9-2023

Test Problem Generation and Metaheuristic Selection for the Multidemand Multidimensional Knapsack Problem

Matthew E. Scherer

Test Problem Generation and Metaheuristic Selection for the Multidemand Multidimensional Knapsack Problem

DISSERTATION

Matthew E. Scherer, Capt, USAF

AFIT-ENS-DS-23-S-020

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-DS-23-S-020

TEST PROBLEM GENERATION AND METAHEURISTIC SELECTION FOR THE MULTIDEMAND MULTIDIMENSIONAL KNAPSACK PROBLEM

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Matthew E. Scherer, BS , MS

Capt, USAF

September, 2023

AFIT-ENS-DS-23-S-020

TEST PROBLEM GENERATION AND METAHEURISTIC SELECTION FOR THE MULTIDEMAND MULTIDIMENSIONAL KNAPSACK PROBLEM

DISSERTATION

Matthew E. Scherer, BS , MS
Capt, USAF

Committee Membership:

Dr. Raymond R. Hill
Chair

Dr. Brian J. Lunday
Member

Dr. Bruce A. Cox
Member

Dr. Edward D. White
Member

Adedeji B. Badiru, PhD
Dean, Graduate School of Engineering and Management

AFIT-ENS-DS-21-S-XXX

Abstract

This work focuses on instance generation methods for the multi-demand multidimensional knapsack problem (MDMKP). Specifically, instance space analysis (ISA) is used to characterize the landscape of existing instances and validate the novelty of new instances generated with a novel problem generation method, the primal problem instance generator (PPIG). The instance generator is capable of producing feasible, diverse, and challenging instances by directly controlling the problem features. PPIG contributes to the previous collections of instances and is validated through instance space analysis. The research presents an in-depth empirical evaluation of existing solution procedures for the MDMKP. The portfolio of metaheuristics examined show promising performance on existing benchmark libraries but lack robustness when the test set of instances are extended using the PPIG method. A machine learning classifier is employed to provide an interpretable link between instance configuration and solution procedure performance. The final aspect of the research is an optimization framework used to provide problem generation parameters to the PPIG methodology to further cover the instance space for the full suite of MDMKP test problems.

To my wife Lauren and family.

Acknowledgements

Without the mentorship of Dr. Raymond Hill, this work would not have been possible. Thank you for guiding me when I needed to walk and letting me run when I had it in me. A dissertation is a massive undertaking and it would not have been possible without you. I am a better analyst, researcher, and person thanks to you.

I would also like to thank my committee members, Dr. Brian Lunday, Dr. Bruce Cox, and Dr. Tony White, for their support and feedback. Some of you instructed me on topics I am building upon in this work and could not have done without strong foundations. As educators, you are a shining example of an instructor who continuously strives to be better each year for the sake of their students. Having this experience as a student under you in a classroom but also as an advisor in the research process is something I will carry with me in my career.

Lastly, I would like to thank my friends gained from my PhD cohort. Graduate school would have been a lot more lonely without my peers suffering along the way with me. I would like to think we learned a lot from each other but I can say that I learned a lot about my future as an officer in the Air Force thanks to you.

Matthew E. Scherer

Table of Contents

Page

Abstract iv

Dedication v

Acknowledgements vi

List of Figures ix

List of Tables xii

I. Introduction 1

1.1 Motivation 3
1.1.1 Complexity 3
1.1.2 Instance Generation 7
1.2 Research Objectives and Scope 8
1.3 Organization of the Dissertation 9

II. Analyzing Problem Instances of the Multidemand Multidimensional Knapsack Problem with Instance Space Analysis 11

2.1 Introduction 11
2.2 Instance Space Analysis 17
2.2.1 Problem Space 18
2.2.2 Subspace of Instances 20
2.2.3 Algorithm Space 22
2.2.4 Performance Space 23
2.2.5 Feature Space 24
2.3 The Instance Space 28
2.3.1 Augmenting the Instance Space 32
2.4 Feasibility Considerations on Robust Problem Construction 37
2.4.1 Feasible Instance Generation 38
2.4.2 Analysis of Feasible Instances 41
2.5 Conclusion 45

III. Instance Based Configuration for Metaheuristic Selection 48

3.1 Introduction 48
3.2 Instance Space Analysis 52
3.2.1 Algorithm Selection Problem 52
3.2.2 Problem Space 53

Page

3.2.3 Subset of Instances 54
3.2.4 Algorithm Space 58
3.2.5 Performance Space 62
3.2.6 Feature Space 62
3.2.7 Instance Space 65
3.3 Metaheuristic Selection Problem 70
3.4 Results 75
3.5 Conclusion 79

IV. An Optimization Framework for Filling the Instance Space of Multidemand Multidimensional Knapsack Problems 81

4.1 Introduction 81
4.1.1 Literature Review 84
4.2 Problem Description 88
4.3 Methodology 94
4.4 Results 98
4.5 Discussion 101
4.5.1 Updating the Instance Space 103
4.6 Conclusion 108

V. Conclusion 110

5.1 Summary 111
5.2 Future Work 112

Appendix A. Distribution of Features Across Instance Spaces 114

Bibliography 119

List of Figures

Figure		Page
1	Summary of Algorithm Selection Problem (Rice, 1976) shown in dotted box, extended upon by Smith-Miles et al. (2014)	17
2	Instance space for the MDMKP based on the instances from Beasley (1990), with projection defined by (6)	31
3	Correlation between c_j and w_{ij} for $i = 1, \ldots, m$	33
4	Correlation between c_j and w_{ij} for $i = m+1, \ldots, m+q$	33
5	Instance space for the MDMKP based on the instances from Beasley (1990), augmented with instances generated from Cho (2005)	34
6	Instance space for the MDMKP based on Beasley and Cho instances	35
7	ISA of new instances on the Beasley and Cho instance space	42
8	ISA of Scherer, Beasley, and Cho instance space with theoretical feasible boundary	43
9	Summary of Algorithm Selection Problem (Rice, 1976) shown in dotted box, extended upon by Smith-Miles et al. (2014)	53
10	Distribution of correlation coefficients for knapsack constraints	64
11	Distribution of correlation coefficients for demand constraints	65
12	Instance space for the MDMKP with projection defined by (13)	68
13	Distribution of meta-features across instance space	69
14	SVM predicted best metaheuristic for the MDMKP	71
15	Local Search's footprints across the instance space	72

Figure Page

16 Adaptive Memory Search's footprints across the instance space . . . 72

17 Scatter Search's footprints across the instance space . . . 73

18 Two-Stage Tabu Search's footprints across the instance space . . . 73

19 Optimal sparse decision tree for MDMKP ASP . . . 76

20 GOSDT predicted best metaheuristic for the MDMKP . . . 77

21 Confusion matrix for GOSDT . . . 78

22 Confusion matrix for SVM . . . 78

23 Summary of ASP (Rice, 1976) shown in dotted box, extended upon by Smith-Miles et al. (2014) . . . 89

24 Instance space from Scherer et al. (2023*a*) . . . 93

25 Reduced instance space . . . 98

26 MDMKP instance realizations from PPIG in the reduced instance space . . . 100

27 Footprints of metaheuristics in the reduced instance space . . . 101

28 LS footprints across the instance space . . . 103

29 AMS footprints across the instance space . . . 104

30 SS footprints across the instance space . . . 104

31 TSTS footprints across the instance space . . . 105

32 Updated instance space . . . 106

33 Optimal sparse decision tree for MDMKP ASP . . . 108

34 Distribution of features across Beasley instance space . . . 115

35 Distribution of features across Cho and Beasley instance space . . . 116

Figure Page

36 Distribution of features across Scherer, Cho, and Beasley instance space 118

List of Tables

Table		Page
1	Meta-features for the MDMKP	27
2	Average runtimes (seconds) to identify a within 1% near-optimal solution by objective function type.	32
3	Average runtimes (seconds) to identify a within 1% near-optimal solution by instance source.	44
4	Meta-features for the MDMKP	66
5	Euclidean distance to target for realized instances	100
6	Number of instances with best found solution by metaheuristic	102

TEST PROBLEM GENERATION AND METAHEURISTIC SELECTION FOR THE MULTIDEMAND MULTIDIMENSIONAL KNAPSACK PROBLEM

I. Introduction

This dissertation investigates several aspects in the empirical testing of solution procedures in combinatorial optimization (CO). These investigates resulted in improved problem generation approaches, improved algorithm performance insights and improved visualization of the solution space for a particular type of CO. CO originates from linear programming (LP). LP is a mathematical means for translating a real-world decision making problem into a mathematical framework (a math program) with a desired linear objective function and set of linear constraints. The decisions are in the form of decision variables which align with the decision making construct used by the decision maker. Decision variables can be continuous and/or discrete depending on the scenario being modeled. The formulated model is solved to find an optimal solution. If a heuristic, typically used in settings where the problem is too computationally intensive to find an optimal solution, is implemented then the corresponding solution is not guaranteed to be optimal but a reasonably strong solution is found much quicker. Both solution methods are often evaluated by the time required to obtain their solution and the quality of the solution given.

Integer programming (IP) is based on LP but includes a restriction to a discrete domain in their decision variables. Mixed integer (linear) programming (MIP) includes a combination of continuous and discrete variable domains. Examples of LP formulations include network flow problems such as the maximum flow problem. The maximum flow problem consists of maximizing the flow over a defined graph with

nodes and edges between nodes subject to capacities constraints on associated edges. MIP formulations are seen in facility location problems with a binary fixed cost of building a facility and then an added variable cost for transporting resources from the facility to a customer.

CO refers to problems which contain an optimal solution as some subset of a finite set of feasible solutions to the mathematical program. Therefore, an optimal solution is always obtainable through complete enumeration of all possible solutions, but this is almost always impractical due to the combinatorial number of solutions to find and evaluate. Examples of CO problems include the knapsack problem (KP), which is concerned with filling a knapsack with a fixed capacity with a finite number of items to maximize profit, each item containing a profit coefficient and cost coefficient. A variant of the KP, and the problem of interest in this research, is the multidemand multidimensional knapsack problem (MDMKP). The MDMKP is concerned with filling a knapsack with items with an associated cost/profit such that demands are met across multiple dimensions without violating the supply capacities across multiple dimensions.

Research in CO is usually performed using either a top-down or bottom-up fashion. In the top-down fashion, research is focused on developing solution methods for the most difficult optimization problems such as the traveling salesman problem (TSP). The idea is to assume a solution method for the complex optimization problems will carry over to a larger variety of simpler problems. In contrast, the bottom-up approach focuses on developing methods for the simplest models such as the KP, hoping the techniques generalize to the more complex models. Research focused on the MDMKP strikes a middle-ground approach to these perspectives, serving the KP community through its simplicity but also containing enough complexity to pertain to more complex problem classes.

To test the effectiveness of a solution method for an optimization problem, a collection of test instances are used to evaluate solution procedure performance. The evaluated instances are usually obtained from a synthetic data generation method or existing test libraries. The collections are dependent on the problem of interest, since the set of parameters obtained are assigned to the relevant decision variables and constraints in a formulation. From these given methods, algorithm or heuristic performance and solution quality is measurable across a set of different structured problems.

1.1 Motivation

1.1.1 Complexity

Combinatorial optimization problems are difficult because, in principle, these problems can be solved by enumeration. However when considering the number of feasible solutions, the enumeration is impractical. For example, in the assignment problem there is an additional permutation of assignments for each node n added, therefore there are $n!$ feasible solutions. In set cover problems the number of subsets for consideration is 2^n. In the TSP, the salesperson initially has $n-1$ cities for consideration at city $n=1$, then $n-2$ cities afterwards, and the pattern continues resulting in $(n-1)!$ feasible tours. All of these examples indicate the difficulty in complete enumeration as a solution procedure.

The difficulty of a problem is determined by the performance of the best known algorithm. Performance is generally defined by the running time and the amount of computer memory in the context of an exact algorithm, although solution quality is also a common measure. A heuristic is measured by its solution quality relative to the computational resources required. To have some distinguishing measure of algorithm performance, there are several methods of investigation. The first is to examine a

standardized set of problem instances and measure the performance across different instances. Another approach is to examine the average running time of a solution procedure. In both circumstances a problem arises of confounding factors from the computer specifications or the solution procedure. The most common measure of algorithm performance is to examine the theoretical worst-case analysis of an algorithm. The analysis gives an upper bound on the number of arithmetic operations required to solve any instance of a given size.

The most efficient algorithms are bounded by a polynomial function of computing time and are called polynomial time algorithms. These algorithms are denoted through "big O" notation which denotes the worst-case complexity of an algorithm (Garey and Johnson, 1979). These algorithms scale according a polynomial of the size of the input n such as $O(n), O(n\log n), O(n^k)$ for some constant k. The least efficient algorithms are non-polynomial algorithms because they cannot be bounded by a polynomial function of computing time. Instead these algorithms are bound by an exponential function in n such as $O(2^n)$ or $O(3^n)$. Pseudopolynomial algorithms are bounded asymptotically by a polynomial both in n and at least one other input value. For example the capacity of the knapsack c impacts the algorithm performance of dynamic programming in solving the KP $O(nc)$ (Kellerer et al., 2004*b*). Pseudopolynomial algorithms, such as dynamic programming, are less "attractive" than polynomial time algorithms because of limited computer memory.

The distinction of problems based on their worst case algorithm performance defines the computational complexity of a problem. In computational complexity theory the seminal work of Karp (1972) defined 21 NP-complete problems. NP-complete problems share two characteristics: A solution to an NP-complete problem can be verified in polynomial time and if a polynomial time algorithm exists for any of the NP-complete problems then all NP-complete problems are solvable in polynomial

time. A number of CO problems fall into the class of NP-complete problems after being reduced to a decision problem. However, the implementation of a solution procedure and its runtime determines how "easy" a problem is compared to a difficult problem. A problem such as the KP is an NP-complete problem but is regarded as tractable while the TSP is a intractable problem (Garey and Johnson, 1978). The discussion tends towards the known algorithms to obtain an optimal solution and the efficiency of each algorithm.

The simplest case of CO problems are problems outside of NP-complete and are solvable in polynomial time. For example network flow problems, such as the minimum cost network flow (MCNF) problem without integral flow constraints, are solvable within polynomial time. If the MCNF problem requires integral flow, then the problem is NP-complete (Even et al., 1975). An advantage to solving the MCNF problem is its structure. If the MCNF problem contains integral data in its formulation such as the objective function coefficients, constraint coefficients, and the capacities, the total unimodularity property grants an integral basic feasible solution. In addition, the MCNF problem is solvable through the efficient polynomial time network simplex algorithm (Orlin, 1997). In this instance the structure of the given problem results in handling the complicated integrality portion of the formulation as typically seen in network flow problems.

The MCNF problem is a generalization of the assignment, transport, transshipment, maximal flow, and shortest path problems (Ahuja et al., 1988). Therefore all of these problems are "easy" to solve in practice because there is a polynomial time algorithm for these optimization problems. In some circumstances, the structure within the problem naturally produces integer solutions and also makes combinatorial optimization problems easy to solve. Other problems which are solvable through polynomial time include LP problems. Karmarkar (1984) is the first instance of a

polynomial time algorithm which is also efficient in practice. The addition of a integer constraint to an LP forms a much more difficult problem in an NP-complete problem.

Within NP-complete problems there is no discrimination of problem difficulty by the definition of NP-completeness. The second requirement of an NP-complete problem is if a polynomial time algorithm exists to solve one of the NP-complete problems then the algorithm exists for all NP-complete problems. However there are special cases of NP-completeness which do separate some problems such as the TSP and the 0-1 knapsack problem in the form of strong and weak NP-completeness. These are directly related to the relationship of algorithms which require non-deterministic polynomial time for an optimal solution compared to pseudopolynomial time algorithms.

Strong NP-completeness refers to the set of problems which remain NP-complete even when bounded by a polynomial in the length of the input parameters (Garey and Johnson, 1978). Weak NP-completeness is the complement which indicates the problem no longer remains NP-complete when bounded by a polynomial in the length of the input parameters. For example in network optimization the upper bounds of the arc/edge costs C and capacity U may be bounded by a polynomial (the similarity assumption), i.e., for some integer k, $C = O(n^k)$ and $U = O(n^k)$. Then the network is strong NP-complete if it remains NP-complete even when the similarity assumption is satisfied (Ahuja et al., 1988). Examples of strong NP-complete problems include the TSP and the integer flow MCNF problem. A weak NP-complete problem is the 0-1 KP with integer coefficients. The 0-1 KP is solvable in pseduopolynomial time through dynamic programming ($O(nc)$) but in practice the capacity constraint c is usually large and the problem remains difficult to solve.

1.1.2 Instance Generation

To construct a mathematical program, sets of parameters have to be defined for the associated coefficients for the decision variables and constraints. Different methods for defining parameters in a model include real-world data, benchmark libraries, and synthetic constructions. The complex relationships between parameters designates the objective performance of a solution procedure in empirical testing. Other parameters, such as the size of the problem instance, directly relate to the performance of a solution procedure.

Real-world data is usually unobtainable for most problem classes or lacks a defining number of observations to characterize all potential problem instances. Additionally, real-world data is a less obtainable comparison to subsequent work in a problem class. Benchmark libraries are an attempt to provide standardized results for solution procedure performance which are easily obtainable. However they may not cover the full range of potential problem instances observable in a given problem class, resulting in poor conclusions on a solution procedure performance. Instead the solution procedure is biased towards a benchmark set of data.

In synthetic problem generation, a reoccurring goal is to generate a typical class of problems because real world data is sparse or not diverse enough to test robustness of a solution procedure. The recommendations set forth by Hall and Posner (2010) propose a better approach by generating classes of problems with varying characteristics then focusing on the relationship of problem structure and solution procedure performance as advocated for in the work by Hooker (1995). Instance generation techniques exist for each specific problem class examined but rely on a standardized set of principles. The data generation principles from Hall and Posner (2010) include purpose, comparability, unbiasedness, and reproducibility in instance generation schemes.

1.2 Research Objectives and Scope

This dissertation enhances several aspects in CO by developing methods for controllable instance generation to assist in the empirical evaluation of solution procedures. This work provides a diverse, unbiased, and reproducible instance generation technique for creating synthetic data for the MDMKP. Additionally, this instance generation technique is demonstrated to configure test instances with controllable properties to achieve a diverse set of test instances.

This research contains two threads of research: the first thread examines instance generation techniques and the second thread pertains to the empirical evaluation of metaheuristics. Research in the first thread makes two contributions in Chapters II and IV. The first involves developing an instance generation technique for the MDMKP which is analyzed through the lens of instance space analysis (ISA). After examining the empirical evaluation of metaheuristics in the second contribution the third contribution revisits this instance generation technique to target specific configurations of MDMKP test instances.

The second thread of research pertains to the empirical evaluation of metaheuristics, discovering the relationship between solution procedure performance and metaheuristic performance, and the characterization of test instances. This thread contains a single contribution in Chapter III focused on enhancing the ISA methodology by addressing the algorithm selection problem (ASP) with an alternative machine learning model which provides an interpretable angle to predict the best metaheuristic of a given instance. This contribution is the second contribution of this research and the alternative model for the ASP is revisited in the third contribution of this work for the targeted configurations of MDMKP test instances.

Both of these threads of research are focused on improving existing test instances and conveying of results rather than developing new solution methodologies. Both

threads utilize ISA as the backbone to the methodology. This research shows multiple avenues for ISA to be used. In particular, to identify strengths and weaknesses in the test instances used in empirical testing and to indicate the effective and ineffective solution methodologies for those test instances.

1.3 Organization of the Dissertation

Chapter II examines how to develop problem instances for the MDMKP such that they characterize all possible problem structures to enhance the conclusions of an solution procedure performance as Hooker (1995) advocates. This characterization is motivated by the No Free Lunch (NFL) theorems of Wolpert and Macready (1997) and the inability of an optimization solution procedure to dominate the performance metrics of all possible problem instances.

Using the enhanced collection of test instances, Chapter III implements the existing metaheuristics from the literature and evaluates them through rigorous empirical testing to discover which solution methods perform well across different instance configurations. From these empirical tests, the relationship between the metaheuristic performance and the instance configuration is explored to discover theoretical insights as Hooker (1994) advocates.

Chapter IV explores the effectiveness of generating test instances with controllable features which are unique to the existing collection of test instances. The previously collected set of test instances provide background on the expected performance of the targeted instances, which are then evaluated and compared to their expected performance to characterize the effectiveness of the targeted generated scheme.

Chapter V provides concluding remarks and the appendix provides relevant supplementary material for the first contribution. In addition, Chapter V discusses areas

for future extensions of this work for instance generation techniques and the empirical evaluation of solution procedures.

II. Analyzing Problem Instances of the Multidemand Multidimensional Knapsack Problem with Instance Space Analysis

2.1 Introduction

Current research in optimization too often focuses on finding "novel" solution methods and claiming state-of-the-art performance with marginal improvements using sometimes sparse evidence. In optimization, the success criteria for judging a solution procedure is largely dependent on empirical assessments using a chosen set of test instances. Solution procedures include algorithms to identify a proven optimal solution, approximate algorithms to attain a feasible solution having a proven optimality gap, and heuristic methods whose aim is to obtain a high quality but not necessarily optimal solution. Although a theoretical focus is useful in the comparison of algorithms, e.g., via worst-case or average-case analysis of algorithms, Hooker (1994) prompted the optimization community to use empirical testing for the evaluation of algorithms. The empirical evaluation of algorithms should reveal relationships between problem instance structure and solution procedure performance. Additionally, the empirical study of solution procedures may also illuminate theoretical properties within different solution methods.

In the testing and evaluation of algorithms, researchers propose a solution method and evaluate its performance on instances with data sourced from real-world applications, existing benchmarks, or instances generated randomly through a set of probability distributions. Conclusions are based on comparisons that evaluate the average results of the proposed solution methods to other solution procedures in the literature. Hooker (1995) expressed concerns with this reporting structure because it does not express enough nuance of the efficacy of a proposed solution procedure and instead "resembles track meets more than scientific endeavors". Inferring that the conclusions

hold beyond the tested instances are concerning because the underlying test data are subject to structural issues such as a lack of diversity or bias. Additionally, some test instances are unrepresentative of real world applications or may be trivial to solve.

Wolpert and Macready (1997) proposed the No-Free-Lunch (NFL) Theorem, which states it is impossible for any one algorithm to outperform all other algorithms when averaged over all possible optimization problems. In this manuscript we investigate the NFL over instances of a single optimization problem. We search for the weaknesses and strengths of an algorithm across the variety of possible instance configurations. Current research relating to instance generation is one sided in that it only addresses identifying instances which leverage the strengths of a procedure while ignoring instances which may indicate procedural weaknesses. In coordination with the ideas expressed by Hooker (1995), NFL indicates the need to clarify strengths and weaknesses of a solution procedure. By utilizing both aspects of success and failure, the empirical evidence informs both the theoretical worst-case and best-case settings for a given problem.

Smith-Miles (2019) developed Instance Space Analysis (ISA) to better understand solution procedure performance and its relation to problem structure. ISA addresses the concerns of a fair evaluation of solution procedures in a standardized methodology across problem classes. ISA is presented through a series of publications beginning with Smith-Miles and Lopes (2012) in measuring instance difficulty, followed by Smith-Miles et al. (2014) in examining the algorithmic strengths and weaknesses across the instance space, and Smith-Miles and Bowly (2015) in generating new instances unique to the defined instance space. The visualization methodology was improved with a novel projection algorithm (Muñoz et al., 2018), and the final methodology supporting the online tool MATILDA (Smith-Miles, 2019) is summarized in a recent tutorial paper (Smith-Miles and Muñoz, 2021).

ISA provides a two-dimensional visualization of all test instances and condenses the information of all problem features into a convenient mapping of their structure. In this mapping, the similarities and differences of problem instances based on their structure is shown in a scatterplot defined by a "coordinate system that systematically locates the instances in the two-dimensional plane to facilitate visualization of trends in relationships across the instance space" (Smith-Miles and Muñoz, 2021). Theoretical boundaries of all possible instances are established by discovering the upper and lower bounds of all feature values and projecting them into the same instance space of the existing test instances. These boundary conditions indicate potential areas within the instance space where no instances currently exist, indicating the need for additional methods to generate instances in these underrepresented regions. If the performance of a suite of solution procedures is measured, the instance space enables researchers to discover regions presenting distinguishable performance. These regions, which visualize good performance for a particular solution method across the instance space, are termed *algorithm footprints* (Smith-Miles et al., 2014).

Numerous applications of ISA address optimization problem classes such as the graph coloring problem (Smith-Miles and Baatar, 2014), knapsack problem (Smith-Miles et al., 2021), and the max flow problem (Alipour et al., 2023). ISA is applicable to areas outside of optimization including classification (Muñoz et al., 2018), regression (Muñoz et al., 2021), and anomaly detection (Kandanaarachchi et al., 2020) in machine learning (ML) settings. ISA is applicable to multiple research areas with a combination of algorithmic solution methods and data to derive empirically based conclusions. ISA assists researchers in providing scrutiny to the reporting of results beyond aggregate statistics of performance measures.

Other work examining problem structure and mapping it towards solution procedure performance includes applications in knapsack problem (KP) variants. Hill and

Reilly (2000) examine the correlation structure between the knapsack constraints and the objective function coefficients of the two-dimensional knapsack problem (2KP). The authors examine both an exact approach (CPLEX) and a heuristic (Toyoda, 1975) to measure the impact of correlation structure on solution procedure performance. Results indicate a negative correlation structure between the two knapsack constraints severely degraded the heuristic's performance and CPLEX struggled in scenarios having large differences of correlations between objective function values and constraint coefficients.

Cho et al. (2008) expand upon these findings in the 2KP by showing the lack of diversity in the test problem generation schemes used for analysis of solution procedure performance. They utilize the instance generation technique from Hill and Reilly (2000) that covers the full range of correlation structures between the objective function values and constraints. Leveraging the new collection of test instances, the authors propose two heuristics for the 2KP which outperform existing heuristics on both the legacy test instances and newer, more diverse test instances. Similar findings on the implications of poorly formed test instances in KP variants are discussed by Hill et al. (2011).

Hall and Posner (2007) examine several problem characteristics which impact the performance of solution procedures for the 0-1 KP. The authors summarize the key features as problem size, characteristics of item values, characteristics of item sizes, relationships between item value and item size, knapsack capacity, and characteristics of the linear relaxation solution. These features are used in the feature space defined by Smith-Miles et al. (2021).

Smith-Miles and Lopes (2012) explore multiple types of combinatorial optimization problems and characteristics to distinguish between trivial and difficult problems. These characteristics define the feature space and include features which are either

problem independent or problem dependent. The authors discuss problem specific features for a variety of combinatorial optimization problems including the assignment problem, the traveling salesman problem (TSP), the KP variants, the bin-packing problem, and optimization problems on graphs. Certain aspects of graph problems provide measures stemming from spectral graph theory (Chung and Graham, 1997).

Hill and Reilly (2000) demonstrate instance generation using an explicit correlation induction (ECI) method. Their explicit method imposes a specified correlation structure across the population of instances in contrast to an implicit correlation induction (ICI) method which implies a correlation structure but does not guarantee it. Reilly (2009) discusses the shortcomings of implicit correlated induction generation methods for the 0-1 KP by revealing the lack of a full range of correlation coefficients between classes of generated test instances. The long accepted correlation classes of weak, almost strong, strong, correlated, and others all exist within the rather narrow range of 0.97 to 1. Similar findings from Pisinger (2005) motivated the creation of new correlation classes for the 0-1 KP.

Synthetic data generation requires a generation scheme adhering to principles of correctness, applicability, and reproducibility (Hall and Posner, 2010). A method for instance generation with ISA proposed by Smith-Miles and Bowly (2015) produces controllable instance characteristics by filling observed gaps in the instance space using a genetic algorithm (GA). The authors demonstrate their instance generation method in the context of the graph coloring problem and has been expanded to the 0-1 KP by Smith-Miles et al. (2021). Their methods include modifying the strategy for selection of target points in the instance space and using a customized fitness function for evaluation in the GA. The results indicate a significant portion of the instance space area filled by evolved points.

Bowly et al. (2020) propose a method for an instance generation scheme for linear

programs (LP) which is expandable to an integer-restricted case. Their construction method relies on an alternative encoded space to form an optimal solution and associated instance parameters, subsequently transforming the encoded instance into a feasible, bounded LP. The authors explore the trade-offs of different instance generation parameters and their effect on the distribution of meta-features for the class of LPs. In comparison, this research offers an alternative, sampling-based approach focused on discrete optimization problems.

This paper extends the use of ISA and provides a novel problem generation method based on sampling methods that identify difficult but feasible instances of the multidemand, multidimensional knapsack problem (MDMKP). We demonstrate ISA as a tool to observe where the current benchmark test instances lie within the instance space and to validate the novelty of generated test instances. We introduce two new sets of test instances for the MDMKP: one set of test instances focused on correlation structure and another set of test instances focused on feasibility. Our findings indicate the proposed generation method covers the instance space while maintaining feasibility for the generated MDMKP instances.

The remainder of this work is as follows. Section 2.2 decomposes the problem of interest into its components of ISA, after which Section 2.3 shows the current collection of instances lie within the instance space. Within Section 2.4, this instance space is augmented with a new set of instances to fill gaps therein, noting its impact on the ISA methodology. Further discussion associates the difficulties with instance generation methods in terms of feasibility. These issues are addressed with a new instance generation technique able to generate feasible instances across a robust set of problem settings. Finally, Section 2.5 summarizes the contributions of this work and proposes directions for future work.

2.2 Instance Space Analysis

ISA is motivated by the Algorithm Selection Problem (ASP) proposed by Rice (1976). The ASP is concerned with selecting the best algorithm in the algorithm space ($\mathcal{A}$) to maximize the performance in the performance space ($\mathcal{Y}$) given a problem space ($\mathcal{P}$). From this problem space, a set of measurable problem features of interest then defines the feature space ($\mathcal{F}$). Each of these sets form the collection of meta-data $\{\mathcal{P}, \mathcal{F}, \mathcal{A}, \mathcal{Y}\}$. The framework for ISA is illustrated in Figure 1, with the addition of a problem subset ($\mathbf{I} \subset \mathcal{P}$) being inferred by Rice (1976) rather than the problem space. The focus of ISA is to assist in a well constructed problem subset of the problem space ($\mathbf{I}$) to better inform the construction of the remaining meta-data.

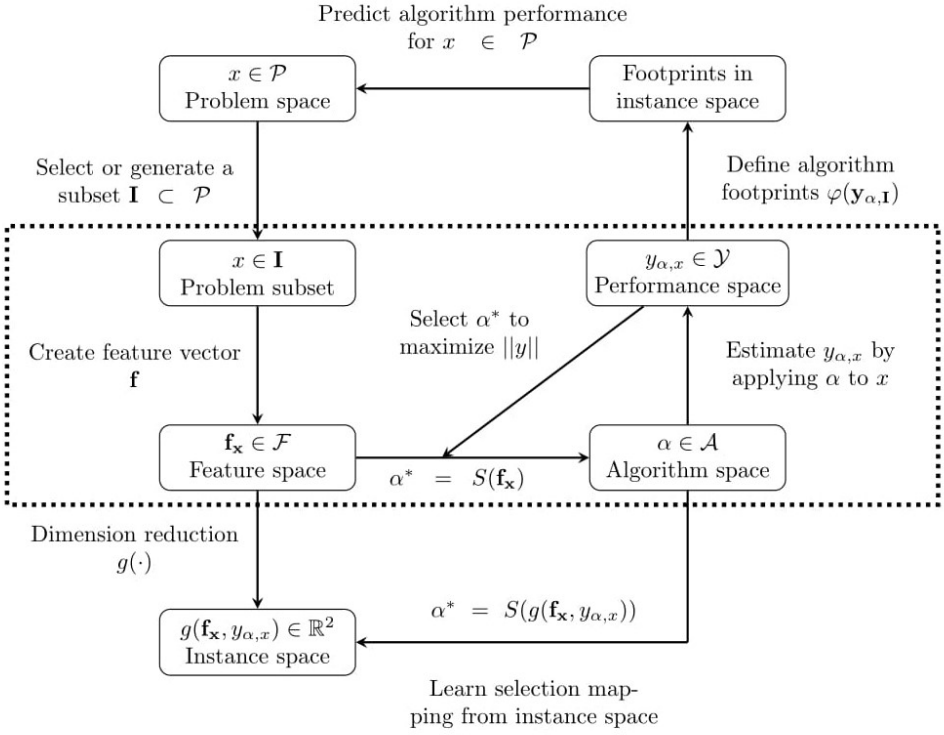

Figure 1. Summary of Algorithm Selection Problem (Rice, 1976) shown in dotted box, extended upon by Smith-Miles et al. (2014)

The Melbourne Algorithm Test Instance Library with Data Analytics (MATILDA) helps posture ISA as an accessible tool for researchers across multiple domains (Smith-Miles, 2019). In MATILDA the steps for conducting ISA are outlined as:
Initialization - Generate the meta-data $\{\mathbf{I}, \mathcal{F}, \mathcal{A}, \mathcal{Y}\}$ shown in Figure 1.

1. Generate the instance space through the selection of key features and then project into a two-dimensional space through a dimension reduction technique.

2. Visualize the existing problem instances to examine the diversity, unbiasedness, and span of existing problem instances.

3. Visualize the algorithm footprints by examining the results of the algorithm performance in different regions of the projected space.

4. Explain strengths and weaknesses of each algorithm across the distribution of key features and the algorithm performance.

5. Use a ML classification algorithm to determine the proper algorithm portfolio for the Algorithm Selection Problem.

6. Generate new instances to fill any gaps present in the instance space. Return to Step 1.

2.2.1 Problem Space

This work considers an extension of the 0-1 multidimensional knapsack problem (MKP) in which there are greater-than-or-equal-to inequalities added to the usual MKP formulation. Additionally, the objective function coefficients are not constrained in sign. This KP variant is referred to as the 0-1 multidemand, multidimensional knapsack problem (MDMKP) (Cappanera and Trubian, 2005).

Given a set of n items to decide whether or not to include in a knapsack with integer profits/costs c_j and integer weights w_{ij} across dimensions $m + q$, solving the MDMKP involves finding the optimal selection of items such that sum of their profits/costs are maximized/minimized. The knapsack has capacity limits across m dimensions and demand requirements across q dimensions such that total weight does not exceed capacity across all m dimensions while meeting demand across all q dimensions. The MDMKP is formulated as:

$$\max \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_{ij} x_j \leq b_i, \qquad \text{for} \quad i = 1, \ldots, m, \tag{2}$$

$$\sum_{j=1}^{n} w_{ij} x_j \geq b_i, \qquad \text{for} \quad i = m+1, \ldots, m+q, \tag{3}$$

$$x_j \in \{0, 1\}, \qquad \text{for} \quad j = 1, \ldots, n, \tag{4}$$

where x_j is a binary decision variable to determine whether to include item j in the knapsack. Additionally, the right-hand-side (RHS) capacity/requirement values are positive, i.e., $b_i > 0$ for $i = 1, \ldots, m + q$, and the weights are assumed to be nonnegative, i.e., $w_{ij} \geq 0$ for $i = 1, \ldots, m + q, \quad j = 1, \ldots, n$.

An MDMKP is *well-stated* if $\sum_{i=1}^{n} w_{ij} > b_i$ for $i = 1, \ldots, m + q$; $\max_j\{w_{ij}\} \leq b_i$ for $i = 1, \ldots, m$; and $\min_j\{w_{ij}\} \leq b_i$ for $i = m + 1, \ldots, m + q$. Meeting these conditions result in an MDMKP instance in the absence of redundant constraints and without a feasible solution when fixing all decision variable values to zero. Each of the m constraints in (2) are referred to as a *knapsack constraint* and each of the q constraints in (3) are a *demand constraint*.

Similar to the MKP, the MDMKP has practical applications by being embedded within other optimization problems such as facility location problems, capital bud-

geting, and portfolio-selection. In terms of computational complexity, the decision version of the 0-1 KP is weakly NP-complete, and by consequence the MKP and the MDMKP as well, unless P = NP (Kellerer et al., 2004*a*). In practical computation, the MDMKP is more difficult than the MKP because feasibility is no longer trivial. The demand constraints and classic knapsack constraints are in direct opposition to each other, removing a heuristic initialization of a null solution. A survey of the developments in the multiple, multidimensional, and quadratic knapsack problems in addition to other variants is discussed in Cacchiani et al. (2022).

2.2.2 Subspace of Instances

The primary source of MDMKP instances is the OR-Library (Beasley, 1990). These instances, referred to as the *Beasley set of instances*, originate from the instance generation method set forth by Cappanera and Trubian (2005). The authors utilize knapsack constraints from MKP instances defined by Chu and Beasley (1998) augmented with demand constraints and objective function coefficients. The generation procedure is as follows: The number of knapsack constraints m are set to 5, 10, and 30, while the number of demand constraints q are dependent on m. After fixing m, q may be set to $q = 1$, $q = m/2$, or $q = m$. The number of variables n is set to 100, 250, or 500, as implemented by Chu and Beasley (1998).

The constraint coefficients are based on a random uniform distribution. The parameters used in the knapsack constraints for the constraint coefficients, as well as the RHS values, are adopted from the corresponding instances defined by Chu and Beasley (1998). The parameters for the demand constraints utilize the same generation method. In particular, each coefficient a_{ij} is selected from a discrete uniform generator `UniformInt(0,1000)`. The RHS b_i are dependent on the tightness ratio α, defined as $b_i = \alpha \sum_{j=1}^{n} a_{ij}$ where $\alpha \in \{0.25, 0.5, 0.75\}$.

The objective function coefficients c_j are constructed to produce an implicit correlation between both types of constraints and the objective function. Since the MDMKP allows c_j to be unconstrained, Cappanera and Trubian (2005) create two cases of MDMKP instances: cost coefficients assumed to be all positive and cost coefficients unconstrained in sign. The positive cost coefficient case are defined as

$$c_j = \sum_{i=1}^{m} \frac{a_{ij}}{m} - \sum_{i=m+1}^{m+q} \frac{a_{ij}}{q} \quad \text{for} \quad j = 1, \ldots, n,$$

while the coefficients are then shifted by a Δ-value to enforce the positive case as

$$c_j = c_j + \Delta + 500u_j.$$

The shift value Δ is defined as

$$\Delta = \begin{cases} 1 + |\min_j\{c_j\}| & \text{if} \quad \min_j\{c_j\} < 0 \\ 0 & \text{otherwise,} \end{cases}$$

and the coefficient is scaled by u_j, a real number drawn from a uniform generator `Uniform(0,1)`. The unconstrained objective function coefficients are referred to as mixed cost coefficients and defined as

$$c_j = \sum_{i=1}^{m} \frac{a_{ij}}{m} - \sum_{i=m+1}^{m+q} \frac{a_{ij}}{q} + 500u_j \quad \text{for} \quad j = 1, \ldots, n,$$

where u_j is a real number drawn from a uniform generator `Uniform(0,1)`. Because the number of demand constraints varies according to fixed m, the set of coefficients are scaled such that there are $nq/(m+q)$ negative coefficients. Cappanera and Trubian (2005) select the first half of the Chu and Beasley MKP instances, then generate six problems from each instance: three from the positive case with $q = 1$, $q = m/2$, or

$q = m$ and three from the mixed case with $q = 1$, $q = m/2$, or $q = m$. There are 810 instances and all are found in the OR-Library (Beasley, 1990).

2.2.3 Algorithm Space

The portfolio of solution methods used in MDMKP research is limited to metaheuristic methods. Cappanera and Trubian (2005) generate instances of the MDMKP in addition to proposing a two-stage tabu search metaheuristic where an outer and inner search method is performed. The outer search oscillates between the feasible boundaries through a constructive and destructive phase to find feasible solutions. The outer search method is based on strategic oscillation described by Glover and Kochenberger (1996). The inner search mimics a standard tabu search method with the collected feasible solutions from the outer search.

Arntzen et al. (2006) propose another method rooted in tabu search. Their metaheuristic also examines the infeasibility of a solution but measures it against the objective function value. This method performs well in the "tight" MDMKP formulations found in the OR-Library but is less desirable for instances where feasibility is not an issue.

Lai et al. (2019) also utilize a nested tabu search framework by first performing an exploratory search for feasible solutions and a second search to exploit the collected list of candidate solutions. Both stages of the search process are guided by a penalty-based evaluation function. Their results indicate the approach performs well in settings with a larger number of knapsack and demand constraints.

Hvattum and Løkketangen (2007) leverage a scatter search method on the MDMKP. Gortazar et al. (2010) also utilize a scatter search method on a variety of general classes of binary integer programs including the MDMKP. The authors apply a penalization approach for constraint violations of the MDMKP. An exact method was

proposed by Hvattum et al. (2010) that employs an alternating control tree (ACT) search method which may also yield inexact solutions, depending on the user's specifications.

A key aspect of ISA is the comparison of algorithmic performances across the instance space. In this work, the focus is not on evaluating different solution procedures but instead using ISA as a tool to demonstrate the inadequacy of current benchmark problem sets. Therefore, the collection of instances are solved with the Gurobi commercial solver to provide a performance measure to help characterize instance difficulty.

2.2.4 Performance Space

The MDMKP is not a trivial problem and has yielded varying results in terms of finding optimal solutions in reasonable time. Song et al. (2022) demonstrate this by utilizing CPLEX version 12 with a 7200 seconds (2 hours) hard limit on each of the 810 instances from Beasley (1990). Of the 810 instances, CPLEX solved 722 to optimality or terminated after 1 hour of no improvement of the optimality gap, 67 instances had feasible but not guaranteed optimal solutions, and in 21 instances CPLEX was unable to find a feasible solution. We have similar findings in Gurobi; it solved 602 instances to within 1% of optimality, identified feasible solutions to 200 instances within 15 minutes, and 8 instances without finding a feasible solution within 15 minutes.

Other KP variants instances tested in the literature are solvable within several seconds (Lu and Vasko, 2020), stressing the importance of classifying instances as difficult or easy before using a solver. Solving a problem with an estimated time based on problem structure is beneficial in real-time decision making settings. Since the majority of the benchmark instances are solvable to within a 1% optimality gap

in a reasonable time, the wall-clock time is used as the measure of instance difficulty. For instances where a feasible solution is not found, the wall-clock time is set to the hard limit (15 minutes). This limit is selected because, in preliminary testing for a subset of Beasley instances, Gurobi produced optimal solutions 5 to 10 times faster than CPLEX. For those instances, previous work by Lu and Vasko (2020) used CPLEX with a 2-hour time limit to identify optimal solutions to MDMKP instances. We utilize an Intel© Core™ i7-11700 CPU at 2.5GHz with 32 GB RAM for each of our tests.

Alternative measures of instances' difficulty include setting the wall-clock time to a fixed constant and then observing the optimality gap across instances, but such a method also requires a user decision regarding how to handle the case in which no feasible solution is identified. Therefore, we select our runtime performance measure as the wall-clock time to get a solution within 1% relative optimality gap between the primal bound z_P and the dual bound z_D, as defined in (5).

$$\text{Relative Gap} = \frac{z_P - z_D}{|z_P|} \tag{5}$$

2.2.5 Feature Space

The feature space is defined by characteristics which capture the structure present within a given instance. The meta-features are defined in Table 1 and are based on research by Hall and Posner (2007) and Smith-Miles et al. (2021) on the 0-1 KP in addition to work by Hill et al. (2012) and Hill and Reilly (2000) on the MKP. A difficulty in defining meta-features for the MKP is the multidimensional number of constraints which typically results in a vector of meta-feature values with varying dimension. Therefore, summary statistics such as the minimum, maximum, and mean of the vector of meta-feature values are reported for the respective meta-features in

Table 1.

The MDMKP is a unique MKP variant due to its additional class of constraints and unrestricted objective function coefficients. The added demand constraints are not linked to the same dimension as the knapsack constraints, therefore these items are treated as "required" amounts of items in each dimension. The unconstrained sign of the profit coefficients also eliminates most heuristic approaches used in the KP. For example, in a greedy heuristic the 0-1 KP the profit coefficients are divided by the associated weight coefficients to generate "efficiency" measures of each item (i.e., $e_i = p_i/w_i$). In the MDMKP, the efficiencies become difficult to define due to the possible negative numerator while also considering the weight associated across multiple dimensions in the knapsack constraints. The demand constraints complicate these heuristic measures further in the case wherein some items with lower weights in the knapsack constraints (efficient) are correlated having lower weights in the demand constraints (inefficient).

Discovering meta-features requires significant domain knowledge over the particular problem of interest and the existing instance generation methods. The MDMKP contains meta-features related to the other KP variants, but the instance generation methods are susceptible to generating instances with concealed biases. For example, the instance generation procedure of Chu and Beasley (1998) for the MKP does not vary the constraint tightness across dimensions for a given instance configuration. If the constraint tightness does not vary across constraint dimensions within a given instance configuration, it is less likely to be a key meta-feature in the ISA methodology. For example, if the constraint tightness varies across constraint dimensions then meta-features are needed to summarize the distribution of tightness values measured across instances. Due to the arbitrary process of engineering meta-features, knowledge in the problem domain and of the existing biases in the instance generation

methods is a requirement to discover meaningful meta-features.

Table 1. Meta-features for the MDMKP

Feature Name	Description - Constraint Type - Statistic
# Decision Vars	Number of decision variables.
# Constraints	Number of constraints - Knapsack and demand.
Constraint Tightness	$\frac{b_i}{\sum_j w_{ij}}, \forall i$ - Both - Min, max, and mean.
Prop Part Dominance	Proportion of items pairs i, j where $p_i \geq p_j$ and $w_{ik} \leq w_{jk}$, or $p_j \geq p_i$ and $w_{jk} \leq w_{ik}$ for any k constraint - Knapsack and demand.
Obj to Constraint Correlation	Pearson correlation coefficient between objective function coefs and constraint coefs for dimension k - Knapsack and demand - Min, max, and range.
Within Constraint Correlation	Pearson correlation coefficient within same class of constraint coefficients for dimension k - Knapsack and demand - Min, max, and range.
Across Constraint Correlation	Pearson correlation coefficient across classes of constraint coefficients for dimension k - Knapsack and demand - Min, max, and range.
Coefficient of Var Weights	Coefficient of variation of weight coefficients across k dimensions - Knapsack and demand.
Coefficient of Var Profits	Coefficient of variation of profit coefficients.

2.3 The Instance Space

The collection of instances, their respective features, and their difficulty measured by the Gurobi solver constitute the meta-data necessary for ISA. The ISA toolkit is executed using MATLAB and is available to download on GitHub (Muñoz and Smith-Miles, 2020*b*). The methods used in the toolkit are described in work by Smith-Miles and Muñoz (2021). The toolkit is automated to perform necessary data preprocessing, feature selection, and instance space visualization across features, algorithms, and data sources. The authors provide customization through an options file to match the users preferences for automatically selecting features, selection criteria, and number of iterations in the functions employed in different stages of the toolkit.

The following description is an overview of the ISA process to reproduce the analysis used in this section. The instance space is constructed by sequentially performing three methods: the Preparation for Learning of Instance Meta-data (PRELIM), the Selection of Instance Features to Explain Difficulty (SIFTED), and Projecting Instances with Linearly Observable Trends (PILOT).

Initially, PRELIM is conducted for the meta-features through a Box-Cox variance stabilization transformation followed by standardizing the meta-features to a standard normal distribution with a z-transformation. From this collection of meta-features, SIFTED is required to distinguish the relevant meta-features and thereby reduce the number of meta-features to project into the two-dimensional instance space.

SIFTED uses the correlation ρ between the meta-features and the performance metrics to find informative features. The performance metric may either be absolute, indicating the performance is good if it exceeds a specified threshold value ϵ, or relative, for which the performance is good if it is marginally close to the best solution procedure y^*. We use an absolute measure of performance with $\epsilon = 0.2$, indicating the solution procedure performance is good if it is the top 20% of the observed response

values. We set our minimum requirement for the correlation between meta-features and wall-clock time as $\rho = 0.3$.

After collecting a reduced set of meta-features, an optimal subset of features is found by enumerating combinations of meta-features and their predictive capability towards the performance metric. In particular, SIFTED clusters features based on their dissimilarity measured by the correlation between (i, j) feature pairs (i.e., $1 - \rho_{i,j}$). The number of clusters is user specified, and we use a default value of 10 clusters. With the 10 clusters of similar features, a single feature is obtained from each cluster, the subset of features are projected to a temporary two-dimensional instance space using principal component analysis, then fed into a random forest classifier model (Breiman, 2001) to distinguish between easy and hard instances. The "optimal" subset is chosen based on the predictive accuracy of the random forest model. From this subset selection method, we reduce our meta-data dimensionality to 10 key meta-features which constitute the instance space.

The configured instance space is then projected into a two-dimensional instance space for visualization purposes. PILOT is a dimensionality reduction scheme to take the important subset of features and produce an optimal two-dimensional instance space with a linear trend from edge to edge in the instance space. The projection is based on a convex global optimization problem with infinitely many solutions due to being highly ill-conditioned (Muñoz et al., 2018). Therefore PILOT is solved numerically using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization algorithm with a user specified number of starting points N_{try} and yields the projection which results from the solution to the global optimization problem found with the best topological preservation. Topological preservation is measured by the correlation between the distances in the ten-dimensional feature space and the distances in the two-dimensional instance space. PILOT resulted in the projection matrix shown as

(6) for the final set of 10 meta-features to a two-dimensional instance space (Z_1, Z_2).

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} -0.3081 & -0.3040 \\ 0.3860 & -0.2429 \\ 0.0309 & 0.1971 \\ 0.1977 & 0.3588 \\ -0.3773 & 0.2181 \\ -0.2576 & -0.2814 \\ 0.2652 & 0.1901 \\ -0.2353 & -0.1835 \\ -0.2794 & -0.0911 \\ -0.2953 & 0.2224 \end{bmatrix}^T \begin{bmatrix} \text{\# Constraints KP} \\ \text{Prop Part Dominant Demand} \\ \text{Max Obj to Constraint Corr KP} \\ \text{Min Obj to Constraint Corr KP} \\ \text{Min Obj to Constraint Corr Demand} \\ \text{Range Within Constraint Corr KP} \\ \text{Min Within Constraint Corr Demand} \\ \text{Range Within Constraint Corr Demand} \\ \text{Max Across Constraint Corr} \\ \text{Coefficient of Var Weights Demand} \end{bmatrix} \tag{6}$$

Figure 2 shows the resulting instance space for the collection of instances from Beasley (1990). The instance space consists of two bands separated by a gap in the instance space, an attribute Hooker (1995) labeled as problem diversity. The two bands are determined mostly by the type of objective function, as defined by Cappanera and Trubian (2005). The first three categories of objective function types are the positive cost coefficient cases defined in Section 2.2 with the respective number of demand constraints dependent on the number of knapsack constraints m i.e., with $q = 1$, $q = \frac{m}{2}$, and $q = m$. The last three categories of objective function types are the mixed cost coefficient cases with the same respective relationships between the demand and knapsack constraints. The two groups are largely characterized by the number of demand constraints. The lower-right cluster is comprised of the $q = 1$ objective function type for both cases with close to half of the $q = \frac{m}{2}$ instances in the upper-right region of the instance space. The upper-left cluster is comprised of the $q = m$ objective function type for both cases with the $q = \frac{m}{2}$ instances spread across

the cluster.

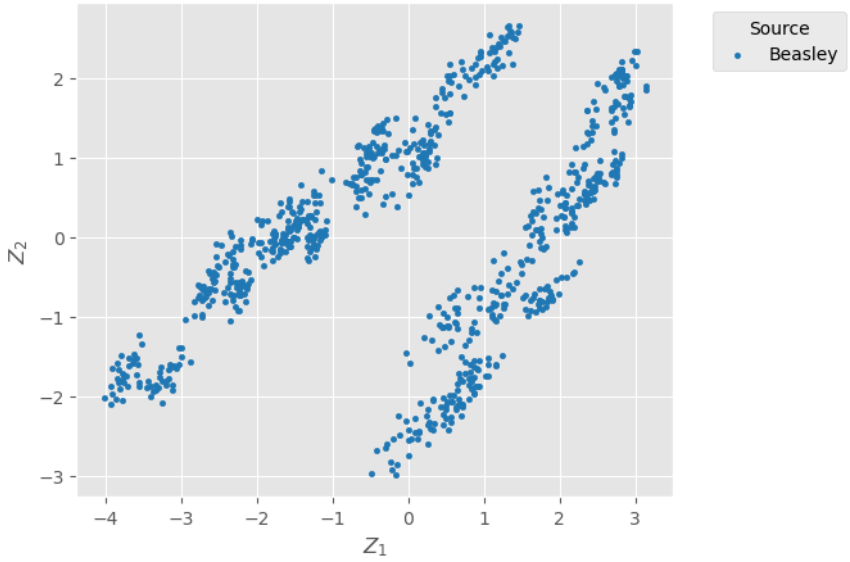

Figure 2. Instance space for the MDMKP based on the instances from Beasley (1990), with projection defined by (6)

PILOT provides an optimal solution to the projection from a ten-dimensional to two-dimensional visualization with linearly observable trends between instance difficulty and the feature values. The resulting visualization conveys these relationships from one edge of the instance space to another. Table 2 shows the difficulty of instances, grouped by the type of objective function, reporting the average solution time required for each objective function type. For the positive coefficient case, the average time required to identify a within 1%, near-optimal solution is 188 seconds, while the mixed case required an average of 382 seconds. However these averages do not capture the variability within each problem class, which exhibit heavy-tailed distributions with standard deviations over 150 seconds for each objective function type.

Since PILOT creates linear trends and clusters through instance difficulty, we report lower solution times for the single demand constraint objective functions, with larger difficulties as the number of demand constraints is increased. ISA reports the

difficult instances lie in the lower-left region of the instance space.

Table 2. Average runtimes (seconds) to identify a within 1% near-optimal solution by objective function type.

	Positive	Mixed
$q = 1$	100.1	64.1
$q = \frac{m}{2}$	286.1	359.3
$q = m$	302.9	404.8
Average	188.3	382.1

2.3.1 Augmenting the Instance Space

Previous research on the 0-1 KP and the MKP investigating problem structure found a lack of a range of correlation structures across instances. Reilly (2009) discusses the lack of ranges of correlation values with the 0-1 KP and Hill et al. (2012) in the MKP. In the MKP, the correlation between the objective function and each dimension of the knapsack constraint is computed. From this standpoint, we examine the correlations for the current collection of MDMKP instances respectively in Figures 3 and 4 for the knapsack constraints and the demand constraints, for which we report similar findings. Ideally the range should contain instances with correlations from -1 to 1.

In the correlation between both types of constraints, across each objective function type, the current collection of instances is lacking the full range of correlation values. For the knapsack constraints, the range is limited to -0.25 to 0.25. The demand constraints show a different pattern across objective function type; the first and fourth types having $q = 1$ exhibit highly negative correlations, whereas the other objective function types; correlations trends towards zero because of the instance generation technique adopted from Cappanera and Trubian (2005). In this subset of meta-features for the MDMKP, the correlation between the objective function coefficients and constraint coefficients poses concerning issues due to their limited range

of correlation values observed.

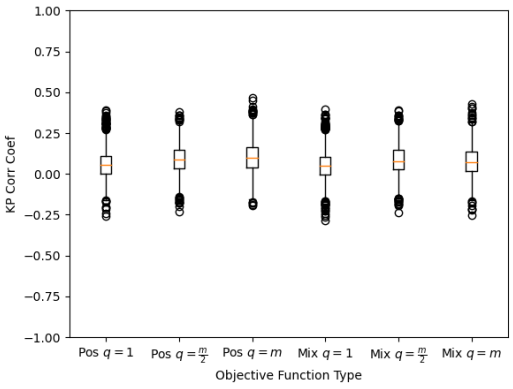

Figure 3. Correlation between c_j and w_{ij} for $i = 1, \ldots, m$

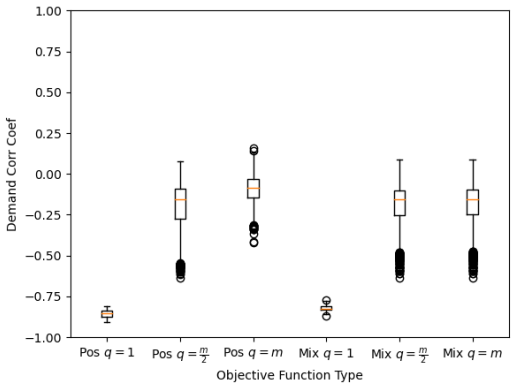

Figure 4. Correlation between c_j and w_{ij} for $i = m+1, \ldots, m+q$

Cho (2005) develop a new set of instances for the MKP with a set correlation structure between the objective function coefficients and the knapsack constraints. These MKP instances were employed to test the effectiveness of heuristic methods for the MKP in Hill et al. (2012). The test instances from Cho (2005) also allow for varying tightness for each RHS value across each dimension of the MKP. From this MKP instance, an MDMKP instance is constructed by adding demand constraints from a discrete uniform generator `UniformInt(0,1000)` and setting the RHS based on the same tightness ratios used in Cappanera and Trubian (2005). The MDMKP instance is then checked for feasibility and is removed if an infeasible instance is generated. We generate 165 feasible MDMKP instances to add to our problem space.

This instance generation method is an ECI method because the correlation structure is set *a priori* and the instance parameters are augmented to match. Of note, the method from Cappanera and Trubian (2005) is also considered to be an ICI method because the correlation is assumed, but not guaranteed, to match the desired correlation structure. By using the ECI method on the set of knapsack constraints for our 165 instances, we generate instances with correlation values from -1 to 1.

The unique aspects of these new MDMKP instances, referred to as the *Cho set*

of instances (Cho, 2005), are the correlation structure and the varying tightness of the RHS values for the knapsack constraints. The *Beasley set of instances* (Beasley, 1990) rely on an induced correlation structure and fixed tightness of the RHS values for both classes of constraints. We utilize the constructed instance space from the Beasley instances to see the impact of the newly constructed Cho instances in filling the instance space. By keeping the ISA methods from Section 2.2 the same, we utilize the projection matrix in (6) to create Figure 5.

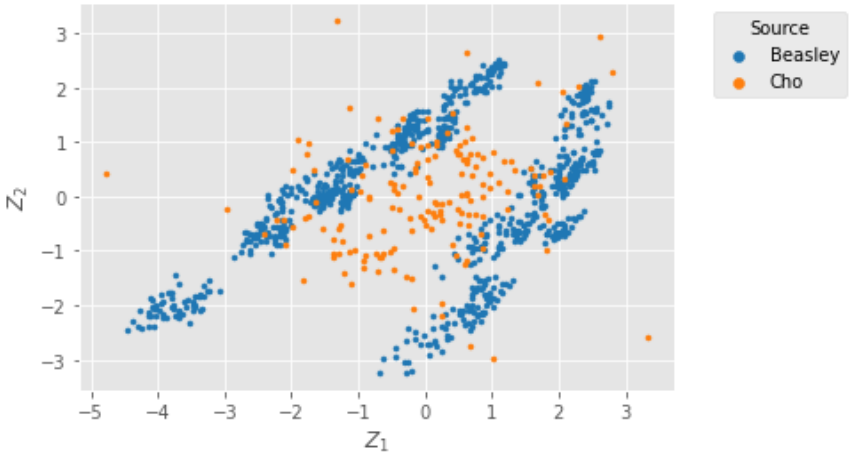

Figure 5. Instance space for the MDMKP based on the instances from Beasley (1990), augmented with instances generated from Cho (2005)

The generated Cho instances cover some of the gaps present within the instance space. The space between the two large clusters of MDMKP instances from the Beasley set of instances is partially filled by the newly generated instances. Unfortunately, the projection in (6) is optimized for the Beasley set, making it difficult to project the Cho set. Figure 6 portrays an updated instance space with a projection optimized for both the Beasley set and Cho set. The instance space is separated into two clusters defined by each instance source.

The updated instance space in Figure 6 portrays a different story than exhibited by Figure 5. Due to the introduction of a different set of instances, the projection matrix shown in (7) reveals a different grouping of important meta-features relevant

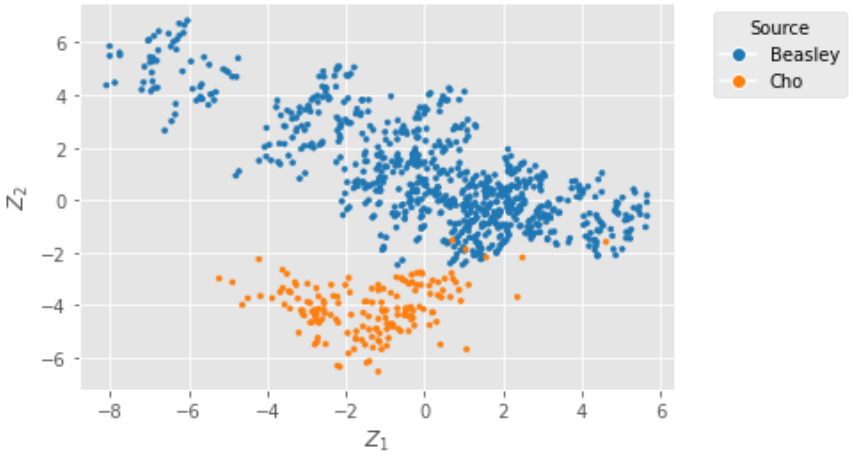

Figure 6. Instance space for the MDMKP based on Beasley and Cho instances

across both sets. There is not a lot of overlap between the two problem sets shown in Figure 6, which is expected given their varied problem generation methods.

The impact of the added set of instances on the feature selection process is illustrated in the differences of their projection matrices. Originally, no meta-features pertaining to the tightness of the constraints were found to be important for characterizing the instance space. After adding the Cho instances, which vary the tightness across each dimension of the knapsack constraints, the feature is found to be important. However, of the new set of ten features shown in (7), five remained the same, albeit with different weights.

To assess the distribution of meta-features across the instance space, the features are individually selected and shown as their value increases where they lie within the instance space. These figures are provided in A. The lower-right portion of the instance space consists of the easier instances in the instance space. The easier instances are correlated with less knapsack constraints and demand constraints, a smaller range of objective function to demand constraint correlation values, and decreased values for the max across constraint correlation. While difficult instances are found with increased values of max across constraint correlation, and lower values of the min

within constraint correlation for demand constraints.

ISA fits a support vector machine (SVM) as a classification model to the meta-data with k-fold cross validation to predict the best algorithm to use for each instance. In our work with only one algorithm, the SVM performs a binary classification of whether or not Gurobi performs well on a given instance. With our augmented instance space we predict classes with 89.80% accuracy with $k = 5$ folds. For comparison, with only the original Beasley instances and projection from (6), we obtain an accuracy of 90.10%. In both cases, our current collection of meta-features sufficiently characterize the MDMKP.

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} 0.5091 & -0.3914 \\ 0.0064 & -0.6332 \\ -0.4568 & -0.5807 \\ -0.4224 & -0.6215 \\ 0.0399 & -0.5695 \\ 0.7947 & -0.0912 \\ -0.7691 & -0.0116 \\ -0.4456 & 0.3111 \\ 0.4997 & -0.2804 \\ -0.4682 & -0.5941 \end{bmatrix}^T \begin{bmatrix} \text{\# Constraints KP} \\ \text{\# Constraints Demand} \\ \text{Max Constraint Tightness KP} \\ \text{Max Obj to Constraint Corr KP} \\ \text{Range Obj to Constraint Corr Demand} \\ \text{Min Within Constraint Corr Demand} \\ \text{Range Within Constraint Corr Demand} \\ \text{Max Across Constraint Corr} \\ \text{Min Across Constraint Corr} \\ \text{Coefficient of Var Weights KP} \end{bmatrix} \tag{7}$$

Augmenting the instance space through instances defined for the MKP introduces several problems illustrated in this work. While the instance space shown in Figure 6 indicates the new instances are exploring separate regions of the instance space, they lack a full exploration of possible meta-feature values. Of the 165 feasible MDMKP instances from the Cho set, 160 contained one demand constraint, and the remaining 5 contained only two demand constraints. The varying constraint tightness ratios

across each knapsack constraint dimension impairs the number of demand constraints to construct a feasible instance. Because of the lack of difficult demand constraints, the average solve time for an instance from the Cho set of instances is 0.07 seconds.

2.4 Feasibility Considerations on Robust Problem Construction

Constructing feasible instances is a large issue in instance generation schemes. In optimization problems defined on a graph, such as a shortest path problem, a feasible path must exist from the source to sink node. Therefore, the adjacency matrix or adjacency list must characterize a directed graph with at least one feasible path. In algorithmic testing settings, it is desirable to examine instances having more than one feasible solution. Finding the number of feasible solutions for a given instance itself is a hard problem which requires enumeration.

Feasibility becomes less difficult to achieve for less complex problems such as the TSP or 0-1 KP because there are fewer competing aspects to consider. Relative to the 0-1 KP, the MKP addresses an increased number of dimensions. However the null solution of selecting no items remains feasible for both KP variants. A difficult aspect of the MDMKP is the introduction of the demand constraints, which removes the guarantee of a feasible null solution. Increasing the number of dimensions for the demand constraints also creates a more complex problem for which to construct feasible instances. Lastly, the consideration of creating a *well-stated* MDMKP is another difficult aspect to creating MDMKP instances.

Feasibility also competes with other desirable properties (e.g., diversity) in an instance generation scheme. Given an instance generation method which produces feasible instances, such as the method set forth by Cappanera and Trubian (2005) for the MDMKP, the generation scheme may not be flexible enough to fully explore the instance space. In previous work of KP variants, the correlation structure and

knapsack constraint tightness were important for determining problem difficulty (Hill et al., 2011). In the Cho set of instances, our work involved changing the tightness across each dimension of the knapsack constraints in the MDMKP but resulted in a number of infeasible solutions. Whereas the Beasley instances demonstrated an easier method to create feasible MDMKP instances, they result in a lack of variation in constraint tightness across the dimensions of the knapsack.

2.4.1 Feasible Instance Generation

To address these competing issues of feasibility and diversity, we introduce the Primal Problem Instance Generator (PPIG). PPIG is a sampling-based instance generation method which begins in the primal feasible space. Detailed in Algorithm 1, the method involves generating a set of k solutions where each decision variable is set active (i.e., to a value of 1) with probability p. The instance's left-hand-side (LHS) coefficients for each constraint are generated in some predetermined fashion to obtain the problem meta-features desired. Each of the k solutions, when applied to the constraint coefficients, yield the required RHS values for each of the k instances, providing an empirical distribution of right-hand side values, which is then used to set the final RHS values for each constraint for the problem instance.

Algorithm 1 Primal Problem Instance Generator

Input: Number of items n; Number of knapsack constraints m; Number of demand constraints q; Number of simulations K;

Output: RHS values b_i;

$\mathbf{W}^{le} \leftarrow$ `UniformInt(0,1000)` ▷ LHS values for knapsack constraints $[m \times n]$

$\mathbf{W}^{gr} \leftarrow$ `UniformInt(0,1000)` ▷ LHS values for demand constraints $[q \times n]$

for $k = 1, \ldots, K$ **do**

- $p \leftarrow$ `Uniform(0,1)` ▷ Fix probability level
- $\mathbf{x}_k \leftarrow$ `Bernoulli`(n, p) ▷ Generate a vector of Bernoulli trials as a solution
- $\mathbf{r}_k^{le} \leftarrow \mathbf{W}^{le}\mathbf{x}_k^T$ ▷ Calculate knapsack resources used $[m \times 1]$
- $\mathbf{r}_k^{gr} \leftarrow \mathbf{W}^{gr}\mathbf{x}_k^T$ ▷ Calculate demand resources used $[q \times 1]$

for $i = 1, \ldots, m$ **do**

- $b_i \leftarrow 50\{\mathbf{r}_k^{le}\}$ ▷ Median RHS value for knapsack constraints

for $i = m+1, \ldots, m+q$ **do**

- $b_i \leftarrow 25\{\mathbf{r}_k^{gr}\}$ ▷ 25th percentile RHS value for demand constraints

After generating the RHS values, the remainder of PPIG includes selecting the w_{ij}-values from the $\mathbf{W}$ matrices. The objective function values follow the same procedure as the mixed cost coefficient case used by Cappanera and Trubian (2005). The c_j-values are based on a weighted average between the knapsack and demand constraint coefficients, shifted through a random number u_j, drawn from a uniform generator `Uniform(0,1)`, and scaled so the number of negative c_j-values is proportional to the number of demand constraints q.

We utilize parameter settings similar to the Beasley and Cho sets of instances. In particular, we generate instances with $K = 100$, $n = \{100, 150,$

$200, 250, 500\}$, $m = \{5, 10, 30\}$, and $q = \{1, \frac{m}{2}, m\}$. This results in 45 new instances, referred to as the *Scherer set of instances*, to examine within the instance space. The selection for the number of items in the knapsack n is different than the other two sets of instances because the initial results with $n = 500$ indicated most instances did not solve to a 1% relative optimality gap within the maximum allowable time (15 minutes). Generating instances with $n = 150$ or $n = 200$ results in instances with wall-clock times less than the maximum time allowed. Additionally, the current collections of instances are either solvable within 5 seconds or reach the maximum time of 15 minutes. Therefore, the number of items is varied in smaller sets to obtain distinguishing performance measures within these two extremes.

Compared to Bowly et al. (2020), PPIG samples the distribution of generated LHS and RHS values to complete the instance generation. In contrast, Bowly et al. (2020) focus on a single LP instance originating from an alternative encoded space and transforming into the instance space with desirable meta-features. The approach presented in this work is specific to the MDMKP and KP variants by allowing any configuration between the objective function coefficients and constraint coefficients. Both PPIG and the instance generation approach from Bowly et al. (2020) utilize a construction method by starting from a solution and building an instance around it. However, we characterize PPIG as a *sampling-based approach* because of its use of parameter distributions from constructed parameter distributions, distinguishing it from the method set forth by Bowly et al. (2020), which we characterize as an inherently *constructivist approach*.

PPIG can efficiently generate feasible instances for a robust set of input parameters. The time complexity of Algorithm 1 is $O(N)$ which permits efficient construction of any range of instances without feasibility issues. We empirically did not examine any feasibility issues with our settings used for sampling the RHS values. If the pa-

rameters (n,m,q,p) are set to match the parameters (n,m,q,α) in the Beasley set of instances, the resulting RHS values are feasible.

PPIG is more likely to construct infeasible instances at the extremes of parameter settings. For example, the probability level p is chosen uniformly from 0 to 1, if the distribution is bimodal with $p < 0.1$ or $p > 0.9$, the resulting RHS distributions will produce instances with respectively tight or loose constraints. Another parameter setting is the percentile of the distribution of RHS values from which to sample the b_i-values. We utilize the median and 25th percentile for the respective knapsack and demand constraints because they allow a varying degree of constraint tightness while preserving feasibility in the cases we observe. When we utilized the "best-case" of both RHS distributions, i.e., the maximum observed $\mathbf{r}_k^{le}$-value and minimum observed $\mathbf{r}_k^{gr}$-value for each repetition k, we would guarantee feasible instances but obtain trivial instances with no constraint slackness across any dimension.

2.4.2 Analysis of Feasible Instances

The Scherer set of 45 MDMKP instances contain both easy and difficult to solve problem configurations. Of the 45 instances, 40 were solvable to the optimality gap in less than 15 minutes with an average solution time of 21 seconds. The remaining 5 instances where in either of two other categories; 4 instances had identified a feasible solution, but Gurobi could not identify a 1% gap, nearly optimal solution within 15 minutes; and 1 instance eluded the identification of a feasible solution in the allotted runtime.

To verify the Scherer set of instances are novel to the instance space, the instances are projected into the instance space of the Beasley and Cho instances via (7) and shown in Figure 7. The 45 instances help fill gaps within the instance space between the Beasley and Cho sets. Unlike the Cho instances, the new instances are

more difficult to solve and visually spread across the instance space with the Beasley set. Additionally, the new instances contain similarity to both existing sources by containing instances within the two clusters, predominantly the Beasley set.

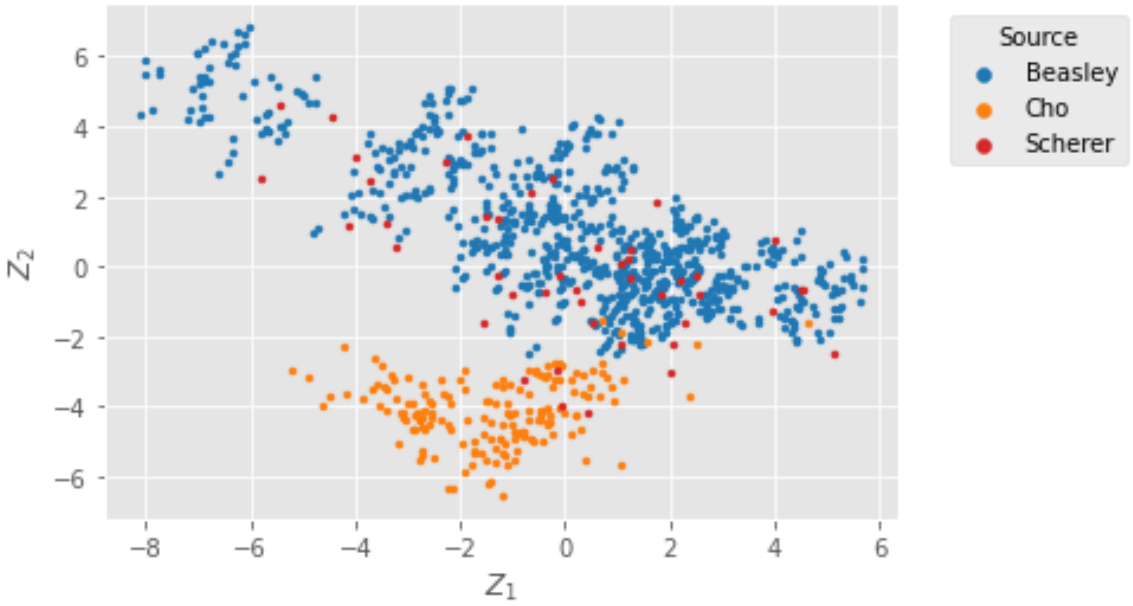

Figure 7. ISA of new instances on the Beasley and Cho instance space

In particular, the instance space of the Beasley and Cho sets shown in Figure 6 show two clusters separated by their correlation structure: the Beasley set resulting from an ICI method and the Cho set from an ECI method. The Scherer set is spread throughout the Beasley set because they also are constructed using an ICI method. However, their variation in constraint slackness helps fill gaps present in the instance space. In addition, the ECI method used by Cho (2005) can be integrated into PPIG to create feasible and difficult MDMKP instances with controlled correlation structure and constraint tightness.

After performing the ISA for all three sets of test instances, we obtain a projection matrix shown in (8). Figure 8 depicts the resulting instance space, along with the theoretically feasible boundary of instances. Similar to Figure 7, the new instances are spread across the instance space comprising the Beasley set of instances.

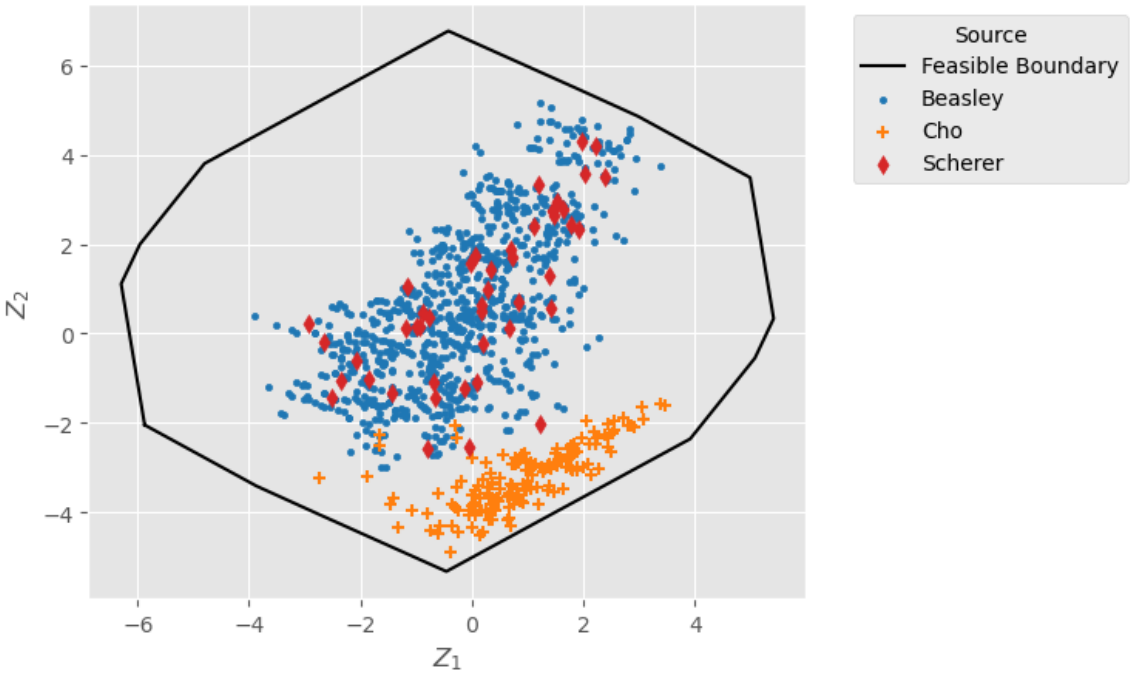

Figure 8. ISA of Scherer, Beasley, and Cho instance space with theoretical feasible boundary

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} 0.132 & 0.364 \\ 0.179 & 0.357 \\ 0.406 & -0.230 \\ 0.498 & -0.279 \\ 0.239 & 0.303 \\ 0.057 & -0.407 \\ -0.044 & 0.343 \\ -0.452 & -0.282 \\ 0.430 & 0.319 \\ 0.360 & -0.246 \end{bmatrix}^T \begin{bmatrix} \text{\# Constraints KP} \\ \text{\# Constraints Demand} \\ \text{Max Constraint Tightness KP} \\ \text{Max Obj to Constraint Corr KP} \\ \text{Range Obj to Constraint Corr Demand} \\ \text{Min Within Constraint Corr Demand} \\ \text{Range Within Constraint Corr Demand} \\ \text{Min Across Constraint Corr} \\ \text{Range Across Constraint Corr} \\ \text{Coefficient of Var Weights KP} \end{bmatrix} \tag{8}$$

Table 3 reports the average wall-clock time of Gurobi to solve the test instances

across each source. The average runtime performance of Gurobi is lower for the Scherer set compared to the Beasley set, but we note that both distributions of runtimes contain instances which are easy to solve but also require the maximum runtime to identify a near-optimal solution. A visualization of the instance space including all three sources is shown in Figure 8. The Scherer instances align within the Beasley instances. In this instance space configuration, the difficult instances lie in the lower-right portion of the instance space. The difficult region of the instance space is characterized by the distribution of meta-features displayed in Figures 34-36 within A. In general, the generation of more difficult instances relies on higher values of the max across constraint correlation between the knapsack and demand constraints, higher values of max within constraint correlation for the demand constraints, and higher values for the number of demand constraints.

Table 3. Average runtimes (seconds) to identify a within 1% near-optimal solution by instance source.

Beasley	Cho	Scherer
252.9	0.07	147.15

To explore the regions of the instance space, we adopt PPIG to control specific meta-features which dictate the region in which the instances lie. In particular, we seek to generate instances on the current frontier of instances shown in Figure 8. By examining the distribution of features for each meta-feature defined in (8), instances having higher values of max within constraint correlation for the demand constraints, in addition to higher values of max constraint tightness for knapsack constraints, will yield instances in the lower-right portion of the instance space, i.e., closer to the point (5,0). Since PPIG is sampling-based strategy over a distribution of MDMKP parametric characteristics, the parameters in Algorithm 1 may be modified to target this region. For example, specifying the quantile for the RHS value selection to force higher values of max constraint tightness and manipulating the constraint correlation

structure of the demand constraints through the ECI method from Iman and Conover (1982).

Overall, PPIG presents a new class of instances in the instance space with controllable meta-features without losing feasibility of the generated instance. The PPIG approach develops an empirical distribution of resources by sampling from $\mathbf{r_k}$-values to calculate the final RHS values. We utilized set percentiles for both types of constraints, which ensures feasibility across the current set of input parameters. Future work includes adapting PPIG for worst-case of both types of constraints, and ensuring feasibility by relaxing the constraint tightness across each dimension. PPIG allows for a target region in the instance space by controlling for meta-features directly through the input parameters. PPIG is a flexible instance generation method for a variety of optimization problems with feasibility assisted by building instance constraints relative to primal feasible instances.

2.5 Conclusion

This research demonstrates a methodology for characterizing the current collection of instances for the MDMKP, contributes new sets of test instances to fill the instance space, and provides a method to generate feasible instances of the MDMKP across any range of meta-features. The conducted ISA leveraged knowledge from the 0-1 KP and MKP to find meaningful features to characterize the instances, as well as to examine current instance generation methods and identify potential weaknesses in approaches. The identified weaknesses include the constraint tightness varying across dimensions or the correlation structure, both of which were varied in a flexible approach used in MKP construction from previous work (Cho, 2005). However, when constructing an MDMKP instance from an MKP instance, feasibility becomes an issue and is addressed through PPIG. The 810 original instances were examined

in a two-dimensional visualization of the instance space and were augmented with additional instances. The resulting instance space identified the importance of varying tightness across constraint dimensions in addition to specifying the correlation structure through an explicit correlation induction technique.

In addition to demonstrating the impact of a different instance generation method for the MDMKP on the instance space, this work discusses the shortcomings of instance generation methods for problems with nontrivial feasible regions. The MDMKP is unique in that it contains competing constraints across multiple dimensions. We find there a number of trade-offs that occur in instance generation methods to produce feasible instances. Approaches which guarantee feasible solutions, such as the generation method used by Cappanera and Trubian (2005), sacrifice completely filling gaps in the instance space, whereas the methods used in this work do not guarantee feasibility but help to cover unexplored regions of the instance space. The method employed by Cho (2005) for the MKP is difficult to generalize for the MDMKP and maintain feasible instances. In contrast, the instance generation method presented in this work allows controllable characteristics in the MDMKP without sacrificing feasibility.

The instances from PPIG demonstrate an instance generation method which emphasizes feasibility by constructing primal feasible instances and building the parameters from this foundation. The approach is flexible to be used in further work, as an even wider array of instances of the MDMKP but also for other KP variants. Integrating the generator into an ISA framework allows for objective measures of diversity in newly generated instances, as well as a measure of difficulty.

The work presented here highlights the important of ISA in instance generation methodologies. ISA is used to validate the novelty of an instance generation method by examining where it lies in the current collection of instances in the instance space.

While the MDMKP is the problem of interest, the application to any KP variant is straightforward. For example, the correlation structure, constraint tightness, and coefficient distribution are measurable for the 0-1 KP, the MKP, and the multiple choice knapsack problem (MCKP).

To directly extend this work, it is of merit to examine in detail the tailorability of the proposed PPIG method's sampling of instance parameter distributions to target specific subregions for instance generation rather than seeking a broad distribution of instance characteristics. For example, there may be interest in generating feasible instances within a certain subregion within Figure 8. Additionally, future work may incorporate the similarities and differences of the instance space for optimization problems belonging to the same class of problem, but with modified formulations. Also worth examining is the extent to which PPIG can be leveraged to generate feasible LP test problems, in comparison to existing methods.

III. Instance Based Configuration for Metaheuristic Selection

3.1 Introduction

Newer optimization solution methods usually claim to produce either better quality solutions then existing methods, or equal quality solutions more efficiently yet both often use less than definitive testing methods. Given the testing is the basis for verifying the contribution, computational testing is "critical" to the paper per the guidelines introduced by Greenberg (1990). Whether through theoretical analysis or empirical testing, the quality and efficiency of a solution approach are two measures of the effectiveness of a solution approach in optimization. The works by Hooker (1994) and Hooker (1995) advocate for a rigorous treatment in empirical testing beyond theoretical worst-case or average-case analysis urging the optimization community to push the envelope of empirical testing by not only showing *How does the solution procedure perform?* but also answering *Why does the approach perform well?*

The focus of most research papers is competitive testing against a fixed set of benchmarks, rather than focusing on controlled experimentation to uncover empirical and theoretical insights. According to the No-Free-Lunch (NFL) Theorem by Wolpert and Macready (1997), it is impossible for any one algorithm to outperform all other algorithms when averaged over all possible optimization problems. Any "state-of-the-art" approach to a stated problem will perform well on some fixed set of instances, without any indication of the success being generated from the solution approach, or random chance.

A better use of research is through rigorous testing under a variety of empirical circumstances and relating findings to the original intention of research: To increase the capacity of knowledge over a given topic. In optimization, many advancements

are made in solution methodologies with promises of higher quality solutions with less time required to find these solutions. However these advancements often fail to mention conditions where the proposed methodology performs poorly and is not expected to succeed. In this work, we provide a methodology to demonstrate where those conditions occur through the use of an expanded set of test instances.

A new approach called Instance Space Analysis (ISA) (Smith-Miles, 2019) helps describe the spectrum of problem conditions where solution methods have strengths and weaknesses. ISA provides a methodology to evaluate the current collection of test instances, compare algorithm performance across this collection, and summarize this information through low-dimensional visualizations. ISA emanates from a series of works beginning with Smith-Miles and Lopes (2012) measuring instance difficulty across a variety of well-known optimization problems. Subsequent work by Smith-Miles et al. (2014) develops the idea of an instance space and examining algorithmic strengths and weaknesses across the instance space. The instance space also presents gaps where the current collection of instances fail to provide test instances, resulting in Smith-Miles and Bowly (2015) proposing a method to fill the gaps in the instances space with an instance generation method. After these three main pieces constructing the visualization methodology is improved with a novel projection algorithm (Muñoz et al., 2018). The final methodology supporting the online tool MATILDA (Smith-Miles, 2019) is summarized in a recent tutorial paper (Smith-Miles and Muñoz, 2021).

ISA is applicable to many settings including optimization, machine learning (Muñoz et al., 2018, 2021), anomaly detection (Kandanaarachchi et al., 2020), and time series forecasting (Kang et al., 2017). Optimization is the most prevalent, with the recent work by Smith-Miles et al. (2021) revisiting *Where are the hard knapsack problems?* originally questioned by Pisinger (2005). Further work includes ISA applied to course timetabling (De Coster et al., 2022), max flow (Alipour et al., 2023), and black-box

optimization (Muñoz and Smith-Miles, 2020*a*).

To characterize the structure of a given problem instance, ISA utilizes meta-features which summarize the relationship between defined problem parameters in an specific formulation. Examples of meta-features are provided in the methods set forth by Smith-Miles and Lopes (2012) and are divided into two groups: Problem independent features and problem specific features.

Problem independent features are mostly borrowed from fitness landscape analysis. The fitness landscape is defined by a set of solutions, a fitness function, and a neighborhood based on the set of solutions. From this fitness landscape, multiple features such as the fitness distance correlation, the distribution of local minima, and the ruggedness of a solution generated by a random walk process. In general, fitness landscape analysis has not sufficiently helped characterize optimization problems (Bierwirth et al., 2004).

In contrast to problem independent features, problem specific features help measure instance difficulty and characterize the instance space (Smith-Miles et al., 2014). For problems based on graphs, where a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is defined by a set of N vertices $\mathcal{V}$ connected by a set of edges $\mathcal{E}$, there are measures defined by graph theory. For knapsack problems (KP), with the goal of maximizing the value of items put into a constrained knapsack, there are relationships defined between the value and weight of an item (Cho et al., 2008; Hall and Posner, 2007; Smith-Miles et al., 2021).

In this work, the problem of interest is the 0-1 multidemand, multidimensional knapsack problem (MDMKP). The MDMKP is an extension of the 0-1 multidimensional knapsack problem (MKP), where the objective is to fill a knapsack with items which contain positive and negative contributions to the overall objective. Each item contains a corresponding weight across each dimension; the total sum of each dimension cannot exceed the stated capacity for each dimension. In contrast to the MKP,

there are an added set of demands, or covering constraints, which must be satisfied across multiple dimensions. For example, the MDMKP may model the limitations and requirements modeled in portfolio optimization. Specifically the model may include demands of a portfolio requiring levels of environmental, social and governance (ESG) scores while limiting the percentage of total investments in a certain sector.

This novel research examines MDMKP meta-heuristic performance under the lens of ISA, specifically examining the relationship between problem specific meta-features and select metaheuristic performance. The available problem set is expanded through the use of a newly developed instance generation technique that produces feasible MDMKP instances with controllable problem characteristics and varying degrees of difficulty. The generated instances are then combined with currently available instances established in the literature to constitute the new instance space explored. A machine learning classifier is used to examine the "algorithm footprints", the areas of the instance space with better solution procedure performance, which is mapped to the exploited structure in the problem. A classification tree model provides an interpretive link between problem structure and recommended metaheuristic selection. A variety of instances are visualized in the instance space with their respective footprints to demonstrate the effectiveness of the classification tree. The results demonstrate the utility of the ISA framework for rigorous empirical testing and provide areas of the instance space where current solution methods need improvement.

The remainder of this work is structured as follows. Section 3.2 outlines each component defined in ISA, beginning with the problem space and ending with the instance space visualization. Section 3.3 describes the metaheuristic performance across the instance space and the strengths and weaknesses of each for different problem configurations. Section 3.4 provides the results and analysis of machine learning classification model for the MDMKP Algorithm Selection Problem (ASP). Section 3.5

discusses the results and summarizes the contributions of this research.

3.2 Instance Space Analysis

This section examines the component spaces of ISA for the MDMKP starting with an overview of the ASP, a breakdown of ISA into its respective component spaces, and an introduction to the problem space, the subset of available test instances, the algorithm space, the performance space, and the feature space. These five areas are used to construct the instance space, which gives a two-dimensional plane to visualize the current collection of instances, positioned by their instance structure and algorithmic performance.

3.2.1 Algorithm Selection Problem

ISA originates from the ASP, defined in the seminal work by Rice (1976) for differential equations, but adopted for an optimization problem framework. The ASP approach is summarized in Figure 9, where the section encapsulated by the dotted line is Rice's work extended by Smith-Miles et al. (2014).

The ISA framework depicted in Figure 9 divides the ASP into six component spaces. The initial requirement for ISA is some problem of interest which is defined by the problem space, $\mathcal{P}$, and all possible feasible instances of the problem. In this work, the MDMKP is the problem of interest and the subset of instances, $\mathbf{I}$, are the test instances used in empirical testing. The feature space, $\mathcal{F}$, encapsulates the structure of the MDMKP using meta-features to quantify the relationship of performance and problem configuration. The algorithm space, $\mathcal{A}$, constitutes the algorithms found in the literature for the MDMKP, of which a representative subset are used in this work. The solution procedures are measured in the performance space, $\mathcal{P}$, where $y_{\alpha,x}$ is the performance metric of algorithm $\alpha \in \mathcal{A}$ for instance $x \in \mathbf{I}$. The result of defining

each component space allows for a dimension reduction, $g(\cdot)$, to visualize the instance space, the algorithm footprints, $\varphi(\mathbf{y}_{\alpha,\mathbf{I}})$, to see trends of algorithm performance, and the objective of the ASP, $\alpha^* = S(g(\mathbf{f}_\mathbf{x}, y_{\alpha,x}))$, to find the best algorithm for a given instance.

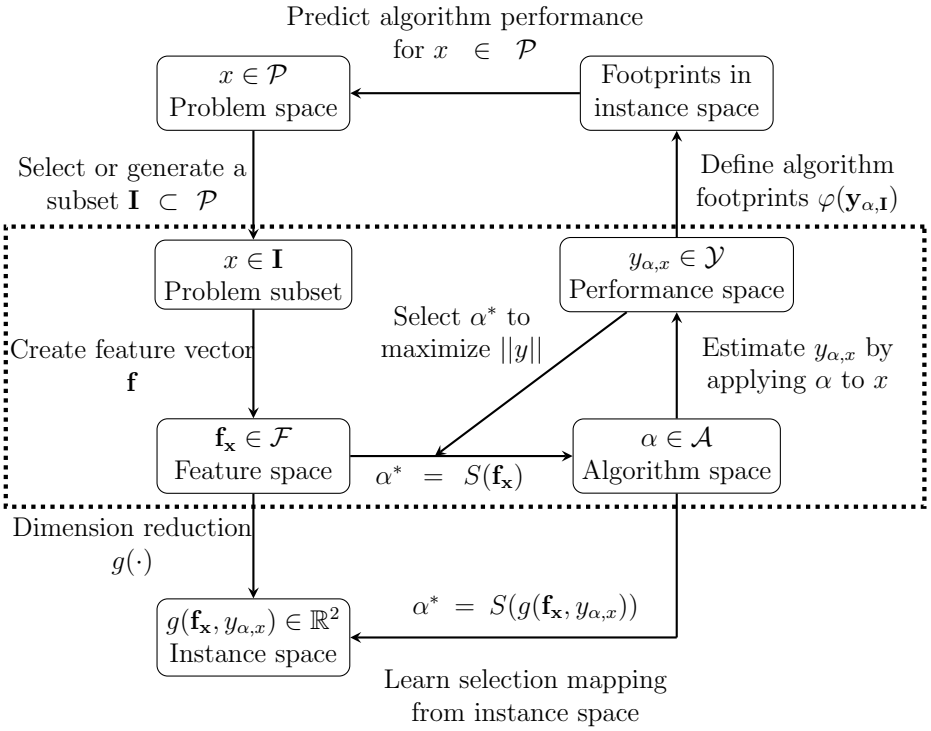

Figure 9. Summary of Algorithm Selection Problem (Rice, 1976) shown in dotted box, extended upon by Smith-Miles et al. (2014)

3.2.2 Problem Space

The MDMKP appears to have been originally defined by Cappanera and Trubian (2005) and emerged from previous work in discrete facility location problems (Cappanera et al., 2003). The MDMKP is formulated as:

$$\max \quad \sum_{j=1}^{n} c_j x_j \tag{9}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_{ij} x_j \leq b_i, \qquad \text{for} \quad i = 1, \ldots, m, \tag{10}$$

$$\sum_{j=1}^{n} w_{ij} x_j \geq b_i, \qquad \text{for } \; i = m+1, \ldots, m+q, \tag{11}$$

$$x_j \in \{0, 1\}, \qquad \text{for } \; j = 1, \ldots, n, \tag{12}$$

where x_j is a binary decision variable to determine whether to include item j in the knapsack. Additionally, the right-hand-side capacity/requirement values are positive, i.e., $b_i > 0$ for $i = 1, \ldots, m+q$, and the weights are assumed to be nonnegative, i.e., $w_{ij} \geq 0$ for $i = 1, \ldots, m+q, \quad j = 1, \ldots, n$. A *well-stated* MDMKP instance results in the absence of redundant constraints and a feasible solution without fixing the decision variable values to zero. The conditions are stated as: $\sum_{i=1}^{n} w_{ij} > b_i$ for $i = 1, \ldots, m+q$; $\max_j\{w_{ij}\} \leq b_i$ for $i = 1, \ldots, m$; and $\min_j\{w_{ij}\} \leq b_i$ for $i = m+1, \ldots, m+q$. Each of the m constraints in (10) are referred to as a *knapsack constraint* (K) and each of the q constraints in (11) are a *demand constraint* (D).

The MDMKP is faintly similar to a variety of KP variants. By removing the set of demand constraints and forcing the objective function coefficients to be positive, the multidimensional knapsack problem (MKP) becomes a special case of the MDMKP. An expansion of the MDMKP is the multidemand multiple-choice multidimensional knapsack problem (MDMMKP) defined by Lamine et al. (2012), where the decision variables become x_{ij}: the selection of exactly one item j out of each class of i items. The methodology presented by Lu and Vasko (2020) demonstrates the effectiveness of converting an MDMKP instance to an MDMMKP instance when using CPLEX to verify an optimal solution.

3.2.3 Subset of Instances

The theoretical component space of all possible MDMKP instances constitutes the problem space, while the empirical collection of test instances is the problem subset of interest. In an ideal statistical setting, the set of test instances should adequately

represent the problem space. The original collection of test instances, as set forth by Cappanera and Trubian (2005), begin by establishing the left-hand-side (LHS) constraint coefficients for the knapsack and demand constraints w_{ij} drawn uniformly from the range [0,1000]. The problem right-hand-side (RHS) coefficients b_i are determined by summing the LHS values across each dimension i and multiplying that sum by a "tightness ratio" α. Specifically, $b_i = \alpha \sum_{j=1}^{n} w_{ij}$ where $\alpha \in \{0.25, 0.5, 0.75\}$.

The remaining portion of an MDMKP formulation is the objective function coefficients. The instance generation method set forth by Cappanera and Trubian (2005) define two types of cost coefficients: positive cost coefficients or mixed cost coefficients. Specifically, the positive cost coefficients are defined as

$$c_j = \sum_{i=1}^{m} \frac{a_{ij}}{m} - \sum_{i=m+1}^{m+q} \frac{a_{ij}}{q} \quad \text{for} \quad j = 1, \ldots, n,$$

which are shifted by a Δ-value to enforce the positive case as

$$c_j = c_j + \Delta + 500u_j.$$

The shift value Δ is defined as

$$\Delta = \begin{cases} 1 + |\min_j\{c_j\}| & \text{if} \quad \min_j\{c_j\} < 0 \\ 0 & \text{otherwise,} \end{cases}$$

and the coefficient is scaled by u_j, a real number drawn uniformly from the range [0,1]. The mixed cost coefficients are defined as

$$c_j = \sum_{i=1}^{m} \frac{a_{ij}}{m} - \sum_{i=m+1}^{m+q} \frac{a_{ij}}{q} + 500u_j \quad \text{for} \quad j = 1, \ldots, n,$$

where u_j is a real number drawn uniformly from the range [0,1]. In addition, the set

of coefficients are scaled such that there are $nq/(m+q)$ negative coefficients to offset the difference in the number of knapsack and demand constraints.

The MDMKP instance generation method by Cappanera and Trubian (2005) mimics the MKP instance generation method by Chu and Beasley (1998). Cappanera and Trubian (2005) create six problems from existing MKP instances (Chu and Beasley, 1998) by modifying the number of demand constraints and the type of cost coefficient. The six problems are broken down from each instance as the positive case with $q = 1$, $q = m/2$, or $q = m$ or mixed case with $q = 1$, $q = m/2$, or $q = m$. Cappanera and Trubian (2005) use the first half of the instances in Chu and Beasley's test set to create six additional MDMKP instances, for a total of 810 instances available in the OR-Library (Beasley, 1990). These test instances are referred to as the *Beasley set of instances*.

Previous research conducted by Reilly (2009) summarizes the shortcomings of current instance generation methods for the KP by showing most generation methods only cover a small range of correlation coefficient values. Reilly (2009) defines two types of instance generation methods: *implicit correlation induction* (ICI), and *explicit correlation induction* (ECI). ICI methods rely on an implied correlation structure through the definitions of the model parameters, while ECI methods articulate the correlation structure before crafting the model parameters. MKP instances from Cho et al. (2008) utilize an ECI method based on the generation method defined by Iman and Conover (1982) to create test instances that show the ineffectiveness of various well known, competitive heuristics across the full range of correlation structures. The currently available MDMKP test instances do not cover an adequate range of correlation structures, something this work mitigates by expanding the number of test instances as described below.

Unlike the KP or the MKP, the MDMKP is not trivially feasible meaning gen-

erating feasible instances can be difficult. To address the issue of infeasible instance generation methods, the newly developed Primal Problem Instance Generator (PPIG) from the work by Scherer et al. (2023*b*) is used to generate MDMKP instances which are unique and feasible. PPIG is a flexible instance generation approach which constructs sampling distributions of RHS values from a fixed LHS weight matrix. In this work, the distributions are constructed from different initial solutions of MDMKP instances which follow a Bernoulli distribution with a uniformly drawn probability of an item being included in the solution p. The RHS values for the knapsack and demand constraints are selected from different percentiles of the sampling distribution from each constraint and constraint type. The sampling of RHS values from a distribution allows for variation in the tightness ratio of each constraint across all dimensions. In this work, we utilize the 50^{th} percentile for the knapsack constraints and the 25^{th} percentile for the demand constraints. The LHS follows the definition from Cappanera and Trubian (2005) and is drawn uniformly from the range [0,1000]. The 45 MDMKP instances generated are referred to as the *Scherer set of instances*.

PPIG is a flexible instance generation method that can be used to directly manipulate meta-features in the construction of MDMKP test instances. In this work, we use the ECI method by Hill et al. (2012) in conjunction with PPIG to generate MDMKP instances with explicit correlation structure mapped between the objective function coefficients and the LHS values for the knapsack and demand constraints. The parameters for PPIG remain the same for the RHS sampling, resulting in instances that cover the full range of correlation coefficients while also varying the tightness ratio across each dimension. These 135 MDMKP instances are referred to as the *Scherer Correlated set of instances*.

3.2.4 Algorithm Space

The algorithm space consists of any solution procedure used to provide solutions to the MDMKP. However, the current MDMKP literature is focused just on metaheuristic methods. Metaheuristics are heuristic "solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping local optima and performing a robust search of a solution space" (Glover and Kochenberger, 2006). The two classes of metaheuristics used to find solutions for the MDMKP in this work are tabu search methods and scatter search methods, resulting in four metaheuristics implemented in the testing.

The tabu search procedure used, referred to as the *Local Search* (LS) method, is from the methods set forth by Cappanera and Trubian (2005) and features two overarching phases: the outer phase and the inner phase. The outer phase performs a search for a diverse set of feasible solutions while the innder phase uses the collection of feasible instances and intensifies the search procedure to find improving solutions. The outer phase borrows the idea of *strategic oscillation* from Glover and Kochenberger (1996) in the MKP. Strategic oscillation focuses on finding the boundary between feasible and infeasible solution, and relaxing that boundary. The search begins with a constructive phase that continues until the feasibility boundary is crossed and the solution is stored. The variables changed are stored as tabu in the tabu memory. The crossing of the boundary is identified as a *critical event* and changes the phase of the search from the constructive phase to the destructive phase. The depth of the search into the feasible or infeasible region is controlled by the `span` parameter. The stopping criteria for the outer phase is as any of a specified length of time, a number of repetitions, or enough feasible solutions have been identified to improve upon in the second phase.

The inner phase of the tabu search is a local search procedure applied to the stored

collection of feasible solutions which examines two neighborhoods of solutions. The first neighborhood N_1 is made up of all solutions generated through three moves. The first move is changing a variable set to zero to one, the second move is changing a variable set to one to zero, and the third is to swap the values between a variable set to zero and a variable set to one. The second neighborhood N_2 is made up of solutions generated by a double swap move. The double swap move is taking two variables set to zero and two variables set to one and interchanging their values. In the first neighborhood search, a move is considered tabu if the variable is in tabu memory, while the second neighborhood search requires all four variables to non-tabu.

A simpler approach proposed by Arntzen et al. (2006) includes a tabu search method, referred to as the *Adaptive Memory Search* (AMS) method, for the MDMKP which focuses on an adaptive move evaluation to guide the search process. The search consists of a simple neighborhood of flips by changing a variable set to zero to one or a variable set to one to zero. The move is evaluated through the change in the objective function value and a weighted change in the infeasibility of the knapsack and/or demand constraints. The search is less sophisticated than the previous tabu search method (Cappanera and Trubian, 2005) but requires more parameter tuning to obtain high quality solutions.

The adaptive weights are updated depending on the current feasibility condition of the search. The conditions are divided into four disjoint regions: feasible solutions, knapsack feasible but demand infeasible solutions, knapsack infeasible but demand feasible solutions, and doubly infeasible solutions. The move is selected in a greedy fashion as long as it is not tabu or an aspiration criteria based on a new best feasible solution is met. The tabu memory includes a dynamic `tabu tenure` parameter randomly selected from a fixed, pre-defined, interval. The search incorporates diversification by using random solution restarts after a fixed set of processing time.

The *Scatter Search* (SS) method proposed by Hvattum and Løkketangen (2007) is the only metaheuristic examined not based on tabu search. In contrast to tabu search, which is a single-solution-based metaheuristic, scatter search is population-based and examines multiple solutions using an evolutionary process to recombine solutions from a reference set to build other solutions (Laguna and Martí, 2003). The scatter search implementation set forth by Hvattum and Løkketangen (2007) uses the five common SS components of: the diversification generation method, the improvement method, the reference set update method, the subset generation method, and the solution combination method. In the improvement method, the authors utilize the objective function value and the sum of the violations of each constraint to measure the quality of solutions. The solution are improved upon to create a single flip neighborhood by changing a variable set to zero to one or a variable set to one to zero. In the subset generation method, the authors produce five trial solutions for each pair of solutions used for input in the improvement method. The authors modify the neighborhood structure by allowing more flips to occur in the search process.

A more recent solution method, referred to as the *Two-Stage Tabu Search* (TSTS) method (Lai et al., 2019), utilizes two stages of tabu search. The two stages use methods from previous attempts in the literature in addition to a unique hash vector for tabu memory of solutions. The first stage is similar to the outer phase method defined by Cappanera and Trubian (2005), where the focus is on finding feasible solutions through exploration of two simple neighborhood structures. The first neighborhood consists of all non-tabu solutions available by changing a variable set to zero to one or a variable set to one to zero. The second neighborhood consists of all non-tabu solutions available by swapping a variable set to zero with a variable set to one. These neighborhoods of solutions are evaluated through a penalized evaluation function similar to the improvement method introduced by Hvattum and Løkketangen

(2007). The search space explored by the first stage contains both feasible and infeasible solutions but both indicate promising regions to improve upon in the second stage.

The second stage aims to improve upon the collection of feasible and infeasible solutions from the first stage. The main difference between the first stage is the neighborhood structure, which focuses con a constrained swap neighborhood. Specifically, the neighborhood consists of all non-tabu solutions available by swapping a variable set to zero with a variable set to one such that their objective function value is better than the current best objective function value. The best objective function value is updated when a higher value is obtained from a feasible MDMKP solution. The search continues until a maximum time limit is reached.

This work adopts the parameter settings recommended in the relevant publications for all metaheuristics employed. For instance, the weights of the penalization term for change in infeasibility of the knapsack and/or demand constraints require careful tuning, and we use the recommended `tabu tenure` settings for each instance. In the scatter search method proposed by Hvattum and Løkketangen (2007), different improvement techniques are used to expand the search space, but we adopt the simplest criterion of steepest ascent local search using single-flip moves only. Another crucial aspect that affects the quality of the solutions obtained is the termination criteria for each metaheuristic. Following the methods set forth by Lai et al. (2019), we set the maximum runtime for each metaheuristic proportional to the number of decision variables in the test instance, and the recommended number of iterations from each publication is also used. The focus in this work is to gain insight into solution procedure performance thus the lack of focus on the optimal tuning of metaheuristic parameters to the test problem set.

3.2.5 Performance Space

The performance space contains measures to identify solution procedure effectiveness on a given test instance. In heuristic methods common metrics include the quality of the solutions found and the best feasible objective function value found. The difficulty of the MDMKP is clearly demonstrated when seeking an optimal solution using exact solution procedures such as CPLEX (Song et al., 2022) and Gurobi. In some test instances, a feasible solution is not identified within several hours of processing time in CPLEX or Gurobi. We utilize an Intel© Core™ i7-11700 CPU at 2.5GHz with 32 GB RAM for each of our tests.

The best objective function value of a feasible solution obtained by a metaheuristic in saved as the performance metric. Due to the inability to find feasible solutions with state-of-the-art commercial solvers, we remove instances from our problem subset that were not deemed feasible by Gurobi within 1 hour of processing time. By introducing this criteria, the metaheuristics which do not identify a feasible solution in empirical testing are fairly compared using instances with known feasible solutions. This criteria removes 4 test instances from the Beasley set while none of test instances from the Scherer and Scherer Correlated sets were infeasible. In total, 986 test instances are retained for empirical testing.

3.2.6 Feature Space

Meta-features are quantitative measures that capture different aspects of an optimization problem beyond its objective function parameters and constraint parameters. Meta-features are a set of characteristics that can be extracted from an optimization problem to provide useful information for algorithm selection, performance prediction, and understanding problem properties. This work presents a comprehensive study of various meta-features and how they can be used to characterize an opti-

mization problem. The examined meta-features are used to build a two-dimensional visualization of the instance space to see where the current collection of test instances lie in terms of their problem structure. Their physical location in the instance space is then mapped to the meta-feature values it contains.

Previous literature on features to characterize optimization problems includes the work by Smith-Miles and Lopes (2012). The authors distinguish two types of problem features: problem independent features and problem dependent features. Problem independent features are measures of the objective function parameters and constraint parameters to summarize their distribution of values or by borrowing methods from landscape analysis (Reeves, 1999). For example, the problem dimensionality can be determined by analyzing the number of input variables or dimensions of the problem. Problem dependent features allow the use of specific measures pertaining to a problem, such as the various measures seen in the literature for KP variants. Smith-Miles et al. (2021) include an in depth description of the feature space for the 0-1 KP in addition to the work by Hall and Posner (2007). Their meta-features are measures exploited in heuristic methods for the 0-1 KP such as the efficiency of an item defined as: $e_i = p_i/w_i$. Work by Cho et al. (2008) on the MKP focused on examining the correlation structure between the objective function parameters and constraint parameters.

Previous studies on the 0-1 KP discovered that the range of correlation coefficients explored with some test sets was limited to a narrow range of 0.97 to 1 (Reilly, 2009), even though the description of those test problems implied a much more robust range. The impact of the narrow range of problems emanating from ICI generation methods went largely unnoticed until compared to an ECI method. Figures 10 and 11 display the correlations between the objective function and constraint parameters for a subset of instances selected from the ICI method used to generate the Beasley set

of instances, contrasted with the ECI method used to produce the Scherer Correlated set of instances. Notably, the correlation values for both the knapsack and demand constraints exhibit significant differences in range, except when $q = 1$, leading to instances with more prominent correlation values than when $q = 30$. This is an important problem feature that has not been examined in past MDMKP work, but has been a focus in past MKP work (Hill et al., 2012) and past characterization of MKP-type problems (Hill et al., 2011).

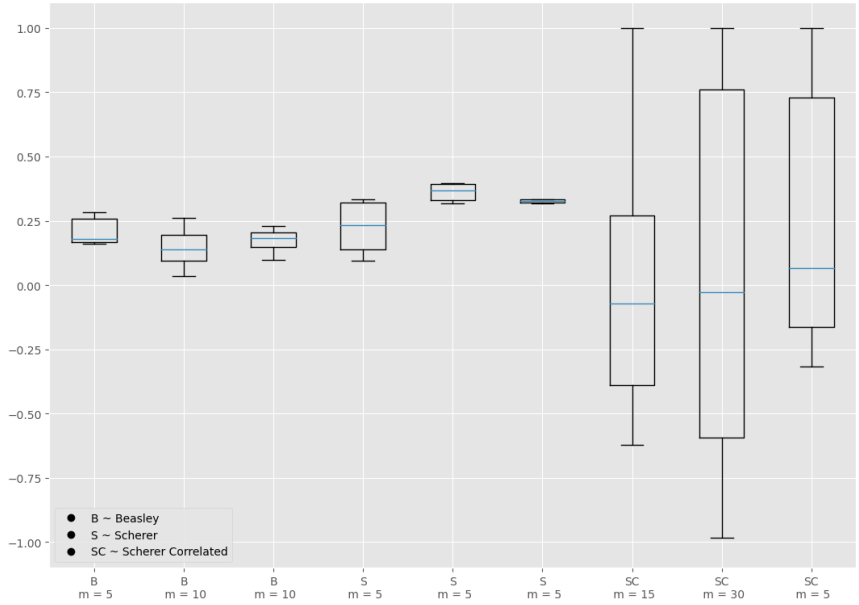

Figure 10. Distribution of correlation coefficients for knapsack constraints

This work adopts meta-features previously used in the literature for the 0-1 KP and the MKP. The meta-features are augmented for the MDMKP by using separate measures for the knapsack and demand constraints in addition to using summary statistics to capture the multiple i constraint dimensions. The meta-features are described in Table 4 for a total of 29 meta-features to choose from to construct the instance space. ISA uses Preparation for Learning of Instance Meta-data (PRELIM) to pass the meta-features through a Box-Cox variance stabilization transformation followed by standardizing the meta-features to a standard normal distribution with a

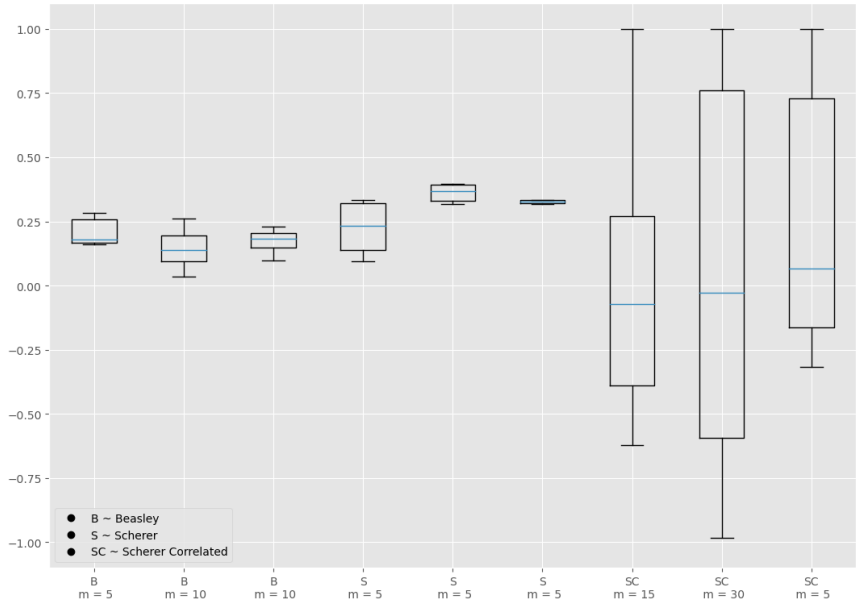

Figure 11. Distribution of correlation coefficients for demand constraints

z-transformation. From this collection of transformed meta-features, the Selection of Instance Features to Explain Difficulty (SIFTED) is required to discover the relevant meta-features and reduce the quantity of meta-features to characterize the instance space.

3.2.7 Instance Space

SIFTED reduces the number of meta-features from 29 to 10. The 10 meta-features portray the instance space in a ten-dimensional space. To visualize the instance space, Projecting Instances with Linearly Observable Trends (PILOT) occurs to reduce the dimensionality of the instance space to two-dimensions. The projection in PILOT is based on a convex global optimization problem with an objective to preserve the topology of the instance space (Muñoz et al., 2018). PILOT measures topological preservation by the correlation between the distances in the ten-dimensional feature space and the distances in the two-dimensional instance space. The resulting projection to the two-dimensional instance space (Z_1, Z_2) matrix is shown in (13).

Table 4. Meta-features for the MDMKP

Feature Name	Description - Constraint Type - Statistic
# Decision Vars	Number of decision variables.
# Constraints	Number of constraints - K & D.
Constraint Tightness	$\frac{b_i}{\sum_j w_{ij}}, \forall i$ - Both - Min, max, and mean.
Prop Part Dominance	Proportion of items pairs i, j where $p_i \geq p_j$ and $w_{ik} \leq w_{jk}$ for any k knapsack constraint, ($p_i \geq p_j$ and $w_{ik} \geq w_{jk}$ for any k demand constraint)
Obj to Constraint Correlation	Pearson correlation coefficient between objective function coefs and constraint coefs for dimension k - K & D - Min, max, and range.
Within Constraint Correlation	Pearson correlation coefficient within same class of constraint coefficients for dimension k - K & D - Min, max, and range.
Across Constraint Correlation	Pearson correlation coefficient across classes of constraint coefficients for dimension k - K & D - Min, max, and range.
Coefficient of Var Weights	Coefficient of variation of weight coefficients across k dimensions - K & D.
Coefficient of Var Profits	Coefficient of variation of profit coefficients.

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} -0.2742 & 0.0232 \\ -0.1321 & 0.2524 \\ -0.2695 & -0.3597 \\ 0.4183 & -0.2882 \\ 0.0334 & 0.509 \\ 0.2937 & -0.2537 \\ 0.1802 & -0.5974 \\ -0.115 & -0.359 \\ 0.1624 & 0.2651 \\ 0.221 & -0.379 \end{bmatrix}^T \begin{bmatrix} \text{\# Decision Vars} \\ \text{Average Constraint Tightness - D} \\ \text{Prop Part Dominance - K} \\ \text{Max Obj to Constraint Corr - K} \\ \text{Min Obj to Constraint Corr - K} \\ \text{Range Obj to Constraint Corr - D} \\ \text{Range Within Constraint Corr - K} \\ \text{Max Within Constraint Corr - D} \\ \text{Min Within Constraint Corr - D} \\ \text{Range Across Constraint Corr} \end{bmatrix} \tag{13}$$

The collection of problem instances and the projection from (13) yields the instance space depicted in Figure 12. Analysis of the distribution of test instance sources indicates that the Beasley and Scherer sets of instances share a similar region, while the 135 Scherer Correlated set of instances are mostly clustered towards the lower-right region of the instance space. The collective result is better coverage of the instance space and thus a more comprehensive empirical study, which as discussed below, yields remarkable insights.

Figure 13 displays the distribution of meta-features across the instance space. The cluster observed in the lower-right region of the instance space is mainly attributed to the correlation structure among the instances. Notably, the other meta-features, such as the number of demand constraints and decision variables, follow a left-to-right trend of increasing magnitude, whereas the proportion of partial dominant items for demand constraints exhibits higher values on the left and lower values on the right. This visualization provides valuable insights into the underlying relationship between

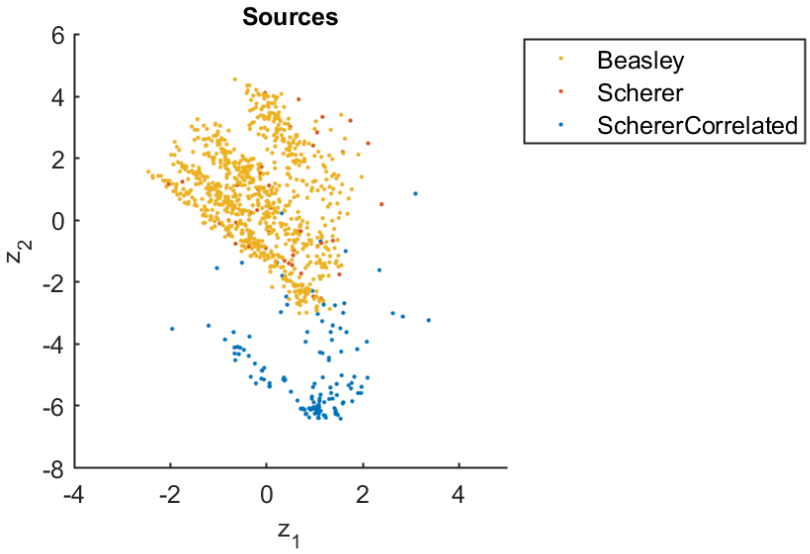

Figure 12. Instance space for the MDMKP with projection defined by (13)

problem structure and solution procedure performance.

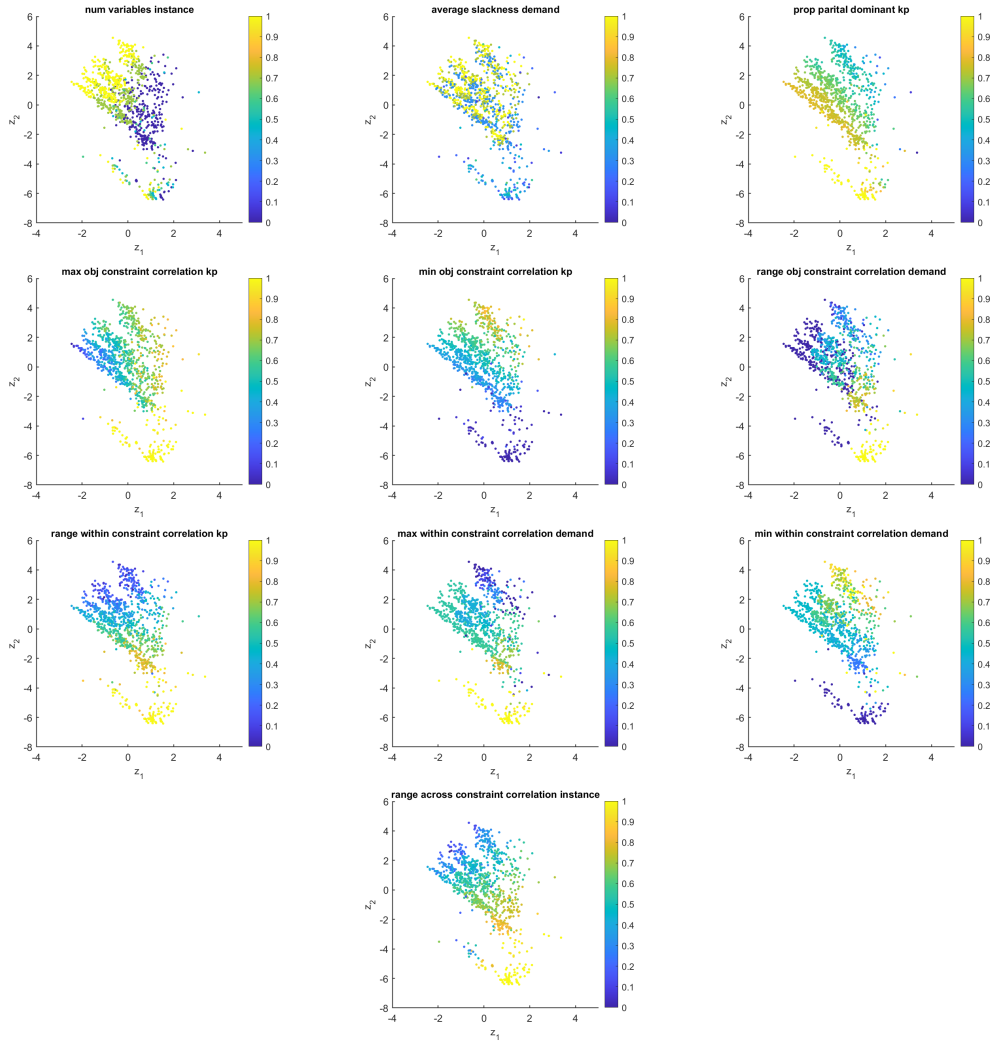

Figure 13. Distribution of meta-features across instance space

3.3 Metaheuristic Selection Problem

The ASP, first introduced by Rice (1976) and extended by Smith-Miles et al. (2014), establishes a mapping between the feature space and the algorithm performance space. This goal involves predicting the best-performing algorithm for a given test instance, which in the current setting, is focused on metaheuristics. The full suite of available test instances, including those generated with an explicit correlation structure, are used to examine which metaheuristics to use in specific areas of the instance space. The defined meta-features are used as inputs to a machine learning classification model. The model then assigns labels to instances with the recommended metaheuristic, and the results are visualized in the two-dimensional instance space.

Figure 14 displays the predicted best metaheuristic in the instance space using the support vector machine (SVM) classifier from the ISA toolbox (Muñoz and Smith-Miles, 2020*b*). The lower-right region of the instance space, consisting mostly of instances from the Scherer Correlated set, indicates poor performance among all examined metaheuristics. Past work has found correlation structure in optimization problems significantly affects algorithm performance, and this work demonstrates the same holds for the MDMKP. Conversely, the upper-left region, mainly composed of instances from the Beasley set, indicates best performance with the Local Search method. Additionally, a small subset of instances, sourced from all three sets, is predicted to achieve the best performance with the Two-Stage Tabu Search method.

Figures 15-18 provide specific graphical insights into the performance of the suite of metaheuristics considered. A careful examination of the algorithm footprints in Figures 15-18 confirms that the lower-right region of the instance space lacks any exceptionally performing solution procedure. Specifically, the Scherer Correlated set of instances do not exhibit a prominent method in the metaheuristic portfolio for

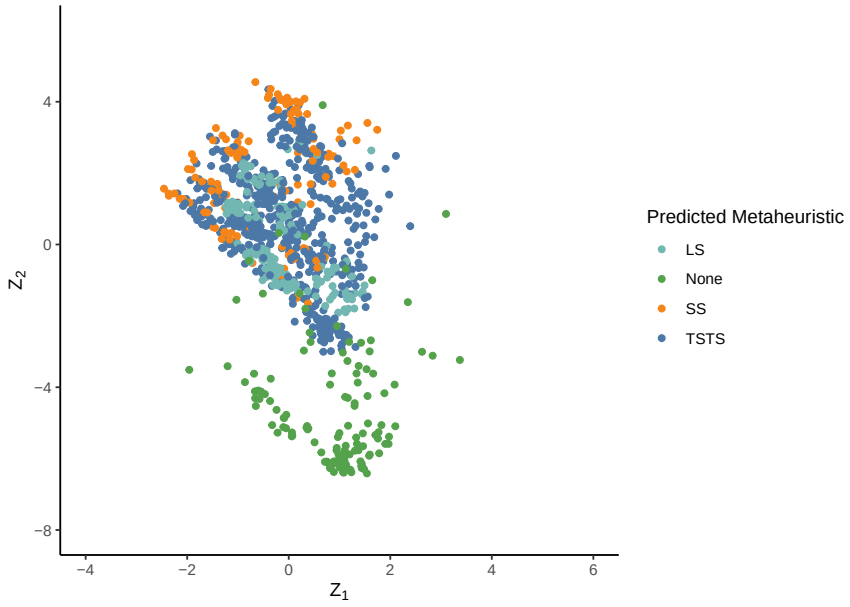

Figure 14. SVM predicted best metaheuristic for the MDMKP

finding high-quality solutions for the MDMKP. Since correlation structure within an optimization problem is a reasonable expectation, this finding underscores the significance of using an adequate source of test instances. The better test problem set used in this work allows further enhancement of the ASP with a classification model.

There are additional insights gained by analyzing the distribution of meta-features in Figure 13 and the footprints in Figures 15-18. Clearly, problem structure matters. There is a lack of effective metaheuristics for problems involving the ECI method, as evidenced by the absence of corresponding footprints in the lower-right region of the instance space. This is the region dominated by meta-features associated with correlation structure, spanning the entire range of correlation coefficients.

The footprints of the LS and TSTS method cover the largest area of the instance space, indicating their robustness to a variety of instance configurations outside of the correlation structure. Conversely, the footprints of the AMS and SS method, which are the weakest performing metaheuristics, cover only a small portion of the

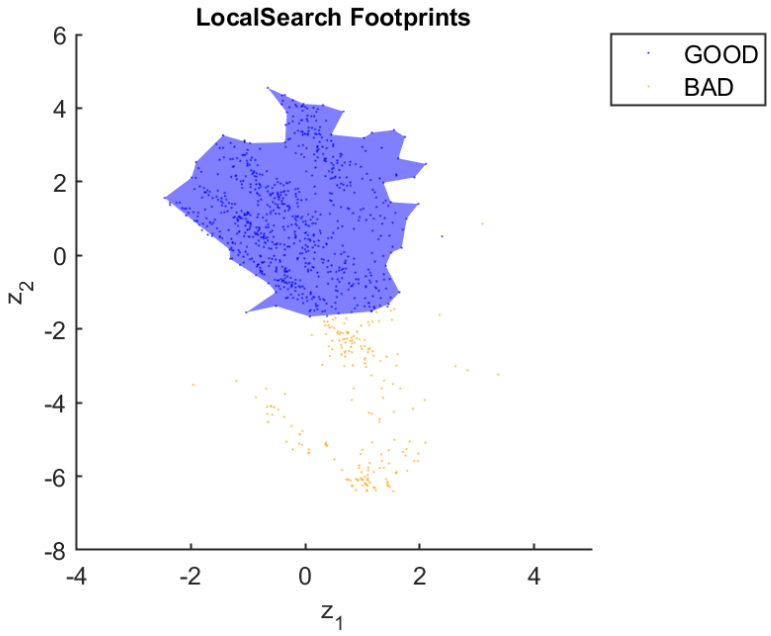

Figure 15. Local Search's footprints across the instance space

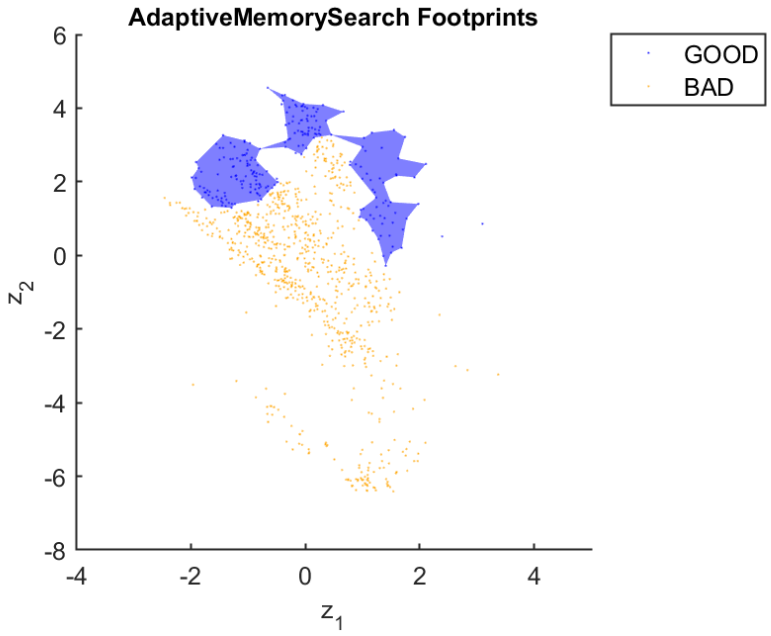

Figure 16. Adaptive Memory Search's footprints across the instance space

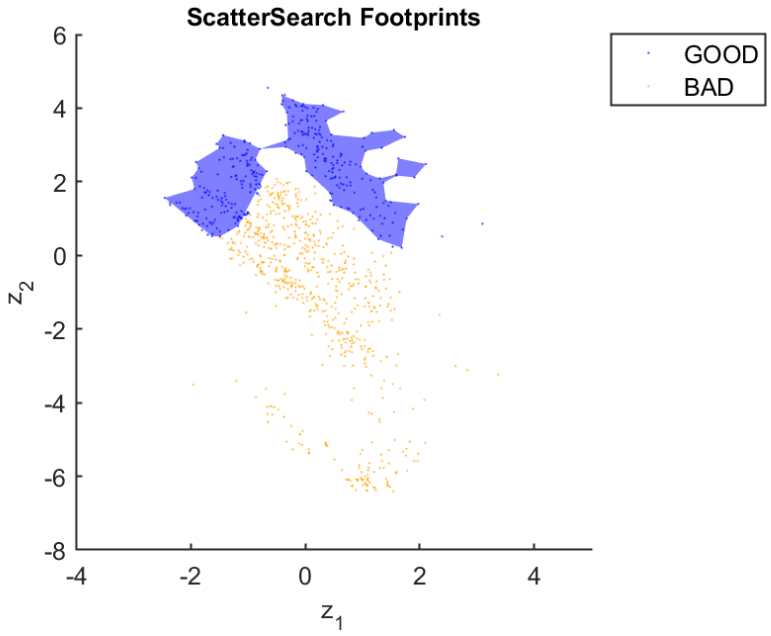

Figure 17. Scatter Search's footprints across the instance space

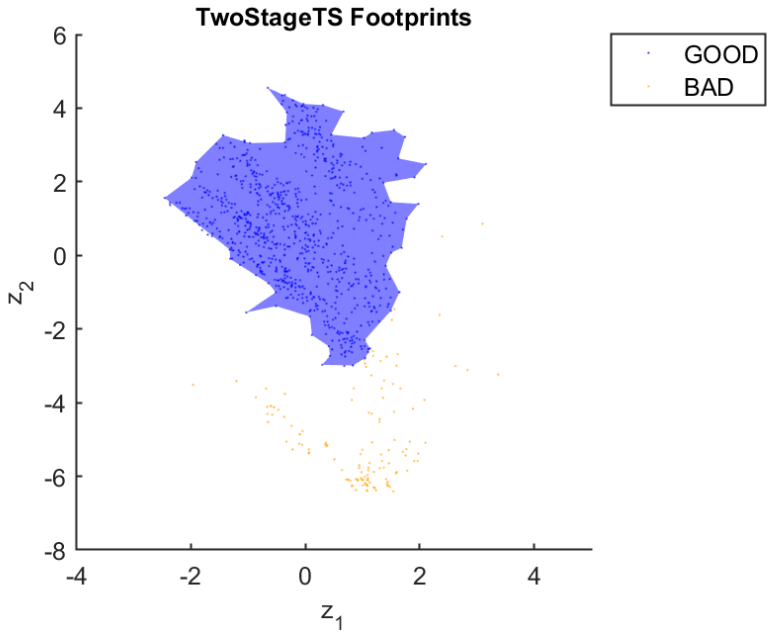

Figure 18. Two-Stage Tabu Search's footprints across the instance space

instance space. However, the SS method performs relatively well in instances with a higher proportion of partially dominated items for the demand constraints. Overall, the combination of insights from the distribution of meta-features and the footprints provides valuable information for understanding the performance of different meta-heuristics in the context of the MDMKP.

One limitation of using the SVM classifier is the lack of inferential methods (James et al., 2013). To address this limitation, previous research has focused on elucidating the decision-making process of machine learning models (Rudin, 2019). This study overcomes this challenge by utilizing a decision tree model due to its flexibility and interpretability in the decision-making process. The decision tree model is constructed in an optimal manner, rather than the typical greedy approach, to ensure a more robust and reliable interpretation of the model's decisions. The decision tree model is also forced to be sparse, therefore creating a smaller tree which prevents overfitting and makes the model easier to visualize.

Decision trees are popular machine learning models because of their simplicity and effectiveness. While their structure is simple, constructing a decision tree to represent data is NP-hard (Laurent and Rivest, 1976). Originally decision trees are built using greedy heuristic methods which iteratively proceed in a top-down manner from the tree with local objective functions (Breiman et al., 1984). With recent improvements in computational power, optimal decision trees have become within practical reach with better generalization on unseen data than heuristic methods (Bertsimas and Dunn, 2017).

The Generalized and Scalable Optimal Sparse Decision Trees (GOSDT) as proposed in the original work by Hu et al. (2019), further refined in the subsequent study by Lin et al. (2020), and computationally accelerated by McTavish et al. (2022) is used in this work. GOSDT employs a specialized branch and bound method to effi-

ciently find the optimal sparse decision tree, mitigating the combinatorial explosion of possible trees. This is achieved through analytical bounds and intermediary storage of prior paths explored, eliminating the need for recomputation of bounds (Hu et al., 2019). The loss function is based on misclassification error. To enforce sparsity, the loss function is regularized by the number of leaves in the tree scaled by a regularization term λ. The task is multiclass classification, where the decision tree predicts the best metaheuristic based on the 10 meta-features used in the instance space.

A training data set consisting of the majority of instances (80%), with a smaller subset (20%) reserved for testing, is used. Additionally, 25% of each training set was further partitioned for validation purposes. Comparing these results to the tuned SVM classifier demonstrates the effectiveness of optimal decision trees by examining a greedy decision tree. The performance and generalization capability of all three methods were compared based on their accuracy on the testing split of instances in the meta-data.

3.4 Results

Figure 19 displays the decision tree resulting from GOSDT, using the best hyperparameter settings found: $\lambda = 0.01$ and a depth budget of 5 nodes. Although setting $\lambda = 0.001$ results in a slight improvement in training performance, the corresponding optimal decision tree is not sparse, with 14 leaves. The decision tree shown in Figure 19 has only 8 leaves. The lack of sparsity present in larger decision trees hinders the interpretability of the machine learning model, therefore the sparse decision tree is preferred.

The optimal sparse decision tree model for the ASP is used to evaluate the subset of test instances and their corresponding recommended metaheuristic. For each test instance, only 6 meta-feature values are required for the model to make a reliable

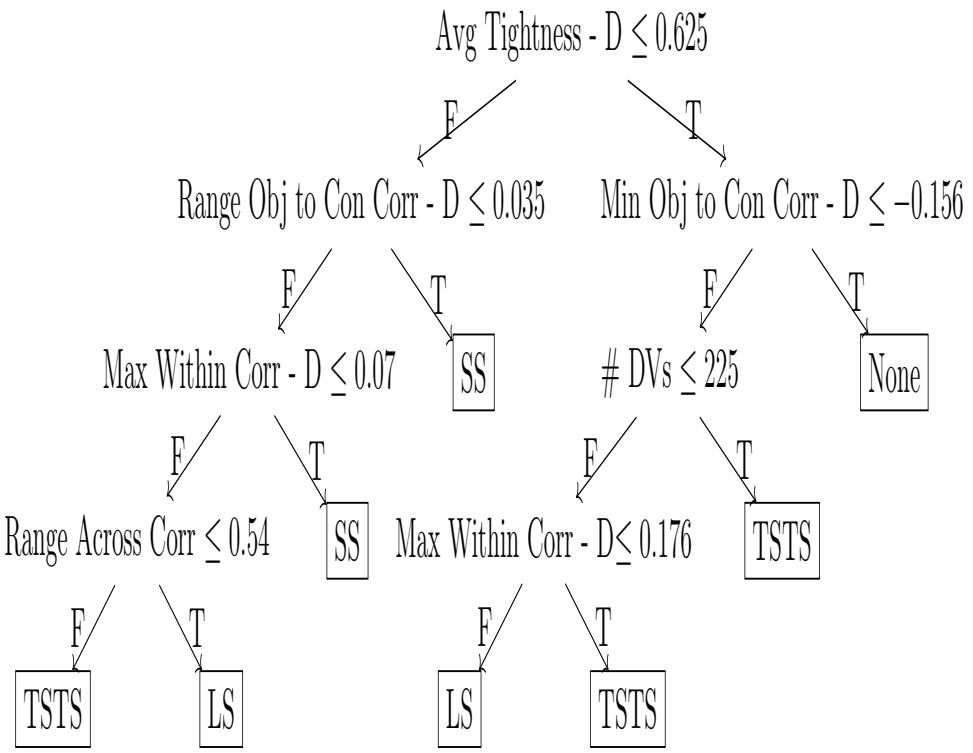

Figure 19. Optimal sparse decision tree for MDMKP ASP

prediction. The first meta-feature examines is the average tightness of the demand constraints. If the average tightness is less than 0.625 and the minimum objective to constraint correlation of the demand constraints is less than 0.156, then the decision tree predicts that none of the examined metaheuristics will perform well on the instance. An effective ECI instance generation method should produce a distribution of correlation coefficients across the entire possible range, indicating that no examined metaheuristic is recommended. Of the 986 MDMKP instances examined, 196 are observed to not have an effective metaheuristic.

Examining the optimal decision tree model through the instance space in Figure 20 shows a similar story to Figure 14. The predictions for both models are similar, such as the predictions for the bottom-right collection of instances to not perform well for any examined metaheuristic. The SVM classifier does predict SS more than GOSDT, while GOSDT favors the LS.

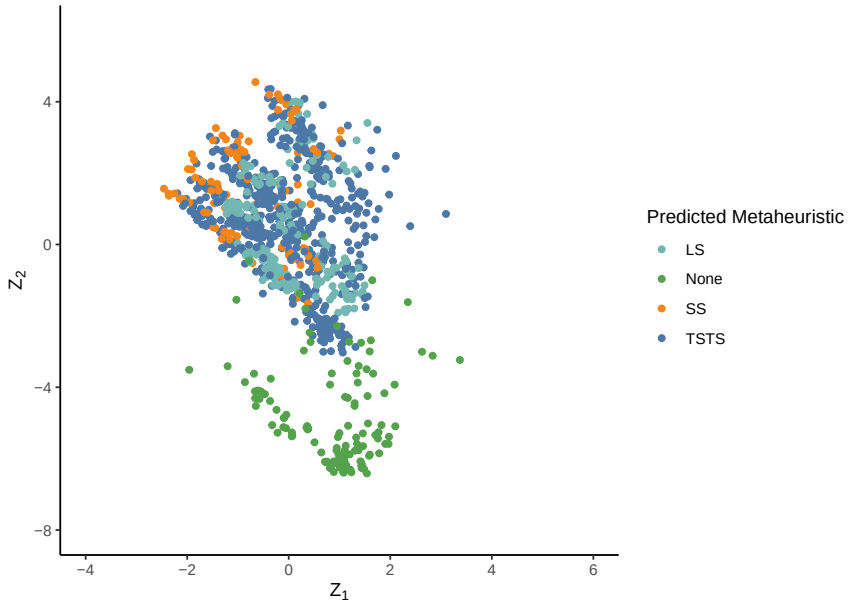

Figure 20. GOSDT predicted best metaheuristic for the MDMKP

The performance of two classification models was evaluated on testing data. The GOSDT classifier demonstrated an accuracy of 85.35%, while the SVM classifier had an accuracy of 87.37%. There is no evidence of outfitting occurring between either model, but the sparsity component of GOSDT is useful for generalizing the classifier to unseen test instances (Lin et al., 2020). If instead a greedy decision tree model is employed without pruning, the risk of overfitting to the training data increases, leading to suboptimal performance when analyzing unseen data (Bertsimas and Dunn, 2017).

The results involving the entire subset of MDMKP instances are summarized in the confusion matrices presented in Figures 21 and 22. Both models make similar misclassifications, with the primary errors arising from predicting no metaheuristics as the best option when LS is observed as the best choice, and predicting TSTS when no good metaheuristics are identified. The dominant observed and correctly predicted best metaheuristic is TSTS.

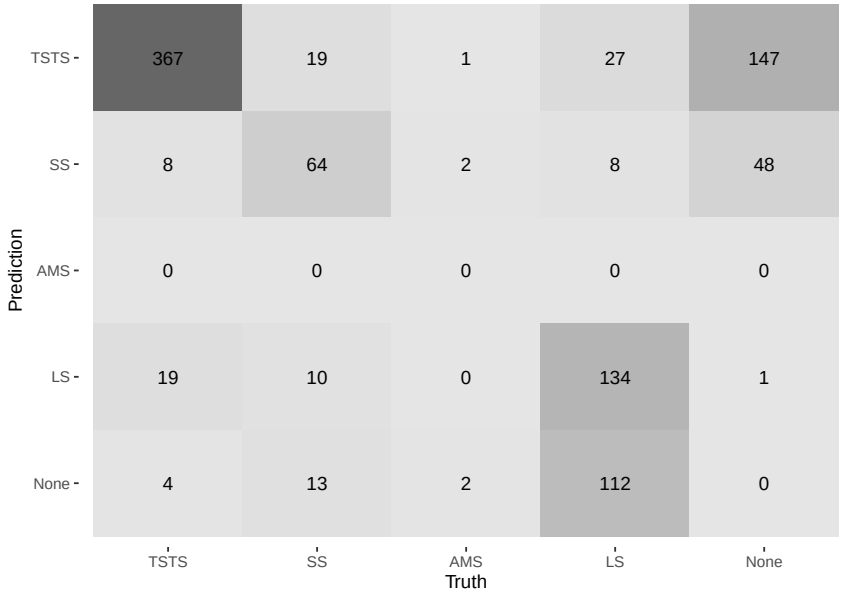

Figure 21. Confusion matrix for GOSDT

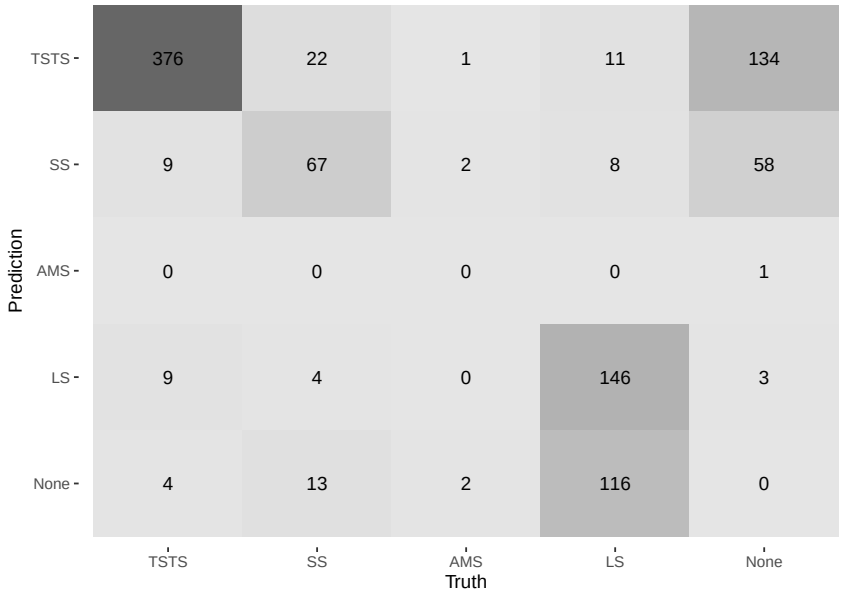

Figure 22. Confusion matrix for SVM

3.5 Conclusion

This work has examined the ASP for the MDMKP under the lens of ISA with a new set of test instances. Using ISA, the instance space is shown and the performance of selected metaheuristics are examined in the instance space. To address the ASP an alternative classification model is proposed. By combining the interpretability and simplicity of GOSDT with ISA we have demonstrated another method to add insights into the conditions under which a given metaheuristic is expected to perform. The comparisons between the classification model included in the ISA methodology and the proposed alternative GOSDT reveal no loss in predictive capabilities, but with a more interpretable model.

The proposed method of integrating a optimal classification tree into ISA provides an alternative machine learning classifier within the ISA methodology as opposed to a replacement. A decision tree model removes ambiguity between the mapping of problem structure to metaheuristic selection in the ASP. Previously, the approach within ISA is to examine the distribution of meta-features across the instance space, as shown in Figure 13, then combine the insights gained from the algorithm footprints across the instance space to discover the relationship between meta-features and algorithm performance. By implementing a sparse optimal classification tree, the most influential meta-features and their inflection points for the ASP are revealed.

There are two main limitations to this work. First, the focus was on metaheuristic solution procedures thus excluding work established by Hvattum et al. (2010) since it involves an exact approach through CPLEX. Second, there was no attempt to optimally tune the metaheuristics to the problem set. Admittedly, changing metaheuristic parameter settings will affect performance, but the focus of this work was using a improved test problem space within an ISA framework to demonstrate new insights into algorithm performance. Future work will expand the subset of the algo-

rithm space examined, examining optimally tuned algorithms, and further problem generation methods to further cover the problem instance space.

IV. An Optimization Framework for Filling the Instance Space of Multidemand Multidimensional Knapsack Problems

4.1 Introduction

When evaluating solution procedures in optimization, a common approach is to test the portfolio of solution procedures empirically through a variety a test instances. Using test instances that are randomly generated, selected from a benchmark library, or drawn from real-world scenarios, the performance of a solution procedure is effectively examined and compared to existing methods. An issue with empirical testing is the potential bias of the underlying structure of test instances. Across optimization problem types, varying the number of decision variables or constraints in a problem does not guarantee problem diversity in empirical tests.

Empirical testing of solution procedures as advocated by Hooker (1994) is a viable alternative to theoretical worse-case and average-case results when determining whether an approach is successful in practice. By utilizing a rigorous experimental design and analysis in empirical testing, theories may be derived by mapping test instance configuration to solution procedure performance. This idea is expanded upon by Hooker (1995) by arguing for competitive testing to highlight "why" a solution procedure is better and the need to clarify strengths and weaknesses of a solution procedure.

The No-Free-Lunch (NFL) Theorem by Wolpert and Macready (1997), which states it is impossible for any one algorithm to outperform all other algorithms when averaged over all possible optimization problems, is the motivation of this work. The root of NFL lies in the test instances provided for empirical testing. Test instances should be diverse, unbiased, challenging, practical, and discriminate algorithm performance in coordination with NFL. Hooker (1995) argued that randomly generated test

instances lack in problem diversity and common benchmark libraries of instances include intrinsic biases which do not stress test algorithms to a sufficient degree. These issues prevalent within the empirical testing of algorithms results in the over-tuning of algorithms to a small subset of instances while ignoring the potential trade-offs when the test instance subset is expanded as stated in NFL.

Culminating from the ideas expressed by Hooker (1995) and inspired by NFL, Smith-Miles (2019) developed Instance Space Analysis (ISA) to improve the understanding between solution procedure performance and the structure of test instances. ISA provides a systematic method to describe a problem of interest by compartmentalizing all aspects of the problem into component spaces. These spaces encompass the test instances employed, the features utilized to characterize these test instances, the solution procedures employed for evaluation, and the performance measures used to assess the efficacy of the solution procedures. ISA extends the framework outlined in the Algorithm Selection Problem (ASP) of the seminal work by Rice (1976), while placing emphasis on the associations between solution procedure performance and instance configuration.

Current research in the empirical evaluation of solution procedures in optimization assumes the distribution of test instances is sufficient to generalize to all instances of a problem. This assumption is implicitly declared by drawing on historical instances found in benchmark libraries, real-world scenarios, or randomly generated while ignoring any validation of the structure of the collection of test instances. Therefore, it is important to utilize the tools provided in ISA in coordination with the challenge presented by Hooker (1995) to discover insights of the strengths and weaknesses of solution procedures. This work focuses on the instance generation procedure typically ignored in the research process and presents a method for targeting specific configurations of test instances to enlarge the scope of existing collections of test

instances.

In this work, the problem of interest in the Multidemand Multidimensional Knapsack Problem (MDMKP). The MDMKP is concerned with selecting an optimal subset from n available items with integer profits/costs $\{c_1, c_2, \ldots, c_n\}$ such that their sum total is maximized and integer weights w_{ij} across dimensions $m + q$. The knapsack is constrained m dimensions to a set capacity such that total weight does not exceed capacity across each dimension while also requiring q dimensions of demands to be met such that total weight meets a threshold across each dimension. The MDMKP is unique from other knapsack variants in that the objective function coefficients are unconstrained in sign. The MDMKP is formulated as:

$$\max \quad \sum_{j=1}^{n} c_j x_j \tag{14}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_{ij} x_j \leq b_i, \qquad \text{for} \quad i = 1, \ldots, m, \tag{15}$$

$$\sum_{j=1}^{n} w_{ij} x_j \geq b_i, \qquad \text{for} \quad i = m+1, \ldots, m+q, \tag{16}$$

$$x_j \in \{0, 1\}, \qquad \text{for} \quad j = 1, \ldots, n, \tag{17}$$

where x_j is a binary decision variable to determine whether to include item j in the knapsack. Additionally, the right-hand-side (RHS) capacity/requirement values are positive, i.e., $b_i > 0$ for $i = 1, \ldots, m + q$, and the weights are assumed to be nonnegative, i.e., $w_{ij} \geq 0$ for $i = 1, \ldots, m + q, \quad j = 1, \ldots, n$. Each of the m constraints in (15) is called a *knapsack constraint*, while each of the q constraints of family (16) is referred to as a *demand constraint*.

4.1.1 Literature Review

Test instance generation is usually not a focus of most research in optimization. Empirical evaluations of solution procedures are tested on benchmark test instances or a real-world scenario of interest. If test instances are generated, they are usually a short portion of the methodology while the majority of the section is directed towards the solution procedure. However, the idea of examining the underlying distribution of test instances is not new and has been explored across multiple problem classes in optimization.

In the 0-1 Knapsack Problem (KP), the MDMKP with $m = 1$, the instance generation methods set forth by Balas and Zemel (1980) defined an original classes of randomly generated instances of the KP as uncorrelated, weakly correlated, strongly correlated, and subset sum. Additional classes are defined by Martello et al. (1999) including inverse strongly correlated, almost strongly correlated, similar uncorrelated weights, even-odd subset-sum, and even-odd strongly correlated. The correlated classes are based upon linear combinations of uniformly distributed random variables to induce the correlation structure such as the weakly correlated defined by setting the weights to a discrete uniform distribution with range [0,R] and profits as a function of their weights minus one tenth of their upper bound.

Building upon these classes, the work by Pisinger (2005) explored the original classes of the KP and asked the question "where are the hard knapsack problems?". The problem classes lacked a full exploration of potentially difficult test instances and visualized the minor differences between multiple problem classes such as the almost strongly correlated and strongly correlated instances. From this initial exploration, Pisinger (2005) defined six more classes of test instances which were difficult for dynamic programming algorithms to find optimal solutions for and posed unique structure from the previous classes.

Smith-Miles et al. (2021) revisits the question "where are the hard knapsack problems?" through the lens of ISA. Additionally, they propose four new classes of KP instances including quadratic fit, cubic fit, random ceiling, and profit ceiling large only. These four classes, in addition to the previous 16, constitute known set methods to generate KP instances with their generation procedures outlined in Table 1 of Smith-Miles et al. (2021). Other KP instances are generated through an genetic algorithm (GA) which attempts to fill gaps in the instance space based on the procedure outlined by Smith-Miles and Bowly (2015).

Instance generation methods have also been explored for other knapsack problem variants. Reilly (2009) finally examined the correlation induced by instance generation methods for the KP, generalized assignment problem, multidimensional knapsack problem (MKP), and set-covering problem. The majority of test instances examined lacked a full range of correlation coefficients and instead are insufficiently qualified as "weakly correlated" or "almost strongly correlated". The poorly defined classes of test instances are based on an implied correlation structure. Instance generation methods with an implied correlation structure are referred to as an implicit correlation induction (ICI) method. The implications of an ICI method are a lacking of quantifiable measures of correlation and the lack of systemic control over the correlation in a computational study.

In contrast to an ICI method, Reilly (2009) defines explicit correlation induction (ECI) methods and their advantages over ICI methods. ECI methods instead target a predefined correlation structure and provide the following benefits: (1) ECI methods allow the user to specify the correlation structures in experiments, (2) the correlation structure is controllable for systemic changes needed in a computational study, (3) the distributions of coefficient values are no longer confounded with correlation structure.

ECI methods for the MKP include Cho et al. (2008), where the full range of cor-

relation coefficients are explored across each dimension. When compared to existing heuristics for the MKP by Hill et al. (2012) the explicit correlation structure with increasing problem size degrades performance of existing heuristics. These types of results would not have been possible if only the test library of instances were used. Hill et al. (2012) demonstrates several heuristics which would appear insignificant if the predominant set of test instances were used for the empirical study. Similar findings by Hiremath and Hill (2013) occur in the multiple-choice multidimensional knapsack problem (MCMKP).

Beyond the KP and its variants, an instance generation scheme for job shop scheduling problems with desirable properties and ECI is presented by Hall and Posner (2001). The properties of correctness, applicability, and reproducibility are expanded upon in the work by Hall and Posner (2010). Other examples of instance generation schemes which contain desirable properties include the works by Romeijn and Morales (2001) in supply chains, Laguna and Martí (2001) in sparse graphs, and Arthur and Frendewey (1988) in the traveling salesman problem (TSP). While these examples include desirable properties in their instance generation method, few have been examined in an ISA framework such as the job shop scheduling problem (Smith-Miles et al., 2009), the graph coloring problem (Smith-Miles et al., 2014), the max-cut problem (McAndrew, 2020), the curriculum-based course timetabling (CB-CTT) problem (De Coster et al., 2022), and the TSP (Smith-Miles and Lopes, 2012).

The methods described by Lopes and Smith-Miles (2010) explore the discrepancies between generated instances for the CB-CTT by Burke et al. (2008) and real-world instances from the University of Udine. The instance generator is redefined in subsequent work by Lopes and Smith-Miles (2013) to address these issues by focusing on minimizing the differences between the generated synthetic instances and the real-world instances while providing discriminating performance for two distinct solution

methods. A recent work by De Coster et al. (2022) views the generated instances in the defined instance space. The results indicate the instance space is not adequately explored and the random generator from Lopes and Smith-Miles (2013) is extended by performing a principal component analysis on parameters of the real-world instances, sampling from the resulting distribution to obtain unique instances, then performing the inverse of the principal component analysis to obtain parameters in the original parameter space.

Ultimately, none of the various instance generation methods in the literature combine the desired principles of correctness, applicability, and reproducibility, an ECI method, and view the collection of instances through ISA. In particular, for the MDMKP the literature is sparse for solution methods, applications, and instance generation. By contrast, the KP contains examples of each distinct aspect listed but not in combination with each other. The KP literature serves as a complement to our work, in addition to the different ISA formulations across different problem classes. We build upon work in validating instance generation methods for the MDMKP by Scherer et al. (2023*b*) and the ASP by Scherer et al. (2023*a*).

The remainder of this paper is organized as follows. Section 4.2 depicts the problem by utilizing previous work from the known literature to generate the instance space for the known sources of MDMKP instances. Section 4.3 explains the methodology to target specific regions of the instance space through a goal programming method. To accomplish this, several modifications of the ISA methodology are needed without losing any information that is captured in the instance space. From this methodology, the resulting instance space is shown in Section 4.4 with the targeted instances. Section 4.5 discusses the implications of the targeted instances on the instance space in terms of metaheuristic performance and the mapping of instance structure to predicted performance. Finally, Section 4.4 summarizes the contributions

of this work and directions for future research.

4.2 Problem Description

ISA serves as a method to break down an optimization problem into distinct component spaces as illustrated in Figure 23. This concept was first introduced as ASP in Rice's seminal work (Rice, 1976), as depicted within the dotted box in the figure. Smith-Miles et al. (2014) expanded on this idea. However, the ISA framework goes beyond the original scope of the ASP, which primarily focuses on identifying the most suitable algorithm from a given portfolio based on measurable features derived from a collection of test instances. Instead the ISA framework allows a closer examination of the current set of test instances by examining algorithm footprints and regions where new test instances are needed to fill gaps present within the instance space.

For the MDMKP, Scherer et al. (2023*b*) decomposes the problem into its component spaces, borrowing from the literature of KP variants and adjusting measures to handle the multidimensional aspects of the knapsack and demand constraints. Subsequent work by Scherer et al. (2023*a*) examine the MDMKP across the known solution methods present in the literature. Beginning with the seminal work by Cappanera and Trubian (2005) with a tabu search method which borrows from work by Glover and Kochenberger (1996) in the method of *strategic oscillation* across the feasibility boundary. Followed by a simpler adaptive memory search by Arntzen et al. (2006) which focuses on tuning parameters to balance the level of feasibility against the quality of a solution. A scatter search implementation set forth by Hvattum and Løkketangen (2007) tackles the MDMKP from a evolutionary, population based approach. The most recent approach by Lai et al. (2019) utilizes multiple stages of tabu search combined with a balanced evaluation function between the quality of solution

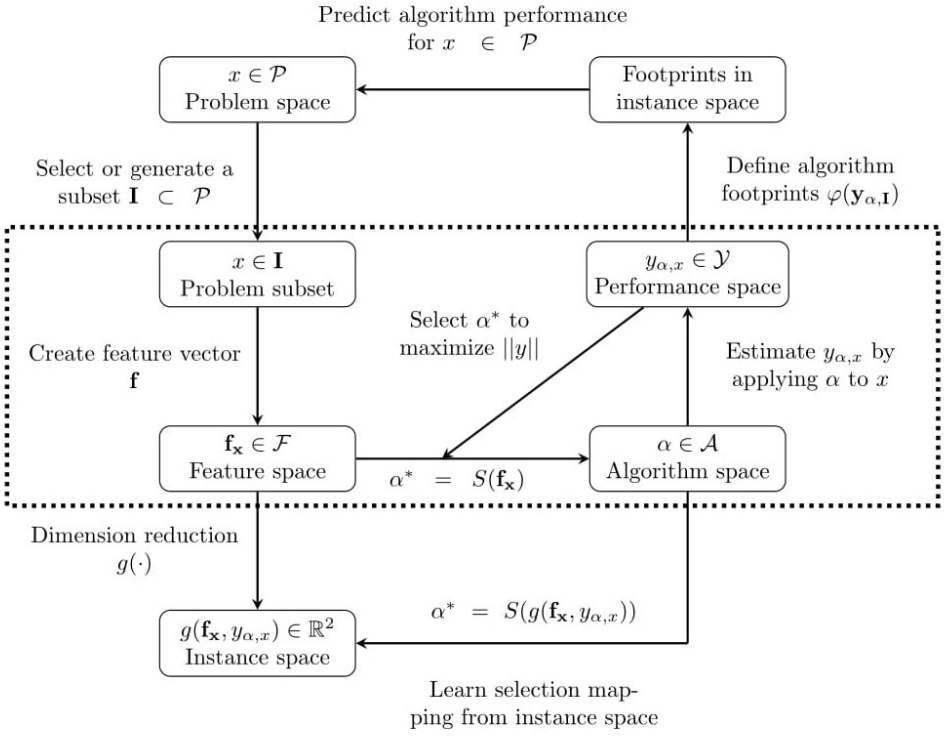

Figure 23. Summary of ASP (Rice, 1976) shown in dotted box, extended upon by Smith-Miles et al. (2014)

and the degree of infeasibility present in a solution.

The main source of test instances which constitute the problem subset of the MDMKP are from a known test library (Beasley, 1990), referred to as the *Beasley set of instances*, and originate from the instance generation method set forth by Cappanera and Trubian (2005). The authors employ knapsack constraints derived from MKP instances as defined by Chu and Beasley (1998). These constraints are further enhanced by the inclusion of demand constraints and objective function coefficients. The generation process follows a specific procedure: initially, the number of knapsack constraints, denoted as m, is set to either 5, 10, or 30. The number of demand constraints, denoted as q, is determined based on the value of m. Once m is fixed, the value of q can be set to one of three options: $q = 1$, $q = m/2$, or $q = m$. Additionally, the number of variables, denoted as n, is set to either 100, 250, or 500. The objective function coefficients are divided into two cases: the positive case where the objective function coefficients are nonnegative and the mixed case where the objective function coefficients are unconstrained in sign.

The secondary source of test instances utilized is derived from the Primal Problem Instance Generator (PPIG) set forth by Scherer et al. (2023*b*). PPIG employs a sampling-based approach for generating instances starting from the primal feasible space. PPIG involves generating a set of k solutions, where each decision variable has a probability p of being set as active (i.e., assigned a value of 1). The left-hand-side (LHS) coefficients of each constraint are generated in a predetermined manner to capture the desired problem meta-features. By applying each of the k solutions to the constraint coefficients, the corresponding right-hand side (RHS) values for each instance are obtained, resulting in an empirical distribution of RHS values. These empirical distributions are then used to set the final RHS values for each constraint in the problem instance. The instances generated in this fashion are referred to as

the *Scherer set of instances*.

By utilizing the method set forth by Cho et al. (2008) in the MKP, PPIG is augmented to explicitly induce desired correlation to the MDMKP instances. The procedure involves using the rank correlation induction method from Iman and Conover (1982), adapted for the MKP in Cho et al. (2008) and augmented for the MDMKP in Scherer et al. (2023*a*). This extended procedure enables us to generate the left-hand-side (LHS) coefficients in PPIG while explicitly maintaining the desired correlation coefficient values between the objective function coefficients and the LHS coefficients. The resulting instances generated through this approach are referred to as the *Scherer Correlated set of instances*, highlighting the incorporation of explicit correlation within the problem structure.

From these three sources of instances, we obtain 990 test instances. Of which, we removed 2 test instances because the Gurobi solver was unable to find an integer feasible solution after 6 hrs of runtime. The performance metric is the quality of the solution, measured by examining the best objective function value observed for an integer feasible solution. We utilize an Intel© Core™ i7-11700 CPU at 2.5GHz with 32 GB RAM for each of our tests.

After collecting the meta-feature values, measuring metaheuristic performance, and assigning the source of instances, we obtain the necessary inputs known as the meta-data needed to conduct ISA on the MDMKP by the Instance Space Analysis Toolkit, a MATLAB-based set of tools which automatically conducts ISA (Muñoz and Smith-Miles, 2020*b*). The toolkit is the underlying engine in the Melbourne Algorithm Test Instance Library with Data Analytics (MATILDA) online analysis tool. The detailed procedures used in the ISA methodology are discussed by Smith-Miles and Muñoz (2021) and the specific meta-features used are discussed by Scherer et al. (2023*b*).

As the output of the Projecting Instances with Linearly Observable Trends (PILOT) procedure used in ISA, the instance space is described through a projection matrix of the meta-feature values to a two-dimensional instance space. The projection is shown in (18). The output are coordinates of the instance space for an observed test instance, where the coordinates are designated based on a linearly observed trend of the difficulty of the test instance for the portfolio of metaheuristics to find high quality solutions for.

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} -0.2742 & 0.0232 \\ -0.1321 & 0.2524 \\ -0.2695 & -0.3597 \\ 0.4183 & -0.2882 \\ 0.0334 & 0.509 \\ 0.2937 & -0.2537 \\ 0.1802 & -0.5974 \\ -0.115 & -0.359 \\ 0.1624 & 0.2651 \\ 0.221 & -0.379 \end{bmatrix}^T \begin{bmatrix} \text{\# Decision Vars} \\ \text{Average Constraint Tightness - D} \\ \text{Prop Part Dominance - K} \\ \text{Max Obj to Constraint Corr - K} \\ \text{Min Obj to Constraint Corr - K} \\ \text{Range Obj to Constraint Corr - D} \\ \text{Range Within Constraint Corr - K} \\ \text{Max Within Constraint Corr - D} \\ \text{Min Within Constraint Corr - D} \\ \text{Range Across Constraint Corr} \end{bmatrix} \tag{18}$$

The resulting instance space is shown in Figure 24, color-coded by the source of the test instances. The linearly observable trend of difficult instances is located in the lower-right of the instance space, while the upper-left are the easier instances. The difficulty is measured by the performance space, where the majority of the metaheuristics failed to obtain high quality solutions for the Scherer Correlated instances, while multiple metaheuristics found high quality solutions for the original Beasley instances.

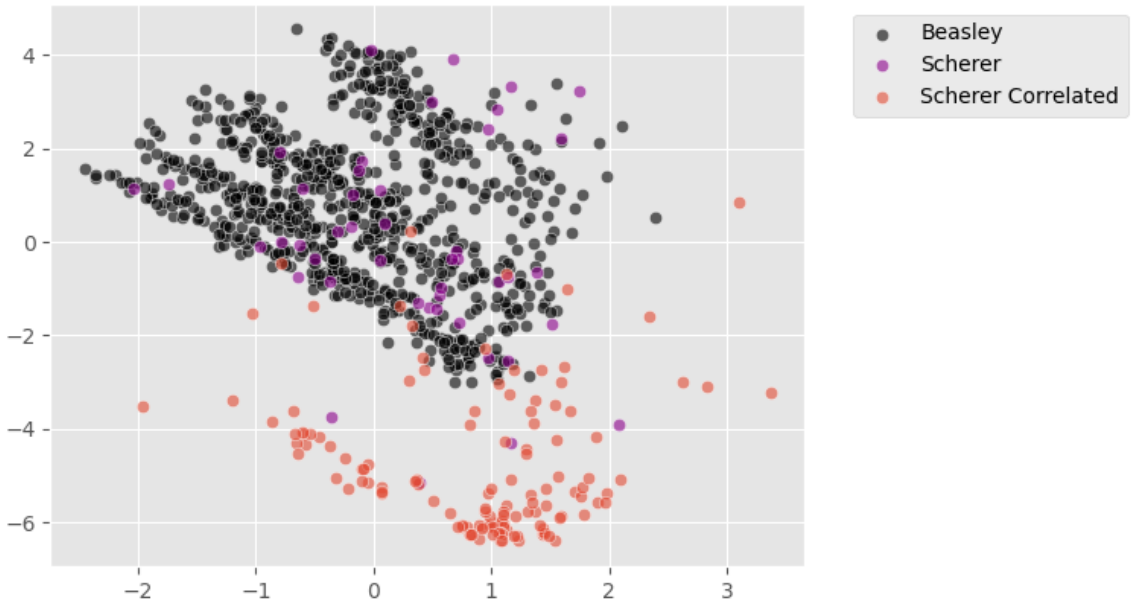

Figure 24. Instance space from Scherer et al. (2023*a*)

ISA performs algorithm selection through Performance Prediction and Automated Algorithm Selection (PYTHIA) which predicts the regions of strengths or weaknesses of each metaheuristic in the instance space. PYTHIA is based on a Support Vector Machine (SVM) that uses the coordinates of an instance in the instance space and outputs a binary measure of performance (i.e., good or bad) for each metaheuristic. The predictive performance of PYTHIA allows a measure of the adequacy of the gathered meta-features to characterize the MDMKP. We obtain an 84.7% accuracy of correctly selecting a metaheuristic with good performance for a given instance.

Through the combination of the projection in (18) and the instance space shown in Figure 24, the regions of particular meta-feature values are distinguishable. In particular, the region in the lower portion of the instance space is characterized by instances with the full range [-1,1] of correlation coefficients due to the instance generation method utilizing an explicit correlation. Of the 10 meta-features used in (18), 7 are related to the correlation structure of an MDMKP instance. The instances in the upper region of the instance space contain demand constraints with less tightness

on average compared to the lower region.

ISA is particularly useful for exposing regions of the instance space lacking in diverse instances (Smith-Miles and Bowly, 2015). To target a region of the instance space, Smith-Miles and Bowly (2015) use a GA with multiple strategies to obtain novel instances which fill the gaps present in an instance space of the graph coloring problem. The authors adopt four strategies to vary the fitness function of the GA and the selection of target points. The first rewards increased distance from the set of known instances, the second adopts a grid of target points across the valid instance space with a fitness function minimizing distance to the point, the third uses target points along the theoretical boundary of the valid instance space, and the final strategy chooses points arbitrarily close to known instances but spread throughout the valid instance space.

We adopt a similar method to target different regions of the instance space, but we select target points based on gaps present in the instance space shown in Figure 24. Instead of a GA, we use a direct approach by formulating the problem in a mathematical optimization framework. In particular a goal programming approach aiming to find the meta-feature values needed to obtain instances in the desired region of the instance space. Then through PPIG, we generate instances with controllable meta-feature values for the MDMKP. We examine the newly generated instances realization in the instance space and test their difficulty by applying the aforementioned metaheuristics.

4.3 Methodology

To target a region of the instance space, we adopt a goal programming approach to find the necessary meta-feature values to construct a desirable instance. ISA performs several tasks before PILOT to project the instances into the instance space. In

particular, a data preprocessing method which transforms the raw feature values into inputs for the transformed features stated in (18). Additionally, the preprocessing methods must be invertible to take necessary transformed feature values from an optimization framework into raw feature values to target with instances generated from PPIG.

ISA executes a number of data preprocessing tasks to prepare the meta-data for additional tasks such as meta-feature selection and PILOT for dimension reduction. Defined in Smith-Miles and Muñoz (2021), Preparation for Learning of Instance Meta-data (PRELIM) converts the performance measure of each solution procedure into a binary measure of "good" performance for binary classification, then bounds and scales the features to reduce the effect of outliers. The bounding criteria is the median plus or minus five times its interquartile range (IQR). To stabilize the variance of the features and performance measures, one parameter Box-Cox transformation. Following these transformations, the inputted performance measures and features undergo standardization such that their mean is zero and standard deviation is one. These converted values for the instance features are used as input to the projection defined in (18).

The goal programming formulation is based on using the transformed feature values, the projection in (18), and the deviations in the instance space in the (Z_1, Z_2) coordinate plane. Specifically, the decision variables include the values to set for the transformed feature values F_i, for $i = \{1, \ldots, 10\}$ and the positive and negative deviations from the desired target point (X_1, X_2) for the first dimension $\{Z_1^+, Z_1^-\}$ and second dimension $\{Z_2^+, Z_2^-\}$. The constraints are based on the projection matrix coefficients, $\{p^2, p^2\}$ for i = {1,...,10}, from PILOT and maintaining the feasible boundary of possible transformed feature values, $\{l_i, u_i\}$ for i = {1,...,10}, provided by utilizing the bounding and scaling procedures in PRELIM. This results in the

mathematical optimization framework formulated as:

$$\min \quad Z_1^+ + Z_2^+ + Z_1^- + Z_2^- \tag{19}$$

$$\text{s.t.} \quad \sum_{i=1}^{10} p_i^1 F_i + Z_1^+ - Z_1^- = X_1 \tag{20}$$

$$\sum_{i=1}^{10} p_i^2 F_i + Z_2^+ - Z_2^- = X_2 \tag{21}$$

$$l_i \leq F_i \leq u_i \quad \text{for} \quad i = \{1, \ldots, 10\} \tag{22}$$

$$Z_1^+, Z_2^+, Z_1^-, Z_2^- \geq 0 \tag{23}$$

The formulation using (18) presents an issue with PPIG. Several of the constraint correlation meta-features are dependent on the initial objective to constraint correlation distribution chosen. The correlation structure for the MDMKP is in the general form shown in (4.3). The first column and row are defined explicitly to match the desired correlation structure, while the subsequent entries in the matrix are elementwise products between the corresponding column and row entries. Therefore a solution presented by the goal programming formulation may recommend impossible combinations of meta-feature values. For example, the goal program may find an optimal solution by setting both the maximum and minimum objective to constraint correlation for the knapsack constraints of 0, while the maximum within constraint correlation for the knapsack constraints must be 1.

$$\begin{bmatrix} 1 & \rho(c, A_1) & \cdots & \rho(c, A_{m+q}) \\ \rho(c, A_1) & 1 & \cdots & \rho(c, A_1) * \rho(c, A_{m+q}) \\ \vdots & \vdots & \ddots & \cdots \\ \rho(c, A_{m+q}) & \rho(c, A_1) * \rho(c, A_{m+q}) & \vdots & \ddots \end{bmatrix}$$

By reducing the final meta-features selected in the Instance Space Analysis Toolkit

settings to the minimum recommended setting of 4 (Smith-Miles and Muñoz, 2021), the projection matrix shown in (24) is obtained. The resulting instance space shown in Figure 25 appears to distinguish performance, sources, and feature values distributed across the instance space. Upon further examination of PYTHIA the classifier performs similar to the higher dimensional projection with an accuracy of 85.2%. The difficult instances still lie within the lower-right region of the instance space. Therefore we retain most of the properties of ISA while reducing the feature space to a manageable number of meta-features that can be used within the PPIG generation method.

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} 0.5004 & 0.2304 \\ 0.0043 & 0.3854 \\ -0.3554 & 0.3986 \\ 0.125 & -0.1931 \end{bmatrix}^T \begin{bmatrix} \text{\# Constraints - D} \\ \text{Average Constraint Tightness - D} \\ \text{Min Obj to Constraint Corr - K} \\ \text{Max Within Constraint Corr - D} \end{bmatrix} \tag{24}$$

After manually selecting a target point in the instance space and obtaining the required meta-feature values by using inverse PRELIM, PPIG naturally fits into the instance generation methodology. PPIG allows manipulations in its parameters to address the average constraint tightness by changing the sampling from the distribution of RHS values and the correlation structure through its explicit correlation induction method. The resulting instances are then processed through PRELIM, projected into the instance space, and their realization in the instance space compared to the desired target point is easy to visualize.

In this work, we choose target points based on existing gaps within the feasible region of instances while remaining on the boundary of the current collection of instances. The purpose of this work is to demonstrate the controllable properties

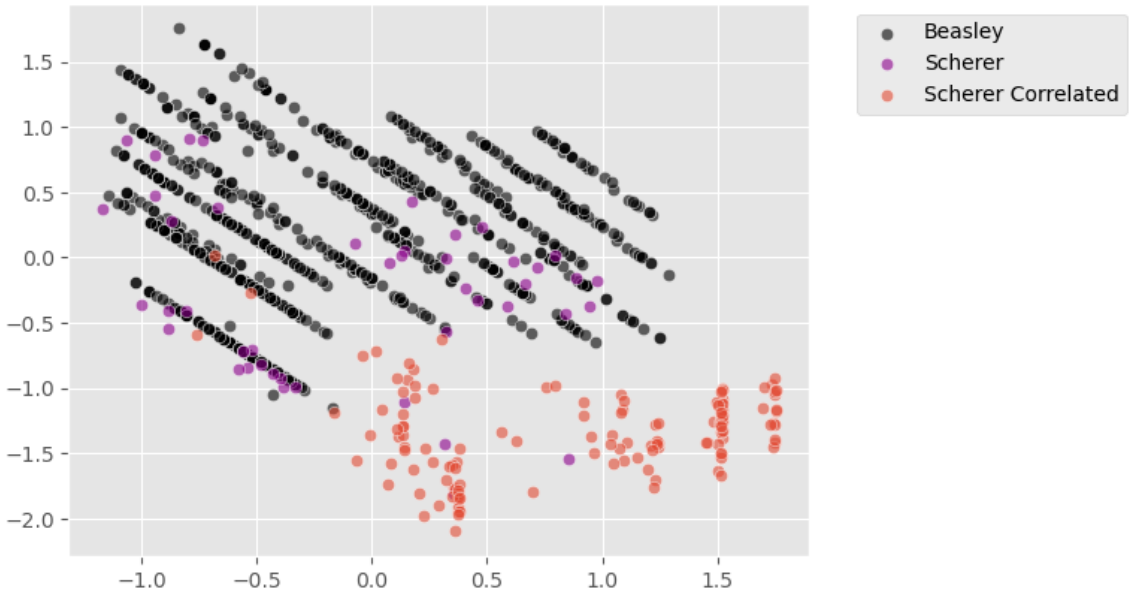

Figure 25. Reduced instance space

of targeting an instance space through an optimization framework then generating instances close to the targeted region through PPIG. Therefore we choose to target 3 separate regions of the instance space and generate 45 instances from PPIG to examine the proximity of the instance realizations in the instance space. We target (-0.5,-1.5), (1.5,-0.5), and (0,1.5) because two target points are based on finding instances between the Scherer Correlated set of instances and the Beasley set of instances while the third target point is to demonstrate our methodology for expanding the original Beasley set of instances.

4.4 Results

The recommended meta-feature settings to obtain an instance realization at (-0.5,-1.5) in the reduced instance space are: containing one demand constraint, setting the constraint tightness of demand constraints to zero, setting the minimum objective to constraint correlation for the knapsack constraints to -0.237, and setting the maximum within constraint correlation for the demand constraints to 0.01. A similar process is

used for the remaining two target points which are then matched by PPIG.

To control the meta-features in PPIG, the number of demand constraints is trivial because it is an input parameter to the instance generation method. The average tightness is obtained by intuitively selecting from the distribution of the RHS values generated in PPIG. For example, to obtain an average tightness of zero we select the maximum value of our observed RHS value distribution. If we need a nontrivial constraint tightness, such as the recommended setting of 0.397 to target (1.5,-0.5), then we select a random sample of RHS values such that their average is 0.397, resulting in different combinations of RHS values in the instance realization.

For the correlation values, we utilize a uniform distribution for the explicit declaration of correlation coefficients between the objective function and constraint coefficients. For example, the minimum objective to constraint correlation for the knapsack constraints to target point (-0.5,1.5) must be -0.237, therefore we set this to be the lower bound of our distribution of explicitly declared correlation coefficients. While the maximum within constraint correlation for the demand constraints is set by utilizing a uniform distribution for the explicit declaration of correlation coefficients between the objective function and constraint coefficients such that their element wise product is no more than the recommended value.

The results of the procedure are shown in Figure 26, where the deviations of the instance realizations from the target points are mostly due to fluctuations by using the uniform distribution for the ECI portion of PPIG. If the correlation coefficients were set to exactly the recommended levels without variation, then the newly generated instances would lack diversity from each other in the instance space. By allowing a degree of variance in the correlation, we obtain unique MDMKP instances which help to fill the gaps present in the instance space while showing the effectiveness of the proposed methodology.

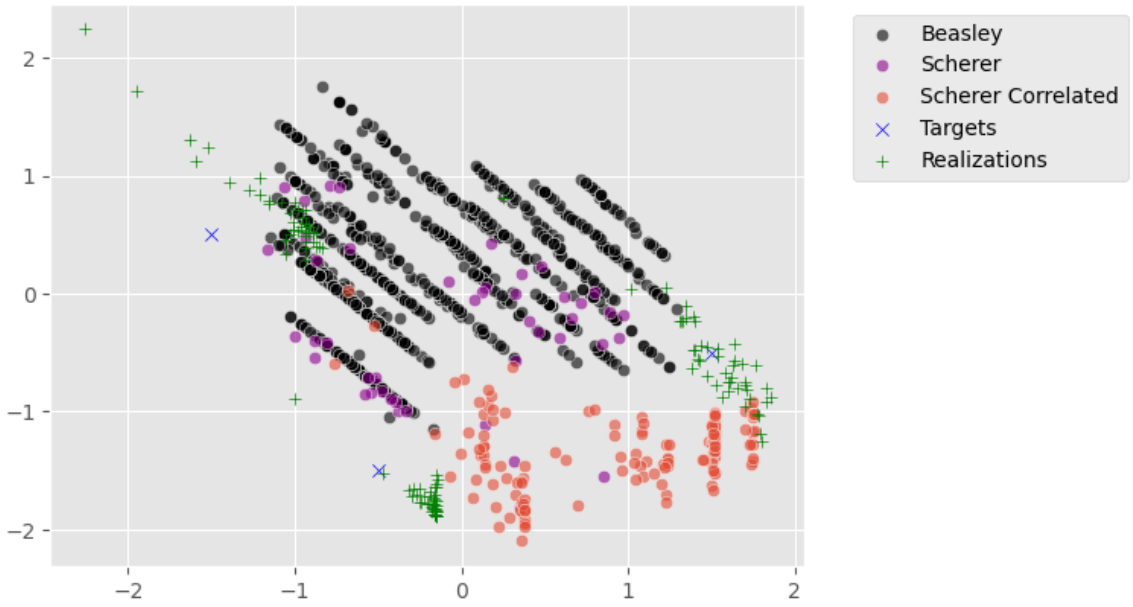

Figure 26. MDMKP instance realizations from PPIG in the reduced instance space

To measure the effectiveness of PPIG to target the specific regions of the instance space, we measure the average euclidean distance for the 45 generated instances for each targeted point. The results are shown in Table 5, with the largest distance occurring from targeting the point (-1.5,0.5), due to the minimum objective to constraint correlation for the knapsack constraint following a uniform distribution with the lower bound set to zero. Therefore we observe instances, such as the outliers appearing in the upper-left region of the instance space, which contain minimum correlation values further away from zero.

Table 5. Euclidean distance to target for realized instances

	Target Points		
	(0.5,-1.5)	(1.5,-0.5)	(-1.5,0.5)
Average	0.42	1.25	2.12
SD	0.11	0.91	1.09
Median	0.43	1.95	2.56

4.5 Discussion

To examine the new instances in the ASP for the MDMKP, we use previous work by Scherer et al. (2023*a*) to produce the algorithm footprints of the four examined metaheuristics. Shown in Figure 27, the two-stage tabu search method by Lai et al. (2019) is expected to perform best for the instances located in the upper-left region of the instance space. While the instances located in the lower-right section of the instance space have been observed to obtain better solutions through the scatter search method from the work by Hvattum and Løkketangen (2007) and the local search method from the work by Cappanera and Trubian (2005). The instances in the lower-right region of the instance space however are not expected to contain any good performance from the implemented metaheuristics.

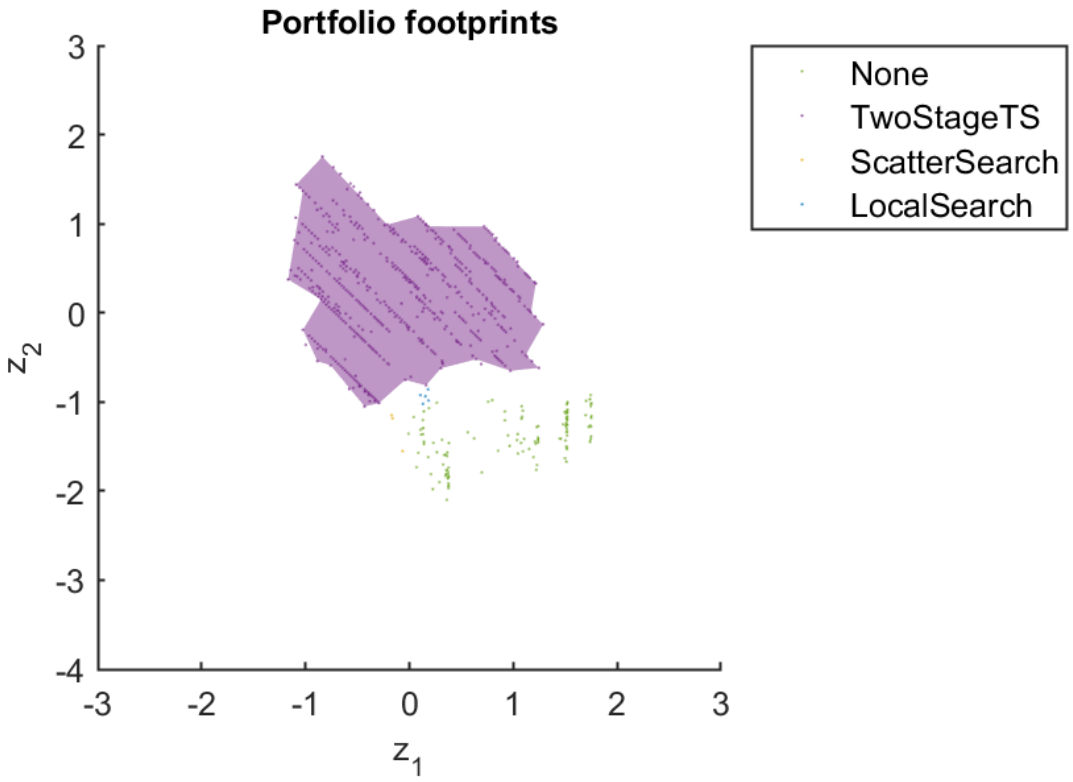

Figure 27. Footprints of metaheuristics in the reduced instance space

The metaheuristics are applied to the new realizations of the targeted instances, then compared to the estimated performances through the footprints shown in Figure 27 to gauge the effectiveness of ISA to characterize new instances of the MDMKP. The results are summarized in Table 6. In some instances, the metaheuristics reported the same solution, therefore Table 6 includes counts for ties between the best found metaheuristic. We refer to the local search method (Cappanera and Trubian, 2005) as LS, the adaptive memory search method (Arntzen et al., 2006) as AMS, the scatter search method (Hvattum and Løkketangen, 2007) as SS, and the two-stage tabu search method (Lai et al., 2019) as TSTS.

Table 6. Number of instances with best found solution by metaheuristic

	LS	AMS	SS	TSTS
(-0.5,-1.5)	31	14	19	0
(1.5,-0.5)	30	9	13	3
(-1.5,0.5)	19	10	16	9

For the 45 instances generated to target (-0.5,-1.5), LS found the best solution for 31 instances, while SS found the best solution for 19 instances and AMS for 14 instances. For the 45 instances generated to target (1.5,-0.5), similar results occurred with LS finding the best solution for 30 instances, but SS and AMS did slightly worse. Results were more spread across the metaheuristic portfolio for (-1.5,0.5), where TSTS began to show its effectiveness compared to the other targets, but largely LS performed best.

The results of the portfolio footprints compared to the results observed in Table 6 differ in several aspects, most notably the dominance of TSTS compared to the other metaheuristics. However, when examining the performance of the metaheuristics for targets (-0.5,-1.5) and (1.5,-0.5), the observed performance is mostly in agreement with the established footprints. SS and LS performed well on instances in the lower-right region of the instance space, with LS performing well on instances between the

Beasley and Scherer Correlated sources of test instances.

The largest difference is from the upper-left region of the instance space, where TSTS is found to be most effective in the original set. However the results in Table 6 for the target point (-1.5,0.5) indicate LS found the best solution for the most number of the 45 generated instances. The ISA toolkit also provides the footprints for each solution method when separately evaluated. The resulting footprints are shown in Figures 28-31. Including these perspectives, the similar performances across the portfolio of metaheuristics in Table 6 now align with the expectations of the metaheuristic footprints. These instances are characterized as easier to solve, given the ability of the known solution methods to all find the best known solutions for most instances.

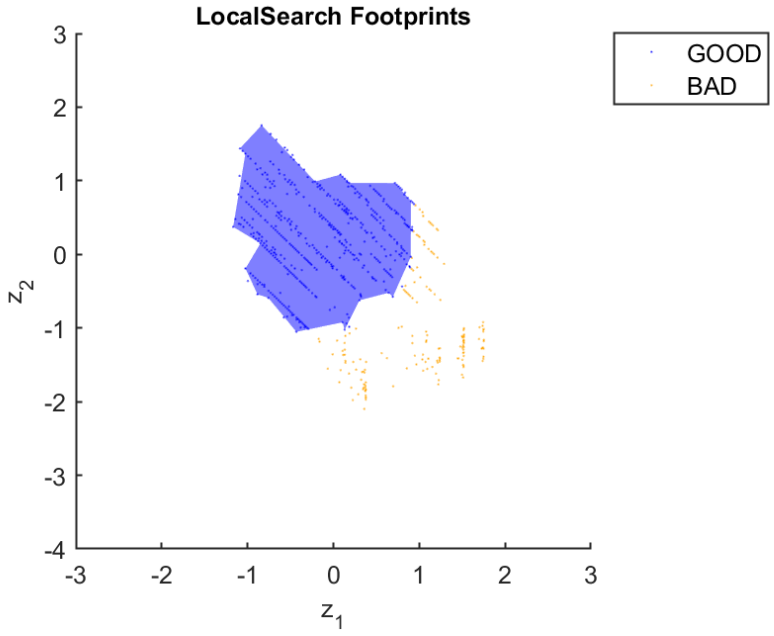

Figure 28. LS footprints across the instance space

4.5.1 Updating the Instance Space

The ISA methodology is an iterative process, thus the added instances need to be reevaluated to update the instance space. The instances generated to fill the gaps

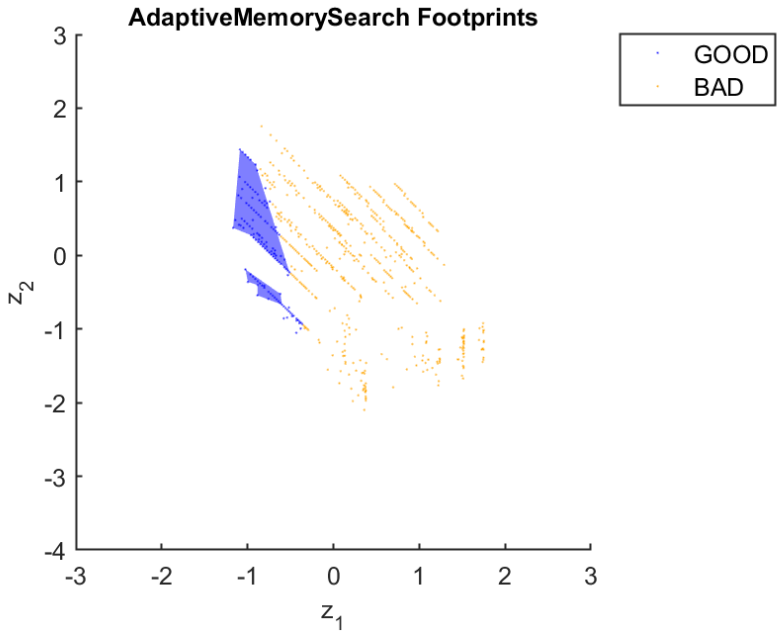

Figure 29. AMS footprints across the instance space

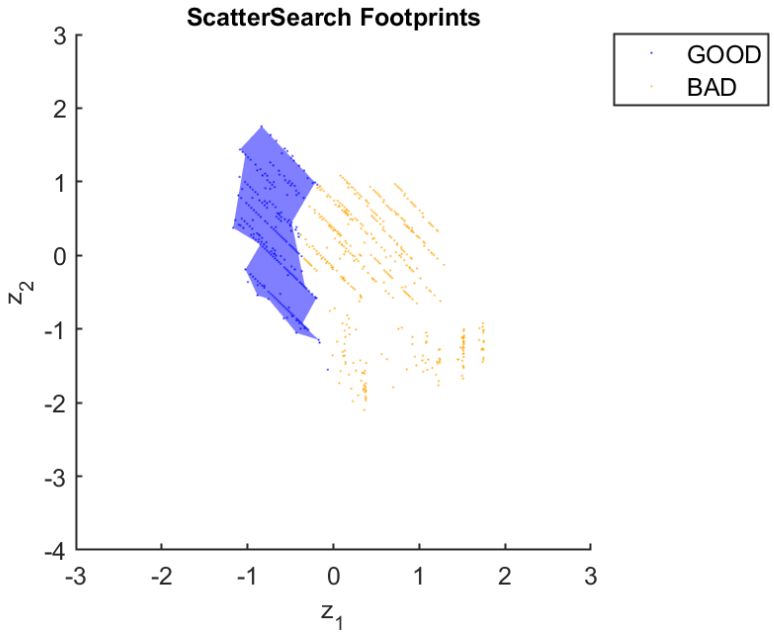

Figure 30. SS footprints across the instance space

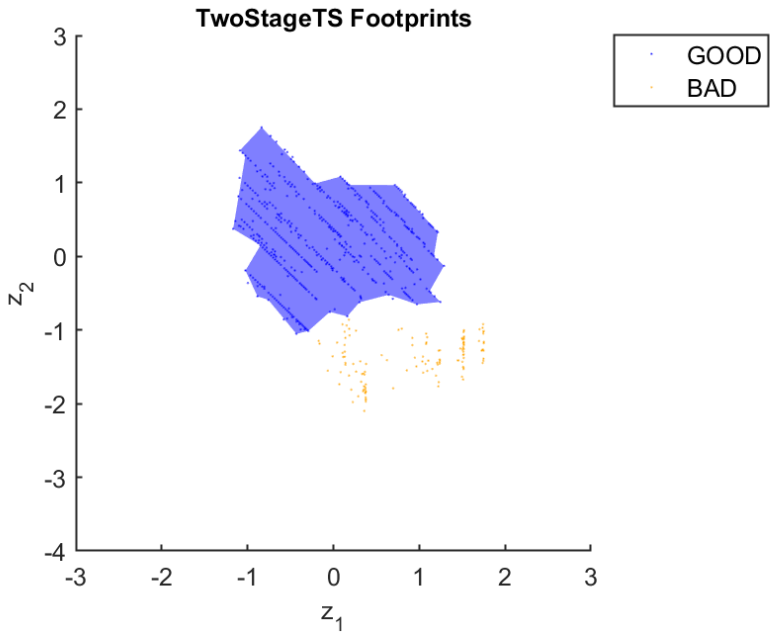

Figure 31. TSTS footprints across the instance space

present within the instance space in Section 4.4 can be assessed in a new instance space with a possibly new set of features to describe their difficulty. After adding the 135 instances generated from PPIG to target specific regions of the instance space, the updated ISA produces the projection equation in (25) and the two-dimensional instance space shown in Figure 32.

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} 0.2656 & 0.0315 \\ -0.1966 & -0.3061 \\ 0.0926 & -0.4124 \\ 0.1889 & 0.0021 \end{bmatrix}^T \begin{bmatrix} \text{Average Constraint Tightness - D} \\ \text{Proportion Partial Dominant Pairs - K} \\ \text{Max Obj to Constraint Corr - D} \\ \text{Profits Coefficient Variation Normalized} \end{bmatrix} \tag{25}$$

A different subset of meta-features is selected from the original four used in the reduced instance space from Section 4.3. As a measure of instance difficulty, the average constraint tightness for the demand constraints and a statistic relating the objective to constraint correlation values remained. However, the other meta-features

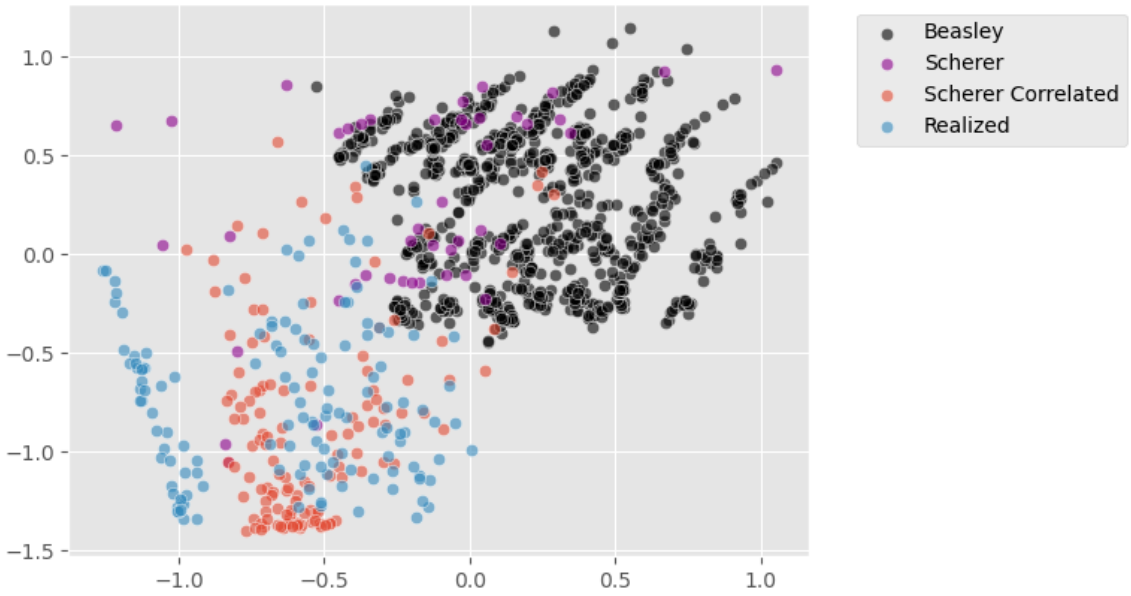

Figure 32. Updated instance space

added are the proportion of partially dominant pairs for the demand constraints, and the coefficient of variation for the objective function coefficients. The proportion of partially dominant pairs is adapted from the work on the KP by Hall and Posner (2007), but is directly tied to the objective to constraint correlation since the ECI method in PPIG is based on rank correlation. Therefore the two meta-features are not perfectly collinear but exhibit similar patterns across the instances observed.

In terms of the distribution of meta-features in Figure 32, the lower-left region of the instance is characterized by difficult instances. In this region the average tightness for the demand constraints and the coefficient of variation for the objective function coefficients are lower values. While the proportion of partially dominant pairs for the knapsack constraints are high. The lower region is where higher values for the maximum objective to constraint correlation coefficients for the demand constraints are observed, trending directly upwards in the instance space where instances with lower correlation coefficients lie.

The footprints of the metaheuristics exhibit a similar pattern as reported in Sec-

tion 4.5, with the Scherer Correlated and the newly realized instances largely lacking any effective metaheuristics. In comparison LS and TSTS perform best for the majority of the Beasley and Scherer instances. To help relate the problem structure and the recommended metaheuristic for the ASP, an optimal sparse decision tree (McTavish et al., 2022) adopted from the methodology set forth by Scherer et al. (2023*a*) is shown in Figure 33.

To evaluate the model and avoid overfitting, the metadata for the final four metafeatures is evaluated with a 75/25 training/testing split. The testing data accuracy of the Generalized Optimal Sparse Decision Tree (GOSDT) model to predict the best metaheuristic given an MDMKP instance is 0.87. The sparse decision tree model provides an interpretable answer to the ASP originally posed by Rice (1976). The decision tree contains the properties of sparsity and optimality, compared to a greedy decision tree (Hu et al., 2019).

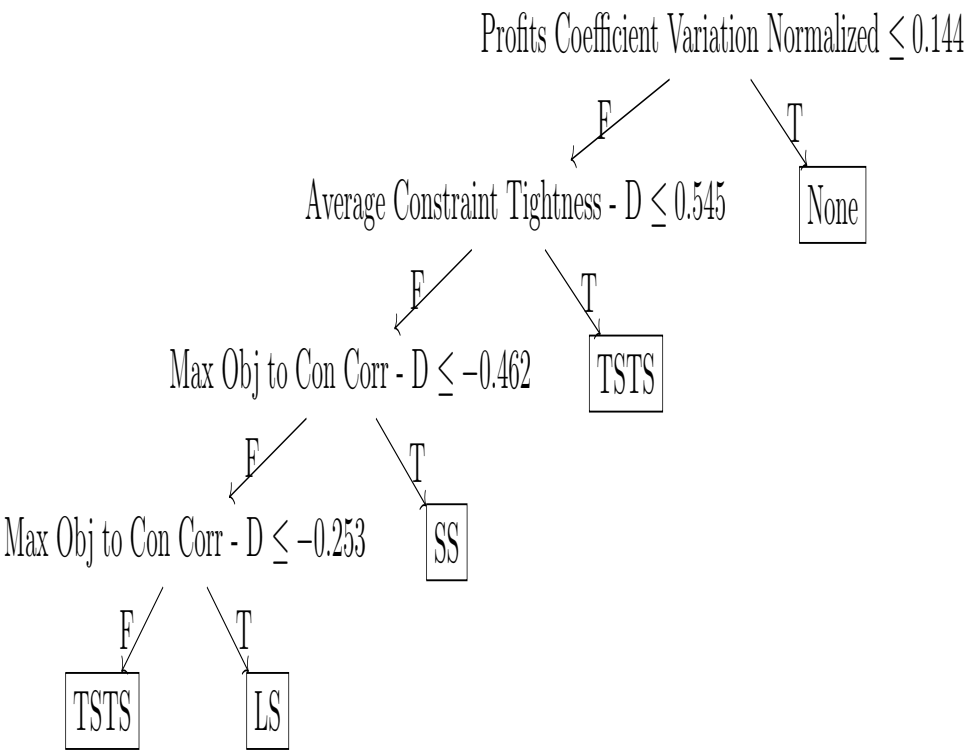

Figure 33. Optimal sparse decision tree for MDMKP ASP

4.6 Conclusion

This work has provided an optimization framework to generating MDMKP test instances which target regions of the instance space. Using ISA the MDMKP is separated into component spaces to ultimately answer the ASP. By producing a two-dimensional visualization of the collection of instances, the metaheuristics may be evaluated in terms of their effectiveness and the adequacy of the test set of instances. Using a goal program to find desired meta-features and utilizing PPIG to generate instances with the desired meta-features, the methodology presented in this work shows promise to directly targeting regions of the instance space.

The instance realizations shown in the two-dimensional instances space provide a visual measure of effectiveness of the methodology presented in this work. The goal of the realized instances are to help present a diverse set of instances to use for future

empirical testing of the MDMKP. A final instance space is presented which utilizes the added instances which are evaluated by the portfolio of metaheuristics examined.

While PPIG can generate feasible MDMKP instances, the limitations of using a goal programming approach outlined in this work are from the recommended meta-feature values. For example, if targeting certain regions of the instance space, the goal program recommends meta-feature values which are at their extreme values. In particular, the constraint tightness being set to maximum for either the knapsack or demand constraints. Without loss of generality, for the knapsack constraints a maximum tightness for each dimension of the constraints renders only a null solution to be feasible while the demand constraints are ineffective.

In this work the final meta-features selected were reduced from 10 to 4. This reduction provided independent meta-features which were controllable through PPIG. However other methodologies extending this approach may result in meta-feature combinations which are impossible to produce because of dependencies of meta-feature values such as the correlation structure discussed in Section 4.3. Other meta-features which do not appear controllable under PPIG should be examined with scrutiny, as this work found that the relationship between the objective and constraint correlation is directly tied to the proportion of dominant pairs meta-feature. Therefore the two may be interchangeable when attempting to generate targeted instances.

Future work may entail using PPIG on other knapsack variants. A natural extension would be towards the MKP or a multiple choice knapsack variant. The combination of using instance generation procedures which are based on sampling strategies and targeting regions of the instance space through goal programming is another possible avenue of future research. The purpose of this work was to demonstrate the methodology on a particular problem which has been mostly overlooked in the literature but is easily extendable to other applications.

V. Conclusion

This research develops methods and applications for instance generation methodologies in the multidemand multidimensional knapsack problem (MDMKP). Initially a literature review is conducted to examine the background for different subject areas pertaining to this research. This broad review then focused into instance generation methodologies and instance space analysis (ISA) to become foundational aspects of this work. After gathering information over multiple subject areas, this research applied ISA to the MDMKP by defining all aspects required to fully investigate the problem of interest. Followed by multiple variants of an instance generation method based on a sampling of distribution of test instances. In addition, this research conducted multiple empirical tests for the MDMKP including exact and inexact approaches. Lastly, to address directly targeting specific configurations of an MDMKP instance, this research combined the instance generation procedure developed into a goal programming framework to produce instance realizations with desired characteristics.

The key findings of this research pertain to instance generation procedures, using ISA in empirical research and providing algorithm insights into MDMKP performance. These areas of research are combined in this dissertation to help generalize to other problem classes in optimization and machine learning. The findings include the use of ISA to verify the effectiveness of an instance generation procedure, the implementation of interpretable machine learning methods to help uncover the relationship between test instance structure to solution procedure performance, and targeting regions of the instance space through a controllable optimization framework.

5.1 Summary

Chapter II adopts ISA for the MDMKP and demonstrates the ISA methodology as a verification of the diversity of the current collection of test instances and proposes augmentations of the current test set. The ISA is conducted by breaking down the component spaces for the MDMKP, most importantly defining the meta-features which characterize the problem. Following this dissection of the problem, the original test instances are evaluated through the Gurobi solver and shown in their instance space. The multidimensional knapsack (MKP) instances of Cho (2005) and adding an additional set of $q \geq$ constraints to generate MDMKP instances. The added instances create issues with maintaining feasibility in the generation procedure. Subsequently the Primal Problem Instance Generator (PPIG) is defined to address this feasibility issue and demonstrated in the defined instance space. This contribution includes defining the component spaces of the ISA for the MDMKP, analyzing added test instances impacts on the instance space, and an instance generation procedure which provides feasible and diverse instances.

Chapter III utilizes the existing solution methods for the MDMKP from the literature and gains perspective on the relationship between instance configuration and solution procedure performance. This is accomplished by modifying the ISA and constructing an interpretable generalized optimal sparse decision tree (GOSDT) model to predict the best metaheuristic for the algorithm selection problem (ASP). This model replaces the existing support vector machine (SVM) classifier within the ISA toolkit and provides a direct mapping between the decision boundaries for recommending particular metaheuristics for a given instance. The findings include a lack of effective metaheuristics when examining correlated instance generated by PPIG. This contribution includes the first use of ISA with metaheuristics constituting the algorithm space, an interpretable approach to answer the ASP, and exposing the lack

of high quality metaheuristics for instances outside of the original training set.

Chapter IV is a culmination of the two previous contributions into a targeting procedure for generating instances to fill gaps present in the instance space. Utilizing the ISA defined in the previous contributions and the metaheuristic performance for the previous test instances, this contribution introduces an optimization framework to target regions of the instance space. Initially a full instance space is shown to be infeasible due to some meta-features being directly dependent on others, forcing the recommended configurations to be impossible. To address this, a reduced instance space is proposed which does not degrade the quality of the instance space to characterize the test instances while granting controllable meta-features. The integration of PPIG then produces instances which minimize the distance between the generated instances realized in the instance space compared to the target points. This contribution includes a new goal programming model which gives optimal recommendations for meta-feature values to target then demonstrates the capability of PPIG to produce instances which target the recommended meta-feature values and result in filling the gaps present in the instance space.

5.2 Future Work

This dissertation provides a substantial body of work to draw upon for future research. Multiple aspects drawing from each contribution could be extended. For the use of ISA on a given problem, the MDMKP is the selected problem of interest in this research due to its applications for capital budgeting, aircraft loading, and the location of undesirable facilities. However, other avenues to explore are the other knapsack variants or other interesting classes of optimization problems. ISA is a relatively new approach to characterizing a problem, with the majority of the problem classes explored detailed by Smith-Miles (2019).

For instance generation procedures, there are many future directions of this work. To begin, the optimization research community largely ignores the analysis of test instances used in empirical testing; an issue addressed by Reilly (2009). ISA serves to fill this need by showing the current collection of test instances in a convenient two-dimensional visualization. Building upon this realization of inadequate test instances, instance generation techniques are a large area to explore for multiple problem classes. The principles of a well-defined instance generation procedure are outlined by Hall and Posner (2010) but the properties of feasibility and diversity are the direct extensions of this work.

The literature reviews, methodological advances, and avenues explored in this dissertation vastly improve the empirical testing of solution procedures. The contributions fulfill the needs for improving the test set of instances which exist for the MDMKP, the ISA methodology for the ASP, and the use of instance generation procedures to generate instances with controllable configurations. As numerous new solution procedures are proposed, the empirical testing of their effectiveness will remain a key aspect in research.

Appendix A. Distribution of Features Across Instance Spaces

Figure 34 depicts the instance space defined by (6) as it relates to the distribution of meta-features. The lower-right portion of the instance space constitutes the region with more difficult instances. The generation of more difficult instances relies on higher values of the max across constraint correlation between the knapsack and demand constraints, higher values of max within constraint correlation for the demand constraints, and higher values for the number of demand constraints.

The upper-right region of the instance space consists of instances with lower values for: the number of knapsack constraints, the min objective to knapsack constraint correlation coefficient, the range within the knapsack constraint correlation coefficients, and the max across constraint correlation coefficient. The meta-features which separate the left band from the right band include: the proportion of partially dominant items in the demand constraints, the min objective to constraint correlation coefficient for the demand constraints, and the coefficient of variation for the weights in the demand constraints.

Figure 35 depicts the instance space defined by (7) as it relates to the distribution of meta-features. The lower-right region of the instance space comprises of easier instances with the most difficult instances clustered in the upper-left region. The generation of more difficult instances relies on higher values of the number of knapsack and demand constraints, higher values for the range between the objective function and demand constraints correlation coefficients, higher values of the range within the demand constraints correlation coefficients, higher values of max across constraint correlation, lower values for the min within constraint correlation for the demand constraints, and lower values for the min across constraint correlation.

The lower-right portion of the instance space consists of instances with: a lower

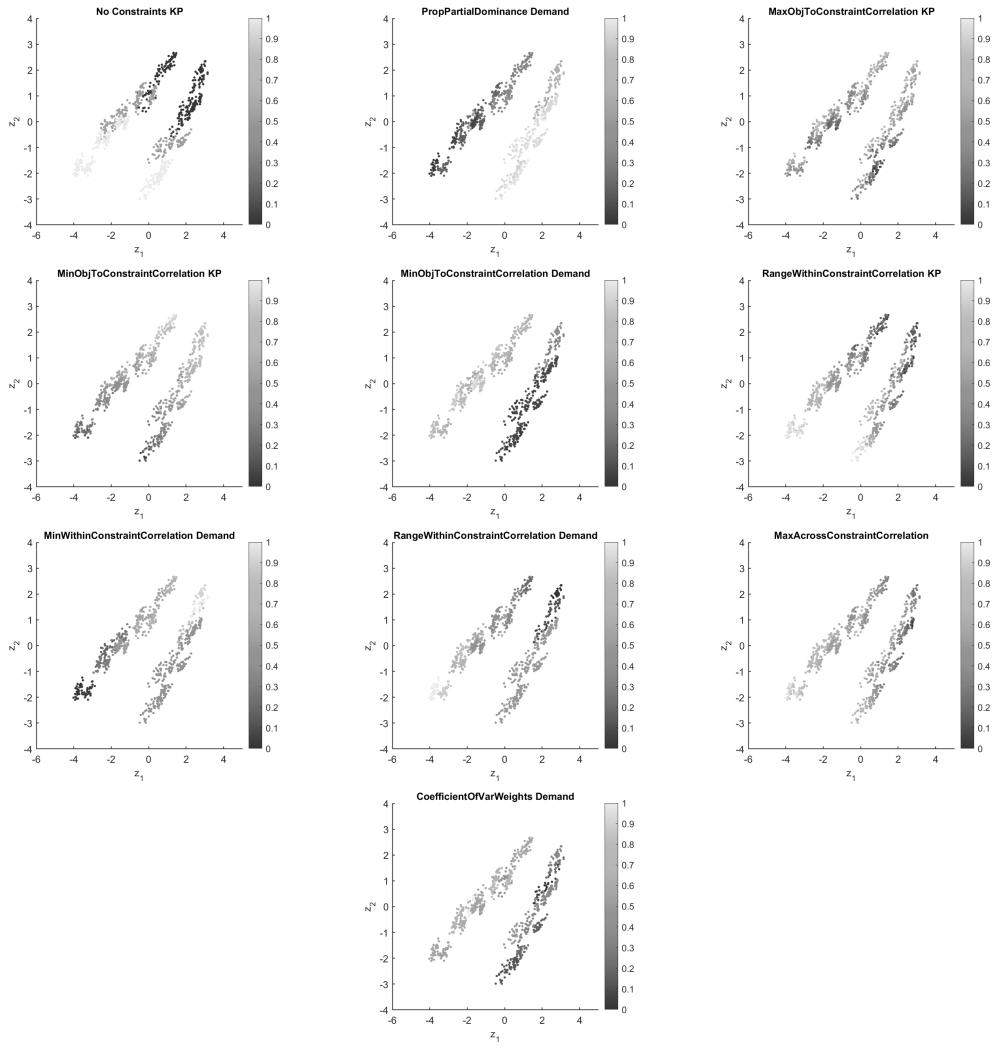

Figure 34. Distribution of features across Beasley instance space

number of knapsack and demand constraint, higher values of max constraint tightness for the knapsack constraints, higher values of max objective to constraint correlation for the knapsack constraints, and lower values of the range objective to constraint correlation for the demand constraints. The cho set of instances defines the smaller cluster located in the lower portion of the instance space, which are characterized by a smaller number of demand constraints due to infeasibility issues from adding more constraints.

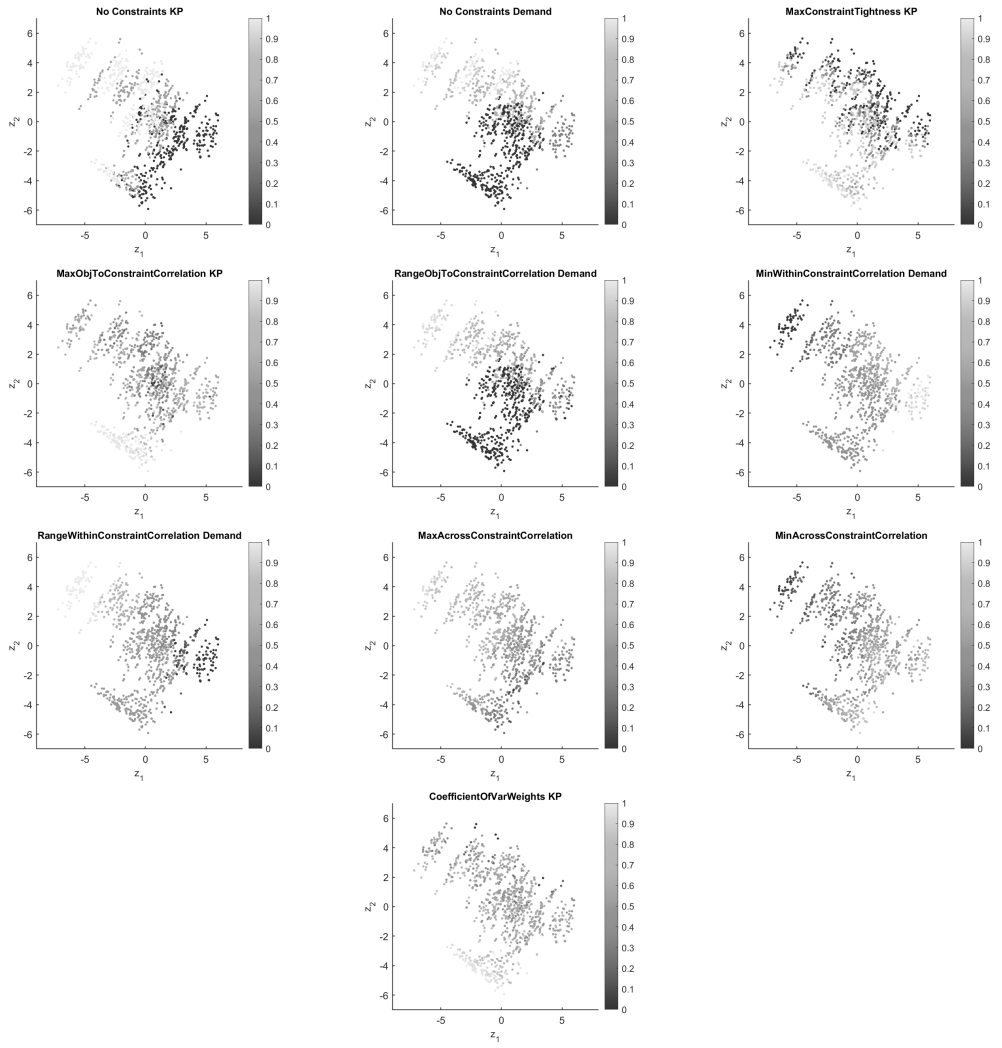

Figure 35. Distribution of features across Cho and Beasley instance space

Figure 36 depicts the instance space defined by (8) as it relates to the distribution

of meta-features. The lower region of the instance space comprises of easier instances with the most difficult instances in the upper region. The generation of more difficult instances relies on higher values of the number of knapsack and demand constraints, higher values for the range between the objective function and demand constraints correlation coefficients, higher values for the range within constraint correlation for the demand constraints, higher values of range across constraint correlation, lower values for the min within constraint correlation for the demand constraints, and lower values for the min across constraint correlation.

The cluster depicted in the lower region of the instance space is defined by instances with lower values of the number of demand constraints, the range between the objective function and demand constraint correlation coefficients, and the range within the constraint correlation coefficients for the demand constraints. The lower region is also defined by higher values of the max constraint tightness for the knapsack constraints, the max between the objective and constraint correlations for the knapsack constraints, the min within the constraint correlation coefficients for the demand constraints, and the coefficient of variation for the knapsack constraint weights.

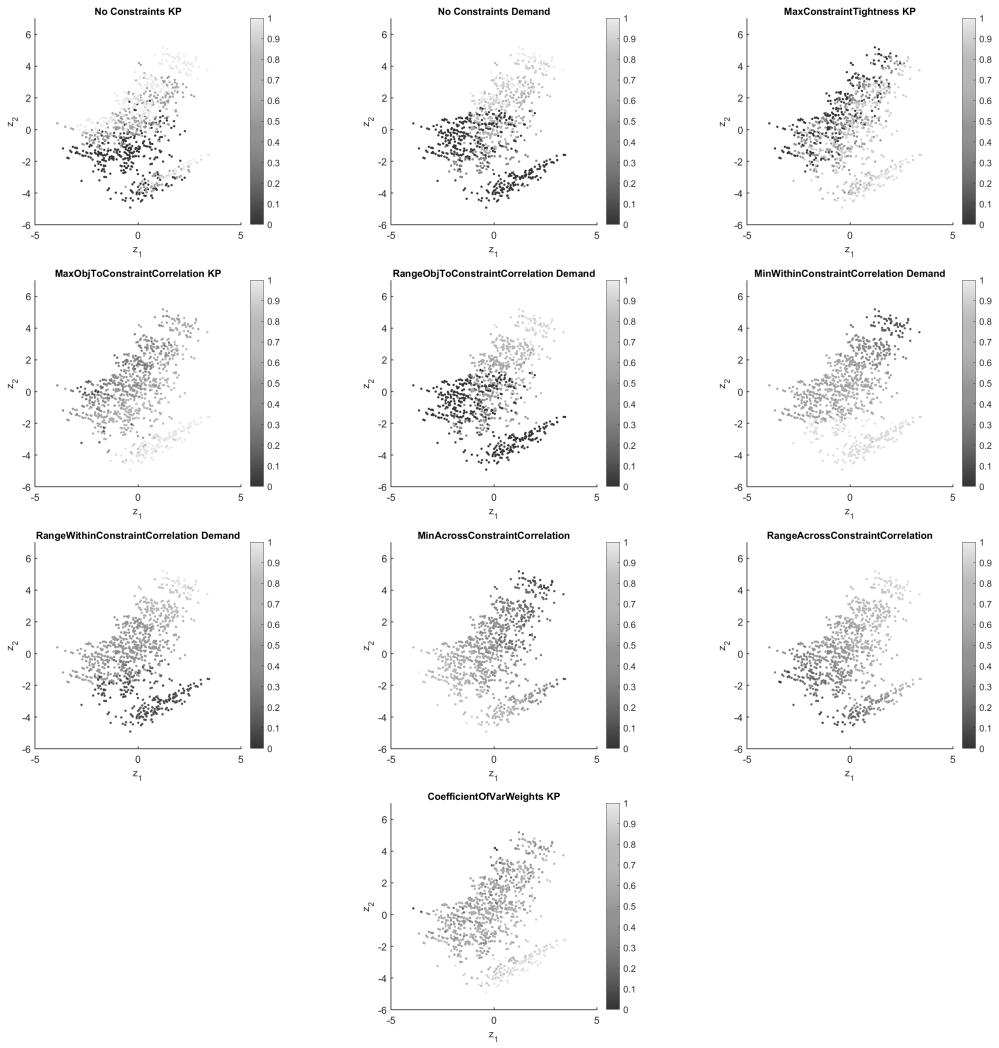

Figure 36. Distribution of features across Scherer, Cho, and Beasley instance space

Bibliography

Ahuja, R. K., Magnanti, T. L. and Orlin, J. B. (1988), 'Network flows'.

Alipour, H., Muñoz, M. A. and Smith-Miles, K. (2023), 'Enhanced instance space analysis for the maximum flow problem', *European Journal of Operational Research* **304**(2), 411–428.

Arntzen, H., Hvattum, L. M. and Løkketangen, A. (2006), 'Adaptive memory search for multidemand multidimensional knapsack problems', *Computers & Operations Research* **33**(9), 2508–2525.

Arthur, J. L. and Frendewey, J. O. (1988), 'Generating travelling-salesman problems with known optimal tours', *Journal of the Operational Research Society* **39**(2), 153–159.

Balas, E. and Zemel, E. (1980), 'An algorithm for large zero-one knapsack problems', *operations Research* **28**(5), 1130–1154.

Beasley, J. E. (1990), 'Or-library: distributing test problems by electronic mail', *Journal of the Operational Research Society* **41**(11), 1069–1072.

Bertsimas, D. and Dunn, J. (2017), 'Optimal classification trees', *Machine Learning* **106**, 1039–1082.

Bierwirth, C., Mattfeld, D. C. and Watson, J.-P. (2004), Landscape regularity and random walks for the job-shop scheduling problem, *in* 'European Conference on Evolutionary Computation in Combinatorial Optimization', Springer, pp. 21–30.

Bowly, S., Smith-Miles, K., Baatar, D. and Mittelmann, H. (2020), 'Generation techniques for linear programming instances with controllable properties', *Mathematical Programming Computation* **12**(3), 389–415.

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**(1), 5–32.

Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984), 'Classification and regression trees. wadsworth int', *Group* **37**(15), 237–251.

Burke, E. K., Mareček, J., Parkes, A. J. and Rudová, H. (2008), Penalising patterns in timetables: Novel integer programming formulations, *in* 'Operations Research Proceedings 2007: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR) Saarbrücken, September 5–7, 2007', Springer, pp. 409–414.

Cacchiani, V., Iori, M., Locatelli, A. and Martello, S. (2022), 'Knapsack problems-an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems', *Computers & Operations Research* p. 105693.

Cappanera, P., Gallo, G. and Maffioli, F. (2003), 'Discrete facility location and routing of obnoxious activities', *Discrete Applied Mathematics* **133**(1-3), 3–28.

Cappanera, P. and Trubian, M. (2005), 'A local-search-based heuristic for the demand-constrained multidimensional knapsack problem', *INFORMS Journal on Computing* **17**(1), 82–98.

Cho, Y. K. (2005), Developing new multidimensional knapsack heuristics based on empirical analysis of legacy heuristics, PhD thesis, Air Force Institute of Technology.

Cho, Y. K., Moore, J. T., Hill, R. R. and Reilly, C. H. (2008), 'Exploiting empirical knowledge for bi-dimensional knapsack problem heuristics', *International Journal of Industrial and Systems Engineering* **3**(5), 530–548.

Chu, P. C. and Beasley, J. E. (1998), 'A genetic algorithm for the multidimensional knapsack problem', *Journal of heuristics* **4**(1), 63–86.

Chung, F. R. and Graham, F. C. (1997), *Spectral graph theory*, number 92, American Mathematical Soc.

De Coster, A., Musliu, N., Schaerf, A., Schoisswohl, J. and Smith-Miles, K. (2022), 'Algorithm selection and instance space analysis for curriculum-based course timetabling', *Journal of Scheduling* pp. 1–24.

Even, S., Itai, A. and Shamir, A. (1975), On the complexity of time table and multi-commodity flow problems, *in* '16th annual symposium on foundations of computer science (sfcs 1975)', IEEE, pp. 184–193.

Garey, M. R. and Johnson, D. S. (1978), '"strong"np-completeness results: Motivation, examples, and implications', *Journal of the ACM (JACM)* **25**(3), 499–508.

Garey, M. R. and Johnson, D. S. (1979), *Computers and intractability*, Vol. 174, Freeman San Francisco.

Glover, F. and Kochenberger, G. A. (1996), Critical event tabu search for multidimensional knapsack problems, *in* 'Meta-heuristics', Springer, Boston, MA, pp. 407–427.

Glover, F. W. and Kochenberger, G. A. (2006), *Handbook of metaheuristics*, Vol. 57, Springer Science & Business Media, New York, NY.

Gortazar, F., Duarte, A., Laguna, M. and Martí, R. (2010), 'Black box scatter search for general classes of binary optimization problems', *Computers & Operations Research* **37**(11), 1977–1986.

Greenberg, H. J. (1990), 'Computational testing: Why, how and how much', *ORSA Journal on Computing* **2**(1), 94–97.

Hall, N. G. and Posner, M. E. (2001), 'Generating experimental data for computational testing with machine scheduling applications', *Operations Research* **49**(6), 854–865.

Hall, N. G. and Posner, M. E. (2007), 'Performance prediction and preselection for optimization and heuristic solution procedures', *Operations Research* **55**(4), 703–716.

Hall, N. G. and Posner, M. E. (2010), The generation of experimental data for computational testing in optimization, *in* 'Experimental methods for the analysis of optimization algorithms', Springer, Boston, MA, pp. 73–101.

Hill, R., Moore, J., Hiremath, C. and Cho, Y. (2011), 'Test problem generation of binary knapsack problem variants and the implications of their use', *Int. J. Operational Quantitative Management* **18**(2), 105–128.

Hill, R. R., Cho, Y. K. and Moore, J. T. (2012), 'Problem reduction heuristic for the 0–1 multidimensional knapsack problem', *Computers & Operations Research* **39**(1), 19–26.

Hill, R. R. and Reilly, C. H. (2000), 'The effects of coefficient correlation structure in two-dimensional knapsack problems on solution procedure performance', *Management Science* **46**(2), 302–317.

Hiremath, C. S. and Hill, R. R. (2013), 'First-level tabu search approach for solving the multiple-choice multidimensional knapsack problem', *international Journal of Metaheuristics* **2**(2), 174–199.

Hooker, J. N. (1994), 'Needed: An empirical science of algorithms', *Operations research* **42**(2), 201–212.

Hooker, J. N. (1995), 'Testing heuristics: We have it all wrong', *Journal of Heuristics* **1**(1), 33–42.

Hu, X., Rudin, C. and Seltzer, M. (2019), 'Optimal sparse decision trees', *Advances in Neural Information Processing Systems* **32**.

Hvattum, L. M., Arntzen, H., Løkketangen, A. and Glover, F. (2010), 'Alternating control tree search for knapsack/covering problems', *Journal of Heuristics* **16**(3), 239–258.

Hvattum, L. M. and Løkketangen, A. (2007), Experiments using scatter search for the multidemand multidimensional knapsack problem, *in* 'Metaheuristics', Springer, Boston, MA, pp. 3–24.

Iman, R. L. and Conover, W. J. (1982), 'A distribution-free approach to inducing rank correlation among input variables', *Communications in Statistics-Simulation and Computation* **11**(3), 311–334.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013), *An introduction to statistical learning*, Vol. 112, Springer.

Kandanaarachchi, S., Muñoz, M. A., Hyndman, R. J. and Smith-Miles, K. (2020), 'On normalization and algorithm selection for unsupervised outlier detection', *Data Mining and Knowledge Discovery* **34**(2), 309–354.

Kang, Y., Hyndman, R. J. and Smith-Miles, K. (2017), 'Visualising forecasting algorithm performance using time series instance spaces', *International Journal of Forecasting* **33**(2), 345–358.

Karmarkar, N. (1984), A new polynomial-time algorithm for linear programming, *in* 'Proceedings of the sixteenth annual ACM symposium on Theory of computing', pp. 302–311.

Karp, R. M. (1972), Reducibility among combinatorial problems, *in* 'Complexity of computer computations', Springer, pp. 85–103.

Kellerer, H., Pferschy, U. and Pisinger, D. (2004*a*), Introduction to np-completeness of knapsack problems, *in* 'Knapsack problems', Springer, Berlin, Heidelberg, pp. 483–493.

Kellerer, H., Pferschy, U. and Pisinger, D. (2004*b*), Knapsack problems, Springer, Berlin, Heidelberg.

Laguna, M. and Martí, R. (2001), 'A grasp for coloring sparse graphs', *Computational optimization and applications* **19**(2), 165–178.

Laguna, M. and Martí, R. C. (2003), *Scatter search: methodology and implementations in C*, Springer Science & Business Media, Boston, MA.

Lai, X., Hao, J.-K. and Yue, D. (2019), 'Two-stage solution-based tabu search for the multidemand multidimensional knapsack problem', *European Journal of Operational Research* **274**(1), 35–48.

Lamine, A., Khemakhem, M. and Chabchoub, H. (2012), 'Knapsack problems involving dimensions, demands and multiple choice constraints: generalization and transformations between formulations', *International Journal of Advanced Science and Technology* **46**, 71–94.

Laurent, H. and Rivest, R. L. (1976), 'Constructing optimal binary decision trees is np-complete', *Information processing letters* **5**(1), 15–17.

Lin, J., Zhong, C., Hu, D., Rudin, C. and Seltzer, M. (2020), Generalized and scalable optimal sparse decision trees, *in* 'International Conference on Machine Learning', PMLR, pp. 6150–6160.

Lopes, L. and Smith-Miles, K. (2010), Pitfalls in instance generation for udine timetabling, *in* 'Learning and Intelligent Optimization: 4th International Conference, LION 4, Venice, Italy, January 18-22, 2010. Selected Papers 4', Springer, pp. 299–302.

Lopes, L. and Smith-Miles, K. (2013), 'Generating applicable synthetic instances for branch problems', *Operations Research* **61**(3), 563–577.

Lu, Y. and Vasko, F. J. (2020), 'A comprehensive empirical demonstration of the impact of choice constraints on solving generalizations of the 0–1 knapsack problem using the integer programming option of cplex®', *Engineering Optimization* **52**(9), 1632–1644.

Martello, S., Pisinger, D. and Toth, P. (1999), 'Dynamic programming and strong bounds for the 0-1 knapsack problem', *Management science* **45**(3), 414–424.

McAndrew, F. (2020), 'Adiabatic quantum computing to solve the maxcut graph problem', *University of Melbourne School of Mathematics* .

McTavish, H., Zhong, C., Achermann, R., Karimalis, I., Chen, J., Rudin, C. and Seltzer, M. (2022), Fast sparse decision tree optimization via reference ensembles, *in* 'Proceedings of the Thirty Sixth AAAI Conference on Artificial Intelligence', pp. 9604–9613.

Muñoz, M. A. and Smith-Miles, K. (2020*a*), 'Generating new space-filling test instances for continuous black-box optimization', *Evolutionary computation* **28**(3), 379–404.

Muñoz, M. A. and Smith-Miles, K. (2020*b*), 'Instance space analysis: A toolkit for the assessment of algorithmic power.'.
URL: *https://doi.org/10.5281/zenodo.4484107*

Muñoz, M. A., Villanova, L., Baatar, D. and Smith-Miles, K. (2018), 'Instance spaces for machine learning classification', *Machine Learning* **107**(1), 109–147.

Muñoz, M. A., Yan, T., Leal, M. R., Smith-Miles, K., Lorena, A. C., Pappa, G. L. and Rodrigues, R. M. (2021), 'An instance space analysis of regression problems', *ACM Transactions on Knowledge Discovery from Data (TKDD)* **15**(2), 1–25.

Orlin, J. B. (1997), 'A polynomial time primal network simplex algorithm for minimum cost flows', *Mathematical Programming* **78**(2), 109–129.

Pisinger, D. (2005), 'Where are the hard knapsack problems?', *Computers & Operations Research* **32**(9), 2271–2284.

Reeves, C. R. (1999), 'Landscapes, operators and heuristic search', *Annals of Operations Research* **86**, 473–490.

Reilly, C. H. (2009), 'Synthetic optimization problem generation: show us the correlations!', *INFORMS Journal on Computing* **21**(3), 458–467.

Rice, J. R. (1976), 'The algorithm selection problem', *Advances in Computers* **15**, 65–118.

Romeijn, H. E. and Morales, D. R. (2001), 'A probabilistic analysis of the multi-period single-sourcing problem', *Discrete applied mathematics* **112**(1-3), 301–328.

Rudin, C. (2019), 'Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead', *Nature machine intelligence* **1**(5), 206–215.

Scherer, M., Hill, R. R., Lunday, B. J., Cox, B. A. and White, E. D. (2023*a*), 'Instance based configuration for metaheuristic selection', *Computers & Operations Research* . Under Second Review.

Scherer, M., Hill, R. R., Lunday, B. J., Cox, B. A. and White, E. D. (2023*b*), 'Verifying new instances of the multidemand multidimensional knapsack problem with instance space analysis', *Computers & Operations Research* . Under Second Review.

Smith-Miles, K. (2019), 'Matilda: melbourne algorithm test instance library with data analytics', *URL: https://matilda.unimelb.edu.au* .

Smith-Miles, K. A., James, R. J., Giffin, J. W. and Tu, Y. (2009), A knowledge discovery approach to understanding relationships between scheduling problem structure and heuristic performance, *in* 'Learning and Intelligent Optimization: Third International Conference, LION 3, Trento, Italy, January 14-18, 2009. Selected Papers 3', Springer, pp. 89–103.

Smith-Miles, K. and Baatar, D. (2014), 'Exploring the role of graph spectra in graph coloring algorithm performance', *Discrete Applied Mathematics* **176**, 107–121.

Smith-Miles, K., Baatar, D., Wreford, B. and Lewis, R. (2014), 'Towards objective measures of algorithm performance across instance space', *Computers & Operations Research* **45**, 12–24.

Smith-Miles, K. and Bowly, S. (2015), 'Generating new test instances by evolving in instance space', *Computers & Operations Research* **63**, 102–113.

Smith-Miles, K., Christiansen, J. and Muñoz, M. A. (2021), 'Revisiting where are the hard knapsack problems? via instance space analysis', *Computers & Operations Research* **128**, 105184.

Smith-Miles, K. and Lopes, L. (2012), 'Measuring instance difficulty for combinatorial optimization problems', *Computers & Operations Research* **39**(5), 875–889.

Smith-Miles, K. and Muñoz, M. A. (2021), 'Instance space analysis for algorithm testing: Methodology and software tools'.

Song, M. S., Emerick, B., Lu, Y. and Vasko, F. J. (2022), 'When to use integer programming software to solve large multi-demand multidimensional knapsack problems: a guide for operations research practitioners', *Engineering Optimization* **54**(5), 894–906.

Toyoda, Y. (1975), 'A simplified algorithm for obtaining approximate solutions to zero-one programming problems', *Management Science* **21**(12), 1417–1427.

Wolpert, D. H. and Macready, W. G. (1997), 'No free lunch theorems for optimization', *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
20–07–2023	Dissertation	March 2020 — September 2023

4. TITLE AND SUBTITLE

Test Problem Generation and Metaheuristic Selection for the Multidemand Multidimensional Knapsack Problem

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Scherer, Matthew E., Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-DS-23-S-020

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Agency of Command Name
POC Rank & Name within Agency
Street Address
City, ST ZIP
email of agency POC

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution A: Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This work focuses on instance generation methods for the multi-demand multidimensional knapsack problem (MDMKP). Specifically, instance space analysis (ISA) is used to characterize the landscape of existing instances and validate the novelty of new instances generated with a novel problem generation method, the primal problem instance generator (PPIG). The instance generator is capable of producing feasible, diverse, and challenging instances by directly controlling the problem features. PPIG contributes to the previous collections of instances and is validated through instance space analysis. The research presents an in-depth empirical evaluation of existing solution procedures for the MDMKP. The portfolio of metaheuristics examined show promising performance on existing benchmark libraries but lack robustness when the test set of instances are extended using the PPIG method. A machine learning classifier is employed to provide an interpretable link between instance configuration and solution procedure performance. The final aspect of the research is an optimization framework used to provide problem generation parameters to the PPIG methodology to further cover the instance space for the full suite of MDMKP test problems.

15. SUBJECT TERMS

Multidemand multidimensional knapsack problem (MDMKP), empirical testing, instance space analysis, instance generation, algorithm selection problem

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Brian J. Lunday, AFIT/ENS
U	U	U	UU	139	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4624; brian.lunday@afit.edu