unconstrained binary quadratic programming problem,

m	Decomposition	Branching Rule	Quadratic Box Problem		Continuous Quadratic Knapsack	
			Ave Solve Time (sec)	Num Solved	Ave Solve Time (sec)	Num Solved
1	BASIC	-	0.165	10	0.120	10
1	GREEDY	-	0.345	10	0.212	10
1	STRATI.	-	0.342	10	0.213	10
2	BASIC	GAP	2.932	10	3.004	10
2	BASIC	CONTRI.	600.228	0	600.178	0
2	BASIC	ANGLE	540.580	1	495.843	3
2	GREEDY	GAP	10.363	10	10.388	10
2	GREEDY	CONTRI.	542.789	1	488.377	2
2	GREEDY	ANGLE	542.649	1	488.293	2
2	STRATI.	GAP	10.973	10	9.628	10
2	STRATI.	CONTRI.	486.898	2	497.913	2
2	STRATI.	ANGLE	543.897	1	434.822	3
3	BASIC	GAP	28.569	10	17.464	10
3	BASIC	CONTRI.	600.476	0	600.239	0
3	BASIC	ANGLE	600.627	0	600.321	0
3	GREEDY	GAP	103.558	10	116.902	9
3	GREEDY	CONTRI.	600.204	0	600.166	0
3	GREEDY	ANGLE	600.263	0	600.335	0
3	STRATI.	GAP	126.799	9	100.012	10
3	STRATI.	CONTRI.	546.232	1	600.307	0
3	STRATI.	ANGLE	600.422	0	600.426	0

Table 5.1: Average solve time in seconds, and number of problems out of 10 solved (5 of $n = 50$ and 5 of $n = 100$) across the continuous instances using a 600 second time limit. We separate by number of positive eigenvalues, decomposition and branching strategy.

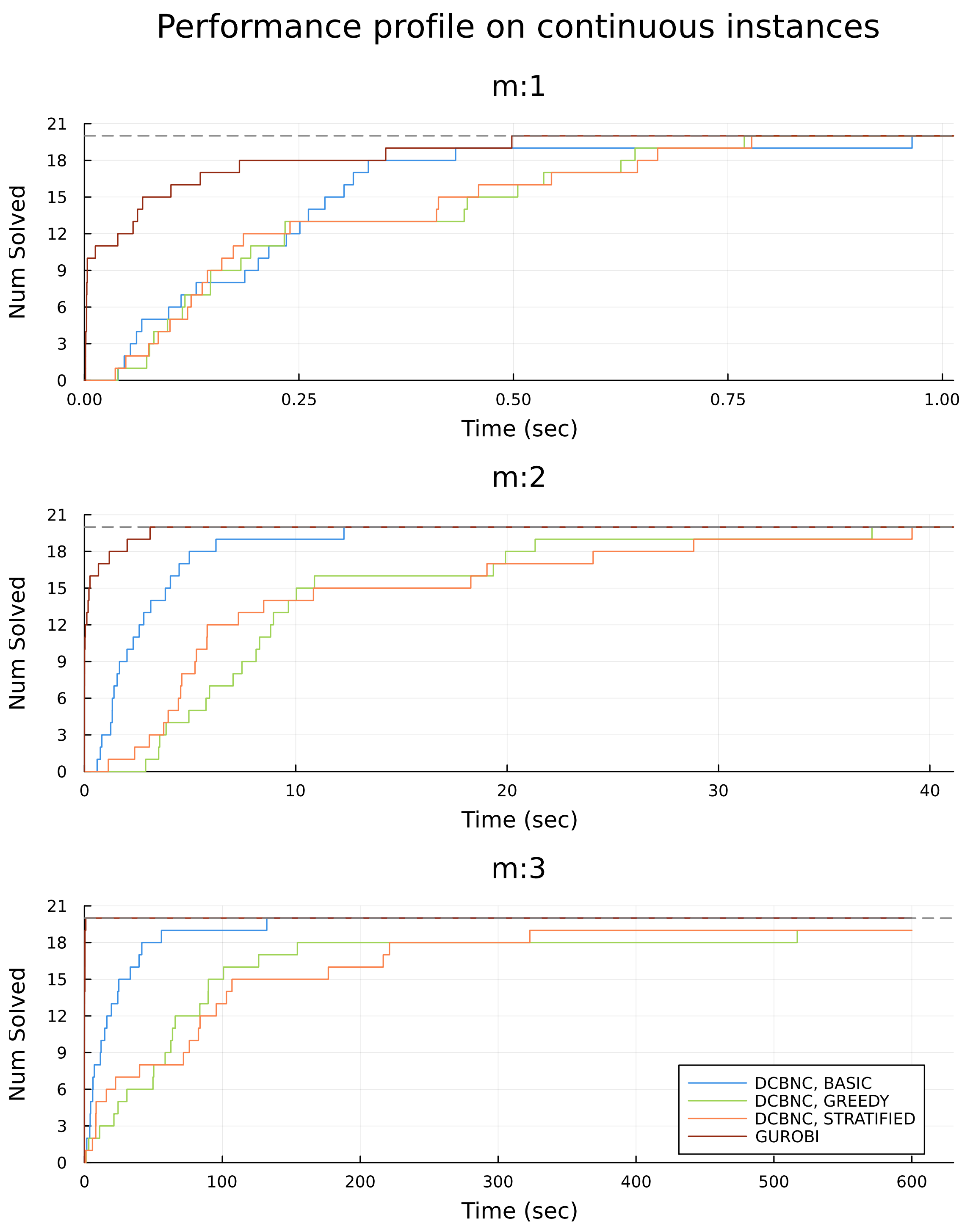

Figure 5.4: Performance profile on continuous instances, broken down by number of positive eigenvalues. We show performance of Gurobi, and the Directional Concave Branch and Cut method (DCBNC) with each of the three decomposition strategies, using the largest gap branching strategy.

given by

$$\begin{aligned} \max \quad & \langle Qx, x\rangle \\ \text{s.t.} \quad & x \in [0,1]^n. \end{aligned}$$

As before, we randomly generate five matrices with $n = 50$ and five with $m = 100$, with $m = 1$, 2, or 3, and $l = 10$, making for a total of 30 test instances. In addition to the unconstrained problem, we tested the algorithm on the diversity problem, whereby a single cardinality constraint is included. We set the cardinality to $p = \lfloor 0.1n \rfloor$. Finally, we also look at the standard quadratic knapsack problem, whereby the knapsack constraint is generated using the same procedure as the previous section.

The results from the binary test instances, over a 600 second time limit, are shown in Table 5.2. Many of the observations are similar to those of the continuous instances. As before, the choice of branching rule plays an important role, as poor decisions can lead to branching down unhelpful subtrees, thereby exaggerating runtime. Furthermore, the number of positive eigenvalues has a major effect on runtime, with the algorithm experiencing many timeouts at $m = 3$.

Rather interestingly, there appears to be a better performance, in general, on the binary instances compared to the continuous instances. Each node subproblem is essentially an easy integer program with a lazy constraint callback. Gurobi is especially good at these problems, and hence has no challenge solving node-level subproblems to integer optimality. The improvement on the binary test instances comes from the solution diversity between a node and its children. In the continuous case, child nodes can have solutions very close to the cut of its parent, yet still pass the bounding rules and hence the algorithm must dive further down this subtree, even though all solutions are very similar. However, on binary instances this is not always the case. The solution of a child node will either return the same solution as its parent, in which case the bounding rules fail and hence the branch is pruned, or a new solution is generated that has at least one x_i that has changed from 0 to 1 (or vice versa). As such, there is less chance of fully exploring an unhelpful subtree.

Figure 5.5 compares the performance of Algorithm 11 using the gap branching on the binary instances, broken down by number of positive eigenvalues. Additionally, it includes the results from Gurobi as a benchmark. For low m, all functional decompositions perform far better than Gurobi. In fact, even when $m = 3$, there remains a large number of problems solved by our branch and cut algorithm that were unable to be solved by Gurobi within the time limit. This represents an impressive performance by Algorithm 11.

As we have seen, the correct choice in functional decomposition and branching strategy can lead to substantial improvements in runtime. However, our implementation has focused primarily on the key concepts of directionally concave upper planes, and less on the overall performance of our branch tree. Gurobi, on the other hand, is a very mature

			Unconstrained Binary		Diversity Problem		Quadratic Knapsack	
m	Decomposition	Branching Rule	Ave Solve Time (sec)	Num Solved	Ave Solve Time (sec)	Num Solved	Ave Solve Time (sec)	Num Solved
1	BASIC	-	0.184	10	0.234	10	0.160	10
1	GREEDY	-	0.212	10	0.134	10	0.167	10
1	STRATI.	-	0.194	10	0.138	10	0.155	10
2	BASIC	GAP	1.298	10	1.608	10	2.587	10
2	BASIC	CONTRI.	70.351	9	124.214	8	210.968	7
2	BASIC	ANGLE	182.582	7	34.670	10	187.984	7
2	GREEDY	GAP	123.324	8	1.592	10	182.834	7
2	GREEDY	CONTRI.	136.030	8	4.023	10	292.258	6
2	GREEDY	ANGLE	136.344	8	4.005	10	292.407	6
2	STRATI.	GAP	2.896	10	1.508	10	3.799	10
2	STRATI.	CONTRI.	17.804	10	2.910	10	52.109	10
2	STRATI.	ANGLE	43.572	10	6.162	10	80.889	9
3	BASIC	GAP	67.529	9	8.939	10	14.640	10
3	BASIC	CONTRI.	600.083	0	600.165	0	494.161	2
3	BASIC	ANGLE	600.161	0	499.628	2	424.897	3
3	GREEDY	GAP	540.826	1	14.044	10	82.269	9
3	GREEDY	CONTRI.	600.328	0	579.572	1	439.58	3
3	GREEDY	ANGLE	600.140	0	466.657	3	434.82	3
3	STRATI.	GAP	34.697	10	8.843	10	49.171	10
3	STRATI.	CONTRI.	600.207	0	481.235	3	446.654	3
3	STRATI.	ANGLE	600.188	0	478.313	3	428.38	3

Table 5.2: Average solve time in seconds, and number of problems out of 10 solved (5 of $n = 50$ and 5 of $n = 100$) across the binary instances over a 600 second time limit. We separate by number of positive eigenvalues, decomposition and branching strategy.

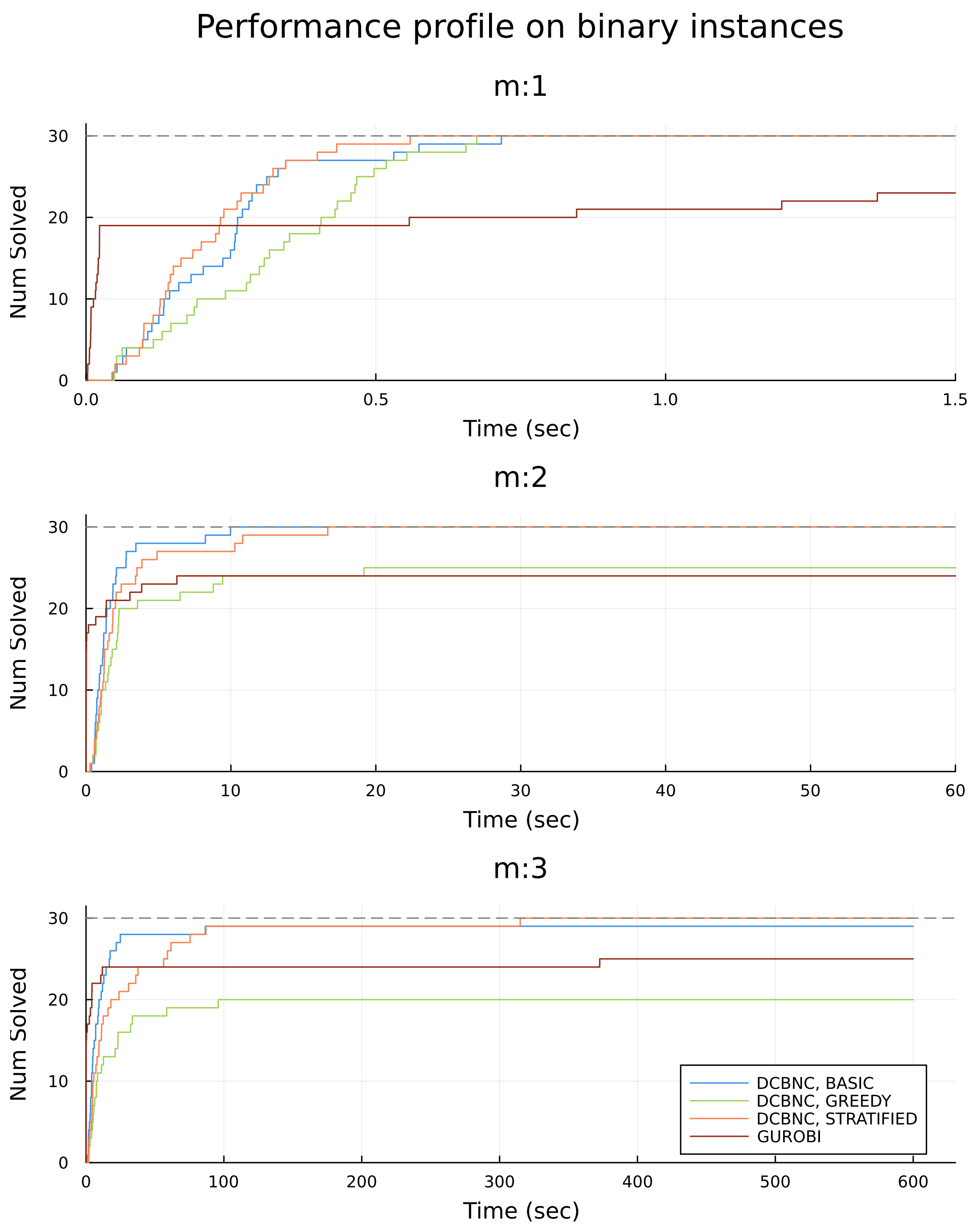

Figure 5.5: Performance profile on binary instances, broken down by number of positive eigenvalues. We show performance of Gurobi, and the Directional Concave Branch and Cut method (DCBNC) with each of the three decomposition strategies, using the largest gap branching strategy.

and state-of-the-art software with sophisticated branching, node selection and heuristic tools under the hood. While these tools do not affect the global convergence, they do have a significant impact on overall runtime.

To therefore conduct a fairer comparison, we show in Table 5.3 the best runtime for each binary instance of size $n = 100$ tested, as well as which decomposition and branching rule achieved this result. We compare these best case run times with the results from Gurobi. Firstly, it is interesting to note that while basic functional decomposition and gap branching are most often the best performer, every decomposition and branching strategy contributes to the best runtime in at least one instance. For the unconstrained binary quadratic problem Gurobi performs particularly well, consistently outperforming our proposed algorithm. However, this is not the case on the diversity problem instances, where it timed out without solving a single instance. Algorithm 11 performs very well on these instances, with several problems solved in less than a second. On the knapsack instances the performance comparison is varied. While Gurobi performs better in most cases, the branch and cut algorithm improves in several instances (see 40, 41 and 44). This all indicates that the key ideas underpinning our algorithm are effective, and with improved branching, node selection and heuristics tools they should be competitive with Gurobi across a wider range of test instances.

Table 5.3: Best achieved runtime over the various decomposition and branching strategies across binary test instances with $n = 100$.

Problem Type	Instance	m	Best Solve Time (sec)	Best Decomposition	Best Branch	Solve Time Gurobi (sec)
unconstrained	1	1	0.283	GREEDY	GAP	0.017
unconstrained	2	1	0.207	BASIC	ANGLE	0.023
unconstrained	3	1	0.152	BASIC	ANGLE	0.021
unconstrained	4	1	0.215	BASIC	ANGLE	0.023
unconstrained	5	1	0.256	BASIC	GAP	0.023
unconstrained	6	2	1.105	BASIC	CONTRI.	0.019
unconstrained	7	2	2.052	BASIC	GAP	0.019
unconstrained	8	2	1.168	BASIC	GAP	0.021
unconstrained	9	2	2.756	BASIC	GAP	0.019
unconstrained	10	2	0.744	BASIC	GAP	0.017
unconstrained	11	3	86.987	STRATI.	GAP	0.019
unconstrained	12	3	16.878	BASIC	GAP	0.018
unconstrained	13	3	24.886	BASIC	GAP	0.019
unconstrained	14	3	4.655	BASIC	GAP	0.021
unconstrained	15	3	9.023	BASIC	GAP	0.021

Problem Type	Instance	m	Best Solve Time (sec)	Best Decomposition	Best Branch	Solve Time Gurobi (sec)
diversity	16	1	0.237	GREEDY	ANGLE	600.001
diversity	17	1	0.158	GREEDY	ANGLE	600.001
diversity	18	1	0.133	BASIC	GAP	600.001
diversity	19	1	0.185	GREEDY	CONTRI.	600.012
diversity	20	1	0.210	GREEDY	ANGLE	600.014
diversity	21	2	1.267	STRATI.	GAP	600.013
diversity	22	2	1.818	STRATI.	GAP	600.024
diversity	23	2	1.245	STRATI.	GAP	600.013
diversity	24	2	1.673	BASIC	GAP	600.013
diversity	25	2	2.775	BASIC	GAP	600.023
diversity	26	3	15.941	STRATI.	GAP	600.013
diversity	27	3	11.301	STRATI.	GAP	600.016
diversity	28	3	11.976	BASIC	GAP	600.013
diversity	29	3	7.973	STRATI.	GAP	600.023
diversity	30	3	7.008	BASIC	GAP	600.014
knapsack	31	1	0.296	BASIC	CONTRI.	0.012
knapsack	32	1	0.126	STRATI.	GAP	0.019
knapsack	33	1	0.215	STRATI.	CONTRI.	0.003
knapsack	34	1	0.211	BASIC	CONTRI.	0.002
knapsack	35	1	0.174	GREEDY	GAP	0.016
knapsack	36	2	1.002	BASIC	GAP	0.043
knapsack	37	2	1.803	BASIC	ANGLE	0.168
knapsack	38	2	1.222	BASIC	GAP	0.003
knapsack	39	2	9.970	BASIC	GAP	0.003
knapsack	40	2	8.241	BASIC	GAP	268.255
knapsack	41	3	86.500	BASIC	GAP	372.621
knapsack	42	3	8.588	BASIC	GAP	0.331
knapsack	43	3	11.027	BASIC	GAP	4.353
knapsack	44	3	6.992	BASIC	GAP	10.694
knapsack	45	3	17.512	BASIC	GAP	11.850

5.3.4 Performance Analysis

To better understand the overall performance of Algorithm 11, we now examine how functional decomposition, achieved angles, and the number of positive eigenvalues affects run time. Figure 5.6 shows, for each binary instance tested, the average angle achieved

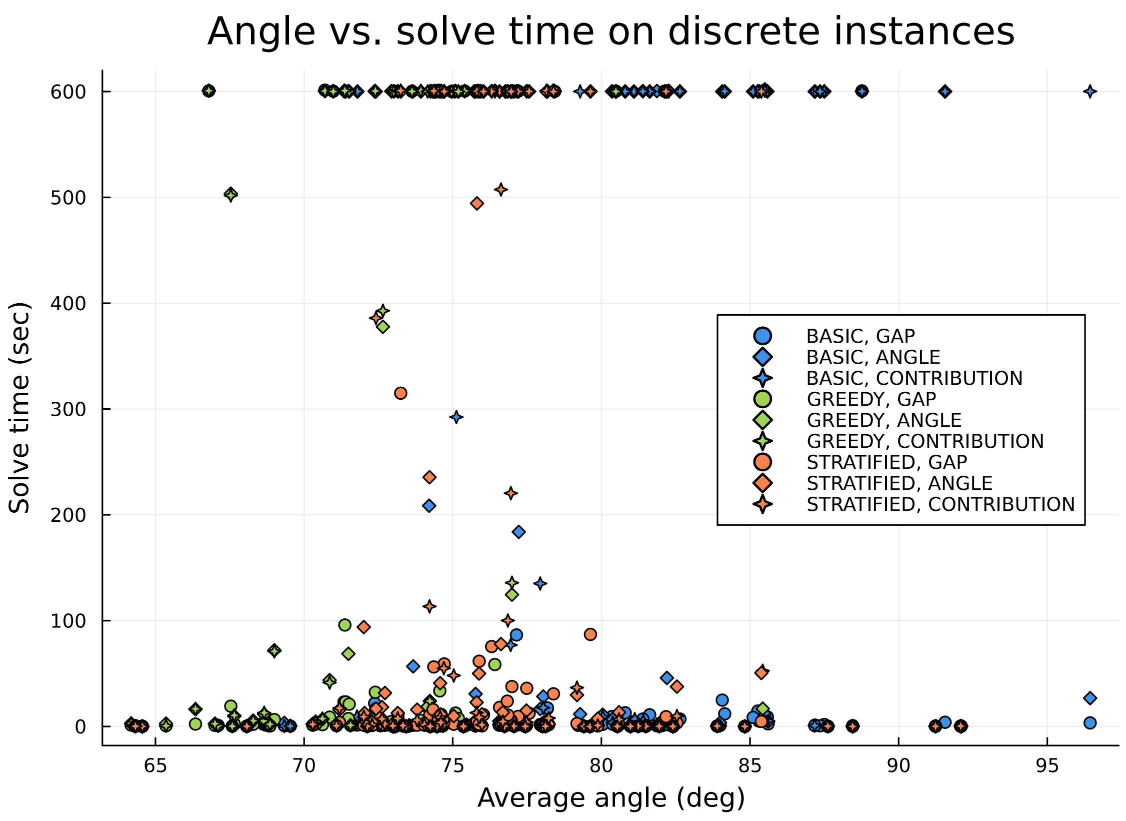

Figure 5.6: Average angle achieved after functional decomposition and resultant solve time in seconds across the binary test instances with a 600 second time limit.

after functional decomposition, and the resultant runtime. Firstly, we notice that the angles achieved by each decomposition align with the results seen in Figure 5.3. However, rather interestingly, there seems to be no discernible trend between measured angle and run time. There are clearly examples of low average angle instances that achieve a good runtime, as well as high angle instances with poor run times. This suggests the achieved angle is not necessarily a key indicator of algorithm performance.

In Figure 5.7 we plot the average runtime for each of the strategy combinations for the three values of m on the binary instances. The trend here is much clearer, indicating a strong positive relation between number of positive eigenvalues and average runtime. This is somewhat expected, as this increases the number of branching directions, and hence branches, of Algorithm 11. Therefore, without more sophisticated node selection and branching rules, this trend will likely remain.

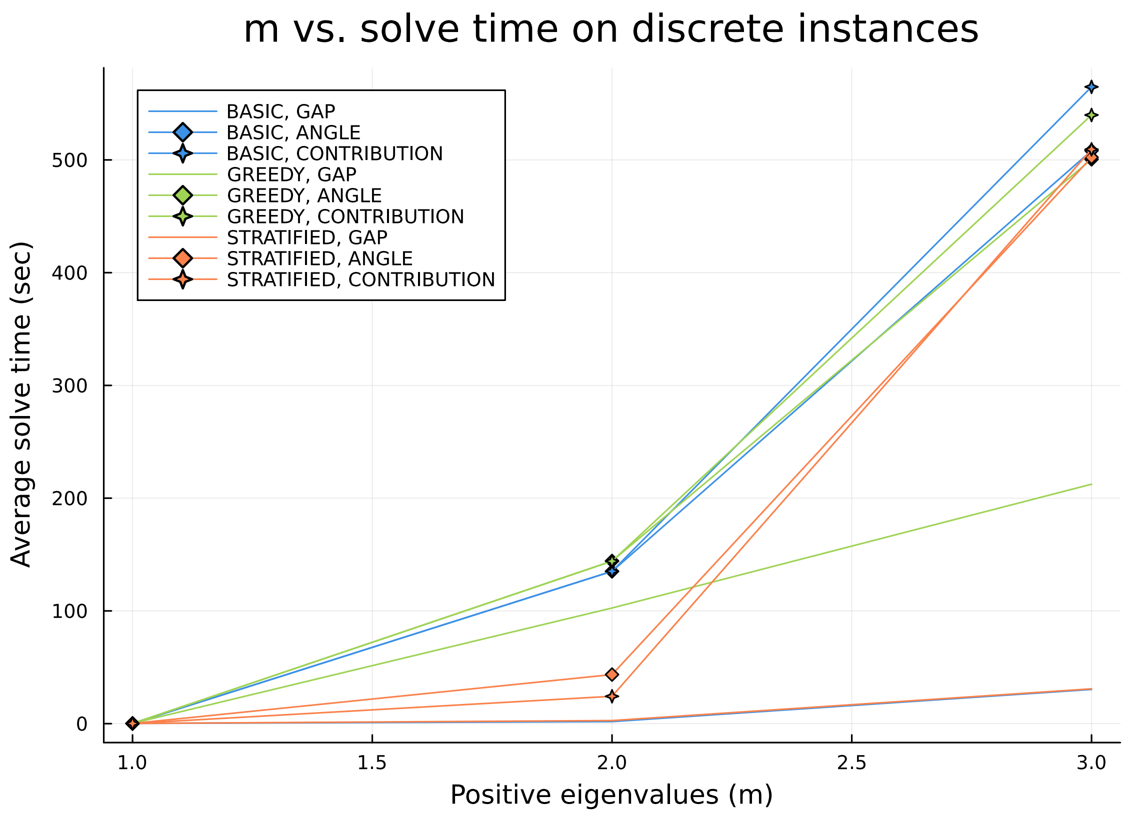

Figure 5.7: Number of positive eigenvalues against average run time for the binary test instances, broken down by decomposition and branching strategy.

Problem Type	n	Average Positive Evals (m)	DCBNC Num Solved	DCBNC Ave Solve Time (sec)	DCBNC Ave Gap (%)	Gurobi Num Solved	Gurobi Ave Solve Time (sec)	Gurobi Ave Gap (%)
boolean	50	25.0	0	600.136	4339.5	7	207.191	3.096
boolean	100	50.0	0	601.048	9283.51	0	600.011	13.752
diversity	50	22.3	0	600.017	13143.7	10	119.803	0.000
diversity	100	46.2	0	600.095	20026.1	0	600.007	290.104
knapsack	50	22.6	0	600.016	718.371	10	2.925	0.000
knapsack	100	46.5	0	600.087	639.524	4	375.42	7.511

Table 5.4: Number solved out of ten, average solve time (seconds) and average gap (%) on the highly convex instances. Additionally, we show the average number of positive eigenvalues m of each instance.

5.3.5 Highly Convex Instances

Thus far we have only experimented with matrices with 1, 2 or 3 positive eigenvalues. However, majority of literature on the test problems mentioned do not assume the number of positive eigenvalues, and hence may take any value. We finish this section with a repeat of the binary variable experiments from before, altering the way we generate the matrix so as not to control the number of positive eigenvalues. For the unconstrained quadratic programming problem, each element of Q gets a uniformly randomly generated integer in the range $[-100, 100]$, aligning with the procedure from Pardalos and Rodgers (1990). For the diversity and knapsack problems, each integer element of Q is randomly generated in the range $[0, 100]$. All other parameters are kept the same.

The results over a 10 minute time limit are shown in Table 5.4. As we can see, with a large m these instances become incredibly difficult for Algorithm 11 to solve. In fact, the algorithm is unable to solve a single instance within the time limit, and on average achieves a very poor objective bound. However, this poor objective bound more likely due to poor branching rules and tree formulation, rather than weak cuts. On the other hand, Gurobi performs similarly to the previous experiments, and is able to solve a majority of instances.

5.4 Conclusion

In this chapter, we introduced a new method for solving quadratic programming problems through a unique algorithm that focuses on the concept of directional concavity. This concept allows us to determine whether the tangent plane of y is valid at an x, even when

$f(x)$ is nonconcave. We outlined the key mechanisms to assert directional concavity on general matrices and showed how, when Q contains one positive eigenvalue, we can combine directionally concave tangent planes with convex over-envelopes to form an upper approximation of $f(x)$ everywhere. By dividing $\mathbb{R}^n$ into four subspaces, this approximation can be linearised and then used in our globally convergent branch and cut algorithm.

Extensive numerical results proved the effectiveness of this new approach. When Q contains few positive eigenvalues and thus a small convex component, our proposed algorithm is effective and can solve many large scale quadratic programming problems. In particular, on diversity problem instances, the approach proved very efficient and presented significant improvements compared to the state-of-the-art Gurobi benchmark. However, on matrices with many positive eigenvalues, the algorithm generally struggled due to the large number of branching directions and functional components.

6 Application: Bayer Digestion Maintenance Optimisation[1]

This final chapter describes a maintenance scheduling model for digester banks. Digester banks are network-connected assets that lie on the critical path of the Bayer process, a chemical refinement process that converts bauxite ore into alumina. The banks require different maintenance activities at different due times. Furthermore, the maintenance schedule is subject to production-related constraints and resource limitations. Given the complexity of scheduling maintenance for large fleets of digester banks, a continuous-time, mixed-integer linear program is formulated to find a cost-minimising maintenance schedule that satisfies all required constraints. A solution approach that employs lazy constraints and Benders decomposition is proposed to solve the model. Unlike generic implementations of Benders decomposition, we show that the subproblems can be solved explicitly using a specialist algorithm. We solve the scheduling model for realistic scenarios involving two Bayer refineries based in Western Australia.

6.1 Introduction

Refining operations are, by their nature, asset-intensive endeavours. Under harsh environmental and operational conditions, all assets and equipment inevitably face some form of degradation. Without maintenance interventions, degrading asset health can lead to safety concerns, equipment failure, and loss of production. Maintenance operations were once considered retroactive tasks conducted after a failure, but now proactive maintenance strategies have become mainstream within the resources industry and make up a substantial proportion of the operating costs of large-scale refineries (de Jonge & Scarf, 2020). Determining the optimal maintenance strategy can reduce running costs and enable more sustainable production, giving organisations a vital competitive advantage. This chapter looks at building a maintenance scheduling model for digester banks, a critical asset used in the Bayer process.

The Bayer process is a chemical refinement process that converts bauxite ore into alumina. In the digestion phase of the Bayer process, bauxite slurry is mixed with hot caustic liquor in large banks of pressure vessels that act like pressure cookers (Li et al.,

[1]This chapter is based on Spiers, Bui, Loxton, et al. (2023).

2015). Once processed, the slurry leaves the bank as supersaturated alumina in solution, also known as green liquor. This liquor is then passed on to the production units, where the Bayer process is continued.

In this chapter, we outline the practical scheduling requirements for digester bank maintenance and formulate an appropriate scheduling model. We use a continuous time framework to ensure timing accuracy and robustness to increasing time horizons. To assist in solving the model at large dimensions, a Benders decomposition algorithm is introduced where the subproblems are solved using a specialist algorithm. Additionally, lazy constraints are used to better handle a proportion of the constraint set. Finally, model performance is evaluated on two real-life case studies involving Bayer refineries in Western Australia, as well as several test instances.

6.1.1 Digester Bank Maintenance Activities

Routine preventative maintenance activities are crucial to maintaining the health of the digester banks. The maintenance activities generally fall into two categories: bank cleanings and bank services. Due to a chemical reaction in the Bayer process, scale builds up on the digester equipment. Scale build-up is unavoidable and is often thick and very hard (Cheng et al., 2021). To manage scale build-up, digester banks require frequent cleanings. A cleaning activity involves taking a bank offline and mechanically removing the scale until it is all removed. Bank cleanings are relatively small maintenance activities that must be planned frequently to sustain bank health and avoid unplanned failures. Furthermore, due to some physical requirements regarding scale build up, a cleaning must be scheduled immediately after a bank is take offline, regardless of how long the bank was previously in operation for. On the other hand, a bank service is a major activity that may only occur following a bank cleaning. During a bank service, worn or broken components may be replaced or repaired. In practice, it is common for digester banks to be cleaned several times in between services.

Scale only builds up when a bank is operational, meaning maintenance due times are calculated based on operational time, not elapsed time. As services are major activities that occur less frequently than cleanings, it is common for a bank to be cleaned several times between services, and therefore have several distinct operational periods. Changing the schedule of cleanings affects a bank's operational time and hence changes when its next service is due. This presents a significant challenge for schedulers, as minor changes in the schedule of cleaning activities can lead to significant changes in the operational time of a bank, and hence when its next service should occur.

A service can only occur immediately after a cleaning, meaning aligning these activities so they are due at similar times is beneficial. The best alignment of maintenance activities is a common challenge found in the resources sector, where assets may be subject to different levels of maintenance activities that are based on different cycle times (Seif

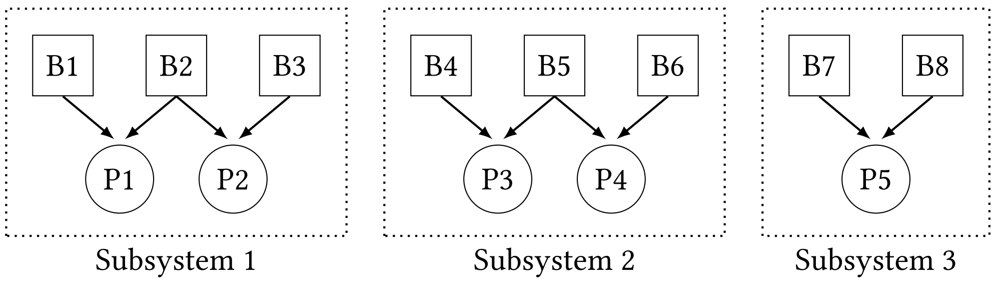

Figure 6.1: Example digester setup with eight digester banks feeding five production units, split into three subsystems.

et al., 2020). A trade-off must be made when the due times of two activities that must be scheduled together are misaligned. This compromise is often encountered when scheduling digester bank maintenance, as cleaning and service activities are rarely aligned, but must be completed together.

Digester banks sit on the refineries' critical path of production, meaning unplanned failures or downtimes can significantly impact site-wide production. To mitigate the risk of production loss due to digester bank failure, redundant digester banks are introduced. In a redundancy-based system, the number of available assets in a system exceeds the minimum required (Olde Keizer et al., 2016; Siopa et al., 2015). This allows production to be kept at full capacity, even when a bank is offline and receiving maintenance. Moreover, in the event of a failure of an operational bank, a standby bank can be brought online to rectify any potential production loss quickly. To demonstrate this, consider the example shown in Figure 6.1, where the entire digestion system is split into several independent subsystems. In each subsystem, the number of available banks is more than production units, and hence not every bank is required to be operational at all times. From a scheduling perspective, managing the redundant digester banks is crucial. The redundant units allow the production load to be shared across all banks. This is a cost-effective strategy for refineries, as bank utilisation can be maximised without halting production. However, finding the optimal operational balance between banks and usage of the redundant bank is a challenging task.

Digester bank maintenance should align with the maintenance activities of the downstream production units, the most important of which is known as a *valve change*. Valve changes are relatively small activities that are essential to the operations of the entire refinery and are planned well in advance. After a valve change has been completed, the associated production unit must connect to a freshly cleaned digester bank. The difficulty with valve changes is that they are not always consistent, and are planned at fixed times. It is therefore very challenging to find a stable, repeatable maintenance pattern, as the schedule may be affected by an upcoming valve change. Subsequently, we need the maintenance schedule for digester banks to align with these significant activities.

Generally speaking, the revenue from production far outweighs the cost of preventative maintenance, meaning schedulers may be willing to over maintain to avoid the possibility of an unexpected failure. The tendency to over maintain is often exacerbated when the maintenance schedule is governed by complex or difficult-to-satisfy restrictions. This chapter aims to build an optimisation model to plan the maintenance activities for a fleet of digester banks. Given the importance of digestion in the Bayer process, improvements in its maintenance strategy can lead to cost savings and improved sustainability of production.

6.1.2 Scheduling Optimisation

The application of optimisation techniques to maintenance scheduling problems is a well-studied subject in the literature. For a recent review on maintenance optimisation including maintenance scheduling, the reader is directed to de Jonge and Scarf (2020). The vast majority of maintenance scheduling models can be separated into discrete or continuous-time models. In a discrete-time model, the desired planning or time horizon is broken up into many time windows or intervals. This allows for constraints to be formulated easily, as the schedule takes on a grid-like structure, thereby giving control to each discrete time point (Floudas & Lin, 2004). Discrete-time modelling has been successfully applied to many optimisation problems, such as job shop scheduling (Manne, 1960), the resource-constrained scheduling problem (Kopanos et al., 2014) and more complex parallel machine scheduling (Heydar et al., 2021). A significant challenge in discrete-time modelling is managing the dimension growth of time-indexed variables when the time horizon increases, or when greater time precision is required (Hooker, 2007). One method to overcome this difficulty is to move to a continuous-time model with discrete events.

Continuous-time models have often been proposed to overcome the challenges of their discrete-time counterparts (Kopanos et al., 2014). In continuous-time models, the decision variables are the continuous start times for the set of discrete events (Maravelias & Grossmann, 2003). This makes continuous time models more robust to increasing time horizons and can provide greater precision on event start times. Continuous-time modelling has been successfully applied to general problems such as the resource-constrained scheduling problem (Kopanos et al., 2014) and the multi-product batch process scheduling problem (Floudas & Lin, 2004; Maravelias & Grossmann, 2003), as well as to more complicated assign and schedule problems (Hooker, 2007). The continuous-time framework has recently been extended to event-based models. In event-based modelling, the time horizon is broken up into a set of events. Each event is then given a continuous variable denoting its start time, and binary variables are used to match activities to events. Koné et al. (2011) formulate two event-based models for the resource-constrained project scheduling problem and show that these models outperform others on a specific set of test instances. While continuous-time and event-based models are more stable when increasing the desired planning horizon, their scale increases with the number of discrete events that are to be scheduled. Furthermore, to attain the same level of control on the timing of events as discrete-time models, continuous-time models may require the inclusion of big-M constraints, which create poor LP relaxations (Maravelias & Grossmann, 2003). Overcoming these challenges can be achieved through the use of an appropriate solution approach.

A suitable solution approach can allow the model to be solved faster and at larger scales. The selection of solution approach often depends on the specific model structure, and the attributes that can be exploited. Lazy constraints are one such approach that can help solve problems that contain large constraint sets (Pearce & Forbes, 2018). The technique selects a subset of constraints called the lazy constraints, which are removed from the original problem to form a reduced problem and a set of lazy constraints. The reduced problem is then solved using a generic procedure, and whenever a solution is found, it is checked to see which lazy constraints are violated. Violated lazy constraints are then returned to the original problem. When the constraint set of the problem is large, but only a small portion are active in the optimal solution, the use of lazy constraints can substantially reduce the problem size and complexity. If only a tiny portion of lazy constraints must be reintroduced to the problem, then the solver benefits from a far simpler problem. This technique has successfully been applied to the classical travelling salesman problem (Miller et al., 1960), the resource-constrained scheduling problem (Lerch & Trautmann, 2019) and a network maintenance scheduling problem (Pearce & Forbes, 2018). For an in-depth analysis of the benefits of lazy constraints, the reader is directed to the PhD thesis Pearce (2019).

Another solution approach commonly used to assist in solving large scale scheduling problems is Benders decomposition. Benders decomposition is a partitioning technique that can break up a complex mixed-integer linear program (MILP) into easier to solve problems (Benders, 1962). Using Benders decomposition, the original problem is broken up into a mixed-integer master problem and several potentially independent continuous subproblems. The technique has been successfully applied to a wide range of optimisation problems. For an in-depth review of Benders decomposition, the reader is directed to Rahmaniani et al. (2017). When the subproblems are independent and easy to solve, Benders decomposition can lead to significant computational advantages. For example, in Fischetti et al. (2016), the Benders subproblem of the uncapacitated facility location problem were shown to reduce to the continuous knapsack problem, which permits a closed-form solution. Similarly, in Pearce and Forbes (2018), the subproblem of a network maintenance scheduling problem was shown to be equivalent to the minimum cut problem, for which many solution algorithms already exist. In some cases, the subproblems may exhibit a unique structure that can be solved using a specialist algorithm, thereby circumventing the need for an LP solver. Contreras et al. (2011) show that the subproblems of the uncapacitated hub location problem were semi-assignment problems that could be solved efficiently with a specialist algorithm. Similarly, in Naoum-Sawaya and Buchheim (2016), the subproblems of a critical node selection problem were solved using a specialist algorithm derived from the Floyd Warshall algorithm. Specialist algorithms such as these can provide significant computational improvements when solving the subproblems.

6.2 Problem Formulation

The digester scheduling model should plan cleaning and service activities for a fleet of digester banks such that maintenance cost is minimised and all maintenance, operational and production-related constraints are adhered to. We now describe the key constraints in detail, and outline the assumptions made for the scheduling model.

For this model, we assume that the digestion system consists of multiple independent subsystems, as in Figure 6.1. Within each subsystem, we assume that there is always one redundant bank, i.e., the number of banks is one more than the number of production units. This means that exactly one bank is to be offline in each subsystem at all times. Additionally, we assume that a bank can only enter a standby period after completing a maintenance activity. Therefore, if a bank is to be taken offline for maintenance, the activity should be commenced immediately.

The due times for cleaning and service activities are based on operational time, not elapsed time. To ensure the model conducts maintenance on time, it must keep track of each bank's operational time and periods. In the case of cleaning activities, taking a bank offline requires that it immediately receives a cleaning and therefore, there is only ever one operational period between consecutive cleaning activities. However, this is not the case for service activities. A service is a significant activity that can only occur immediately after a cleaning activity, and has a longer operational due time than a cleaning activity. Therefore there will be several operational periods between consecutive service activities. The number of cleanings, and therefore operational periods, between consecutive service activities is unknown and may vary greatly depending on the specific bank and the required due times.

To demonstrate the complexity of operational due times, consider the example maintenance schedule for a single bank shown in Figure 6.2. The green represents periods where the bank is operational, yellow represents cleaning activities, purple represents services, and grey represents periods where the bank is on standby. The due time of the cleaning at t_5 is not calculated as the elapsed time since the previous cleaning at t_3, but instead the period where the bank is operational, which is given as t_4 to t_5. Similarly, the service activity planned at t_7 is calculated based on the previous three operational periods since the last service at t_1, which are t_2 to t_3, t_4 to t_5 and t_6 to t_7. To ensure these

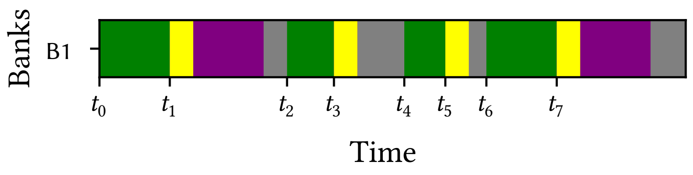

Figure 6.2: Example schedule for a single digester bank. Green represents operational periods, whereas yellow and purple represent cleaning and services activities respectively. Grey represents periods where the bank is on standby.

activities are conducted on time, the model should keep track of each bank's operational time at all points in time.

The maintenance activities should be scheduled such that production levels are always satisfied, and resourcing restrictions are adhered to. To ensure required production levels are always met, exactly one bank must be offline in each subsystem at all times. This means that whenever a bank commences a maintenance activity, the offline bank in that subsystem must come back online to ensure that production levels are satisfied. However, a bank may not come back online until all its planned maintenance activities are completed.

We assume the labour force restrictions for both cleanings and services take two forms; overlap penalties and maximum available resources. A cost penalty is incurred whenever two cleanings (or services) are planned simultaneously. This is known as an overlap penalty. The penalty is incurred only for the period where the two activities overlap. Additionally, we assume that three or more simultaneous cleaning or service activities are not permitted, as this exceeds the maximum available resources. While the model can easily be extended to consider a larger number of maximum available resources, this is considered beyond the scope necessary for practical implementations.

6.2.1 Model Setup

The digester scheduling model is a continuous-time, mixed-integer linear program. For the schedule to be practical, it must plan a considerable amount into the future and decide on the start times with a high level of accuracy. For this reason, a discrete-time model is inappropriate, as the number of discrete time points required would be vast. Rather than being burdened by the enormous scale of a discrete-time model, a continuous-time model can span the same duration and provide a greater degree of accuracy with far fewer decision variables.

Given a set of available maintenance activities, the model should decide on the start time of each activity, which bank is to receive maintenance and whether the activity should be a cleaning or a service. For the entire digestion system, let B and P be the set of banks and production units respectively. We let S be the set of subsystems and hence, for every $s \in S$, $B_s \subset B$ gives the subset of banks and $P_s \subset P$ gives the subset of production units within that subsystem. Note that as each subsystem $s \in S$ has exactly one extra bank, the cardinality of B_s is one more than P_s. The model should determine the maintenance schedule up to a given time horizon, denoted by τ. To do so, let $I = \{0, 1, \dots, n\}$ be the set of available maintenance activities. The model need not use all available maintenance activities, in fact, we intentionally provide an overestimate of the number of maintenance activities needed, allowing the model to decide exactly how many are required. The decision variables and remaining parameters are defined as required when formulating the constraint set, which is done in the remainder of this section. All sets, parameters and decision variables are summarised in Table 6.1.

6.2.2 Scheduling Maintenance Activities

Let t_i be a continuous variable that gives the start time of maintenance activity $i \in I$. As the set I is an overestimate of the number of activities required, let binary decision variable u_i equal 1 if maintenance activity $i \in I$ is required, and 0 otherwise. Let binary decision variable $x_{i,b}$ equal 1 if activity $i \in I$ is a cleaning on bank $b \in B$. As a service can only ever occur immediately after a cleaning activity, we group these into one variable. Hence, let binary decision variable $y_{i,b}$ equal 1 if activity $i \in I$ is a service on bank $b \in B$. Within this activity, the cleaning is completed first and immediately after it the service is commenced. To track a bank's operational periods, let binary variable $z_{i,b}$ equal 1 if bank b has not yet been restarted by activity $i \in I$, following a recently completed maintenance activity.

The timing of the maintenance activities should be such that the schedule starts at zero, activities are completed in order, and the schedule lasts the time horizon, i.e.,

$$t_0 = 0, \qquad t_i \geq t_{i-1}, \qquad t_n \geq \tau, \qquad i = 1, \dots, n. \tag{6.1}$$

The use of $t_0 = 0$ ensures the schedule commences immediately, and thus by having $t_n \geq \tau$, the schedule is guaranteed to last for the desired time horizon. The model should decide exactly how many maintenance activities it requires, and use these up in order until no more are required. This is formulated as

$$u_i \leq u_{i-1}, \quad i = 1, \dots, n. \tag{6.2}$$

Hence, once $u_i = 0$, none of the remaining activities will be used. If $u_i = 1$, then a maintenance activity must be started, therefore,

$$\sum_{b \in B} \left(x_{i,b} + y_{i,b} \right) = u_i, \quad \forall i \in I. \tag{6.3}$$

At any point in time, the number of banks offline, either receiving maintenance or on standby, is equal to number of subsystems (as there is one extra bank in each subsystem).

Sets	
I	Set of available maintenance activities.
S	Set of subsystems.
B	Set of digester banks.
P	Set of production units.
D	Set of pairs of banks that lead to a double bank change.
Parameters	
τ	Time horizon
α	Time to complete a cleaning.
β	Time to complete a service.
Θ	Operational due time of a cleaning.
Φ	Operational due time of a service.
A	Cost of completing a cleaning.
B	Cost of completing a service.
L	Penalty per day when planning simultaneous cleaning activities.
M	Penalty per day when planning simultaneous service activities.
Δ	Largest time between consecutive maintenance start times.
Continuous Decision Variables	
t_i	Start time of maintenance activity $i \in I$.
$\theta_{i,b}$	Cleaning operational time for bank $b \in B$ at the start of activity $i \in I$.
$\phi_{i,b}$	Service operational time for bank $b \in B$ at the start of activity $i \in I$.
λ_i	Amount of cleaning overlap for activity $i \in I$.
μ_i	Amount of service overlap for activity $i \in I$.
Binary Decision Variables	
u_i	1 if maintenance activity $i \in I$ is required.
$x_{i,b}$	1 if maintenance activity $i \in I$ is a cleaning on bank $b \in B$
$y_{i,b}$	1 if maintenance activity $i \in I$ is a service on bank $b \in B$
$z_{i,b}$	1 if bank $b \in B$ is remaining offline at the start of activity $i \in I$.

Table 6.1: Sets, parameters and decision variables.

Therefore, whenever an activity is started, there will be several banks remaining offline, either continuing a maintenance activity, or on standby after a recently completed activity. As these banks are offline, this period should not count towards their operational due times. To keep track of the periods that a bank remains offline after beginning maintenance, but before restarting, let binary variable $z_{i,b}$ equal 1 if bank b is yet to be restarted following a recent maintenance, at the start of activity $i \in I$. Then $z_{i,b}$ may be updated with the following constraints,

$$\begin{aligned}
& z_{i,b} + x_{i,b} + y_{i,b} \leq 1, && \forall b \in B, i \in I, && (6.4)\\
& z_{i,b} \leq x_{i-1,b} + y_{i-1,b} + z_{i-1,b}, && \forall b \in B, i = 1, \dots, n, && (6.5)\\
& z_{i,b} \leq 1 - \sum_{b' \in B_s} \left(x_{i,b'} + y_{i,b'} \right), && \forall s \in S, b \in B, i \in I, && (6.6)\\
& x_{i,b} + y_{i,b} \leq 1 - x_{i-1,b} - y_{i-1,b}, && \forall b \in B, i = 1, \dots, n, && (6.7)\\
& \sum_{b \in B} z_{i,b} = |S| - u_i, && \forall b \in B, i \in I. && (6.8)
\end{aligned}$$

Constraint (6.4) ensures that each bank is in at most one state in each period and (6.5) ensures that $z_{i,b} = 0$ if in the previous period bank b was not cleaned, serviced or remaining offline. Constraint (6.6) ensures that a bank cannot remain on standby if a new maintenance activity is planned in the same subsystem (and therefore forcing the bank to come online). Constraint (6.7) ensures that the same bank does not have maintenance activities planned in consecutive activities. Lastly, constraint (6.8) ensures that the correct number of banks are chosen to have $z_{i,b} = 1$.

Note that for the purpose of our model, the schedule is assumed to start immediately, with the first activity planned at $t_0 = 0$. However, in practice, the scheduling of the first maintenance activity is often influenced by the current state of the bank setup. This includes situations where the schedule should begin with banks on standby, rather than immediately starting a maintenance activity. In such cases, constraints (6.3) and (6.8) can be modified to exclude the case where $i = 0$ and thus allow $x_{0,b} = y_{0,b} = 0$ for all $b \in B$. Additionally, the values of $z_{0,b}$ can be constrained appropriately to match the banks that are currently offline. This allows the model to begin with the correct banks on standby, without affecting subsequent maintenance intervals.

Let α be the time to complete a cleaning activity, and let β be the time to complete a full service, including the cleaning. As a service is a major activity that has a cleaning as one of its subtasks, we assume that $\alpha < \beta$. Maintenance activities must be timed such that there is only ever one bank in each subsystem offline at any time, thus ensuring production levels are always maintained. Additionally, across all banks there may never be three or more simultaneous cleanings or services. Finally, the timing of maintenance activities should account for overlapping activities, such that the overlap can be included in the objective function as a penalty.

Consider first the timing of cleaning activities. As a service always begins with a cleaning activity, a cleaning is undergone at the start of every used maintenance activity. Therefore, to avoid three simultaneous cleanings, whenever an activity i is planned for maintenance, we must have $t_{i+2} \geq t_i + \alpha$, as this ensures the cleaning in i has finished, before the cleaning in $i+2$ starts. This can formulated as so,

$$t_i \geq t_{i-2} + \alpha u_{i-2}, \quad i = 2, \dots, n. \tag{6.9}$$

To avoid overlapping cleanings within the same subsystem, consecutive activities may not overlap if they are planned for the same subsystem. This can be formulated as,

$$t_i \geq t_{i-1} + \alpha \left(\sum_{b \in B_s} \left(x_{i-1,b} + y_{i-1,b} + x_{i,b} + y_{i,b} \right) - 1 \right), \quad \forall s \in S, i = 1, \dots, n. \tag{6.10}$$

Whenever consecutive maintenance activities are planned for the same bank subsystem, i.e., $\sum_{b \in B_s} \left(x_{i-1,b} + y_{i-1,b} + x_{i,b} + y_{i,b} \right) = 2$, constraint (6.10) becomes $t_i \geq t_{i-1} + \alpha$. Therefore, the cleaning must be completed before the next one is started, thus maintaining production levels.

To introduce the overlap penalties for cleaning activities, let λ_i be a continuous variable that gives the amount of overlap between the cleaning activities in $i-1$ and i. Then

$$0 \leq \lambda_i \leq \alpha, \quad \forall i \in I. \tag{6.11}$$

The model may decide how much cleaning overlap to accept with the following constraint,

$$t_i \geq t_{i-1} + \alpha u_{i-1} - \lambda_i, \quad i = 1, \dots, n. \tag{6.12}$$

Whenever the cleaning component of activities i and $i-1$ overlap, λ_i will get the amount of overlap. This is then included as a penalty in the objective function.

Avoiding clashes in service activities is slightly more challenging as a service activity may span several maintenance activities, however the constraints can be formulated in an analogous way to (6.9), (6.10) and (6.12), and are give as

$$t_j \geq t_i + \beta \left(\sum_{b \in B_s} \left(y_{i,b} + x_{j,b} + y_{j,b} \right) - 1 \right), \quad \forall s \in S, i, j \in I : i < j, \tag{6.13}$$

$$t_j \geq t_i + \beta \left(\sum_{b \in B_s} \left(y_{i,b} + y_{j,b} \right) - 1 \right) - \alpha - \mu_j, \quad \forall s \in S, i, j \in I : i < j, \tag{6.14}$$

$$t_k \geq t_i + \beta \left(\sum_{b \in B_s} \left(y_{i,b} + y_{j,b} + y_{k,b} \right) - 2 \right) - \alpha, \quad \forall s \in S, i, j, k \in I : i < j < k. \tag{6.15}$$

Constraint (6.13) ensures that no activity is commenced in a subsystem until enough time has passed since the last conducted service in that subsystem, similar to constraint (6.10).

Constraint (6.14) ensures that whenever two activities have services that overlap, μ_j is forced to get the amount of overlap, which can then be included in the objective function as a cost penalty. As a service includes the cleaning at the start of the activity, and goes into the service immediately afterwards, this should be offset by α such that the overlap only considers the service component. Finally, constraint (6.15) ensures three or more services are never planned simultaneously. While the number of constraints introduced here is large, few are expected to be binding in the optimal solution, as services are major activities that are sparsely scheduled.

6.2.3 Maintenance Due Times

To ensure banks are maintained on time, the model should count the time each bank is operational for. Let $\theta_{i,b}$ and $\phi_{i,b}$ be bank b's operational time since its last cleaning and service respectively, at the start of activity $i \in I$. Then, let Θ and Φ be the operational due times of a cleaning and service activity, respectively. As a service is a major activity, we assume that $\Theta < \Phi$. In practice, we expect Φ to be several times greater than Θ. Lastly, let $\tilde{\theta}_b$ and $\tilde{\phi}_b$ give the starting operational time for bank $b \in B$ since its last cleaning or service respectively, and let Δ be a parameter such that $\Delta \geq t_i - t_{i-1}$ for $i = 1, \dots, n$. Note that we can always choose $\Delta = \Theta$.

Observe that the model does not require the exact values of θ and ϕ at every maintenance activity, it only requires that maintenances are scheduled on time. Furthermore, the model has no benefit from having large values of θ and ϕ, as this leads to more required maintenance activities due to longer bank runtimes. For this reason, rather than forcing the model to calculate the exact values of θ and ϕ, we can instead provide exact lower bounds. For ϕ, this can be achieved with,

$$\phi_{0,b} = \tilde{\phi}_b, \qquad \forall b \in B, \tag{6.16}$$

$$\phi_{i,b} \geq \phi_{i-1,b} + t_i - t_{i-1} - (\Phi + \Delta)\, y_{i-1,b} - \Delta\left(x_{i-1,b} + z_{i-1,b}\right), \qquad \forall b \in B, i = 1, \dots, n, \tag{6.17}$$

$$\phi_{i,b} \geq \phi_{i-1,b} - \Phi y_{i-1,b}, \qquad \forall b \in B, i = 1, \dots, n, \tag{6.18}$$

$$0 \leq \phi_{i,b} \leq \Phi, \qquad \forall b \in B, i \in I. \tag{6.19}$$

Any feasible solution satisfies (6.17) to (6.19), and therefore

$$\phi_{i,b} \geq \max\left\{\phi_{i-1,b} + t_{i+1} - t_i - (\Phi + \Delta)\, y_{i-1,b} - \Delta\left(x_{i-1,b} + z_{i-1,b}\right), \phi_{i-1,b} - \Phi y_{i-1,b}, 0\right\}$$

for $i = 1, \dots, n$. If, during the previous activity, the bank was operational, then $x_{i-1,b} + y_{i-1,b} + z_{i-1,b} = 0$ and hence

$$\phi_{i,b} \geq \max\left\{\phi_{i-1,b} + t_{i+1} - t_i, \phi_{i-1,b}, 0\right\} = \phi_{i-1,b} + t_{i+1} - t_i,$$

as required. Alternatively, if the bank was cleaned or on standby, then $x_{i-1,b} + z_{i-1,b} = 1$ and $y_{i-1,b} = 0$ and hence

$$\phi_{i,b} \geq \max\left\{\phi_{i-1,b} + t_{i+1} - t_i - \Delta, \phi_{i-1,b}, 0\right\} = \phi_{i-1,b},$$

as required. Finally, if the bank was serviced, then $x_{i-1,b} + z_{i-1,b} = 0$ and $y_{i-1,b} = 1$ and hence

$$\phi_{i,b} \geq \max\left\{\phi_{i-1,b} + t_{i+1} - t_i - \Phi - \Delta, \phi_{i-1,b} - \Phi, 0\right\} = 0,$$

as required. The same logic can then be applied to θ,

$$\theta_{0,b} = \tilde{\theta}_b, \qquad \forall b \in B, \tag{6.20}$$

$$\theta_{i,b} \geq \theta_{i-1,b} + t_i - t_{i-1} - (\Theta + \Delta)\left(x_{i-1,b} + y_{i-1,b}\right) - \Delta z_{i-1,b}, \qquad \forall b \in B, i = 1, \dots, n, \tag{6.21}$$

$$0 \leq \theta_{i,b} \leq \Theta, \qquad \forall b \in B, i \in I. \tag{6.22}$$

This provides an appropriate formulation to update the exact lower bounds on θ and ϕ, and thus ensure banks are maintained on time.

6.2.4 Double Bank Changes

Bank switching may only take place between banks that can connect to the same production unit. For example, consider subsystem 1 of Figure 6.1. Bank 1 may not be switched with bank 3, as they do not share a common production unit. However bank 1 may be switched with bank 2. Switching banks 1 and 3 is known as a *double bank switch*, and must be avoided. Let D_s be the set of pairs of banks in subsystem $s \in S$ that cannot connect to the same production unit. For instance, $(b_1, b_2) \in D_s$ if $b_1, b_2 \in B_s$ and they cannot connect to the same production unit. To avoid a double bank change, their maintenance activities should not take place in consecutive intervals. Hence,

$$x_{i,b_1} + y_{i,b_1} \leq 1 - x_{i-1,b_2} - y_{i-1,b_2} - z_{i-1,b_2}, \qquad \forall s \in S, (b_1, b_2) \in D_s, i = 1, \dots, n. \tag{6.23}$$

6.2.5 Estimating Number of Required Maintenance Activities

An appropriate upper bound for the number of maintenance activities required to span a time horizon is given by total number of activities that could potentially be scheduled. This is given by the schedule that assumes constant cleaning activities with full overlap. Let $f : \mathbb{R} \to \mathbb{Z}$ be an integer-valued function that determines an upper bound on the number of activities required to span a given time horizon. Then,

$$f(t) := 2\left\lfloor \frac{t}{\alpha} \right\rfloor, \tag{6.24}$$

where $t \geq 0$ is the time frame the schedule should span.

6.2.6 Valve Changes

On every valve change, the schedule requires that a freshly cleaned digester bank is connected to the associated production unit. To account for valve changes, we can break the schedule up into distinct time windows, separated by the planned valve changes. Within each window a set of maintenance activities are provided that can be used to schedule activities. As before, this set of activities is an over-estimate of the number required. The model should then use these maintenance activities to find a schedule that starts exactly on a valve change, and finishes at the next.

Let $V = (v_1, \dots, v_m)$ be a vector of m planned valve changes, such that $v_1 < v_2 < \cdots < v_m < \tau$ and let $P = (p_1, \dots, p_m)$ denote the associated production unit of each valve change. The time horizon can be broken up into $m+1$ distinct time windows, denoted by the set $W = \{1, \dots, m+1\}$. The first m time windows end at the next valve change, while the last window ends at τ. Let $n_1 = f(v_1)$, $n_w = f(v_w - v_{w-1})$ for $w = 2, \dots, m$, and $n_{m+1} = f(\tau - v_m)$ be the number of activities required to span each time window. Let $I = \{0, \dots, n\}$ be defined as before, except now let $n = \sum_{w \in W} n_w$. Lastly, let $J = \{0, n_1, n_1 + n_2, \dots, n_1 + \cdots + n_m\}$ be the set that denotes the starting activity of each time window. Whereas in the previous formulation the model used up maintenance activities until no more were required over the entire time horizon, this should be amended to within each time window. To formulate this, constraint (6.2) should be updated to the following,

$$u_i = 1, \qquad \forall i \in J, \tag{6.25}$$
$$u_i \leq u_{i-1}, \quad \forall i \in I \setminus J. \tag{6.26}$$

Therefore $u_i = 1$ for the first activity of every time window, given by $i \in J$. Within each time window, activities are used up until no more are required.

Now that it is known what activities are to be planned at valve changes, the following constraints ensure the schedule adheres to the valve change,

$$t_i = v_w, \qquad w = 1, \dots, m, i = \sum_{j=1}^{w} n_j, \tag{6.27}$$
$$\sum_{b \in B_{p_w}} \left(x_{i,b} + y_{i,b} \right) \geq 1, \qquad w = 1, \dots, m, i = \sum_{j=1}^{w} n_j, \tag{6.28}$$
$$\sum_{b \in B_{p_w}} \left(x_{i-1,b} + y_{i-1,b} + z_{i-1,b} \right) \geq 1, \quad w = 1, \dots, m, i = \sum_{j=1}^{w} n_j, \tag{6.29}$$

where $B_{p_w} \subset B$ gives the subset of banks that can connect to production unit $p_w \in P$. Constraint (6.27) ensures that the activities associated with valve changes are time appropriately. Constraint (6.28) ensures that the bank planned for maintenance in $i \in J$ can connect to the appropriate production unit. Similarly, constraint (6.29) ensures that in the previous activity the bank offline can also connect to the appropriate production unit. In tandem, these constraints ensure a bank switch is occurring on the production unit, thus meeting the requirements of a valve change.

6.2.7 Objective Function

The objective of this model is to determine the maintenance schedule that optimises cost. This cost is made up of planned maintenance activities as well as overtime cost from overlapping activities. Let A and B be the cost of a cleaning and service activity respectively and let L and M be the additional cost per day of operating two cleaning and service activities respectively. The full model, denoted by OP, can then be formulated as so,

$$
\begin{aligned}
[OP] \qquad \min \quad & \sum_{i \in I} \left(\sum_{b \in B} (Ax_{i,b} + By_{i,b}) + L\lambda_i + M\mu_i \right), \\
\text{subject to} \quad & (6.1),(6.3) \text{ - } (6.23),(6.25)\text{-}(6.29), \\
& x_{i,b}, y_{i,b}, z_{i,b} \in \{0,1\}, \quad \forall i \in I, b \in B, \\
& u_i \in \{0,1\}, \quad \forall i \in I, \\
& \theta_{i,b}, \phi_{i,b} \geq 0, \quad \forall i \in I, b \in B, \\
& t_i, \lambda_i, \mu_i \geq 0, \quad \forall i \in I,
\end{aligned}
$$

6.3 Solution Algorithm

The original problem OP is solved using a combination of lazy constraints and Benders decomposition. Additionally, valid inequalities are introduced to further tighten the problem formulation.

Lazy constraints are a modelling technique that removes a subset of constraints from the original problem to form a reduced problem. The lazy constraints are then added back to the reduced problem only when the solver deems that the constraint is necessary. If only a small fraction of the lazy constraints are required to be added back in order to find the optimal solution, then the solver benefits from a far simpler problem, as a large number of unnecessary constraints have been removed.

Services are significant activities that are expected to be planned infrequently. In many practical examples, the total number of services planned may be as little as one-fifth of the total number of cleaning activities. However, the constraints required to ensure service clashes, in particular (6.15), represent a significant proportion of total constraints for the model. As only a few services are expected to be scheduled, the number of active constraints in this set is small. Therefore, by formulating (6.15) as lazy constraints, we are able to reduce the constraint set to only those that are necessary.

Benders decomposition is a partitioning technique that can break up the original problem into a mixed-integer master problem and several potentially independent continuous subproblems. Once a solution to the master problem is found, this solution is used to construct dual subproblems. By solving the dual subproblems, feasibility and optimality cuts may be generated and added to the master problem to either remove this solution, or improve the objective value.

By applying Benders decomposition to the variables θ and ϕ, we construct the *lifetime subproblems* using the maintenance due time constraints (6.16)-(6.22). These subproblems are used to identify periods in which a bank is overrun with respect to its cleaning and service lifetimes. As such, the lifetime subproblems are purely feasibility problems. The subproblems can be solved independently by separating them into each bank's cleaning and service lifetime subproblem. If a subproblem is infeasible, i.e., a bank is overrun, appropriate feasibility cuts are added to the master problem to remove this solution and ensure the bank is maintained on time. Finally, to tighten the master problem formulation, a set of valid inequalities based on practical assumptions of the problem is introduced.

6.3.1 Master Problem

The master problem (denoted by *MP*) follows a natural interpretation. It determines a candidate maintenance schedule that minimises the total cost due to planned maintenance activities and overlap penalties. The schedule should avoid double bank changes and be aligned with the planned valve changes. Additionally, the problem should satisfy a set of valid inequalities, added to tighten the master problem formulation. Lazy constraints are added to *MP* whenever a solution that violates a constraint in (6.15) is found. Finally, feasibility cuts are added whenever a schedule does not maintain a bank on time. Therefore, the master problem can be formulated as so,

$$\begin{aligned}
[MP] \quad & \min \quad && \sum_{i\in I}\left(\sum_{b\in B}\left(Ax_{i,b}+By_{i,b}\right)+L\lambda_i+M\mu_i\right),\\
& s.t. && (6.1),(6.3)\text{-}(6.14),(6.23),(6.25)\text{-}(6.29),\\
& && \text{Lazy Constraints},\\
& && \text{Feasibility Cuts},\\
& && \text{Valid Inequalities},\\
& && x_{i,b}, y_{i,b}, z_{i,b} \in \{0,1\}, \quad \forall i\in I, b\in B,\\
& && u_i \in \{0,1\}, \quad \forall i \in I,\\
& && t_i, \lambda_i, \mu_i \geq 0, \quad \forall i\in I,
\end{aligned}$$

where the feasibility cuts and valid inequalities are described in the following sections.

6.3.2 Lifetime Subproblems

The lifetime subproblems determine whether a candidate solution of *MP* overruns any of the banks with respect to either cleaning or service activities. If a bank is overrun, a feasibility cut is generated and added to *MP* to remove this solution. The subproblems are broken up into cleaning lifetimes (associated with variable θ) and service lifetimes (associated with variable ϕ) and then separated further by each bank $b \in B$. Therefore the number of lifetime subproblems is $2 \times |B|$. In general, feasibility cuts require using an LP solver to determine an extreme ray. Here, we show that the extreme rays of the lifetime subproblems can be generated using the exact operational time of each bank, thereby circumventing the need for an LP solver.

We begin this section by formulating the dual of the cleaning and service lifetime subproblems. We then show how the candidate schedule from the master problem can be used to calculate the exact operational time of each bank. The exact operational time of each bank is then used as a candidate solution for its lifetime subproblems. Notably, we show how a subproblem is feasible if and only if the exact operational time is feasible. Furthermore, this candidate solution can then be used to generate extreme rays, and thus feasibility cuts.

Given a fixed schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ for all $b \in B$, $i \in I$, the cleaning lifetime subproblems attempt to find values for $\theta_b \in \mathbb{R}^{n+1}$ such that constraints (6.20)-(6.22) are satisfied for each bank $b \in B$. The cleaning lifetime subproblem of bank $b \in B$ (denoted by θ_b-SP) is formulated as follows,

$$
\begin{aligned}
[\theta_b\text{-}SP] \quad \min \quad & 0 && (6.30)\\
s.t. \quad & \theta_0 \geq \tilde{\theta}, && (6.31)\\
& \theta_i - \theta_{i-1} \geq t_i - t_{i-1} - (\Theta + \Delta)\left(x_{i-1,b} + y_{i-1,b}\right) - \Delta z_{i-1,b}, \quad i = 1, \ldots, n, && (6.32)\\
& \theta_i \leq \Theta, \quad i = 0, \ldots, n, && (6.33)\\
& \theta_i \geq 0, \quad i = 0, \ldots, n. && (6.34)
\end{aligned}
$$

The dual problem of θ_b-SP can be written in terms of dual variables $\gamma \in \mathbb{R}^{n+1}$ and $\eta \in \mathbb{R}^{n+1}$ and is denoted by θ_b-DP,

$$
\begin{aligned}
[\theta_b\text{-}DP] \quad \max \quad & \tilde{\theta}\gamma_0 + \sum_{i=1}^{n}\left(t_i - t_{i-1} - (\Theta + \Delta)\left(x_{i-1,b} + y_{i-1,b}\right) - \Delta z_{i-1,b}\right)\gamma_i + \sum_{i=0}^{n}\Theta\eta_i\\
s.t. \quad & \gamma_i - \gamma_{i+1} + \eta_i \leq 0, \quad i = 0, \ldots, n-1,\\
& \gamma_n + \eta_n \leq 0,\\
& \gamma_i \geq 0, \quad i = 0, \ldots, n,\\
& \eta_i \leq 0, \quad i = 0, \ldots, n.
\end{aligned}
$$

Similarly, the service lifetime subproblems attempt to find values for $\phi_b \in \mathbb{R}^{n+1}$ such that constraints (6.16)-(6.19) are satisfied for each bank $b \in B$. The service lifetime subproblem of bank $b \in B$ (denoted by ϕ_b-*SP*) is formulated as follows,

$$
\begin{aligned}
[\phi_b\text{-}SP] \quad & \min \quad 0 && (6.35)\\
& s.t. \quad \phi_0 \geq \tilde{\phi}, && (6.36)\\
& \phi_i - \phi_{i-1} \geq t_i - t_{i-1} - (\Phi + \Delta)\, y_{i-1,b} - \Delta\left(x_{i-1,b} + z_{i-1,b}\right), \quad i = 1, \dots, n, && (6.37)\\
& \phi_i - \phi_{i-1} \geq -\Phi y_{i-1,b}, \quad i = 1, \dots, n, && (6.38)\\
& \phi_i \leq \Phi, \quad i = 0, \dots, n, && (6.39)\\
& \phi_i \geq 0, \quad i = 0, \dots, n. && (6.40)
\end{aligned}
$$

The dual problem of ϕ_b-*SP* can be written in terms of dual variables $\pi \in \mathbb{R}^{n+1}$, $\sigma \in \mathbb{R}^n$ and $\rho \in \mathbb{R}^{n+1}$ and is denoted by ϕ_b-*DP*,

$$
\begin{aligned}
[\phi_b\text{-}DP] \quad \max \quad & \Gamma_b = \tilde{\phi}\pi_0 + \sum_{i=1}^{n} \left(t_i - t_{i-1} - (\Phi + \Delta)\, y_{i-1,b} - \Delta\left(x_{i-1,b} + z_{i-1,b}\right)\right)\pi_i\\
& \quad - \Phi \sum_{i=1}^{n} y_{i-1,b}\sigma_i + \sum_{i=0}^{n} \Phi\rho_i\\
s.t. \quad & \pi_0 - \pi_1 - \sigma_1 + \rho_0 \leq 0,\\
& \pi_i - \pi_{i+1} + \sigma_i - \sigma_{i+1} + \rho_i \leq 0, \quad i = 1, \dots, n-1,\\
& \pi_n + \sigma_n + \rho_n \leq 0,\\
& \pi_i \geq 0, \quad i = 0, \dots, n,\\
& \sigma_i \geq 0, \quad i = 1, \dots, n,\\
& \rho_i \leq 0, \quad i = 0, \dots, n.
\end{aligned}
$$

If for bank $b \in B$, either θ_b-*SP* or ϕ_b-*SP* are infeasible, then the schedule generated by *MP* does not service the bank on time, and therefore feasibility cuts should be added to *MP* to remove this solution. To generate a feasibility cut, the exact operational times of each bank can be calculated, and provide all information required about θ_b-*SP* and ϕ_b-*SP*.

Definition 34. Let $\theta_b^* \in \mathbb{R}^{n+1}$ be the exact operational time of bank $b \in B$ since its' last cleaning, based on the schedule determined by *MP*. This is the operational time of the bank at every $i \in I$, if the schedule was followed in practice. Hence, the operational time increases whenever the bank is in operation, and resets to 0 whenever it is cleaned. Then θ_b^* can be calculated recursively as follows,

$$
\theta_{i,b}^* := \begin{cases} \tilde{\theta}_b, & i = 0,\\ \left(\theta_{i-1,b}^* + t_i - t_{i-1}\right)\left(1 - x_{i-1,b} - y_{i-1,b} - z_{i-1,b}\right), & i = 1, \dots, n. \end{cases} \tag{6.41}
$$

Similarly, let $\phi_b^* \in \mathbb{R}^{n+1}$ be the exact operational time of bank $b \in B$ since its' last service, based on the schedule determined by *MP*. Then ϕ_b^* can be calculated recursively as follows,

$$\phi^*_{i,b} := \begin{cases} \tilde{\phi}_b, & i = 0 \\ \phi^*_{i-1,b}\left(1 - y_{i-1,b}\right) + \left(t_i - t_{i-1}\right)\left(1 - x_{i-1,b} - y_{i-1,b} - z_{i-1,b}\right), & i = 1, \dots, n. \end{cases} \tag{6.42}$$

Lemma 35. *Let $\phi_b \in \mathbb{R}^{n+1}$ be such that it satisfies (6.36) - (6.38) and (6.40), and let $\phi^*_{i,b}$ be defined as in (6.42). Then $\phi_{i,b} \geq \phi^*_{i,b}$ for all $i \in I$.*

Proof. We prove $\phi_i \geq \phi^*_{i,b}$ for all $i \in I$ by induction on dimension i. The base case $i = 0$ holds because $\phi_{0,b}$ satisfies (6.36) and therefore $\phi_{0,b} \geq \tilde{\phi}_b = \phi^*_{0,b}$. Suppose $\phi_{i,b} \geq \phi^*_{i,b}$ holds for $i \leq n - 1$. We now prove that $\phi_{i+1,b} \geq \phi^*_{i+1,b}$ holds by considering three cases. From constraint (6.4) of *MP* either $x_{i,b} = y_{i,b} = z_{i,b} = 0$, or $x_{i,b} + z_{i,b} = 1$ and $y_{i,b} = 0$, or finally $y_{i,b} = 1$ and $x_{i,b} + z_{i,b} = 0$. As $\phi_{i,b}$ satisfies (6.37), (6.38) and (6.40) we have that

$$\phi_{i+1,b} \geq \max\left\{0, \phi_{i,b} + t_{i+1} - t_i - (\Phi + \Delta)\, y_{i,b} - \Delta\left(x_{i,b} + z_{i,b}\right), \phi_{i,b} - \Phi y_{i,b}\right\}. \tag{6.43}$$

Suppose firstly that $x_{i,b} = y_{i,b} = z_{i,b} = 0$, then from (6.43),

$$\begin{aligned} \phi_{i+1,b} &\geq \max\left\{0, \phi_{i,b} + t_{i+1} - t_i, \phi_{i,b}\right\} \\ &= \phi_{i,b} + t_{i+1} - t_i \\ &\geq \phi^*_{i,b} + t_{i+1} - t_i \\ &= \phi^*_{i+1,b}. \end{aligned}$$

Alternatively, suppose $x_{i,b} + z_{i,b} = 1$ and $y_{i,b} = 0$, then from (6.43),

$$\begin{aligned} \phi_{i+1,b} &\geq \max\left\{0, \phi_{i,b} + t_{i+1} - t_i - \Delta, \phi_{i,b}\right\} \\ &= \phi_{i,b} \\ &\geq \phi^*_{i,b} \\ &= \phi^*_{i+1,b}. \end{aligned}$$

Finally, suppose $y_{i,b} = 1$ and $x_{i,b} + z_{i,b} = 0$, then from (6.43),

$$\begin{aligned} \phi_{i+1,b} &\geq \max\left\{0, \phi_{i,b} + t_{i+1} - t_i - (\Phi + \Delta), \phi_{i,b} - \Phi\right\} \\ &= 0 \\ &= \phi^*_{i+1,b}. \end{aligned}$$

Therefore, by induction, $\phi_{i,b} \geq \phi^*_{i,b}$ for all $i \in I$. □

Lemma 36. *Let $\theta_b \in \mathbb{R}^n$ be such that it satisfies (6.31), (6.32), and (6.34), and let $\theta^*_{i,b}$ be defined as in (6.41). Then $\theta_{i,b} \geq \theta^*_{i,b}$ for all $i \in I$.*

Proof. The proof follows analogously from the proof of Lemma 35. □

We now show how θ_b^* and ϕ_b^* can be used to precisely determine the feasibility of θ_b-*SP* and ϕ_b-*SP*. The proofs are shown to hold for ϕ^* first, as this is the more complicated subproblem and the proofs for θ^* follow analogously.

Proposition 37. *ϕ_b-SP is feasible if and only if $\phi_{i,b}^* \le \Phi$ for all $i \in I$.*

Proof. We first prove the forward statement. If ϕ_b-*SP* is feasible, then there exists a feasible solution $\phi_{i,b}$ that satisfies (6.36) - (6.40). From Lemma 35 we therefore have $\phi_{i,b} \ge \phi_{i,b}^*$ for all $i \in I$. As $\phi_{i,b}$ satisfies (6.39), we have that $\phi_{i,b} \le \Phi$ for all $i \in I$ and therefore $\phi_{i,b}^* \le \Phi$ for all $i \in I$.

We now prove the reverse statement, by showing that if $\phi_{i,b}^* \le \Phi$ for all $i \in I$, then $\phi_{i,b}^*$ is a feasible solution to ϕ_b-*SP*. Firstly, if $\phi_{i,b}^* \le \Phi$ for all $i \in I$, then ϕ_b^* satisfies (6.39). From (6.42), we have that $\phi_{0,b}^* \ge \tilde{\phi}_b$, thereby satisfying (6.36). We now prove that $\phi_{i,b}^*$ satisfies (6.40) by induction on dimension i. The base case $i = 0$ holds because $\phi_{0,b}^* \ge \tilde{\phi}_b \ge 0$. Suppose $\phi_{i,b}^* \ge 0$ holds for $i \le n-1$. We now prove that $\phi_{i+1,b}^* \ge 0$ also holds. Recall from (6.42),

$$\phi_{i+1,b}^* = \phi_{i,b}^*\left(1 - y_{i,b}\right) + \left(t_{i+1} - t_i\right)\left(1 - x_{i,b} - y_{i,b} - z_{i,b}\right). \tag{6.44}$$

From constraint (6.1) we have $t_i - t_{i-1} \ge 0$. Additionally, from constraint (6.4) of *MP*, we can see that $1 - x_{i,b} - y_{i,b} - z_{i,b} \ge 0$ and $1 - y_{i,b} \ge 0$. Therefore, from (6.44) we can see that $\phi_{i+1,b}^* \ge 0$ as $\phi_{i,b}^* \ge 0$. Thus, by induction, we have proved $\phi_{i,b}^* \ge 0$ holds for $i \le n$ and hence $\phi_{i,b}^*$ satisfies (6.40). Recall from (6.42), for $i = 1, \dots, n$,

$$\begin{aligned}
\phi_{i,b}^* &= \phi_{i-1,b}^*\left(1 - y_{i-1,b}\right) + \left(t_i - t_{i-1}\right)\left(1 - x_{i-1,b} - y_{i-1,b} - z_{i-1,b}\right), \\
&= \phi_{i-1,b}^* - \phi_{i-1,b}^* y_{i-1,b} + t_i - t_{i-1} - \left(t_i - t_{i-1}\right)\left(x_{i-1,b} + y_{i-1,b} + z_{i-1,b}\right), \\
&\ge \phi_{i-1,b}^* - \Phi y_{i-1,b} + t_i - t_{i-1} - \Delta\left(x_{i-1,b} + y_{i-1,b} + z_{i-1,b}\right),
\end{aligned}$$

as $\phi_{i-1,b}^* \le \Phi$ and $t_i - t_{i-1} \le \Delta$, hence $\phi_{i,b}^*$ satisfies (6.37). Similarly for $i = 1, \dots, n$,

$$\begin{aligned}
\phi_{i,b}^* &= \phi_{i-1,b}^*\left(1 - y_{i-1,b}\right) + \left(t_i - t_{i-1}\right)\left(1 - x_{i-1,b} - y_{i-1,b} - z_{i-1,b}\right), \\
&\ge \phi_{i-1,b}^* - \phi_{i-1,b}^* y_{i-1,b}, \\
&\ge \phi_{i-1,b}^* - \Phi y_{i-1,b},
\end{aligned}$$

and therefore $\phi_{i,b}^*$ satisfies (6.38). As ϕ_b^* satisfies (6.36) - (6.40) it is a feasible solution to ϕ_b-*SP*, meaning the problem is feasible. □

Proposition 38. *θ_b-SP is feasible if and only if $\theta_{i,b}^* \le \Theta$ for all $i \in I$.*

Proof. Both forward and reverse proofs follow analogously from the proofs of Proposition 37. □

We now show that in addition to determining the feasibility of the lifetime subproblems, θ_b^* and ϕ_b^* can outline periods of bank overruns, which can in turn be used to generate valid feasibility cuts.

Definition 39 (Cleaning Overruns)**.** Let X_b denote the set of maintenance activities where bank $b \in B$ was operational at the previous activity,

$$X_b := \left\{ i \in I \; : \; \begin{cases} x_{i-1,b} + y_{i-1,b} + z_{i-1,b} = 0, & \text{if } i \geq 1, \\ x_{i,b} + y_{i,b} + z_{i,b} = 0, & \text{if } i = 0, \end{cases} \right\}. \tag{6.45}$$

This gives the set of activities where the cleaning operational time is non-decreasing. Then, the set of overrun cleaning lifetimes is given as C_b, where

$$C_b := \left\{ (i,j) \in I \times I \; : \; \forall k \in [i,j], k \in X_b, i-1 \notin X_b, \theta^*_{j,b} > \Theta \right\}. \tag{6.46}$$

If $(i,j) \in C_b$, then from activity i to activity j, bank b is always operational and at activity j, the bank is overdue for a cleaning.

Definition 40 (Service Overruns)**.** Let Y_b be the set of maintenance activities where bank $b \in B$ was not serviced in the previous activity,

$$Y_b := \left\{ i \in I \; : \; \begin{cases} y_{i-1,b} = 0, & \text{if } i \geq 1, \\ y_{i,b} = 0, & \text{if } i = 0. \end{cases} \right\}. \tag{6.47}$$

This gives the set of activities where the service operational time is non-decreasing. The set of overrun service lifetimes is given as V_b, where

$$V_b := \left\{ (i,j) \in I \times I \; : \; \forall k \in [i,j], k \in Y_b, i-1 \notin Y_b, \phi^*_{j,b} > \Phi \right\}. \tag{6.48}$$

If $(i,j) \in C_b$, then from activity i to activity j, bank b is never serviced and at activity j, the bank is overdue for a service.

Definition 41 (Standby Periods)**.** Let O_b be the set of activities where bank $b \in B$ was either receiving a cleaning, or on standby at the previous activity,

$$O_b := \left\{ i \in \{1, \dots, n\} \; : \; x_{i-1,b} + z_{i-1,b} = 1 \right\}. \tag{6.49}$$

Proposition 42. *$V_b = \emptyset$ if and only if ϕ_b-SP is feasible.*

Proof. We first prove the forward statement. From constraint (6.7) of *MP*, there always exists an $i \in I$ whereby $y_{i,b} = 0$, and therefore $Y_b \neq \emptyset$. If $Y_b \neq \emptyset$, then for any $j \in \{1, \dots, n\}$ such that $\phi^*_{j,b} > 0$, we must have $y_{j-1,b} = 0$ and therefore from (6.47), $j \in Y_b$. As $j \in Y_b$, there always exists an $i \leq j$ such that $\forall k \in [i,j]$ we have $k \in Y_b$ and $i-1 \notin Y_b$. However, if $V_b = \emptyset$, then for all $j \in Y_b$ such that $\phi^*_{j,b} > 0$, we must have $\phi^*_{j,b} \leq \Phi$, otherwise there exists an i such that $(i,j) \in V_b$. Therefore $\phi^*_i \leq \Phi$ for all $i \in I$, and hence from Proposition 37, ϕ_b-*SP* must be feasible.

The reverse statement is trivial. If ϕ_b-*SP* is feasible then from Proposition 37 we have $\phi^*_i \leq \Phi$ for all $i \in I$, and hence there is no $j \in I$ such that $\phi^*_{j,b} > \Phi$, therefore $V_b = \emptyset$. □

Proposition 43. *$C_b = \emptyset$ if and only if θ_b-SP is feasible.*

Proof. Both forward and reverse proofs follow analogously from the proof of Proposition 42. □

Definition 44 (Unbounded Direction)**.** Let $LP = \max\left\{a^T x \ : \ Ax \leq 0, x \in \mathbb{R}^n\right\}$ be a linear program, where $a \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$. Then $v \in \mathbb{R}^n$ is an *unbounded direction* of *LP* if $Av \leq 0$ and $a^T v > 0$.

Proposition 45. *Fix a schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ for all $b \in B$, $i \in I$. Suppose further that ϕ_b-SP is infeasible with respect to the schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$. Then, for all $(i, j) \in V_b$, define $\pi^*, \rho^* \in \mathbb{R}^{n+1}$ as*

$$\pi_k^* = \begin{cases} 1, & \text{if } k \in [i,j] \text{ and } k \in X_b, \\ 0, & \text{otherwise}, \end{cases} \qquad \rho_k^* = \begin{cases} -1, & \text{if } k = j, \\ 0, & \text{otherwise}, \end{cases}$$

for $k = 0, \dots, n$, and define $\sigma^* \in \mathbb{R}^n$ as

$$\sigma_k^* = \begin{cases} 1, & \text{if } k \in [i,j] \text{ and } k \in O_b, \\ 0, & \text{otherwise}, \end{cases}$$

for $k = 1, \dots, n$. Then $(\pi^*, \sigma^*, \rho^*)$ is an unbounded direction of ϕ_b-DP.

Proof. From Proposition 42, if ϕ_b-*SP* is infeasible with respect to the schedule given by $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ then V_b is non-empty, and therefore the vector $(\pi^*, \sigma^*, \rho^*)$ exists. Moreover, the problem ϕ_b-*DP* is of the form $\max\left\{a^T x \ : \ Ax \leq 0, x \in \mathbb{R}^n\right\}$, and therefore matches that in Definition 44. We now prove $(\pi^*, \sigma^*, \rho^*)$ is feasible solution to ϕ_b-*DP*. The schedule generated by the master problem must satisfy constraint (6.4) and hence $x_{i,b} + y_{i,b} + z_{i,b} \leq 1$ for all $i \in I$. Therefore from (6.45) and (6.49) we have that $X_b \cap O_b = \emptyset$ and hence for all $k \in [i, j]$ we have $\pi_k^* + \sigma_k^* = 1$. Therefore the only non-zero constraints are,

$$\begin{aligned} \pi_k + \sigma_k - \pi_{k+1} - \sigma_{k+1} + \rho_k &= 1 - 1 + 0 = 0 \leq 0, \quad k = i, \dots, j-1, \\ \pi_j + \sigma_j - \pi_{j+1} - \sigma_{j+1} + \rho_j &= 1 - 0 - 1 = 0 \leq 0, \end{aligned}$$

which are all satisfied and hence $(\pi^*, \sigma^*, \rho^*)$ is feasible solution ϕ_b-*DP*.

The objective value Γ_b is given as,

$$\Gamma_b = \tilde{\phi}\pi_0^* + \sum_{k=1}^{n} \left(t_k - t_{k-1} - (\Phi + \Delta)\, y_{k-1,b} - \Delta\left(x_{k-1,b} + z_{k-1,b}\right)\right) \pi_k^* \\ - \Phi \sum_{k=1}^{n} y_{k-1,b}\sigma_k^* + \Phi \sum_{k=0}^{n} \rho_k^*. \quad (6.50)$$

From (6.47) we have that $0 \in Y_b$ if and only if $1 \in Y_b$. Therefore from (6.48), for all $(i,j) \in V_b$, either $i = 0$ or $i \geq 2$. If $i \geq 2$, then (6.50) can be re-written with non-zero components,

$$\Gamma_b = \sum_{\substack{k\in[i,j]\\ k\in X_b}} \left(t_k - t_{k-1} - (\Phi + \Delta)\, y_{k-1,b} - \Delta \left(x_{k-1,b} + z_{k-1,b}\right)\right) - \Phi \sum_{\substack{k\in[i,j]\\ k\in O_b}} y_{k-1,b} - \Phi. \quad (6.51)$$

Now, from (6.49), for all $k \in O_b$ we have $x_{k-1,b} + z_{k-1,b} = 1$ and hence from constraint (6.4) of *MP* we have that $y_{k-1,b} = 0$. Furthermore, if $x_{k-1,b} + z_{k-1,b} = 1$, then from (6.42), $\phi^*_{k,b} - \phi^*_{k-1,b} = 0 = y_{k-1,b}$. Additionally, from (6.45), for all $k \in X_b$ we have $x_{k-1,b} = y_{k-1,b} = z_{k-1,b} = 0$ and therefore from (6.42) we have $\phi^*_k - \phi^*_{k-1} = t_k - t_{k-1}$. Then (6.51) can be simplified as follows,

$$\begin{aligned}\Gamma_b &= \sum_{\substack{k\in[i,j]\\ k\in X_b}} (t_k - t_{k-1}) + \sum_{\substack{k\in[i,j]\\ k\in O_b}} \left(\phi^*_{k,b} - \phi^*_{k-1,b}\right) - \Phi,\\ &= \sum_{\substack{k\in[i,j]\\ k\in X_b}} (\phi^*_k - \phi^*_{k-1}) + \sum_{\substack{k\in[i,j]\\ k\in O_b}} \left(\phi^*_{k,b} - \phi^*_{k-1,b}\right) - \Phi. \quad (6.52)\end{aligned}$$

Given $X_b \cap O_b = \emptyset$ we can simplify (6.52) as follows,

$$\begin{aligned}\Gamma_b = &\sum_{k\in[i,j]} (\phi^*_k - \phi^*_{k-1}) - \Phi,\\ =&\phi^*_j - \phi^*_{i-1} - \Phi,\\ =&\phi^*_j - \Phi > 0,\end{aligned}$$

as $\phi^*_{i-1} = 0$ and $\phi^*_j > \Phi$. Therefore $(\pi^*, \sigma^*, \rho^*)$ is an unbounded direction of ϕ_b-*DP*. The proof for the case where $i = 0$ follows analogously. □

Proposition 46. *Fix a schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ for all $b \in B$, $i \in I$. Suppose further that θ_b-SP is infeasible with respect to the schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$. Then, for all $(i,j) \in C_b$, define $\gamma^*, \eta^* \in \mathbb{R}^{n+1}$ as*

$$\gamma^*_k = \begin{cases} 1, & \text{if } k \in [i,j], \\ 0, & \text{otherwise}, \end{cases} \qquad \eta^*_k = \begin{cases} -1, & \text{if } k = j, \\ 0, & \text{otherwise}, \end{cases}$$

for $k = 0, \ldots, n$. Then (γ^*, η^*) is an unbounded direction of θ_b-DP.

Proof. The proof follows analogously from the proof of Proposition 45. □

Proposition 47. *Fix a schedule $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ for all $b \in B$, $i \in I$. For all banks $b \in B$ where $\phi_b - SP$ is infeasible,*

$$\sum_{\substack{k\in[i,j]\\ k\in X_b}} \left(t_k - t_{k-1} - (\Phi + \Delta)\, y_{k-1,b} - \Delta \left(x_{k-1,b} + z_{k-1,b}\right)\right) - \Phi \sum_{\substack{k\in[i,j]\\ k\in O_b}} y_{k-1,b} \leq \Phi,$$

$$\forall (i,j) \in V_b \, : \, i \geq 2, \quad (6.53)$$

$$\tilde{\phi} + \sum_{\substack{k\in[1,j]\\ k\in X_b}} \left(t_k - t_{k-1} - (\Phi+\Delta)\, y_{k-1,b} - \Delta\left(x_{k-1,b} + z_{k-1,b}\right)\right) - \Phi \sum_{\substack{k\in[1,j]\\ k\in O_b}} y_{k-1,b} \le \Phi, \quad \forall (i,j)\in V_b \,:\, i = 0, \qquad (6.54)$$

are valid feasibility cuts for MP.

Proof. These cuts are derived from the unbounded directions outlined in Proposition 45, and so from Benders (1962), they are valid cuts that remove this infeasible solution. Furthermore, the cuts only remove solutions that cause unbounded directions, and therefore infeasible subproblems. As such, the cuts do not remove other feasible solutions. Hence, they are valid feasibility cuts. □

Proposition 48. *Fix a schedule* $(x_{i,b}, y_{i,b}, z_{i,b}, t_i)$ *for all* $b\in B$, $i\in I$. *For all banks* $b\in B$ *where* $\theta_b - SP$ *is infeasible,*

$$t_j - t_{i-1} - \sum_{k=i}^{j}\left((\Theta+\Delta)\left(x_{k-1,b}+y_{k-1,b}\right) + \Delta z_{k-1,b}\right) \le \Theta, \quad \forall (i,j)\in C_b \,:\, i\ge 2,$$

$$t_j + \tilde{\theta}_b - \sum_{k=1}^{j}\left((\Theta+\Delta)\left(x_{k-1,b}+y_{k-1,b}\right) + \Delta z_{k-1,b}\right) \le \Theta, \quad \forall (i,j)\in C_b \,:\, i = 0.$$

are valid feasibility cuts for MP.

Proof. The proof follows analogously from the proof of Proposition 47. □

6.3.3 Valid Inequalities

Entirely removing the operational due time constraints from the master problem has the potential to make its formulation weak. In such cases, it is common to introduce a set of valid inequalities to tighten the master problem. Doing so can reduce the number of feasibility cuts required, thereby leading to faster convergence. However, if the valid inequalities are weak then the model may perform worse due to the large number of unnecessary constraints included. Hence, having strong valid inequalities is highly desirable. We now formulate a set of valid inequalities based on practical assumptions of the problem, in hopes of tightening the master problem formulation.

The valid inequalities are based on the assumption that at any time, *exactly* one bank is offline in every subsystem. Consider the following formulation for the service due time for a single bank. Let intervals $i, j\in I$ be such that $i<j$. Then the operational time for bank $b\in B$ between intervals i and j is given as

$$\sum_{\substack{k=i\\ x_{k,b}+y_{k,b}+z_{k,b}=0}}^{j-1} \left(t_{k+1} - t_k\right). \qquad (6.55)$$

In other words, it is the sum of the length of intervals between i and j where the bank is operational. Then, to ensure a bank is serviced on time, we must have

$$\sum_{\substack{k=i \\ x_{k,b}+y_{k,b}+z_{k,b}=0}}^{j-1} (t_{k+1} - t_k) \leq \Phi \left(1 + \sum_{k=i}^{j-1} y_{k,b} \right). \tag{6.56}$$

While this formulation ensures banks are serviced on time, the conditional sum makes it inappropriate for use in a mixed-integer linear model, and hence we proposed the formulation shown in Section 6.2.3. However, recall that a practical assumption of the model is that at any time, exactly one bank is offline in each subsystem. Therefore, if we sum (6.55) over all $b \in B_s$ where $s \in S$, we get

$$\sum_{b \in B_s} \sum_{\substack{k=i \\ x_{k,b}+y_{k,b}+z_{k,b}=0}}^{j-1} (t_{k+1} - t_k) = (|B_s| - 1) \left(t_j - t_i \right),$$

where $|\cdot|$ denotes the cardinality. Applying this to (6.56), we get the valid inequality

$$(|B_s| - 1) \left(t_j - t_i \right) \leq \Phi \left(|B_s| + \sum_{b \in B_s} \sum_{k=i}^{j-1} y_{k,b} \right), \quad \forall s \in S, i, j \in I \, : \, i < j, \tag{6.57}$$

and similarly, for cleanings we get

$$(|B_s| - 1) \left(t_j - t_i \right) \leq \Theta \left(|B_s| + \sum_{b \in B_s} \sum_{k=i}^{j-1} x_{k,b} \right), \quad \forall s \in S, i, j \in I \, : \, i < j. \tag{6.58}$$

These inequalities can then be added to the master problem to tighten its formulation.

6.4 Numerical Results and Discussion

We now explore the effectiveness of the proposed solution method using two case studies and several test instances. The case studies aim to provide the reader with a realistic problem setting by examining the scheduling requirements of two Bayer refineries based in Western Australia. The schedules generated in these case studies outline some common characteristics found in practical digester maintenance schedules. Using several test instances, we then assess the sensitivity of the model and solution method to various problem components, with the aim of identifying the factors that contribute to challenging instances. Specifically, we investigate how desired time horizon, service due time, and operational setup impact the model's performance. Furthermore, we analyse the contribution that Benders decomposition, valid inequalities, and lazy constraints make to overall algorithmic performance.

The scheduling model and solution algorithm was implemented in Gurobi version 10.0.1, using the *lazy constraint* callback feature. This feature allows the lazy constraints and Benders feasibility cuts to be integrated into the branch and cut framework. The program was run on a machine with a 2.3GHz AMD EPYC processor with 32 GB of RAM, using a single thread.

6.4.1 Case Study

Alcoa of Australia operates two bauxite mines and three alumina refineries within Western Australia, producing a total of 9 million tonnes of alumina annually, making up approximately 7% of total production worldwide (Alcoa of Australia Limited, 2019). In this case study, we apply the digester scheduling model to the digestion setups in the Wagerup and Pinjarra refineries. The case study aims to provide readers with a realistic parameter selection and demonstrate some of the characteristics of a practical schedule.

In both case studies, the maintenance-related parameters are chosen as follows;

$$\begin{aligned} \alpha &= 35 \text{ days}, & \beta &= 80 \text{ days}, \\ \Theta &= 220 \text{ days}, & \Phi &= 680 \text{ days}, \\ A &= \$10, & B &= \$40. \end{aligned}$$

The cleaning overlap cost L was set at 10% the cost per day of a cleaning activity, and the service overlap cost M was set at 25% the cost per day of a service activity. Each bank's starting operational time since its last cleaning, $\tilde{\theta}_b$ is chosen randomly such that $\tilde{\theta}_b \in [0, \Theta]$. Similarly, each bank's starting operational time since its last service, $\tilde{\phi}_b$ is chosen randomly such that $\tilde{\phi}_b \in [0, \Phi]$. A practical schedule should last for approximately three years. During this time, there are expected to be two valve changes for each production unit.

6.4.1.1 Alcoa Wagerup

Alcoa Wagerup uses three digester banks set up in a single subsystem to complete the digestion phase of the Bayer process. Figure 6.3 outlines an example of this setup. For this setup $S = \{1\}$ and $B = \{1, 2, 3\}$. A double bank switch occurs when bank 1 is switched with bank 3, and vice versa, hence $D_1 = \{(1, 3), (3, 1)\}$.

Table 6.2 summarises the performance of the original model and the decomposition approach proposed in Section 6.3 on the Wagerup case study. The original model was solved in 14 seconds using Gurobi. The use of Benders decomposition and valid inequalities resulted in a nearly 50% reduction in solve time, indicating a substantial improvement due to the application of Benders decomposition.

Figure 6.4 displays the optimal three-year maintenance schedule for the Wagerup case study, which includes fifteen cleanings and three services while meeting all four planned

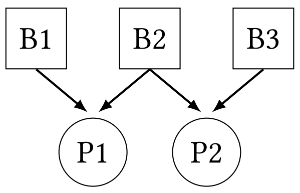

Figure 6.3: Wagerup digestion system with three banks feeding two production units.

	Original			Decomposed		
Case Study	Time (sec)	Gap (%)	Objective Value	Time (sec)	Gap (%)	Objective Value
Wagerup	14.00	0.00	240.0	7.46	0.00	240.0
Pinjarra	7200.02	32.30	1011.0	7200.02	7.52	1010.0

Table 6.2: Performance of the original model and Benders decomposition solution algorithm on the Wagerup and Pinjarra case studies.

valve change days. Notably, to prevent double bank changes, the schedule frequently cleans bank 2. Consequently, the average duration of an operational period for banks 1 and 3 is significantly longer than for bank 2. This is a common characteristic of digester maintenance schedules.

6.4.1.2 Alcoa Pinjarra

The Alcoa Pinjarra refinery contains eight digester banks split into three subsystems. Figure 6.1 outlines an example of this setup. For this setup, $S = \{1, 2, 3\}$ with $B_1 = \{1, 2, 3\}$, $B_2 = \{4, 5, 6\}$ and $B_3 = \{7, 8\}$. Double bank switching occurs when bank 1 is switched with bank 3 or bank 4 is switched with bank 6 and vice versa, hence $D_1 = \{(1, 3), (3, 1)\}$, $D_2 = \{(4, 6), (6, 4)\}$ and $D_3 = \emptyset$.

Table 6.2 presents the performance of the Benders decomposition approach and the original model for the Pinjarra case study model. Apart from Benders decomposition and valid inequalities, a lazy constraint formulation of constraint (6.15) can be utilised, given the presence of multiple subsystems in the Pinjarra setup. Despite two hours of solve time, neither the original model nor Benders decomposition could attain the optimal solution, indicating that the larger and more complex operational setup poses a significantly more challenging problem. Nevertheless, Benders decomposition yields an improved objective value and considerably reduces the optimality gap.

Figure 6.5 displays the best-known three-year maintenance schedule for the Pinjarra setup, which includes a total of 39 cleanings, 7 services and meets all ten planned valve changes. To avoid double bank changes, the schedule frequently plans cleanings for

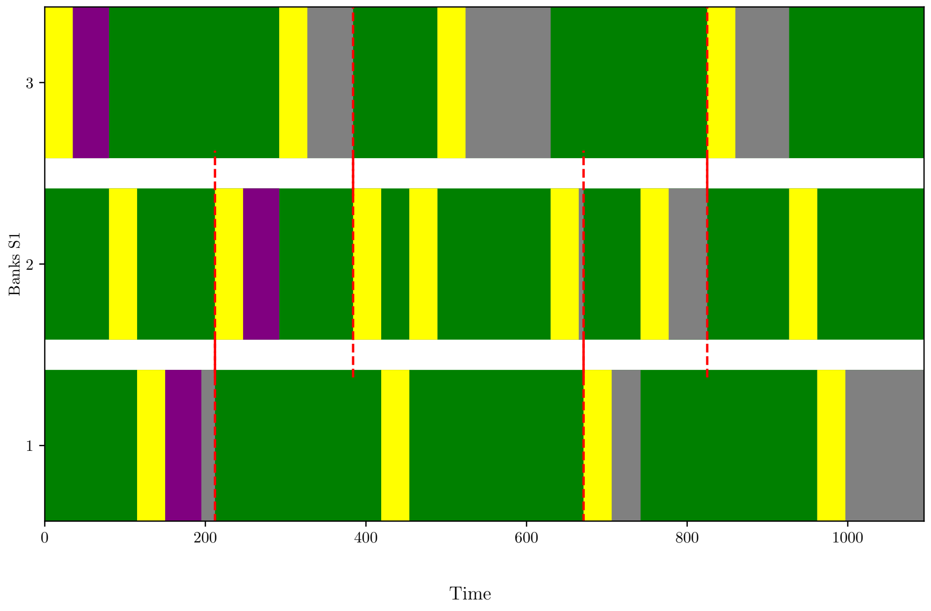

Figure 6.4: Optimal three-year maintenance schedule for Wagerup case study. Green represents operational periods, yellow represents cleaning activities, purple represents service activities and grey represents standby time. Valve changes are shown as red dashed lines that cross over their associated banks. For instance, the valve change on day 212 occurs on production unit 1, as it can connect to banks 1 and 2.

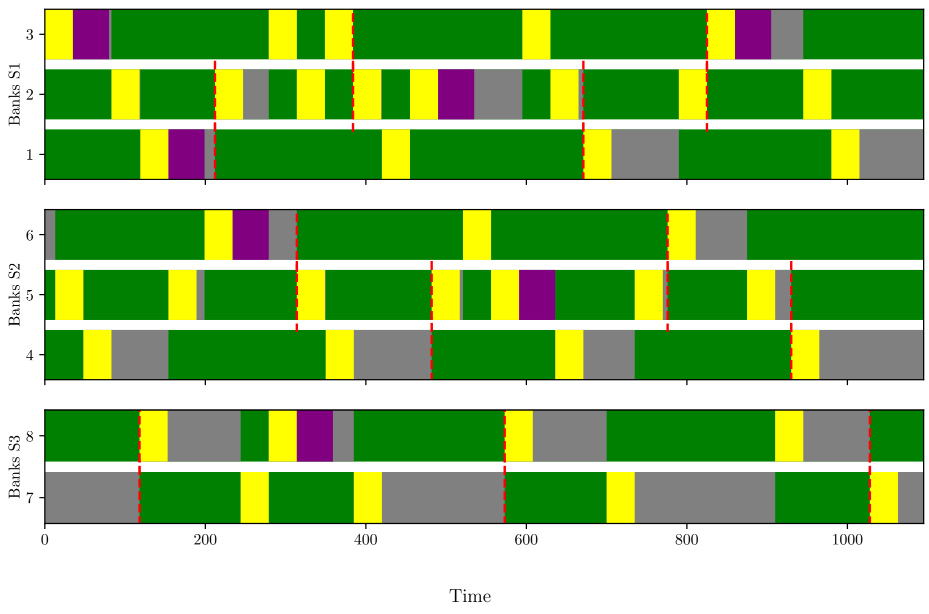

Figure 6.5: Best known three-year maintenance schedule for the Pinjarra case study.

banks 2 and 5. Moreover, we observe that no more than two cleanings or services occur concurrently, and efforts are made to minimise any potential clashes.

6.4.2 Test Instances

The analysis in the previous case study was limited to only two problem settings and only compared the proposed solution algorithm from Section 6.3 with the original model. To provide a more comprehensive understanding of the model's sensitivities and complexities, we introduce several test instances. These instances aim to explore how time horizon, service due time and operational setups impact model performance. Additionally, we investigate the contribution that Benders decomposition, valid inequalities, and lazy constraints make to overall algorithmic performance and attempt to determine which approach better tackles the model complexities.

6.4.2.1 Time Horizon

To explore the effect time horizon has on model performance, we solve the Wagerup and Pinjarra case study models with varying time horizons. Furthermore, we use different combinations of Benders decomposition, valid inequalities, and lazy constraints to gain insights into the most effective technique for handling increasing time horizons.

Table 6.3 outlines the performance of the different solution strategies for the Wagerup scheduling model with time horizons ranging from two to five years. The results show

τ	Benders Decomposition	Valid Inequalities	Time (sec)	Gap (%)	Objective Value	Best Bound	Cleaning Cuts	Service Cuts
730	✓	✓	0.93	0.00	170.0	170.0	61	217
730	✓		0.89	0.00	170.0	170.0	76	212
730		✓	0.61	0.00	170.0	170.0		
730			0.50	0.00	170.0	170.0		
1095	✓	✓	7.46	0.00	240.0	240.0	92	570
1095	✓		19.25	0.00	240.0	240.0	93	1068
1095		✓	21.58	0.00	240.0	240.0		
1095			14.00	0.00	240.0	240.0		
1460	✓	✓	324.80	0.00	340.0	340.0	151	6122
1460	✓		325.66	0.00	340.0	340.0	167	8312
1460		✓	392.17	0.00	340.0	340.0		
1460			276.95	0.00	340.0	340.0		
1825	✓	✓	7200.04	6.98	430.0	400.0	159	34362
1825	✓		7200.11	13.95	430.0	370.0	240	56812
1825		✓	7200.01	9.30	430.0	390.0		
1825			7200.01	11.63	430.0	380.0		

Table 6.3: Performance of various solution approaches for the Wagerup case study for varying time horizons (two to five years) with a two-hour time limit. We report the solve time in seconds, optimality gap, objective value and best bound for each approach. For Benders decomposition, we also report the number of cleaning and service cuts added.

τ	Benders Decomposition	Valid Inequalities	Lazy Constraints	Time (sec)	Gap (%)	Objective Value	Best Bound	Cleaning Cuts	Service Cuts	Lazy Added
730	✓	✓	✓	142.73	0.00	690.0	690.0	560	2524	375
730	✓	✓		209.58	0.00	690.0	690.0	598	2515	
730	✓		✓	105.90	0.00	690.0	690.0	772	2543	651
730	✓			190.70	0.00	690.0	690.0	604	2738	
730		✓		850.65	0.00	690.0	690.0			
730			✓	128.84	0.00	690.0	690.0			90
730				310.39	0.00	690.0	690.0			
1095	✓	✓	✓	7200.02	7.52	1010.0	934.0	979	16965	47
1095	✓	✓		7200.06	10.81	1010.0	901.0	1299	17800	
1095	✓		✓	7200.01	20.42	1087.0	865.0	1321	32947	63
1095	✓			7200.03	19.33	1080.0	871.0	1311	27691	
1095		✓		7200.05	34.99	1011.0	657.0			
1095			✓	7200.00	24.58	1011.0	763.0			290
1095				7200.02	32.30	1011.0	684.0			
1460	✓	✓	✓	7200.01	-	-	1088.0	952	29158	85
1460	✓	✓		7200.05	-	-	1057.0	953	25492	
1460	✓		✓	7200.05	-	-	775.0	1616	79313	64
1460	✓			7200.13	-	-	834.0	1707	69396	
1460		✓		7200.06	39.73	1407.0	848.0			
1460			✓	7200.01	43.10	1385.0	788.0			16
1460				7200.06	60.88	1568.0	613.0			

Table 6.4: Performance of various solution approaches for the Pinjarra case study for varying time horizons (two to four years) with a two-hour time limit. We report the solve time in seconds, optimality gap, objective value, best bound and number of each type of cut added.

that all solution approaches are highly sensitive to increasing time horizons. While all approaches solved the two-year schedule in under a second, the solve time increased exponentially with an increase in time horizon. No solver was able to prove optimality within two hours of solve time for a five-year schedule. For the $\tau = 730$ and $\tau = 1460$, the original model solved the fastest. However, for $\tau = 1095$, Benders decomposition with lazy constraints solved in half the time of the original model. Interestingly, for $\tau = 1095$ and $\tau = 1460$, the introduction of valid inequalities appeared to slow the original model, indicating they may have been ineffective and weighed down the solver. In contrast, valid inequalities improved the performance of Benders decomposition substantially. For instance, for the case where $\tau = 1095$, valid inequalities reduced the number of service cuts by 50%, leading to a far better runtime.

Table 6.4 outlines the performance of the different solution strategies for the Pinjarra scheduling model with time horizons ranging from two to four years. As this problem is more difficult than the Wagerup problem, it is more sensitive to increasing time horizons, with no approach able to solve the three-year schedule within a two-hour time limit.

For $\tau = 730$, the best performance was achieved by Benders decomposition with lazy constraints, solving in 106 seconds. Additionally, the use of lazy constraints appears very effective in this case. For the four-year schedule, Benders decomposition was not able to find an integer feasible solution after two hours. Remarkably, Benders decomposition with valid inequalities achieved a far tighter best bound than any other approach, despite not finding an integer feasible solution. Finally, valid inequalities were highly effective for the larger time horizon, significantly reducing the number of added service and cleaning cuts and tightening the best bound.

Overall, we can conclude that the scheduling model is highly sensitive to the time horizon, with all solution approaches struggling to solve longer schedules within a reasonable time limit. However, the use of valid inequalities and Benders decomposition leads to far tighter best bounds at large time horizons, although these approaches can struggle to find good quality feasible solutions. Furthermore, the use of lazy constraints appears very effective when combined with Benders decomposition.

6.4.2.2 Service Due Time

The case study highlighted the challenges associated with planning services, as operational periods can vary significantly in length, making it difficult to determine when to service a bank. To better understand the effect of this complexity, we explore how different service due times impact model performance. Using the Wagerup operational setup over a three-year time horizon, we set $\Phi = p\Theta$ where $p = 0, 2, 3, 4$. For each value of p, we then examine the performance of the various solution approaches. Note that when $p = 0$, we assume no servicing is required.

Table 6.5 summarises the performance of various solution approaches for each value of p. The findings indicate that the performance of the different solution strategies varies significantly depending on the service due time. For example, when no servicing is required ($p = 0$), the model is easily solved in under 10 seconds, with Benders decomposition proving particularly effective, solving the model to optimality in less than a second. However, when $\Phi = 2\Theta$, Benders decomposition becomes vastly ineffective, achieving only a 9.76% gap after two hours of solve time, whereas the original model solved to optimality in under 400 seconds. On the other hand, when $\Phi = 3\Theta$, the proposed method of Benders decomposition and valid inequalities was the best performer, improving on the original model by 400 seconds. The use of valid inequalities appears very effective in this case, reducing the number of service cuts by almost half. However, when increasing to $\Phi = 4\Theta$, Benders decomposition once again performs worse, and when used without valid inequalities, it is not able to prove optimality in two hours. Finally, when $\Phi = 5\Theta$, the optimal solution contained no services across the three-year time horizon. Nevertheless, Benders decomposition was able to prove optimality far quicker, requiring only 22 service lifetime cuts when used with valid inequalities.

p	Benders Decomposition	Valid Inequalities	Time (sec)	Gap (%)	Objective Value	Best Bound	Cleaning Cuts	Service Cuts
0	✓	✓	1.90	0.00	190.0	190.0	136	
0	✓		0.60	0.00	190.0	190.0	153	
0		✓	9.76	0.00	190.0	190.0		
0			5.66	0.00	190.0	190.0		
2	✓	✓	7200.01	9.76	410.0	370.0	121	22691
2	✓		7200.01	24.39	410.0	310.0	176	32540
2		✓	1100.55	0.00	410.0	410.0		
2			387.05	0.00	410.0	410.0		
3	✓	✓	315.90	0.00	280.0	280.0	157	7082
3	✓		1964.08	0.00	280.0	280.0	172	13007
3		✓	388.61	0.00	280.0	280.0		
3			716.07	0.00	280.0	280.0		
4	✓	✓	2399.56	0.00	260.0	260.0	151	10049
4	✓		7200.01	3.85	260.0	250.0	174	23275
4		✓	1802.64	0.00	260.0	260.0		
4			1159.70	0.00	260.0	260.0		
5	✓	✓	1.89	0.00	190.0	190.0	120	22
5	✓		0.66	0.00	190.0	190.0	156	83
5		✓	36.60	0.00	190.0	190.0		
5			16.66	0.00	190.0	190.0		

Table 6.5: Performance of various solution approaches for varying service due times where $\Phi = p\Theta$. For each Φ and solution approach, we report the solve time in seconds, optimality gap, objective value, best bound and number of each type of cut added.

Service due time represents a complicated and sensitive part of the model. The performances of all solution algorithms vary dramatically when service due time changes with respect to the cleaning due time. Therefore, the decision of which solution algorithm to use should depend on the ratio. Further analysis and research is necessary to better understand this trend and find ways of overcoming this sensitivity.

6.4.2.3 Operational Setup

The previous experiments were limited to the operational setups introduced in the case study. However, upon comparing the results presented in Tables 6.3 and 6.4, it is apparent that there is a significant difference in problem complexity between the Wagerup and Pinjarra setups. To further understand the impact of varying the size and layout of the operational setup, we introduce several new test instances. Specifically, for each instance, we assume there are m subsystems, and within each subsystem, there are n banks, resulting in a total of mn banks. We set up the banks such that there are no double bank changes and assume no planned valve change days. All other maintenance-related parameters are identical to those in the Pinjarra case study. The problem is then solved over a two-year time horizon.

The performance of the various solution approaches on different operational setups are presented in Table 6.6. Interestingly, in all examples where there were two banks per subsystem, the decomposition approach yielded significantly better results. For instance, when two banks were spread across four subsystems, Benders decomposition with valid inequalities achieved optimality in 465 seconds, while the original model only achieved a 75% gap after two hours of solve time. Moreover, including more banks within the same subsystem creates a much more difficult problem. For example, the problem of two banks per subsystem spread across four subsystems (totalling eight banks) was easily solved in under 500 seconds. However, if the eight banks were split across only two subsystems, the performance was far worse, with the best-performing model achieving only a 36% gap after two hours of solve time. Furthermore, for these large and very challenging models, the decomposition approach with valid inequalities consistently outperformed in terms of both objective value and best bound.

6.5 Conclusion

In this chapter, we formulated a maintenance scheduling model for digester banks, a critical asset used in the Bayer process. Our research was motivated by the importance digestion plays in the Bayer process and the difficulty of determining cost-efficient maintenance schedules for fleets of digester banks. Due to the network nature of digestion systems and complex maintenance requirements, scheduling bank maintenance manually can be challenging. Therefore, we propose a scheduling model that can find the cost-

n	m	Benders Decomposition	Valid Inequalities	Lazy Constraints	Time (sec)	Gap (%)	Objective Value	Best Bound	Cleaning Cuts	Service Cuts	Lazy Added
2	2	✓	✓	✓	9.67	0.00	60.0	60.0	562	2	0
2	2	✓	✓		15.04	0.00	60.0	60.0	583		
2	2	✓			47.32	0.00	60.0	60.0	2371	22	
2	2		✓		5690.25	0.01	60.0	60.0			
2	2			✓	2111.90	0.00	60.0	60.0			0
2	2				2896.40	0.01	60.0	60.0			
2	3	✓	✓	✓	76.14	0.00	90.0	90.0	1000	1	0
2	3	✓	✓		97.12	0.00	90.0	90.0	754		
2	3	✓			161.85	0.00	90.0	90.0	3653	39	
2	3		✓		7200.01	35.66	90.0	58.0			
2	3			✓	7200.00	52.52	90.0	43.0			0
2	3				7200.01	57.64	90.0	38.0			
2	4	✓	✓	✓	678.22	0.00	120.0	120.0	1277	1	0
2	4	✓	✓		465.63	0.00	120.0	120.0	1112	2	
2	4	✓			1931.50	0.01	120.0	120.0	4759	48	
2	4		✓		7200.01	52.40	120.0	57.0			
2	4			✓	7200.00	69.18	120.0	37.0			0
2	4				7200.01	73.18	120.0	32.0			
3	2	✓	✓	✓	7200.01	25.30	120.0	90.0	1701	10	0
3	2	✓	✓		7200.01	21.96	120.0	94.0	1486	10	
3	2	✓			7200.01	24.21	120.0	91.0	4006	40	
3	2		✓		7200.01	39.39	120.0	73.0			
3	2			✓	7200.00	65.92	120.0	41.0			0
3	2				7200.01	71.33	120.0	34.0			
3	3	✓	✓	✓	7200.01	41.46	205.0	120.0	2351	9	0
3	3	✓	✓		7200.02	36.62	190.0	120.0	1923	9	
3	3	✓			7200.01	44.43	205.0	114.0	6012	60	
3	3		✓		7200.01	46.78	205.0	109.0			
3	3			✓	7200.00	85.92	235.0	33.0			0
3	3				7200.01	86.26	205.0	28.0			
4	2	✓	✓	✓	7200.01	36.84	190.0	120.0	2574	20	0
4	2	✓	✓		7200.03	41.46	205.0	120.0	2954	20	
4	2	✓			7200.02	43.99	205.0	115.0	5589	56	
4	2		✓		7200.02	41.91	205.0	119.0			
4	2			✓	7200.00	82.05	205.0	37.0			0
4	2				7200.01	84.17	205.0	32.0			
4	3	✓	✓	✓	7200.04	68.14	565.0	180.0	4390	29	0
4	3	✓	✓		7200.02	69.49	590.0	180.0	4462	28	
4	3	✓			7200.01	74.86	590.0	148.0	8369	82	
4	3		✓		7200.02	69.72	590.0	179.0			
4	3			✓	7200.00	94.21	580.0	34.0			5
4	3				7200.01	95.31	600.0	28.0			

Table 6.6: Performance of various solution approaches for varying operational setup. Each setup consists of n banks per subsystem, with m subsystems. For each setup and solution approach, we report the solve time in seconds, optimality gap, objective value, best bound and number of each type of cut added.

optimised maintenance schedule that satisfies all required constraints. While this research focuses on Bayer digestion, many maintenance scheduling problems in refinery settings exhibit similar challenges.

Several strategies were introduced to assist in solving the problem at larger dimensions. Benders decomposition was used to handle the complicated operational lifetime requirement of the banks. We showed how the Benders subproblems could be solved easily using a specialist algorithm. Additionally, valid inequalities based on practical assumptions were introduced to further tighten the master problem. Finally, lazy constraints were used to handle service clash constraints, which make up a substantial proportion of the constraint set, yet only a small proportion are ever active. These strategies were then evaluated on two case studies involving real world digester setups. Several test instances were also generated to further explore the effectiveness of each strategy as well as better understand the key complexities of the problem.

Extensive numerical experiments highlight the key model sensitives and complexities. Parameters such as time horizon, service due time and operational setup each have significant impacts on the performance of the suggested solution strategies. Crucially, the results show that no single strategy is better in all cases. The model's sensitivities mean that particular strategies perform better in particular settings.

7 Conclusion and Future Work

In this thesis, we introduced several innovative approaches to extend cutting plane techniques - traditionally reserved for concave maximisation problems - to nonconcave scenarios. We explored the complex questions of *how* and *when* such an extension can be effectively applied, and demonstrated the improved performance that can arise from its application. Our findings have helped to enhance our understanding of nonconcave quadratic functions and the underlying mechanisms of successful cutting plane algorithms. Consequently, we have been able to develop several state-of-the-art algorithms for some nonconcave quadratic maximisation problems.

The key analysis began in Chapter 2, where we showed how restricting the search domain to regions of concavity allows us to easily apply cutting plane methodologies. In particular, we showed that the Euclidean distance matrix has the interesting property of being conditionally negative definite. This means that the quadratic objective can be effectively treated as concave when we restrict our search to points with the same cardinality. This discovery enables the direct application of cutting plane methods to the Euclidean max-sum diversity problem, even though the problem is inherently nonconcave.

Our resultant algorithm proved to be very efficient, making it the current state-of-the-art for this particular class of problem. It is able to solve large, two-coordinate diversity problem instances of more than 80000 variables. This impressive performance is in part due to the resultant cutting plane model (Θ_A) being fairly easy to solve, as binary variables are subject to a single cardinality constraint. Given (Θ_A) has a tight linear programming relaxation, mixed-integer solvers can solve this subproblem at large scales with relative ease. This represents a great example of one of the benefits of using cutting plane methodologies over general nonlinear programming. As the resultant cutting plane model is easy to handle and tangent planes provide tight approximations of the objective function, we can easily implement a cutting plane algorithm within an integer programming solver and are likely to see impressive overall performance.

Moreover, this chapter introduced the novel approach of analysing the concavity of a quadratic function within the problem's *feasible* domain, rather than the entire *functional* domain. Conventionally, a problem's nonconcavity is assessed based on the objective or constraint functions alone. However, from an optimisation standpoint, our primary interest lies in feasible solutions of the problem, or more specifically, feasible and optimal solutions. Thus, it is more useful to evaluate the properties of the quadratic objective on only feasible solutions. This innovative approach could potentially lead to substantial improvements in solving other quadratic programming problems. Historically, some of the most significant improvements to solution methods for quadratic programming problems have come from the research of classical problems. Future work should look to re-evaluate some of these classical problems with the new perspective of assessing the concavity of the objective within the feasible domain.

One of the downfalls of cutting plane methods is their reliance on tangent planes to provide good approximations of the objective function. If tangents are weak, especially at an optimal solution, then the approach can perform poorly. This was frequently observed on high-coordinate instances of the Euclidean max-sum diversity problem, as seen in Chapter 2.4. In Chapter 3, we conducted an in-depth instance analysis to better understand why the number of coordinates of original locations has a noticeable effect on cut tightness. By examining the effectiveness of a tangent's ability to remove nonoptimal solutions, we showed how these planes are unable to effectively and accurately approximate the objective contribution of locations towards the exterior of the cluster. As such, the cuts become weaker when the interior density decreases. This observation led to the introduction of a new class of challenging instances, where locations are spread around the circumference of a ball, and thus have no interior points.

Instance analysis not only highlights the key complexities of the problem, but can also identify the failings of an algorithm and suggest potential remedies. As a result of our analysis, we developed an improved cutting plane algorithm that uses coordinate partitioning to improve the linear approximation provided by tangent planes. By reconstructing a set of points whose squared Euclidean distances equal that of the original, the objective function becomes separable with respect to each new coordinate space. This allows us to form functional components whose location clusters have high interior densities, and therefore expected to have tighter cuts. Furthermore, we introduced several partitioning strategies. The resultant partitioned Euclidean diversity cut algorithm proved to be very efficient, improving on the original algorithm from Chapter 2.

The application of functional decomposition, such as with coordinate partitioning, is not a new practice, however it has been previously undervalued in the existing literature. One possible reason for this is that developing effective decomposition strategies is, in general, challenging and problem specific. The coordinate partitioning of Chapter 3 is specific to Euclidean distance matrices and hence is not applicable to general problems. A promising avenue for future work is in identifying and developing functional decomposition methods for other nonlinear programming problems. Moreover, quadratic functions can always be decomposed into their eigenvalues and eigenvectors. For instance, let $Q \in \mathbb{R}^{n \times n}$ be a square symmetric matrix and let $\lambda_1, \dots, \lambda_n \in \mathbb{R}$ and $v_1, \dots, v_n \in \mathbb{R}^n$ be its eigenvalues and orthonormal eigenvalues, respectively. Using eigenvalue decomposition we can write

$$Q = \sum_{i=1}^{n} \lambda_i v_i v_i^T,$$

which is effectively the sum of n rank one matrices. Developing an effective functional decomposition method using this basic result could therefore be applied to any quadratic programming problem. If achieved, this could lead to substantial algorithmic improvements across almost all types of quadratic programming problem.

The Euclidean diversity problem highlighted the fact that we can restrict our feasible space to achieve concavity in the objective function. In Chapter 4 we extended this idea by showing how we can augment our search strategy to always stay on concave directions. This is achieved through the development of the novel concept of *directional concavity*, which asserts whether the tangent plane of y provides a valid upper approximation at a point x. Looking specifically at the Euclidean max-sum problem, we developed several sufficient conditions to assert whether directional concavity holds by exploiting various properties of Euclidean distance matrices, such as one positive eigenvalue or being conditionally negative definite. These conditions proved very useful, and allowed us to develop an exact algorithm that searches the entire feasible domain while always staying on concave directions. As such, we can once again extend cutting plane methods to this class of nonconcave maximisation problem.

The algorithms introduced in this chapter could be improved in future research. In particular, Algorithm 8 may benefit for a more sophisticated binary search tree structure. This should allow for more efficient evaluation of the problem at different cardinality levels, as opposed to only searching in a decreasing direction. Furthermore, we note a significant gap in the literature on the application of the (EMSP) to real-world problems. The tests used in this chapter provide interesting conceptual frameworks and future work should expand on these results by applying the methods to real-world datasets and problems. In addition to identifying practical (EMSP) models, we should also attempt to identify difficult instances of these problems. In Chapter 3 we showed how the diversity problem becomes more challenging with a larger number of coordinates. That difficulty was not observed for the CDP or GDP problems, and hence more work is required to identify other difficult instances of the (EMSP). These problems can also help to understand and decide on which algorithm to use in which scenario.

In Chapter 5, we then relaxed the requirement of the objective being a Euclidean distance matrix and extended directional concavity to a general matrix. This involved extending the functional decomposition methodology from Chapter 3 and relaxing some of the original conditions for directional concavity from Chapter 4. By combining directionally concave tangent planes with convex over-envelopes, we formulated an upper approximation of the quadratic objective function everywhere. However, this upper plane is piecewise linear. Therefore, to use it in a linear cutting plane model, the conditions were linearised and a branch and cut algorithm was proposed that allowed us to search the entire space. Extensive numerical experiments proved the validity of this idea and presented it as a promising avenue for future research.

We suggest two key avenues of future improvements for our directionally concave branch and cut algorithm. The first is to improve the decomposition, branching and node selection rules used. Our numerical results showed varied performance based on which strategy was chosen. In particular, poor branching and node selection rules meant the algorithm often wasted time exploring unhelpful subtrees. However, the algorithm achieved very promising best-case results, and hence more work is required to improve the formulation of the branch and cut tree, thereby improving overall performance and robustness.

Furthermore, although the number of positive eigenvalues had a great effect on runtime, the average angle after functional decomposition seemed to have very little effect on overall performance. In fact, there were many examples of large achieved angles with poor run times. Moreover, in Figure 5.3 there is an obvious pattern between decomposition strategy and achieved angle, yet the runtime these strategies achieved showed little pattern. This suggests that there may be some other phenomena effecting the performance of this approach, for which we currently have no explanation. Additionally, future work should look to extend the results of our directionally concave tangent planes to matrices with several positive eigenvalues. Although such a result may mean less space is approximated by tangents, it should reduce the number of potential branching directions, possibly leading to run time improvements.

Finally, in Chapter 6 we highlight the importance of pragmatic optimisation solutions by developing a maintenance scheduling model for digester banks, a critical path asset used in the Bayer process. Motivated by the complexity of creating cost-efficient schedules for these assets, our model aims to optimise maintenance costs while adhering to all constraints. We employ several strategies to tackle large-scale problems, including Benders decomposition for managing operational lifetimes, a specialised algorithm for Benders subproblems, and valid inequalities to strengthen the master problem. Additionally, lazy constraints address service clash constraints, prevalent yet minimally active, enhancing our model's efficiency. Our evaluation includes extensive numerical experiments, including a real-world case study, highlighting the model's sensitivities and complexities. Variables like the time horizon, service due times, and operational setups significantly affect performance. This research not only advances maintenance scheduling in the Bayer process but also sheds light on similar challenges across refinery operations.

Our results demonstrate that no single strategy universally excels, but specific strategies may perform better under certain conditions, underscoring the need to tailor approaches to individual scenarios. As such, future work should look to refine some of the decision making regarding which solution strategy to use for a given problem. Furthermore, it may be the case that there is significant symmetry present in the model, whereby the cleaning and service schedule of a bank subsystem could be swapped without impacting the feasibility or objective value. Finally, a discrete time model could be explored, which may provide a tighter formulation for certain digester bank setups.

In summary, this thesis has successfully demonstrated the application of cutting plane methodologies to quadratic programming problems, notably through the integration of novel concepts like directional concavity and functional decomposition. Our research has opened new pathways for enhancing the efficiency and applicability of cutting plane techniques, particularly in the realm of quadratic programming. As we look forward, there is significant potential to expand upon these foundations. Investigating the application of our approaches to a broader range of quadratic programming challenges, optimising decomposition strategies, and refining our algorithms to harness the full potential of directional concavity will be crucial. These endeavours not only promise to advance the theoretical framework of cutting plane methods but also improve practical outcomes in solving complex real-world optimisation problems, such as in Chapter 6. By continuously pushing the boundaries of what is feasible with cutting plane techniques, we can contribute to the development of more robust, scalable, and efficient optimisation tools that meet the growing demands of its diverse application areas.

Bibliography

Aboudolas, K., Papageorgiou, M., Kouvelas, A., & Kosmatopoulos, E. (2010). A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, *18*(5), 680–694. https://doi.org/10.1016/j.trc.2009.06.003

Abramowitz, M., & Stegun, I. A. (1968). *Handbook of mathematical functions with formulas, graphs, and mathematical tables* (Vol. 55). US Government printing office.

Alcoa of Australia Limited. (2019). 2019 tax transparency report.

Bapat, R. B., & Raghavan, T. E. S. (1997). *Nonnegative matrices and applications*. Cambridge University Press. https://doi.org/10.1017/cbo9780511529979

Bazargan, M. (2016, March). *Airline operations and scheduling* (2nd ed.). Routledge. https://doi.org/10.4324/9781315566474

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*(1), 238–252. https://doi.org/10.1007/bf01386316

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. *Proceedings of the twenty-sixth RAMP symposium*, 16–17.

Bomze, I. M. (2002). Branch-and-bound approaches to standard quadratic optimization problems. *Journal of Global Optimization*, *22*, 17–37.

Bonami, P., Lee, J., Leyffer, S., & Wächter, A. (2013). On branching rules for convex mixed-integer nonlinear optimization. *Journal of Experimental Algorithmics (JEA)*, *18*, 2–1.

Bonami, P., Lodi, A., & Zarpellon, G. (2022). A classifier to decide on the linearization of mixed-integer quadratic problems in CPLEX. *Operations Research*, *70*(6), 3303–3320. https://doi.org/10.1287/opre.2022.2267

Bracewell, R., & Kahn, P. B. (1966). The Fourier transform and its applications. *American Journal of Physics*, *34*(8), 98–101.

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, *99*, 300–313. https://doi.org/10.1016/j.cie.2015.12.007

Bui, H. T., Spiers, S., & Loxton, R. (2024). Solving Euclidean Max-Sum problems exactly with cutting planes. *Computers & Operations Research*, *168*, 106682. https://doi.org/10.1016/j.cor.2024.106682

Burer, S., & Vandenbussche, D. (2009). Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. *Computational Optimization and Applications*, *43*, 181–195. https://doi.org/10.1007/s10589-007-9137-6

Charikar, M., Chatziafratis, V., Niazadeh, R., & Yaroslavtsev, G. (2019). Hierarchical Clustering for Euclidean Data. *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, 2721–2730.

Chen, J., & Burer, S. (2012). Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, *4*, 33–52. https://doi.org/10.1007/s12532-011-0033-9

Cheng, L.-w., Wang, Y.-l., Zhou, Q.-s., Qi, T.-g., Liu, G.-h., Peng, Z.-h., & Li, X.-b. (2021). Scale formation during the bayer process and a potential prevention strategy. *Journal of Sustainable Metallurgy 2021 7:3*, *7*(3), 1293–1303. https://doi.org/10.1007/s40831-021-00417-4

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

Contreras, I., Cordeau, J. F., & Laporte, G. (2011). Benders decomposition for large-scale uncapacitated hub location. *Operations Research*, *59*(6), 1477–1490. https://doi.org/10.1287/opre.1110.0965

Davis, P. J. (1959). Leonhard Euler's Integral: A Historical Profile of the Gamma Function. *The American Mathematical Monthly*. https://doi.org/10.2307/2309786

de Jonge, B., & Scarf, P. A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, *285*(3), 805–824. https://doi.org/10.1016/j.ejor.2019.09.047

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., Kelḿanov, 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* (pp. 541–551). 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

Erkut, E., & Neuman, S. (1989). Analytical models for locating undesirable facilities. *European Journal of Operational Research*, *40*(3), 275–291. https://doi.org/10.1016/0377-2217(89)90420-7

Feo, T., Goldschmidt, O., & Khellaf, M. (1992). One-Half Approximation Algorithms for the k-Partition Problem. *Operations Research*, *40*, S170–s173. https://doi.org/10.1287/opre.40.1.S170

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* (pp. 417–428). Springer International Publishing. https://doi.org/10.1007/978-3-031-07704-3_34

Fischetti, M., Ljubic, I., & Sinnl, M. (2016). Redesigning Benders decomposition for large-scale facility location. *Management Science*, *63*(7), 2146–2162. https://doi.org/10.1287/mnsc.2016.2461

Floudas, C. A., & Lin, X. (2004). Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. *Computers & Chemical Engineering*, *28*(11), 2109–2129. https://doi.org/10.1016/j.compchemeng.2004.05.002

Gallo, G., Hammer, P. L., & Simeone, B. (1980). Quadratic knapsack problems. In M. W. Padberg (Ed.), *Combinatorial Optimization* (pp. 132–149). Springer. https://doi.org/10.1007/BFb0120892

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., Ching-Chung, K., & Dhir, K. S. (1995). A discrete optimization model for preserving biological diversity. *Applied Mathematical Modelling*, *19*, 696–701. https://doi.org/10.1016/0307-904x(95)00083-v

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. *Math. Sci*, *7*(1), 1–14.

Gower, J. C. (1985). Properties of Euclidean and non-Euclidean distance matrices. *Linear algebra and its applications*, *67*, 81–97.

Gupta, O. K., & Ravindran, A. (1983). Nonlinear integer programming and discrete optimization. *Transactions of the ASME, Journal of Mechanisms, Transmissions and Automation in Design*, *105*, 160–164.

Gupta, O. K., & Ravindran, A. (1985). Branch and Bound Experiments in Convex Nonlinear Integer Programming. *Management Science*, *31*(12), 1533–1546. https://doi.org/10.1287/mnsc.31.12.1533

Hammer, P. L., & Rubin, A. A. (1970). Some remarks on quadratic programming with 0-1 variables. *Revue française d'informatique et de recherche opé rationnelle. 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.

Heydar, M., Mardaneh, E., & Loxton, R. (2021). Approximate dynamic programming for an energy-efficient parallel machine scheduling problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2021.12.041

Hochbaum, D. S., Liu, Z., & Goldschmidt, O. (2023, January). A Breakpoints Based Method for the Maximum Diversity and Dispersion Problems. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA23)* (pp. 189–200). Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611977714.17

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations Research*, *55*(3), 588–602. https://doi.org/10.1287/opre.1060.0371

Julianto, D. D., Purwani, A., & Bouguern, S. (2023). Analysis of location determination for temporary waste collection points using p-dispersion method: An application to yogyakarta city, indonesia. *Journal of Novel Engineering Science and Technology*, *2*(02), 59–67.

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, *38*, 3–13. https://doi.org/10.1016/j.cor.2009.12.011

Kopanos, G. M., Kyriakidis, T. S., & Georgiadis, M. C. (2014). New continuous-time and discrete-time mathematical formulation for resource-constrained project scheduling problems. *Computers and Chemical Engineering*, *68*, 96–106. https://doi.org/10.1016/j.compchemeng.2014.05.009

Kronqvist, J., Bernal, D. E., & Grossmann, I. E. (2020). Using regularization and second order information in outer approximation for convex MINLP. *Mathematical Programming*, *180*(1-2), 285–310.

Kronqvist, J., Bernal, D. E., Lundell, A., & Grossmann, I. E. (2019). A review and comparison of solvers for convex MINLP. *Optimization and Engineering*, *20*, 397–455. https://doi.org/10.1007/s11081-018-9411-8

Kronqvist, J., Lundell, A., & Westerlund, T. (2016). The extended supporting hyperplane algorithm for convex Mixed-I nteger Nonlinear Programming. *Journal of Global Optimization*, *64*, 249–272.

Kronqvist, J., Lundell, A., & Westerlund, T. (2018). Reformulations for utilizing separability when solving convex MINLP problems. *Journal of Global Optimization*, *71*, 571–592. https://doi.org/10.1007/s10898-018-0616-3

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

Kuhn, H. W., & Tucker, A. W. (1951, January). Nonlinear Programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* (pp. 481–493, Vol. 2). University of California Press.

Kuo, 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

Lai, X., Yue, D., Hao, J.-K., & Glover, F. (2018). Solution-based tabu search for the maximum min-sum dispersion problem. *Information Sciences*, *441*, 79–94.

Land, A. H., & Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, *28*(3), 497–520. https://doi.org/10.2307/1910129

Lerch, D., & Trautmann, N. (2019). A lazy-constraints approach to resource-constrained project scheduling. *IEEE International Conference on Industrial Engineering and Engineering Management*, 144–148. https://doi.org/10.1109/ieem44572.2019.8978524

Leyffer, S. (1993). *Deterministic methods for mixed integer nonlinear programming* [Doctoral dissertation, University of Dundee].

Leyffer, S. (2001). Integrating SQP and Branch-and-Bound for Mixed Integer Nonlinear Programming. *Computational Optimization and Applications*, *18*(3), 295–309. https://doi.org/10.1023/a:1011241421041

Li, X., Yu, S., Dong, W., Chen, Y., Zhou, Q., Qi, T., Liu, G., Peng, Z., & Jiang, Y. (2015). Investigating the effect of ferrous ion on the digestion of diasporic bauxite in the Bayer process. *Hydrometallurgy*, *152*, 183–189. https://doi.org/10.1016/j.hydromet.2015.01.001

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

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, *28*(2), 129–137.

Lu, C., Deng, Z., & Jin, Q. (2017). An eigenvalue decomposition based branch-and-bound algorithm for nonconvex quadratic programming problems with convex quadratic constraints. *Journal of Global Optimization*, *67*, 475–493.

Lu, Z., Martínez-Gavara, A., Hao, J.-K., & Lai, X. (2023). Solution-based tabu search for the capacitated dispersion problem. *Expert Systems with Applications*, *223*, 119856. https://doi.org/10.1016/j.eswa.2023.119856

Lubin, M., Dowson, O., Dias Garcia, J., Huchette, J., Legat, B., & Vielma, J. P. (2023). JuMP 1.0: Recent improvements to a modeling language for mathematical optimization.

Mathematical Programming Computation, *15*, 581–589. https://doi.org/10.1007/s12532-023-00239-3

Lubin, M., Yamangil, E., Bent, R., & Vielma, J. P. (2018). Polyhedral approximation in Mixed-Integer Convex optimization. *Mathematical Programming*, *172*(1), 139–168.

Lundell, A., Kronqvist, J., & Westerlund, T. (2022). The supporting hyperplane optimization toolkit for convex MINLP. *Journal of Global Optimization*, *84*(1), 1–41.

Madhulatha, T. S. (2012). An overview on clustering methods.

Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research*, *8*(2), 219–223. https://doi.org/10.1287/opre.8.2.219

Maravelias, C. T., & Grossmann, I. E. (2003). New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Industrial and Engineering Chemistry Research*, *42*(13), 3056–3074. https://doi.org/10.1021/ie020923y

Martí, R., Duarte, A., Martínez-Gavara, A., & Sánchez-Oro, J. (2021). The MDPLIB 2.0 library of benchmark instances for diversity problems.

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

Martinez-Gavara, A., Corberan, T., & Marti, R. (2021). GRASP and tabu search for the generalized dispersion problem. *Expert Systems with Applications*, *173*, 114703.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I – Convex underestimating problems. *Mathematical Programming*, *10*, 147–175. https://doi.org/10.1007/bf01580665

Mencarelli, L., & D'Ambrosio, C. (2019). Complex portfolio selection via convex mixed-integer quadratic programming: A survey. *International Transactions in Operational Research*, *26*(2), 389–414. https://doi.org/10.1111/itor.12541

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, *7*, 326–329. https://doi.org/10.1145/321043.321046

Naoum-Sawaya, J., & Buchheim, C. (2016). Robust critical node selection by Benders decomposition. *INFORMS Journal on Computing*, *28*(1), 162–174. https://doi.org/10.1287/ijoc.2015.0671

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*(4), 308–313. https://doi.org/10.1093/comjnl/7.4.308

O'Brien, F. A., & Mingers, J. (1997). A heuristic algorithm for the equitable partitioning problem. *Omega*, *25*, 215–223. https://doi.org/10.1016/s0305-0483(96)00046-1

Olde Keizer, M. C., Teunter, R. H., & Veldman, J. (2016). Clustering condition-based maintenance for systems with redundancy and economic dependencies. *European Journal of Operational Research*, *251*(2), 531–540. https://doi.org/10.1016/j.ejor.2015.11.008

Pardalos, P. M., & Rodgers, G. P. (1990). Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, *45*, 131–144. https://doi.org/10.1007/bf02247879

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

Pearce, R. H., & Forbes, M. (2018). Disaggregated Benders decomposition for solving a network maintenance scheduling problem. *Journal of the Operational Research Society*, *70*(6), 941–953. https://doi.org/10.1080/01605682.2018.1471374

Pearce, R. H. (2019). *Towards a general formulation of lazy constraints* [PhD Thesis]. School of Mathematics and Physics, The University of Queensland.

Peiró, J., Jiménez, I., Laguardia, J., & Martí, R. (2021). Heuristics for the capacitated dispersion problem. *International transactions in operational research*, *28*(1), 119–141.

Pia, A. D., Dey, S. S., & Molinaro, M. (2017). Mixed-integer quadratic programming is in NP. *Mathematical Programming*, *162*, 225–240. https://doi.org/10.1007/s10107-016-1036-0

Pisinger, D. (2006). Upper bounds and exact algorithms for p-dispersion problems. *Computers & 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. *International institute of tropical agriculture, ibadan, Nigeria.*

Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, *259*(3), 801–817. https://doi.org/10.1016/j.ejor.2016.12.005

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

Rodriguez-Lujan, I., Huerta, R., Elkan, C., & Cruz, C. S. (2010). Quadratic Programming Feature Selection. *The Journal of Machine Learning Research*, *11*, 1491–1516.

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 Frechet's Article "Sur La Definition Axiomatique D'Une Classe D'Espace Distances Vectoriellement Applicable Sur L'Espace De Hilbert. *Annals of Mathematics*, *36*(3), 724–732. 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*, *38*, 787–793. https://doi.org/10.2307/1968835

Schoenberg, I. J. (1938a). Metric Spaces and Completely Monotone Functions. *Annals of Mathematics*, *39*(4), 811–841. https://doi.org/10.2307/1968466

Schoenberg, I. J. (1938b). Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, *44*(3), 522–536.

Seif, Z., Mardaneh, E., Loxton, R., & Lockwood, A. (2020). Minimizing equipment shutdowns in oil and gas campaign maintenance. *Journal of the Operations Research Society*. https://doi.org/10.1080/01605682.2020.1745699

Shirkhorshidi, A. S., Aghabozorgi, S., & Wah, T. Y. (2015). A comparison study on similarity and dissimilarity measures in clustering continuous data. *PloS one*, *10*(12), e0144059.

Siopa, J. M. P., Garção, J. E. é. S., & E Silva, J. M. (2015). Component redundancy allocation in optimal cost preventive maintenance scheduling. *Journal of the Operational Research Society*, *66*(6), 925–935. https://doi.org/10.1057/jors.2014.56

Slater, M. (1950). Lagrange multipliers revisited. In *Traces and emergence of nonlinear programming* (pp. 293–306). Springer.

Smith-Miles, K., & Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, *39*, 875–889. https://doi.org/10.1016/j.cor.2011.07.006

Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, *41*, 6:1–6:25. https://doi.org/10.1145/1456650.1456656

Spiers, S., Bui, H. T., & Loxton, R. (2023a). Coordinate partitioning for difficult Euclidean max-sum diversity problems. *Under review*.

Spiers, S., Bui, H. T., & Loxton, R. (2023b). An exact cutting plane method for the Euclidean max-sum diversity problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2023.05.014

Spiers, S., Bui, H. T., Loxton, R., Mansour, M. R., Hollins, K., Francis, R., Martindale, C., & Pimpale, Y. (2023). Bayer digestion maintenance optimisation with lazy constraints and Benders decomposition. *Annals of Operations Research*. https://doi.org/10.1007/s10479-023-05561-6

Tawarmalani, M., & Sahinidis, N. V. (2005). A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, *103*, 225–249. https://doi.org/10.1007/s10107-005-0581-8

Taylor, A. E. (1952). L'Hospital's Rule. *The American Mathematical Monthly*, *59*, 20–24. https://doi.org/10.1080/00029890.1952.11988058

Vandenberghe, L., & Boyd, S. (1996). Semidefinite Programming. *SIAM Review*, *38*, 49–95. https://doi.org/10.1137/1038003

Vielma, J. P., Ahmed, S., & Nemhauser, G. L. (2008). A lifted linear programming branch-and-bound algorithm for Mixed- Integer Conic Quadratic Programs. *INFORMS Journal on Computing*, *20*(3), 438–450.

Wachter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57. https://doi.org/10.1007/s10107-004-0559-y

Westerlund, T., & Pettersson, F. (1995). An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, *19*(Suppl. 1), 131–136. https://doi.org/10.1016/0098-1354(95)87027-x

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82. https://doi.org/10.1109/4235.585893

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

A Appendix

A.1 Proof of Proposition 16

Proof. Firstly, observe that $H(j_1) \subset \cdots \subset H(j_p)$ and hence Proposition 15 is equivalent to

$$|I(x) \cap H(j_k)| \geq k, \text{ for all } k = 1, \dots, p.$$

For $k \geq 2$, this can be rewritten as

$$|I(x) \cap (H(j_k) \setminus H(j_{k-1}))| + |I(x) \cap H(j_{k-1})| \geq k. \tag{A.1}$$

Hence, let

$$h(k) = \begin{cases} H(j_1), & \text{if } k = 1, \\ H(j_k) \setminus H(j_{k-1}), & \text{otherwise} . \end{cases}$$

Then for all k_1, k_2 such that $1 < k_1 < k_2$, we have that

$$\begin{aligned} h(k_1) \cap h(k_2) &= \left(H(j_{k_1}) \setminus H(j_{k_1-1})\right) \cap \left(H(j_{k_2}) \setminus H(j_{k_2-1})\right) \\ &\subset H(j_{k_1}) \cap \left(H(j_{k_2}) \setminus H(j_{k_2-1})\right) \\ &= \left(H(j_{k_1}) \cap H(j_{k_2})\right) \setminus H(j_{k_2-1}) \qquad (A.2) \\ &= H(j_{k_1}) \setminus H(j_{k_2-1}) = \emptyset \qquad (A.3) \end{aligned}$$

where (A.2) comes from the set identity $A \cap (B \setminus C) = (A \cap B) \setminus C$ and (A.3) comes from $H(j_{k_1}) \subseteq H(j_{k_2-1}) \subset H(j_{k_2})$. Note this proof holds analogously for the case where $k_1 = 1$. Therefore all $h(k)$ are independent from one another. We can use the independent sets $h(1), \dots, h(p)$ to rewrite (A.1) as

$$|I(x) \cap h(1)| \geq 1, \tag{A.4}$$

$$|I(x) \cap h(k)| \geq k - \sum_{l=1}^{k-1} |I(x) \cap h(l)|, \quad k = 2, \dots, p, \tag{A.5}$$

where (A.5) comes from the fact

$$|I(x) \cap H(j_{k-1})| = \left| I(x) \cap \left(\bigcup_{l=1}^{k-1} h(l) \right) \right| = \sum_{l=1}^{k-1} |I(x) \cap h(l)| .$$

We can therefore determine the number of solutions that satisfy Proposition 15 by counting the number of possible combinations of selecting elements from the independent sets $h(1), \dots, h(p)$ such that (A.4) and (A.5) hold. This can be achieved with the following recursive sum,

$$R(y, LB) \geq \sum_{w_1=1}^{\min\{|h(1)|,p\}} \sum_{w_2=\max\{2-w_1,0\}}^{\min\{|h(2)|,p-w_1\}} \cdots \sum_{w_p=\max\{p-\sum_{l=1}^{p-1} w_l,0\}}^{\min\{|h(p)|,p-\sum_{k=1}^{p-1} w_l\}} \prod_{k=1}^{p} \binom{|h(k)|}{w_k},$$

which provides a valid lower bound for the number of solutions removed by the tangent plane of y. □

A.2 Proofs of Validity of Decomposition Strategies

The following results prove $f(x) = g(x) + \sum_{i=1}^{m} f_i(x)$ for each decomposition strategy listed in Section 5.2.8.

1. *Basic:* For the basic decomposition, we get that

$$\begin{aligned} g(x) + \sum_{i=1}^{m} f_i(x) &= 0 + \sum_{i=1}^{m} \lambda_i \langle v_i, x\rangle^2 + \sum_{i=1}^{m} \frac{1}{m} \langle Q^- x, x\rangle, \\ &= 0 + \left\langle Q^+ x, x\right\rangle + \langle Q^- x, x\rangle = f(x), \end{aligned}$$

as required.

2. *Greedy:* If $m < l$ we have that

$$\begin{aligned} g(x) + \sum_{i=1}^{m} f_i(x) &= \sum_{i=m+1}^{l} \lambda_i \langle v_i, x\rangle^2 + \sum_{i=1}^{m} \left(\lambda_i \langle v_i, x\rangle^2 + \lambda_{m+l+1-i} \langle v_{m+l+1-i}, x\rangle^2\right) \\ &= \sum_{i=m+1}^{l} \lambda_i \langle v_i, x\rangle^2 + \left\langle Q^+ x, x\right\rangle + \sum_{i=l+1}^{m+l} \lambda_i \langle v_i, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \sum_{i=m+1}^{m+l} \lambda_i \langle v_i, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \langle Q^- x, x\rangle = f(x). \end{aligned}$$

If $m = l$ then

$$\begin{aligned} g(x) + \sum_{i=1}^{m} f_i(x) &= 0 + \sum_{i=1}^{m} \left(\lambda_i \langle v_i, x\rangle^2 + \lambda_{m+l+1-i} \langle v_{m+l+1-i}, x\rangle^2\right) \\ &= \sum_{i=1}^{m} \lambda_i \langle v_i, x\rangle^2 + \sum_{i=1}^{l} \lambda_{m+l+1-i} \langle v_{m+l+1-i}, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \sum_{i=m+1}^{m+l} \lambda_i \langle v_i, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \langle Q^- x, x\rangle = f(x). \end{aligned}$$

Finally, if $m > l$, then

$$\begin{aligned} g(x) + \sum_{i=1}^{m} f_i(x) &= 0 + \sum_{i=1}^{l} \left(\lambda_i \langle v_i, x\rangle^2 + \lambda_{m+l+1-i} \langle v_{m+l+1-i}, x\rangle^2\right) + \sum_{i=l+1}^{m} \lambda_i \langle v_i, x\rangle^2 \\ &= \sum_{i=1}^{m} \lambda_i \langle v_i, x\rangle^2 + \sum_{i=1}^{l} \lambda_{m+l+1-i} \langle v_{m+l+1-i}, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \sum_{i=m+1}^{m+l} \lambda_i \langle v_i, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \langle Q^- x, x\rangle = f(x), \end{aligned}$$

as required.

3. *Stratified:* If $m < l$, then

$$\begin{aligned} g(x) + \sum_{i=1}^{m} f_i(x) &= \sum_{i=m+1}^{l} \lambda_i \langle v_i, x\rangle^2 + \sum_{i=1}^{m} \left(\lambda_i \langle v_i, x\rangle^2 + \lambda_{\max\{m,l\}+i} \left\langle v_{\max\{m,l\}+i}, x\right\rangle^2\right) \\ &= \sum_{i=m+1}^{l} \lambda_i \langle v_i, x\rangle^2 + \left\langle Q^+ x, x\right\rangle + \sum_{i=1}^{m} \lambda_{l+i} \langle v_{l+i}, x\rangle^2 \\ &= \sum_{i=m+1}^{l} \lambda_i \langle v_i, x\rangle^2 + \left\langle Q^+ x, x\right\rangle + \sum_{i=l+1}^{m+l} \lambda_l \langle v_l, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \langle Q^- x, x\rangle = f(x). \end{aligned}$$

If $m = l$ then

$$\begin{aligned} g(x) + \sum_{i=1}^{m} f_i(x) &= 0 + \sum_{i=1}^{m} \left(\lambda_i \langle v_i, x\rangle^2 + \lambda_{\max\{m,l\}+i} \left\langle v_{\max\{m,l\}+i}, x\right\rangle^2\right) \\ &= \left\langle Q^+ x, x\right\rangle + \sum_{i=1}^{l} \lambda_{m+i} \langle v_{m+i}, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \sum_{i=m+1}^{m+l} \lambda_i \langle v_i, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \langle Q^- x, x\rangle = f(x). \end{aligned}$$

Finally, if $m > l$ then

$$\begin{aligned} g(x) + \sum_{i=1}^{m} f_i(x) &= 0 + \sum_{i=1}^{l} \left(\lambda_i \langle v_i, x\rangle^2 + \lambda_{\max\{m,l\}+i} \left\langle v_{\max\{m,l\}+i}, x\right\rangle^2\right) + \sum_{i=l+1}^{m} \lambda_i \langle v_i, x\rangle^2 \\ &= \sum_{i=1}^{m} \lambda_i \langle v_i, x\rangle^2 + \sum_{i=1}^{l} \lambda_{m+i} \langle v_{m+i}, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \sum_{i=m+1}^{m+l} \lambda_i \langle v_i, x\rangle^2 \\ &= \left\langle Q^+ x, x\right\rangle + \langle Q^- x, x\rangle = f(x). \end{aligned}$$

A.3 Attribution

Attribution for Chapter 2, which is based off Spiers, Bui, and Loxton (2023b).

	Conception and Design	Mathematical Development	Results, Data Analysis and Implementation	Write-up	Editing and Reviewing
Sandy Spiers	✓	✓	✓	✓	✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					
Hoa Bui	✓	✓	✓	✓	✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					
Ryan Loxton		✓			✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					

Attribution for Chapter 3, which is based off Spiers, Bui, and Loxton (2023a).

	Conception and Design	Mathematical Development	Results, Data Analysis and Implementation	Write-up	Editing and Reviewing
Sandy Spiers	✓	✓	✓	✓	✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					
Hoa Bui		✓			✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					
Ryan Loxton					✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					

Attribution for Chapter 4, which is based off Bui et al. (2024)

	Conception and Design	Mathematical Development	Results, Data Analysis and Implementation	Write-up	Editing and Reviewing
Sandy Spiers	✓	✓	✓	✓	✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					
Hoa Bui	✓	✓	✓	✓	✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					
Ryan Loxton		✓			✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					

Attribution for Chapter 6, which is based off Spiers, Bui, Loxton, et al. (2023).

	Conception and Design	Mathematical Development	Results, Data Analysis and Implementation	Write-up	Editing and Reviewing
Sandy Spiers	✓	✓	✓	✓	✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					
Hoa Bui	✓	✓			✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					
Ryan Loxton	✓	✓			✓
I acknowledge that these represent my contribution to the above research output and I have approved the final version.					
Signed:					

Attribution for Chapter 6 continued.

	Conception and Design	Mathematical Development	Results, Data Analysis and Implementation	Write-up	Editing and Reviewing
Moussa Mansour	✓		✓		✓

I acknowledge that these represent my contribution to the above research output and I have approved the final version.

Signed:

Attribution for Chapter 6 continued.

	Conception and Design	Mathematical Development	Results, Data Analysis and Implementation	Write-up	Editing and Reviewing
Kylie Hollins	✓				✓

I acknowledge that these represent my contribution to the above research output and I have approved the final version.

Signed:

Attribution for Chapter 6 continued.

	Conception and Design	Mathematical Development	Results, Data Analysis and Implementation	Write-up	Editing and Reviewing
Richard Francis	✓				✓

I acknowledge that these represent my contribution to the above research output and I have approved the final version.

Signed:

Attribution for Chapter 6 continued.

	Conception and Design	Mathematical Development	Results, Data Analysis and Implementation	Write-up	Editing and Reviewing
Chistopher Martindale	✓				✓

I acknowledge that these represent my contribution to the above research output and I have approved the final version.

Signed:

Attribution for Chapter 6 continued.

	Conception and Design	Mathematical Development	Results, Data Analysis and Implementation	Write-up	Editing and Reviewing
Yogesh Pimpale	✓				✓

I acknowledge that these represent my contribution to the above research output and I have approved the final version.

Signed: